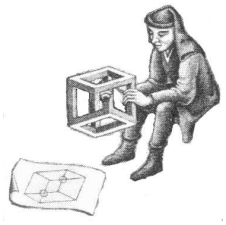
UNIVERSIDAD DE
MURCIA



Sistemas de Percepción y Visión por Computador

Prof. **Alberto Ruiz García**

Dpto. Informática y Sistemas
Facultad de Informática

http://dis.um.es/~alberto

EN CONSTRUCCIÓN. Versión del **2 de febrero de 2015**
Se agradecerá cualquier sugerencia y la notificación de errores.

Índice general

Parte I

Sistemas de Percepción

Capítulo 1

Introducción a la Percepción Artificial

From a syntactic zombie
a semantic and sentient being has emerged!

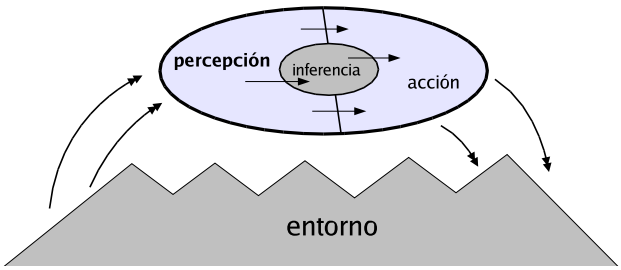
–Hofstadter

¿Qué es la *percepción*? La Real Academia la define como la 'sensación interior que resulta de una impresión material hecha en nuestros sentidos'. También dice que *percibir* es 'recibir por uno de los sentidos las imágenes, impresiones o sensaciones externas'. Curiosamente, percibir también es 'comprender o conocer algo'. Es una de las facultades más sorprendentes de los seres vivos: el sistema nervioso es capaz de interpretar información 'bruta' y transformarla en conceptos abstractos.

Deseamos que las máquinas también se 'den cuenta' de lo que sucede a su alrededor: estamos interesados en dotarlas de algún tipo de *percepción artificial*. Por supuesto, se tratará simplemente de una primera aproximación, muy cruda, a la percepción humana, íntimamente relacionada con la capacidad de *comprender*. Esto se encuentra todavía muy lejos de nuestras posibilidades tecnológicas, incluso en ambientes controlados. El tipo de percepción artificial que podemos conseguir en la actualidad no es más que la detección de propiedades interesantes del entorno. Esta capacidad no es despreciable: nos permite desarrollar aplicaciones prácticas con un impacto cada vez mayor.

1.1. Sistemas Autónomos Artificiales

La figura siguiente muestra un esquema, muy simplificado, de un sistema autónomo que interactúa con su entorno:



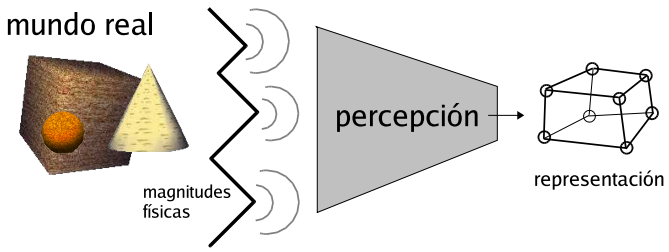
Las flechas indican el flujo de información: los *sensores* captan magnitudes físicas de interés. A partir de ellas, la etapa de percepción trata de obtener, y mantener, una *representación* adecuada del entorno. Tras una etapa de 'razonamiento' más o menos compleja, el sistema 'decidirá' la acción oportuna de acuerdo con sus objetivos (p.ej. modificar el entorno, desplazarse, etc.).

¿Es realmente necesaria una representación abstracta del mundo? Cuando las acciones del sistema dependen directamente de los estímulos decimos que el sistema es *reactivo*. Algunos creen que la acumulación coordinada de comportamientos reactivos elementales puede dar lugar a un comportamiento global que exteriormente puede interpretarse como deliberativo o racional. No construyen representaciones, sostienen que 'el mundo es su propia representación'. El enfoque es atractivo: trata de reducir la complejidad a elementos simples. Sin embargo, los resultados obtenidos hasta el momento (en robots tipo insecto, rebaños, etc., e incluso humanoides [3]) no muestran comportamientos demasiado interesantes. Parece existir una barrera de complejidad que no es fácil atravesar.

Otros piensan que para conseguir un comportamiento inteligente es conveniente mantener algún tipo de representación abstracta explícita, intermedia entre los estímulos y las órdenes de actuación: un modelo del entorno, y del propio sistema dentro de él, que incluya su estado interno, sus objetivos, restricciones, etc.

1.2. La Percepción

La figura siguiente muestra un esquema, también muy simplificado, de la etapa de percepción:



Ante la inmensa complejidad del mundo real nos fijamos en unas ciertas magnitudes físicas, las metemos en una especie de 'embudo' de procesamiento de información y obtenemos una *representación* abstracta. Idealmente la representación debe mantenerse actualizada 'en tiempo real', reflejando continuamente los cambios del entorno. Las múltiples realizaciones posibles de un mismo objeto (o tipo de objeto, o de situación) se reducen a un *concepto*, una 'clase de equivalencia' en la que resumimos su 'esencia', lo que tienen en común.

Es evidente que cualquier representación es una simplificación tremenda de la realidad, que tiene un nivel de detalle potencialmente infinito. La representación se fija solo en los aspectos relevantes para el objetivo del sistema. Si la representación es tan detallada como el entorno pero no está 'estructurada', si no contiene propiedades abstractas, no resuelve el problema (y corremos el riesgo de caer en una regresión infinita), ya que habría que aplicar una nueva etapa de percepción sobre la representación para encontrar en ella los aspectos relevantes.

La percepción trabaja a partir de los *estímulos*, que no son más que medidas de las magnitudes físicas. Los seres vivos codifican los estímulos mediante impulsos nerviosos; en los computadores los digitalizamos y convertimos en variables numéricas. Los estímulos son información bruta, sin elaborar, contaminados con ruido, con elementos irrelevantes, etc. Recibimos un flujo continuo de miles o incluso millones de bits por segundo que tenemos que reorganizar y procesar para extraer las propiedades abstractas que nos interesan en cada momento.

La percepción requiere un gran esfuerzo computacional que, curiosamente, los seres vivos realizamos de manera inconsciente y automática. En la actualidad esta capacidad es difícilmente alcanzable incluso para los más potentes supercomputadores. En contraste, ciertas tareas que son triviales para un computador resultan muy complejas para los seres vivos. La maquinaria biológica está basada en el procesamiento paralelo masivo, que permite considerar simultáneamente un número elevadísimo de hipótesis o conjeturas de interpretación. Algunas se refuerzan, otras se debilitan, hasta que la hipótesis más consistente resulta ganadora. El estilo computacional convencional basado en procesadores muy rápidos pero esencialmente secuenciales no parece el más apropiado para resolver problemas de percepción.

A ver si esto lo mete a derecha.

Los seres vivos reciben la información sobre el mundo exterior en forma de sensaciones 'cualitativas' (*qualia*): sabores, colores, tonos y timbres sonoros, etc., que están

relacionados con magnitudes físicas: composición química, frecuencia de la onda luminosa, espectro de frecuencias de la onda sonora, etc. Podemos incluir también otro tipo de sensaciones, como el dolor, que proporciona información 'urgente' sobre el estado interno del sistema, y otras. Al abordar el problema Mente/Cuerpo desde una perspectiva basada en el procesamiento de información surge una cierta dificultad para explicar este tipo de sensaciones.

1.3. Percepción de Bajo y Alto Nivel

El flujo de información principal es ascendente (*bottom-up*), va desde los estímulos hacia las representaciones. Pero la percepción también utiliza (frecuentemente de manera implícita) cierto conocimiento de las situaciones que pueden ocurrir. No puede trabajar 'en vacío': existe un flujo de información descendente (*top-down*) que orienta y guía la construcción de las representaciones a partir de expectativas, hipótesis o conjeturas razonables.

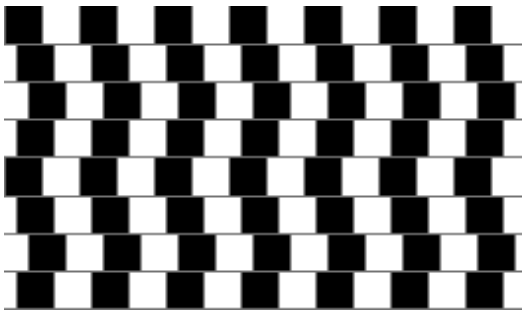
Un ejemplo de procesamiento puramente ascendente se manifiesta en la capacidad de fusionar estereogramas aleatorios, donde las imágenes correspondientes a cada ojo contienen configuraciones aleatorias que no ofrecen ninguna pista sobre el contenido de la imagen. A pesar de ello, el cerebro consigue encontrar correlaciones en la imagen y deducir la profundidad de cada trozo de la escena, que puede ser posteriormente reconocida. Cómo ejemplo, intenta descubrir qué numero hay representado en la figura siguiente: [1]

La existencia del flujo descendente se ilustra en el conocido ejemplo mostrado a

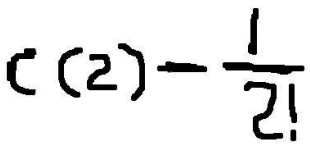
[1] Sitúate a unos 50 cm de distancia y enfoca detrás del papel. (Ten en cuenta que un pequeño porcentaje de la población carece del 'sexto sentido' de la visión estereoscópica y, p. ej., no puede trabajar como piloto de caza…) Este autoestereograma está fabricado con el estilo latex disponible en `http://tex.loria.fr/divers/stereo/index.html`.

continuación, en la que la H y la A se dibujan como una forma intermedia común, que, debido al contexto, se ve en cada palabra como una letra distinta:

'Ver una cosa' es en realidad 'ver algo como esa cosa', es interpretar. Ciertas ilusiones ópticas ponen claramente de manifiesto que existe una 'presión cognitiva' de la hipótesis de interpretación vencedora que inhibe la información ascendente de bajo nivel, afectando –a pesar de todos nuestros esfuerzos conscientes– a nuestra manera de percibir los elementos de una escena. En la figura siguiente no podemos evitar ver muy curvadas líneas rectas y paralelas. [2]



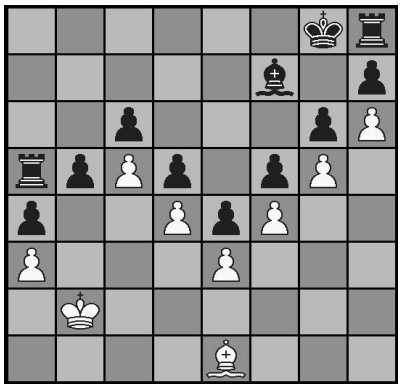
Otro ejemplo interesante es el reconocimiento de expresiones matemáticas manuscritas, muchos de cuyos signos elementales son esencialmente ambiguos:



Su forma y disposición espacial dan pistas sobre los operandos y los operadores y, a su vez, la estructura global de la expresión da pistas sobre la identidad de los símbolos. Aparece la típica explosión combinatoria de interpretaciones posibles. Es preciso combinar los flujos de información ascendente y descendente, que por separado son insuficientes, para obtener eficientemente la interpretación correcta.

La percepción es una de las claves de la inteligencia. Observa la siguiente posicion de ajedrez:

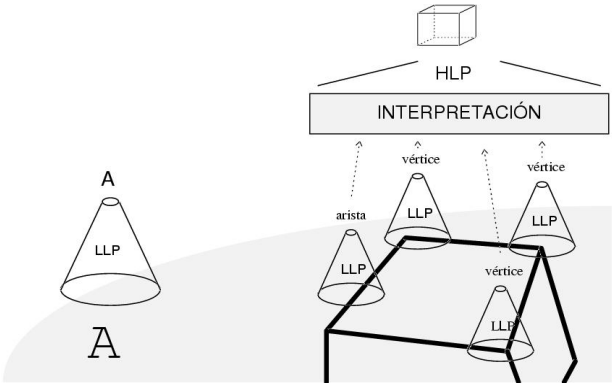
[2] Véase por ejemplo `http://home.wanadoo.nl/hans.kuiper/optillus.htm`.



Se trata de una posición difícil para un programa basado en búsqueda exhaustiva, por fuerza bruta, en el árbol de posibilidades. La ganancia inmediata de material conduce a la derrota. Por el contrario, los ajedrecistas humanos comprenden 'directamente' la naturaleza de una posición, teniendo en cuenta ciertas propiedades abstractas de la configuración de piezas. [3]

Otro ejemplo es la construcción de analogías entre dos situaciones (p.ej. el átomo y el sistema solar), que sólo puede hacerse si existe una gran flexibilidad en la manera de ver una situación, construyendo representaciones que tengan en cuenta los aspectos relevantes entre los que se puede establecer una correspondencia.

Dependiendo del poder expresivo de la representación obtenida, se habla de dos tipos o 'niveles' de percepción. La percepción de *bajo nivel* detecta propiedades sencillas de los estímulos. La percepción de *alto nivel* construye representaciones estructuradas del entorno cuya complejidad no está predeterminada rígidamente por el programador. Las etapas de bajo nivel (LLP) obtienen (en paralelo) conceptos elementales, y las de alto nivel (HLP) imponen estructura a los datos:



[3] Posición tomada de [20]. El análisis del programa *crafty* en mi máquina (hace varios años) el siguiente: depth=24 1/13 -6.97 1. Bxa5 Be6 2. Bd8 Kf7 3. Bc7 Ra8, etc. Nodes: 251373626 NPS: 652850 Time: 00:06:25.04, En 24 niveles y tras analizar (a gran velocidad) más de 250 millones de nodos la evaluación es de casi -7 (muy mala para las blancas), cuando la realidad es que manteniendo la barrera de peones, usando el alfil, se garantiza el empate. La fuerza bruta no consigue captar el concepto de *barrera*.

Cuanto más autónomamente se construya la representación y menos predeterminado esté el resultado, tendremos una percepción de mayor nivel. Si sólo se 'rellenan' variables o etiquetan nodos y arcos de un grafo, la percepción será de bajo nivel.

El problema esencial de la inteligencia artificial es la determinación de los aspectos relevantes de una cierta situación. Esto exige una 'comprensión' del problema que no parece fácil de conseguir si partimos de representaciones predeterminadas por el diseñador del programa. Muchos sistemas de inteligencia artificial tienen éxito porque parten de una representación adecuadamente elegida, de manera que el problema se reduce a un proceso de búsqueda más o menos exhaustiva en un espacio formalizado.

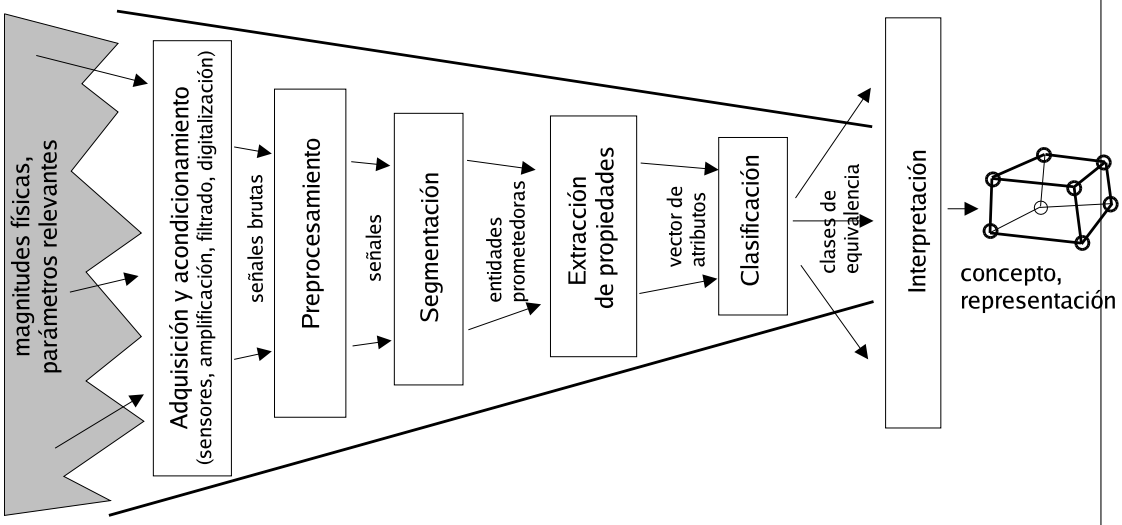
El paso siguiente es conseguir que un sistema elabore automáticamente las representaciones abstractas más apropiadas a partir de los estímulos sensoriales. Deseamos capturar, mediante procesos computacionales, unas relaciones sintácticas lo suficientemente ricas como para dar lugar a una verdadera semántica (comprensión, significado). Algunos piensan que esto es imposible por definición [24, 20]. Otros piensan que un comportamiento externo correcto, independientemente de cómo se lleve a cabo internamente (p. ej., fuerza bruta), es el único objetivo de importancia. Finalmente, hay quien piensa que esa comprensión se puede conseguir, pero probablemente queda todavía un largo camino por recorrer.

Hay que admitir que en la práctica sólo somos capaces de abordar satisfactoriamente problemas de percepción de bajo nivel en condiciones controladas. A medida que se eliminan restricciones y simplificaciones se hace necesaria una comprensión más profunda del problema. La percepción del alto nivel sólo es abordable por el momento en 'microdominios' y los resultados indican que el grado de comprensión alcanzado es todavía muy pequeño. Pero es un primer paso muy valioso en la dirección (creemos) correcta [12].

1.4. Reconocimiento de Modelos

Tras los comentarios anteriores sobre la Percepción con mayúsculas, llega el momento de diseñar sistemas de percepción reales que resuelvan lo mejor posible aplicaciones prácticas. Para esto son de gran utilidad las técnicas de *Reconocimiento de Modelos* (*Pattern Recognition*). Su objetivo es detectar regularidades en los datos con el objetivo de clasificarlos dentro de un conjunto de categorías de interés.

Los sistemas de reconocimiento de modelos descomponen el 'embudo' perceptual en el siguiente conjunto de etapas típicas:



Dependiendo de cada aplicación algunas etapas pueden omitirse, o puede ser necesario añadir algún otro procesamiento específico. En general cada etapa va aumentando progresivamente el grado de abstracción de la información.

Adquisición. Las magnitudes físicas (sonidos, imágenes, etc.) se transforman mediante sensores en señales eléctricas que una vez filtradas, amplificadas y digitalizadas pueden procesarse en el computador. Normalmente utilizaremos equipamiento estándar cuya calidad y prestaciones dependen de nuestra disponibilidad económica y de los avances tecnológicos.

Preprocesamiento. A menudo es conveniente mejorar la calidad de los datos originales (p.ej., para eliminar 'ruido' o entidades irrelevantes). En otros casos interesa transformarlos a una representación más adecuada para su tratamiento matemático (p.ej., el dominio frecuencial es útil en el análisis de la voz). Esta etapa requiere normalmente ciertos conocimientos teóricos sobre la naturaleza de la aplicación. A veces necesitaremos también dispositivos especiales (p. ej., procesadores de señal de alta velocidad), cuyas prestaciones dependerán de nuevo de las disponibilidades económicas y tecnológicas.

Segmentación. Sirve para encontrar dentro del flujo de información los elementos individuales que parecen estar dotados de significado. La segmentación puede ser de tipo temporal, cuando se aislan fragmentos de la secuencia de datos sensoriales recibidos a lo largo del tiempo (p.ej. microfonemas en análisis de voz), o espacial, cuando el conjunto de datos recibidos en un cierto instante contiene varias posibles subunidades relevantes (p.ej. objetos de una escena o partes de un objeto).

Excepto en situaciones triviales en las que el 'fondo' y la 'forma' están claramente diferenciados dentro de la masa de estímulos, la segmentación es en realidad un problema tan complejo como la propia interpretación de la escena. P. ej., en aplicaciones de visión artificial en ambientes poco estructurados existen muchas dificultades para detectar las regiones de interés: ocultaciones y solapamientos parciales, deformaciones, no uniformidad en iluminación, color u otras propiedades visuales, pseudo-objetos irrelevantes, etc. Incluso en situaciones 'de laboratorio' no especialmente complejas, el coste computacional requerido para obtener segmentaciones prometedoras puede resultar enorme, lo que supone un reto para el diseño de sistemas autónomos que operen en tiempo real.

Extracción de propiedades. En esta etapa se calculan propiedades (atributos, características, *features*) de cada entidad segmentada que deben ser, idealmente, discriminantes de las diferentes clases de interés e invariantes a todas sus posibles versiones (p. ej. cambios en posición, tamaño, orientación, intensidad de color, timbre o velocidad de la voz, etc.). Un conjunto de propiedades de mala calidad produce un solapamiento de clases y por tanto una gran probabilidad de error en la clasificación.

Como la anterior, esta etapa también es esencial para el éxito de cualquier sistema de percepción. Desafortunadamente, excepto en casos relativamente sencillos (propiedades lineales o cuadráticas), no existen técnicas automáticas satisfactorias capaces de sintetizar un conjunto reducido de algoritmos que, aplicados a la información original, obtengan resultados discriminantes e invariantes. En la mayoría de los casos el diseñador debe disponer de un conocimiento teórico sobre el problema que le indique propiedades o atributos potencialmente útiles. Posteriormente estas características candidatas sí pueden ser evaluadas y seleccionadas de manera automática.

La dimensión del espacio de propiedades debe ser la menor posible. Como veremos más adelante, es necesario evitar fenómenos de 'sobreajuste' en la etapa de clasificación posterior, usualmente basada en aprendizaje inductivo supervisado, en la que los parámetros ajustables deberán estar sometidos a un número suficiente de restricciones por parte de los ejemplos de entrenamiento disponibles. En caso contrario, no cabe esperar verdadera generalización sobre ejemplos futuros.

La extracción de propiedades adecuadas es esencialmente equivalente al reconocimiento de las categorías de interés.

Clasificación. En esta etapa se decide la categoría más probable (dentro de un conjunto preestablecido) a que pertenece cada observación sensorial elemental caracterizada por su vector de propiedades.

La clasificación tiene un planteamiento matemático bien definido y puede considerarse resuelta desde el punto de vista de la Teoría de la Decisión. Existen diferentes enfoques, entre los que destacamos el probabilístico-estadístico y la optimización de funciones discriminantes. Además, cuando el conjunto de propiedades obtenidas en la

etapa anterior es suficientemente discriminante, la complejidad de esta etapa se reduce sensiblemente. En caso contrario, un diseño correcto del clasificador contribuirá, al menos, a disminuir la proporción de errores.

Los recientes avances en *Máquinas de Kernel*, que estudiaremos más adelante, ofrecen una alternativa muy atractiva para abordar conjuntamente la clasificación –bajo una teoría matemática de la *generalización inductiva*– y la extracción de propiedades, que, curiosamente, se realizará automáticamente, incluso en espacios de propiedades de dimensión infinita (!).

Interpretación. Los conceptos elementales obtenidos en la etapa anterior se organizan en una estructura espacio-temporal requerida por la aplicación. En algunos casos existen algoritmos eficientes (p.ej. programación dinámica), pero en general se basan en algún tipo de búsqueda heurística que trata de evitar la explosión combinatoria de interpretaciones alternativas. En problemas de percepción de bajo nivel esta etapa no suele ser necesaria.

Como vemos, existe una gran variedad en la naturaleza de cada etapa y en el tipo de herramientas matemáticas e informáticas utilizadas para resolverlas. A medida que la información avanza a través de la cadena de proceso las etapas tienden a estar cada vez mejor definidas y a depender menos de la aplicación concreta (excepto la adquisición inicial, que utiliza normalmente dispositivos estándar, y la interpretación, que produce el resultado final del sistema).

Esta descomposición en subtareas aparentemente más sencillas, que trataremos de resolver por separado, es una guía razonable para abordar el diseño de sistemas de reconocimiento. De hecho, la modularización es la manera habitual de atacar la complejidad. Pero no debemos olvidar que las etapas están íntimamente relacionadas unas con otras y no es fácil resolverlas independientemente. Recordemos además que, excepto en casos muy simples, la información se mueve en los dos sentidos, ascendente y descendente. Una de las causas de la naturaleza computacionalmente intratable de la percepción es la dificultad de descomponerla en módulos bien definidos e independientes.

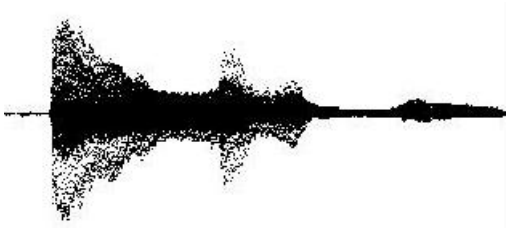
El éxito de un sistema de percepción artificial depende esencialmente de la calidad y eficiencia computacional que podamos conseguir en las etapas de segmentación e interpretación, que dependen de la aplicación concreta y no son fácilmente automatizables. Por el contrario, la clasificación de observaciones dentro de categorías o modelos es un problema resuelto.

Nuestro objetivo es conseguir sistemas de percepción capaces de operar satisfactoriamente en ambientes cada vez menos controlados.

1.5. Ejemplo: análisis de voz

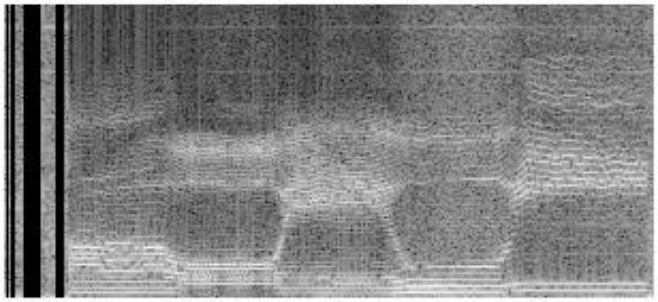
Para ilustrar el tipo de operaciones que se realizan en cada una de las etapas del reconocimiento de modelos, consideremos un sistema –idealizado y muy simplificado– de reconocimiento acústico de vocales. Vamos a suponer que estamos interesados en analizar la señal acústica capturada por un micrófono para detectar la secuencia de vocales pronunciadas por una persona. (Puede establecerse el convenio de que las vocales pronunciadas entre trozos de silencio se escriban en líneas distintas de un fichero de texto.)

La etapa de *adquisición* consiste en comunicarse con la tarjeta de sonido del computador para abrir el dispositivo de captura de onda en el modo adecuado, y hacer lo necesario para acceder a la zona de memoria donde se va guardando la señal acústica muestreada:

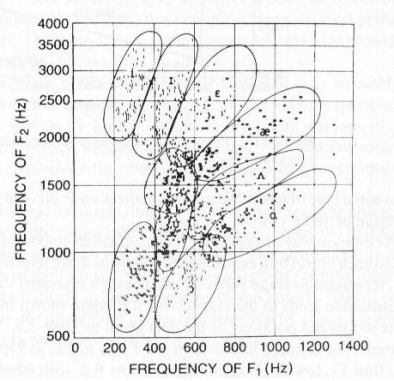


En una etapa de (pre)*segmentación* detectamos los trozos de señal acústica separados por silencio. Esto puede hacerse calculando una medida de la energía de la señal (esencialmente su varianza) en pequeños intervalos de tiempo. Cuando la señal acústica parece contener sonido tomamos una secuencia de *arrays* de, p. ej., 128 ó 256 muestras, cada uno de ellos correspondiente a unos pocos milisegundos de sonido, y los enviamos a la siguiente etapa. Si las vocales se pronunciaran separadas por silencios la segmentación se podría realizar completamente en este momento. Pero como hemos permitido la concatenación de vocales debemos esperar hasta la etapa de interpretación.

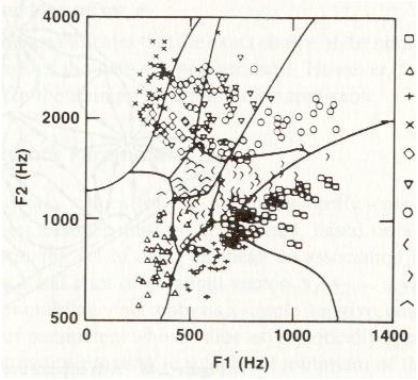
En la etapa de *preprocesamiento* transformamos cada trozo elemental de sonido al dominio de la frecuencia, calculando su transformada de Fourier y manipulándola adecuadamente para obtener el espectro de frecuencias, que proporciona información relevante sobre la naturaleza del sonido (timbre, armónicos, etc.).



En la etapa de *extracción de propiedades* calculamos las dos frecuencias predominantes (*formantes*) del espectro anterior (más adelante estudiaremos otras propiedades más interesantes). De esta forma, el flujo continuo de voz recogido por el micrófono se transforma en una nube de puntos dentro de un espacio vectorial bidimensional correspondientes a las dos propiedades seleccionadas (véase [22]):



La etapa de *clasificación* analiza el espacio de propiedades para determinar las regiones en las que tienden a aparecer los vectores de propiedades de cada vocal (en este caso del idioma inglés). Partiendo de un conjunto representativo de muestras 'etiquetadas' fabricamos una máquina (algoritmo) de clasificación cuya misión es simplemente determinar en qué región del espacio de propiedades se encuentra cada uno de los trozos de sonido:



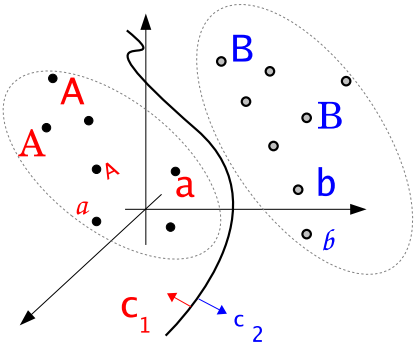
Un método muy simple es caracterizar las regiones simplemente por el valor medio de los vectores de cada clase (prototipos), y clasificar mediante un algoritmo de mínima distancia. Si las regiones de decisión son más complicadas podemos utilizar algoritmos de clasificación más elaborados como los que estudiaremos en capítulos posteriores.

Finalmente, en la etapa de *interpretación* tenemos que analizar la secuencia de trozos de sonido etiquetados por la etapa de clasificación para determinar las vocales que se han pronunciado (p. ej., algo como `aaeaa?e??eeeaeee` puede indicar la secuencia 'A, E'). Obsérvese que esta interpretación final es la que realmente nos permite segmentar correctamente el estímulo en una secuencia de vocales.

1.6. Espacio de Propiedades

Existen dos alternativas para representar la información recogida por los sistemas de reconocimiento. La primera es utilizar estructuras recursivas generales (listas, lógica de predicados, grafos, etc). Estas representaciones son muy flexibles pero su manipulación puede requerir algoritmos poco eficientes (técnicas de matemática discreta, combinatoria, procesamiento simbólico, etc.). Se utilizarán normalmente en la etapa de interpretación.

La segunda opción es limitarnos a un conjunto fijo y predeterminado de propiedades (p. ej., atributos binarios o multivaluados, discretos o continuos, señales muestreadas, arrays multidimensionales, etc.), de manera que la información se representa mediante puntos o vectores en un cierto espacio multidimensional. Las diferentes versiones o realizaciones de un objeto físico se corresponden con diferentes puntos en el espacio de propiedades:



Esta representación es más adecuada para la percepción de bajo nivel, donde podemos aprovechar potentísimas herramientas matemáticas: álgebra, estadística, optimización, etc.

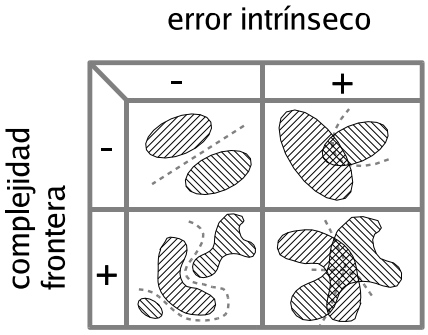
Las *observaciones* son puntos o vectores en ese espacio. Los *ejemplos* son observaciones etiquetadas con la clase a que pertenecen. Las *regiones de decisión* son las zonas del espacio donde suelen caer los ejemplos de cada clase. *Clasificar* es determinar en qué región ha caído una observación. *Aprender* es encontrar y, si es posible, describir adecuadamente, las regiones de cada clase, estableciendo las *fronteras de decisión.*

La clave del reconocimiento de modelos está en encontrar un espacio de propiedades donde los objetos de clases distintas aparezcan muy separados.

Error intrínseco. La complejidad de las fronteras de decisión depende de la forma de las regiones de cada clase. En algunos casos se pueden describir de manera matemáticamente sencilla (p. ej., hiperplanos), y en otros tenemos que recurrir a expresiones –en general, algoritmos– más complejos.

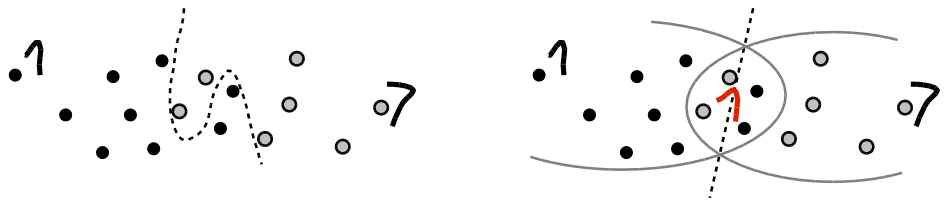
Pero, independientemente de la complejidad del procedimiento de decisión (de la forma de la frontera), existe una característica muy importante de un problema de reconocimiento: su ambigüedad o *error intrínseco*. Cuando las propiedades observadas son de mala calidad (es decir, no son 'discriminantes', no separan bien las clases), existirán regiones ambiguas en las que pueden aparecer observaciones de diferentes clases. En este caso cualquier método de clasificación sufrirá errores inevitables. El tamaño de la zona de *solapamiento* entre regiones nos indica lo bien o mal que puede resolverse la etapa de clasificación.

La complejidad de la frontera y el error intrínseco son dos aspectos diferentes: podemos encontrar cuatro tipos de problemas: fáciles y sin error, fáciles con error, difíciles sin error, y lo peor de todo, difíciles con error. (Los problemas reales se encontrarán en algún punto intermedio de esos cuatro extremos.)



El solapamiento de regiones puede deberse a dos posibles causas. Una es la propia ambigüedad de la información original. P. ej., no es posible distinguir la letra O del número cero, a menos que utilicemos ceros con barra. El número siete manuscrito también lleva una barra para distinguirlo del número uno. Sin ella, a ciertos ángulos e inclinaciones ambos números podrían confundirse. Este tipo de solapamiento no puede remediarse. La única esperanza es que la etapa de interpretación elimine la ambigüedad. La otra posible causa es que las propiedades elegidas en la etapa anterior sean poco discriminantes y sea necesario encontrar otras mejores.

Hemos dicho que la complejidad de la frontera y el error intrínseco son aspectos diferentes de cada problema de reconocimiento. Es cierto, pero debemos añadir que son aspectos relacionados. Ambos dependen de la calidad de las propiedades elegidas: debemos evitar el solapamiento y tratar de conseguir fronteras sencillas. Ahora bien, uno de los resultados más interesantes del aprendizaje automático en el contexto del reconocimiento de modelos es que, en aplicaciones reales en las que la cantidad de ejemplos es necesariamente finita, complejidad y ambigüedad son, en cierto sentido, indistinguibles. Un aparente solapamiento de regiones puede no ser tal, si permitimos que la frontera se curve de la manera adecuada dentro de la región contraria para englobar los ejemplos que parecen estar al otro lado. Pero esto es peligroso porque corremos el riesgo de perseguir fluctuaciones aleatorias:



Nos encontramos con el problema esencial de la cantidad de interpolación que permitimos a nuestras funciones de ajuste. En el capítulo 4 veremos cómo abordar

este problema. En cualquier caso, se manifiesta claramente la necesidad de validar el resultado del ajuste con ejemplos independientes.

1.7. Aprendizaje Automático

El reconocimiento de modelos, especialmente la etapa de clasificación, se apoya en técnicas de *aprendizaje automático* (una denominación atractiva de técnicas más o menos conocidas de estimación de parámetros, optimización, aproximación de funciones, etc.). Si disponemos de conocimiento teórico del dominio de trabajo (en forma de modelos analíticos o, al menos, reglas aproximadas o heurísticas) podemos intentar una solución directa de la clasificación, basada en programación explícita. P. ej., en ambientes con poco ruido podríamos clasificar polígonos a partir del número de sus vértices (hay que tener en cuenta que, en determinadas circunstancias, detectar lo que es verdaderamente un vértice en una imagen digital no es en absoluto algo sencillo...).

Desafortunadamente, en muchos casos la relación entre los conceptos de interés y las posibles realizaciones de los estímulos es desconocida. Nos vemos obligados a recurrir a métodos 'empíricos', que analizan un conjunto de ejemplos resueltos tratando de encontrar alguna regularidad en los estímulos que pueda explotarse para predecir la clase a que pertenecen. Estas técnicas presentan la ventaja de que son independientes de la aplicación concreta y proporcionan soluciones aceptables en problemas prácticos. Sin embargo, las relaciones entrada / salida empíricas obtenidas no suelen arrojar ninguna luz sobre los fenómenos subyacentes.

Podemos distinguir dos tipos de aprendizaje para el reconocimiento de modelos. Un enfoque simbólico, sintáctico o estructural, orientado hacia la percepción de alto nivel y la interpretación (p. ej., el aprendizaje de gramáticas para describir las relaciones entre las partes de un objeto). Y un enfoque 'numérico', orientado hacia la percepción de bajo nivel y al procesamiento sensorial inicial, en el que podemos distinguir dos variantes (íntimamente relacionadas): el probabilístico / estadístico y el de optimización / ajuste.

El problema de la inducción. La inferencia es un proceso por el cual obtenemos información a partir de otra información. Hay varios tipos: la deducción sirve para obtener información cierta a partir de premisas ciertas (conclusiones a partir de hipótesis o premisas). Es la única forma de inferencia lógicamente válida. La abducción o diagnóstico sirve para obtener hipótesis prometedoras a partir de conclusiones ciertas. La analogía sirve para interpretar una situación mediante un isomorfismo aproximado con otra situación que ya está interpretada. La *inducción* tiene como objetivo obtener reglas generales a partir de casos particulares.

La validez de la inducción plantea un interesante problema filosófico relacionado con el *método científico*. Recientemente se han realizado aportaciones muy relevantes

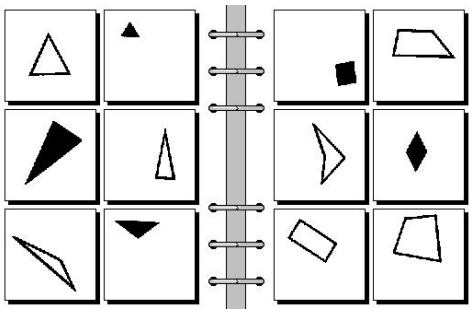
dentro de la Teoría Computacional del Aprendizaje (las revisaremos en el capítulo dedicado a las Máquinas de Kernel), que nos permiten caracterizar las situaciones en las que podemos confiar en el aprendizaje inductivo. La idea esencial es que la máquina de aprendizaje debe evitar la 'memorización'.

Aprendizaje por fuerza bruta. Los sistemas de reconocimiento de modelos basados en aprendizaje automático analizan miles de versiones de un objeto para encontrar lo que tienen en común. A pesar de que la condición que deben cumplir para funcionar correctamente es la de no ser memorísticos (evitando la pura interpolación), lo cierto es la cantidad de ejemplos resueltos necesarios para obtener resultados estadísticamente significativos es absurdamente grande:

Parece que no se consigue ninguna comprensión del dominio de trabajo. La información se manipula ciegamente, independientemente de las propiedades abstractas que pudieran ser relevantes y aprovechables. P. ej., en el caso del reconocimiento de dígitos de la figura anterior, la máquina de aprendizaje obtendrá exactamente los mismos resultados tanto si parte de las imágenes originales como de una permutación aleatoria de los pixels (igual para todas las imágenes), en la que se pierde la coherencia espacial –vecindades– de los mismos.

La figura siguiente muestra un ejemplo del tipo de problemas de reconocimiento de modelos conocido como Problemas de Bongard. [4]. ¿Qué propiedad tienen todos los objetos de la la izquierda, que no cumple ninguno de los de la derecha?

[4] Es el nº 6 (existen cientos de ellos, de muy diferente complejidad). Para más información sobre los problemas de Bongard véase la página web: `http://www.cs.indiana.edu/hyplan/hfoundal/research.html`



Se nos presentan seis instancias positivas de un concepto geométrico y otra seis negativas. En general, ese pequeño número es suficiente para poner claramente de manifiesto el concepto de que se trata; la capacidad de reconocerlo depende mucho más de nuestra capacidad para seleccionar los aspectos relevantes de la figura que de el número de instancias disponibles.

Este tipo de problemas entran de lleno en el dominio de la percepción de alto nivel. No pueden abordarse únicamente a partir de información 'cruda' –pixels de imagen–, sino que es necesario construir dinámicamente la representación más adecuada. En cualquier caso, sería deseable capturar conceptos como la forma de una letra a partir de un número reducido de casos típicos. P. ej., no parece necesario analizar miles de imágenes del nuevo símbolo del *Euro* para reconocerlo correctamente en diferentes estilos o tipos de letra…

1.8. Aplicaciones

Las técnicas de Reconocimiento de Modelos, basadas en aprendizaje automático, permiten abordar numerosas aplicaciones prácticas: P. ej., a partir de imágenes podemos estar interesados en reconocer caracteres manuscritos o impresos, formas geométricas, caras, expresiones faciales, texturas, defectos, objetos, etc. A partir de onda acústica nos gustaría reconocer la identidad del hablante, palabras sueltas, habla continua, etc. A partir de mediciones de vibraciones, fuerzas, temperatura, etc., podemos detectar averías en máquinas, motores, etc., de manera 'no intrusiva'. Mediante radar o ecos ultrasónicos podemos no sólo medir distancias, sino también estimar la forma de objetos, analizando la forma y propiedades del eco (envolvente, etc.). Es posible detectar algunas enfermedades a partir de parámetros fisiológicos (p. ej., de la forma del electrocardiograma).

Las aplicaciones de este tipo deben cumplir una serie de requisitos:

Que la información pueda representarse de manera natural en un espacio de propiedades de dimensión fija, dando lugar a fronteras de decisión 'suaves' (es decir, que el movimiento infinitesimal de la mayoría de los ejemplos no modifique su clase; no tiene sentido permitir situaciones en las que, p. ej., la clasificación dependa del hecho de que un cierto atributo numérico sea racional o irracional).

Que podamos hacer un diseño experimental controlado, basado en muestras aleatorias *i.i.d.* (independientes e idénticamente distribuidas), en las que aparezcan todas las versiones de cada clase, con condiciones de contexto predeterminadas y fijas (p. ej., es necesario especificar si se permite o no la rotación de figuras en el reconocimiento de formas), y sin cambios entre las fases de diseño y uso.

Que seamos capaces de evitar el fenómeno de *sobreajuste* (*overtraining*, subdeterminación, memorización, etc.), que ocurre cuando el número de parámetros libres que hay que ajustar es superior al de restricciones proporcionadas por los ejemplos de entrenamiento. Más adelante estudiaremos las condiciones que debemos cumplir para evitar la interpolación excesiva. De nuevo aparece la necesidad de una etapa final de validación.

Los problemas de percepción artificial más complejos requieren un tratamiento especial. P. ej., en la parte II estudiaremos las particularidades de la *visión por computador*. En realidad, la cadena de proceso típica del reconocimiento de modelos es únicamente orientativa. El diseño de nuevas aplicaciones requerirá técnicas específicas y en decisiones de sentido común difíciles de anticipar.

Capítulo 2

Preprocesamiento y Extracción de Propiedades

Las etapas de preprocesamiento y de extracción de propiedades son fundamentales para el éxito de cualquier sistema de reconocimiento de modelos. Desafortunadamente, dependen muchísimo de las particularidades de cada aplicación, por lo que su diseño no puede automatizarse completamente. A pesar de todo, existen algunas técnicas útiles, ‘de propósito general’, que explicaremos en este capítulo. También comentaremos el preprocesamiento utilizado en varias aplicaciones típicas, con la esperanza de que las ideas esenciales sirvan de inspiración para abordar nuevas situaciones.

Nos enfrentamos a un problema de *reducción de dimensión*. El inmenso flujo de estímulos recibido debe ‘condensarse’ en forma de un pequeño conjunto de propiedades significativas. Tenemos que encontrar una transformación desde un espacio de dimensión muy elevada hasta otro de dimensión pequeña, con la menor pérdida posible de información ‘útil’.

La reducción de dimensión puede abordarse mediante dos métodos: *selección* de propiedades, que simplemente descarta del conjunto de propiedades original aquellas que no son útiles para el objetivo del sistema, y *extracción* de propiedades, que fabrica propiedades nuevas a partir de la información disponible.

2.1. Selección de Propiedades

Es un problema de búsqueda combinatoria. Dada una medida de calidad de un conjunto de propiedades (p. ej., el error de clasificación obtenido por un método determinado sobre un conjunto de ejemplos de prueba), tenemos que encontrar el subconjunto que posee mayor calidad. Como hay un número exponencial de subconjuntos, este tipo de selección ‘global’ de propiedades es computacionalmente intratable.

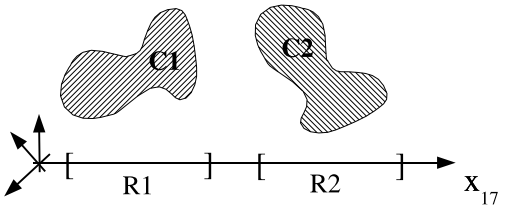
> Si la medida de calidad es ‘monótona’ (esto significa que la incorporación de una nueva propiedad no empeora la calidad de un subconjunto, lo que cumplen

algunos algoritmos de clasificación, pero no todos) se pude realizar una búsqueda eficiente, ordenada, basada en *branch and bound*. (Se construye un árbol de búsqueda eliminando propiedades en cada rama; no se desciende por ramas en las que se garantiza un empeoramiento de la calidad.)

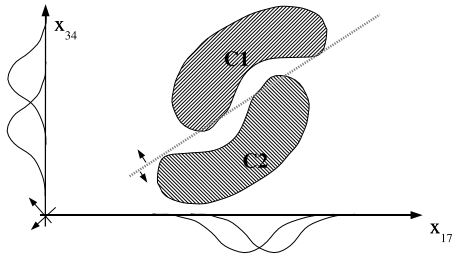
Una alternativa es la selección 'individual': elegimos una medida de calidad de cada propiedad por sí misma (sencilla de calcular y que indique lo mejor posible la 'separación' de las clases de interés), y nos quedamos con las mejores propiedades que sean, a la vez, estadísticamente independientes.

Como medida de calidad individual se puede usar el criterio de Fisher: $|\mu_1 - \mu_2|/(\sigma_1 + \sigma_2)$, que funciona bien cuando las observaciones de cada clase se distribuyen en regiones 'unimodales'. En situaciones más generales es conveniente utilizar una medida no paramétrica (válida para cualquier distribución), como, p. ej., el estadístico del test de Kolmogorov-Smirnov (basado en la máxima diferencia de las distribuciones acumuladas empíricas), o el número de 'cambios' de clase en la secuencia ordenada de muestras, usado en el llamado *Run Test*. Por otro lado, la verdadera independencia estadística no es fácil de determinar. En la práctica se usa el coeficiente de correlación lineal (ver Sección C.2) o, mejor, alguna medida no paramétrica de dependencia como el coeficiente de correlación de rangos (Spearman), en el que las observaciones se sustituyen por su número de orden.

La selección individual de propiedades es, aparentemente, una idea muy prometedora. La calidad del conjunto será tan buena, al menos, como la del mejor componente, y, posiblemente, mejor. Si encontramos una propiedad discriminante, el problema de clasificación puede considerarse resuelto:



El problema está en que la selección individual puede descartar propiedades poco informativas por separado pero que usadas en combinación son altamente discriminantes:



Además, la selección individual produce a veces resultados poco intuitivos: p. ej., dadas tres propiedades p_1, p_2 y p_3, ordenadas de mayor a menor calidad individual, existen situaciones (debido a las dependencias entre ellas) en que el conjunto $\{p_1, p_2\}$ es peor que el conjunto $\{p_2, p_3\}$.

Las medidas de calidad individual pueden ser útiles para fabricar árboles de clasificación.

2.2. Extracción de Propiedades Lineales

La extracción de propiedades lineales es un método excelente para reducir la dimensión de objetos con cierta 'estructura'. (Las propiedades no lineales son más problemáticas, y, a la vista de los recientes avances en *Máquinas de Kernel* (ver Sec. 4.7), no tienen ya demasiado sentido.)

Dado un vector $\boldsymbol{x} = \{x^1, x^2, \ldots, x^n\}$, una función lineal de $\boldsymbol{x}$ es simplemente el producto escalar de $\boldsymbol{x}$ por un cierto (co)vector $\boldsymbol{a}$:

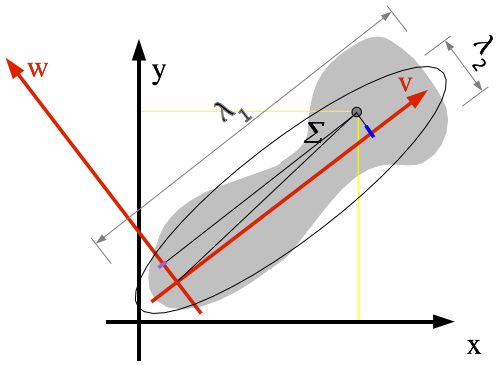
$$f_{\boldsymbol{a}}(\boldsymbol{x}) = \boldsymbol{a}^{\mathsf{T}}\boldsymbol{x} = a_1 x^1 + a_2 x^2 + \ldots + a_n x^n$$

El significado geométrico de una función lineal está relacionado con la proyección del vector $\boldsymbol{x}$ sobre la dirección definida por $\boldsymbol{a}$ (ver Sec. A.1). Las propiedades de una observación que se pueden expresar mediante funciones lineales son las más simples (no triviales) posible: tienen en cuenta cada una de las componentes con un coeficiente o peso que pondera su importancia y ajusta las escalas de las medidas.

La información original se puede condensar atendiendo a dos criterios: expresividad y discriminación. El primero trata de extraer información de las observaciones como tales, que permita *reconstruirlas* de nuevo a partir de la representación condensada. (Podríamos pensar, p. ej., en un compresor del tipo `jpeg`. El problema es que la información comprimida en ese formato es poco apropiada para un procesamiento posterior.) El segundo criterio se preocupa de la información que permita *distinguir* las diferentes clases de observaciones en que estamos interesados.

2.2.1. Propiedades Más Expresivas (MEF) (PCA)

Imaginemos una distribución de observaciones que está bastante 'aplastada' en el espacio:



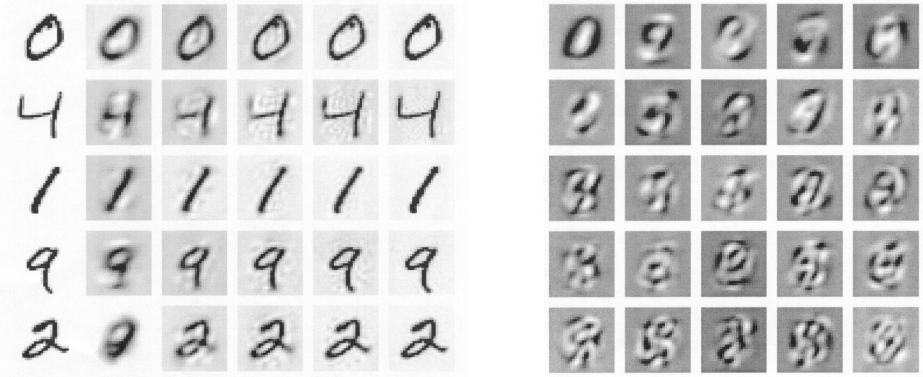
En estas condiciones, mediante un cambio de base que gire los ejes hasta alinearlos con las direcciones principales de la nube de puntos (ver Sec. C.2) conseguimos que las nuevas coordenadas correspondientes a las direcciones de menor varianza tengan siempre valores muy pequeños que podemos descartar. Las coordenadas de las direcciones principales, las 'más expresivas', son suficientes para reconstruir una versión aproximada del vector original.

Si la población tiene media $\boldsymbol{\mu}$ y matriz de covarianza $\mathtt{\Sigma} = \mathtt{V}^\top \mathtt{\Lambda V}$, y siendo $\mathtt{V}_p$ la matriz cuyas filas son los p vectores propios con mayor autovalor, la transformación de compresión es $\boldsymbol{y} = \mathtt{V}_p(\boldsymbol{x} - \boldsymbol{\mu})$ y la de descompresión es $\boldsymbol{x} \simeq \mathtt{V}_p^\top \boldsymbol{y} + \boldsymbol{\mu}$. El error promedio de reconstrucción es la suma de los valores propios (varianzas) de las direcciones principales descartadas.

Esta idea se conoce también como *análisis de componentes principales* (PCA), o transformación de *Karhunen-Loeve*. Muchos otros métodos de compresión de información utilizan el concepto de cambio a una nueva base en la que no todas las coordenadas son necesarias: representación frecuencial, compresión de imágenes JPEG, etc.

Ejemplo. La figura de la izquierda muestra la calidad de la reconstrucción de algunos elementos del conjunto de imágenes de dígitos manuscritos de la base de datos MNIST [1] con un número creciente de componentes principales:

[1] Disponible íntegramente en `http://yann.lecun.com/exdb/mnist`. En la página web de la asignatura existe un subconjunto de 5000 muestras en formato de texto plano.

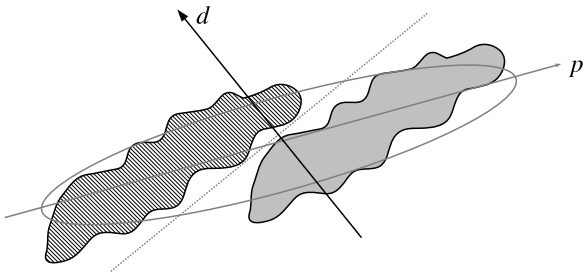


El error de reconstrucción y el número de componentes utilizados son respectivamente 100 % (784), 50 % (10), 80 % (41), 90 % (82), 95 % (141), 99 % (304).

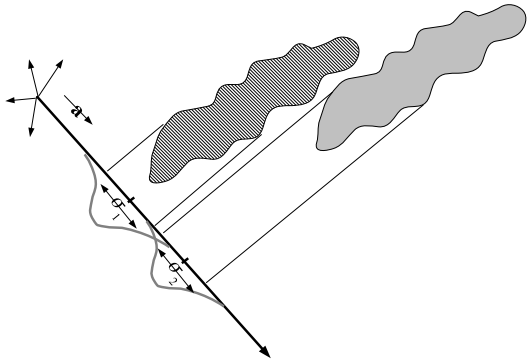
La figura de la derecha muestra algunas *autoimágenes*, que son los vectores propios principales de la población, representados como imágenes. Combinándolos linealmente podemos sintetizar o aproximar con la precisión deseada cualquier ejemplar.

2.2.2. Propiedades Más Discriminantes (MDF)

Las propiedades más expresivas no siempre son apropiadas para *distinguir* objetos. La figura siguiente muestra dos tipos de objetos cuya distribución conjunta tiene una dirección principal global p sobre la que las proyecciones de cada clase quedan muy solapadas:



Nos interesa una dirección d en la que las proyecciones de las clases queden separadas. Cada dirección $\boldsymbol{a}$ da lugar a una poblaciones proyectadas cuya separación se puede cuantificar por ejemplo mediante el criterio de Fisher $F = \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2}$.

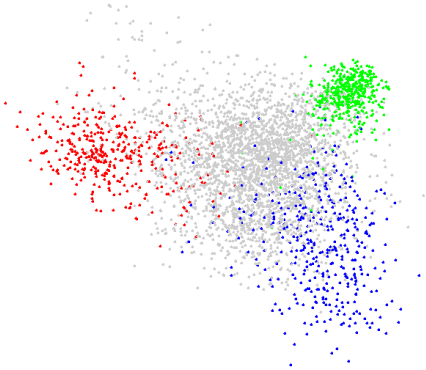


Se puede demostrar que las direcciones más 'discriminantes' (separadoras) en el sentido de Fisher son los vectores propios principales de la matriz $\Sigma^{-1}\Sigma_\mu$, donde Σ_μ es la matriz de covarianza *de las medias* de las clases y Σ es la matriz de covarianza de la población completa. Intuitivamente esa expresión tiene sentido: si hubiera dos clases esféricas, Σ_μ está relacionada con la dirección que une las dos medias, que es la más discriminante. Si no lo son, Σ^{-1} producirá el necesario efecto normalizador. Dadas n clases toda la información de discriminación lineal está contenida en las $n-1$ direcciones principales de $\Sigma^{-1}\Sigma_\mu$.

Si la distribución global está *degenerada* (lo que siempre ocurrirá si tenemos menos observaciones que coordenadas (ver Sec. C.2) no es posible calcular las componentes más discriminantes (Σ es singular). En estos casos puede ser razonable calcular las componentes más discriminantes a partir de un conjunto suficientemente amplio de las más expresivas.

Es importante tener en cuenta que, por su naturaleza lineal, si se usa un buen clasificador que tenga en cuenta la dispersión de las clases, las $n-1$ componentes más discriminantes pueden no aportar ninguna mejora frente a un número suficientemente grande de componentes principales (más expresivas).

Ejemplo. Continuando con el ejemplo anterior, veamos un mapa 2D de los dígitos '0' (rojo),'1' (verde) y '2' (azul) en el espacio de las 2 direcciones más discriminantes de esas tres clases (extraídas de las 20 más expresivas). En gris se muestra el resto de los dígitos.



Ejercicios:

- Intenta reproducir los resultados anteriores.
- Prueba algunos clasificadores sencillos.
- Analiza los autovectores de alguna base de datos de caras, u otro tipo de objetos interesante, disponible en la red.

En todas las prácticas y ejercicios propuestos es conveniente aprovechar, en la medida de lo posible, entornos de cálculo científico como *Matlab/Octave* o *Mathematica* (ver Sec. F).

2.2.3. Componentes Independientes

Cuando expresamos una población aleatoria en la base de direcciones principales conseguimos que las nuevas coordenadas estén descorreladas (tengan covarianza cero). Pero esto no significa que sean independientes. Esto requiere la condición mucho más restrictiva de que la densidad conjunta se pueda descomponer como el producto de las marginales: $p(x, y) = p(x)p(y)$. Encontrar una transformación de variables que describa un problema con componentes independientes es muy útil (p.ej. para separar señales mezcladas).

Si la transformación permitida es lineal entonces el problema se reduce a encontrar una rotación de la muestra "ecualizada" (C.2) que maximice un cierto criterio de calidad que mida la independencia. Un criterio sencillo es la "anormalidad" (la diferencia de cualquier estadístico (p.ej. kurtossis) respecto a lo que obtendría una distribución gaussiana). La justificación es que cualquier combinación lineal de variables aleatorias es más gaussiana que sus ingredientes (Teorema Central del Límite).

→ aplicaciones

2.2.4. Compressed Sensing

2.3. Reconocimiento de Formas

Una vez descritos algunos métodos de reducción de dimensión de propósito general vamos a prestar atención a algunos sistemas de percepción artificial específicos. El reconocimiento de formas (siluetas) es un problema bien definido que entra de lleno en nuestra área de interés. Debemos resolverlo de manera elegante y eficiente antes de abordar situaciones más complejas.

No es un problema trivial. Si partimos de vectores de propiedades que son directamente los pixels de las imágenes binarias que contienen las siluetas de las figuras, la tarea de clasificación requerirá conjuntos de aprendizaje de tamaño enorme para conseguir resultados aceptables. La idea de realizar un simple 'submuestreo' de las imágenes puede conducir a una pérdida de detalle inadmisible. Es necesario un preprocesamiento adecuado de la información original.

Las siluetas son secuencias de puntos que podemos considerar como señales complejas (para 'empaquetar' las dos coordenadas del plano en una única entidad matemática) periódicas. Esto justifica una representación *frecuencial* unidimensional (véase la Sección B): el contorno de una figura puede considerarse como la superposición de varios contornos básicos de diferentes frecuencias (número de 'oscilaciones' en el recorrido completo).

Ya vimos que en todo sistema de reconocimiento hay que especificar lo mejor posible el tipo de transformaciones admisibles, frente a las que las propiedades elegidas deben ser invariantes. En este caso deseamos distinguir formas independientemente de su posición en el plano, de la rotación que hayan podido sufrir, de su tamaño, del punto de partida del contorno, de la distancia concreta entre los puntos del contorno (siempre que estén regularmente espaciados[2]) y, por supuesto, invariantes a pequeñas deformaciones.

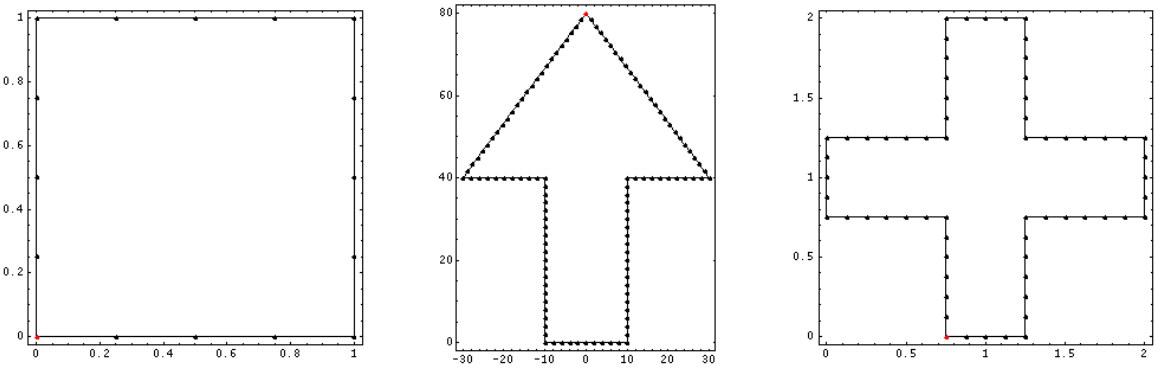
Las componentes frecuenciales obtenidas directamente de la Transformada Discreta de Fourier (ver Sec. B.8) del contorno, F_i, $i = 0..n-1$, no son directamente invariantes a las anteriores transformaciones. Sin embargo es sencillo calcular propiedades derivadas d_i que sí lo son. Para ello es conveniente tener en cuenta la interpretación geométrica de las componentes frecuenciales (B.4) como superposición de elipses. Un conjunto muy sencillo de invariantes es el siguiente:

- Calculamos el valor absoluto de cada componente frecuencial, eliminando la fase, y por tanto la dependencia del punto de partida y de la rotación de la figura: $e_i = |F_i|$.

[2]En caso contrario quizá lo más simple sea remuestrear el contorno a intervalos iguales. En realidad, es posible calcular los invariantes a partir de contornos con puntos irregularmente espaciados (polilíneas, en las que sólo aparecen los puntos de mucha curvatura), pero en este caso no se puede usar directamente la transformada discreta de Fourier y son necesarios cálculos adicionales.

- Juntamos la contribución positiva y negativa de cada frecuencia (véase la sección B):
  $e'_i = e_i + e_{n-i}$, para $i = 1..n/2 - 1$.
- Descartamos la frecuencia cero (valor medio de la señal), que indica la posición del centro de la figura.
- Dividimos todas las componentes por la de frecuencia 1, relacionada con el tamaño global de la figura (elipse aproximadora), consiguiendo así invarianza a escala: $d_i = e'_{i+1}/e'_1$, para $i = 1..m$.
- Nos quedamos con los descriptores de baja frecuencia, consiguiendo así invarianza a pequeñas deformaciones: $m = 10$ ó 15, por ejemplo. El valor concreto de m dependerá de la similitud de las formas que deseamos distinguir. Se puede ajustar observando la calidad de la reconstrucción de los contornos de interés al eliminar diferente número de componentes espectrales.

Ejemplo. Como comprobación, mostramos los invariantes de forma de tres siluetas sencillas:



La silueta cuadrada es una polilínea con los siguientes vértices: {(0., 0.), (0., 0.25), (0., 0.5), (0., 0.75), (0., 1.), (0.25, 1.), (0.5, 1.), (0.75, 1.), (1., 1.), (1., 0.75), (1., 0.5), (1., 0.25), (1., 0.), (0.75, 0.), (0.5, 0.), (0.25, 0.)}. Sus invariantes frecuenciales de forma son los siguientes: {0., 0.123309, 0., 0.0550527, 0., 0.0395661}.

Los 10 primeros invariantes de la silueta de la flecha son {0.153946, 0.238024, 0.150398, 0.0346874, 0.0344614, 0.036184, 0.0380148, 0.0114117, 0.0144718, 0.00547812 }.

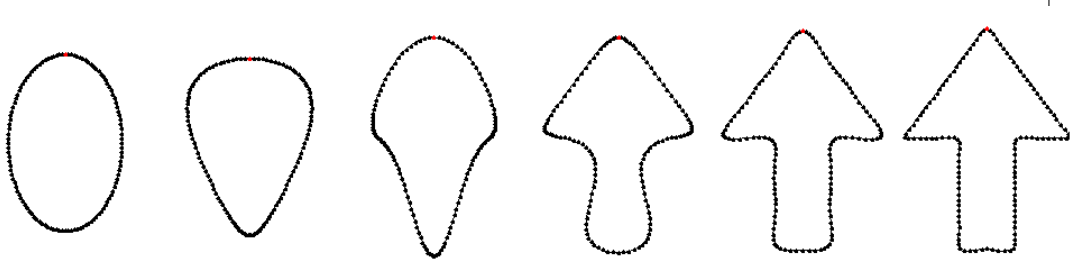
Finalmente, los 10 primeros invariantes de la forma de cruz son {0., 0.234503, 0., 0.182179, 0., 0.0675552, 0., 0.00220803, 0., 0.0132129}.

(Ten en cuenta que los invariantes aplicados a flechas o cruces con brazos de diferente grosor relativo tendrán propiedades con valores numéricos algo diferentes.)

→ ejemplo figuras deformadas

Ejercicios:

- Comprueba que las propiedades propuestas son realmente invariantes a posición, tamaño, orientación, punto de partida y pequeñas deformaciones.
- Comprueba que, a medida que aumenta el número de puntos que describen un contorno dado, las propiedades invariantes tienden a un límite (debido a que la transformada discreta de Fourier se acerca cada vez más a la transformada de Fourier ordinaria de una función continua).
- Escribe un programa que reciba una imagen binaria y devuelva el conjunto de contornos de las componentes conexas encontradas en ella.
- Construye un clasificador de contornos sencillo basado en los invariantes frecuenciales de forma.
- Haz un mapa 2D de las propiedades más expresivas o más discriminantes extraídas de los invariantes frecuenciales de forma de una colección de siluetas.
- Comprueba el método de suavizado de contornos basado en la eliminación de las componentes de alta frecuencia (la parte central de la FFT). La imagen siguiente muestra versiones suavizadas de la forma de flecha en las que hemos dejado respectivamente 1, 2, 3, 5, 10, y 20 componentes frecuenciales:



Existen otras propiedades útiles para el reconocimiento de formas: invariantes basados en momentos estadísticos, propiedades topológicas, técnicas de alineamiento (para calcular la transformación geométrica que convierte el modelo en la figura observada), etc.

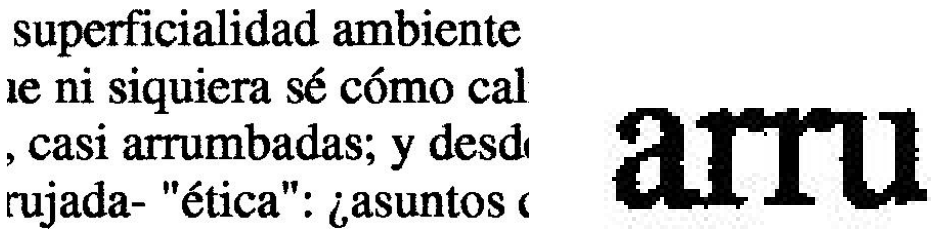
El método de reconocimiento propuesto es invariante frente a transformaciones 'similares' (además de los aspectos técnicos de punto de partida, espaciado de puntos y pequeñas irregularidades debidas al ruido de digitalización). Es posible mejorar el método(B.15) para admitir también transformaciones 'afines' generales, en la que las figuras pueden sufrir también estiramientos de diferente magnitud en dos direcciones arbitrarias.

Quizá la situación más interesante es la del reconocimiento de figuras vistas en perspectiva. Este tema se estudiará en detalle en la parte II de estos apuntes; en este punto solo anticiparemos que el problema es abordable en condiciones idealizadas (contornos perfectos, o de tipo poligonal, o con puntos de curvatura muy pronunciada,

etc.). En condiciones más realistas (con ruido, figuras muy suaves, oclusiones, etc.) el problema es mucho más complicado pero aún así se han propuesto algunas soluciones.

2.4. Reconocimiento de Caracteres Impresos

Los caracteres impresos presentan una menor variabilidad geométrica. En principio son más sencillos de reconocer que los caracteres manuscritos. Sin embargo, las imperfecciones de la digitalización de la imagen pueden dificultar la segmentación de los caracteres individuales, que pueden aparecer 'pegados':



Una forma de atacar el problema es intentar reconocer componentes conexas de pixels (*glyphs*) que serán candidatos a caracteres o secuencias de caracteres. Los renglones, palabras y caracteres se pueden separar a partir de los mínimos locales de las proyecciones horizontales y verticales de la imagen. Si un *glyph* no se parece a ningún prototipo se puede lanzar una búsqueda recursiva de posiciones de corte candidatas, hasta dar con una separación cuyos trozos se reconozcan todos con suficiente confianza. Algunas secuencias de letras que suelen aparecer juntas se pueden considerar como una clase independiente. El resultado obtenido puede corregirse mediante un diccionario.

Como método de clasificación se puede usar una función de distancia sencilla definida en una cuadrícula de p. ej., 7×5 celdas sobre cada *glyph*. Para mejorar los resultados el alineamiento entre candidatos y prototipos debe tener en cuenta la localización y dispersión global del carácter.

$\rightarrow$ ejemplo de subdivisión

Ejercicios:

- Prototipo de OCR sencillo: Escribe un programa que admita la imagen de un texto 'escaneado' y produzca un fichero de texto con el contenido de la imagen.

2.5. Introducción al Reconocimiento del Habla

El reconocimiento del habla es una de las aplicaciones de la percepción artificial con mayor impacto práctico para la comunicación hombre máquina. Existen ya productos

comerciales con resultados prometedores. El texto de Rabiner y Juang [22] es excelente.

La idea esencial consiste en capturar la señal de audio, trocearla en secciones de unos pocos milisegundos, calcular su espectro de frecuencias o, alternativamente, los coeficientes de un predictor lineal, definir una medida adecuada de similitud entre los diferentes trozos de sonido (que puede, hasta cierto punto, precalcularse), y finalmente comparar secuencias de esos trozos elementales para determinar a qué categoría pertenecen. Esta comparación debe tener en cuenta que las secuencias de habla pueden presentar deformaciones (alargamientos, omisión de fonemas, etc.). Algunos aspectos importantes son los siguientes:

Producción del habla. Es conveniente tener una idea de los mecanismos fisiológicos de la producción del habla. El ser humano produce una onda acústica con ciertas propiedades que dependen de la estructura de las cavidades resonantes de la cabeza, punto de articulación, vibración o no de las cuerdas vocales, etc.

Representación frecuencial. Al tratarse de una señal (localmente) periódica, resulta apropiada un análisis frecuencial (Sección B) (aunque algunas propiedades importantes dependen también de la dinámica temporal). La evolución del espectro de frecuencias a lo largo del tiempo (*espectrograma*) es una representación conveniente de la señal de habla. Los picos del espectro se llaman *formantes* y contienen cierta información sobre los fonemas (recordar el ejemplo de la Sección 1.5). En la práctica el espectro se obtiene mediante la transformada rápida de Fourier (FFT). Es conveniente espaciar de manera no uniforme (logarítmicamente) las componentes frecuenciales seleccionadas. Suelen usarse 'ventanas' de suavizado.

Representación LPC. (*Linear Predictive Coding*.) Otra posibilidad es representar cada trozo de señal mediante los coeficientes de un predictor lineal. (Dada una señal periódica, la muestra siguiente se puede obtener a partir de las anteriores simplemente multiplicándolas por determinados coeficientes y acumulando el resultado.) Los coeficientes se ajustan para minimizar el error de predicción. (Pueden obtenerse resolviendo un sistema de ecuaciones lineales por mínimos cuadrados (ver Sección D.2.1) y también a partir de la *autocorrelación* de la señal. Se comprueba que el espectro de la señal original y la producida por el predictor con un número moderado de coeficientes es muy parecido. A partir de los coeficientes del predictor se calcula un conjunto de propiedades derivadas que son las utilizadas finalmente para describir la señal de habla.

Distorsión espectral. Cada trozo de la señal correspondiente a unos pocos milisegundos se representa mediante unas ciertas propiedades espectrales. Lógicamente, si cambian algunas propiedades de la señal cambiará la representación. Pero algunos cambios no son 'perceptualmente relevantes': p. ej., el tono de voz, la intensidad, etc. no afectan a la identidad de los fonemas y palabras pronunciadas. Es importante utilizar una 'función de distancia' entre espectros que refleje, lo mejor posible, la 'similitud perceptual' que nos interesa. Esta función debe ser relativamente invariante frente al tono de voz, atenuación de bajas frecuencias, etc., pero debe responder bien a la posición y anchos de banda de los formantes.

Se han propuesto varias medidas de distorsión espectral que tratan de cumplir estos requisitos.

Cuantización vectorial. Se realiza para ahorrar tiempo en el cálculo de la distorsión espectral de cada posible pareja de trozos elementales de habla (necesario para el reconocimiento de secuencias). La idea es guardar en una tabla la distorsión entre todos los pares de espectros, previamente discretizados. Cada espectro completo se representa mediante un código, el índice de acceso a la tabla de distorsiones espectrales.

Normalización temporal. El problema del reconocimiento de un trozo de habla se reduce a la comparación de secuencias de índices. Hay que encontrar el mejor alineamiento entre las secuencias. Debido a las diferencias en la forma de hablar, algunos fonemas se alargan, acortan, u omiten, por lo que la comparación de secuencias necesita encontrar el mejor alineamiento. Este será no uniforme pero que deberá cumplir ciertas restricciones: p. ej., no se vuelve atrás, no hay saltos bruscos, etc. Existen algoritmos de programación dinámica para resolver este problema.

Modelos de Markov. La técnica de alineamiento anterior permite encontrar la secuencia prototipo más parecida a nuestra entrada. Pero no tiene en cuenta la variabilidad de las secuencias correspondientes a un mismo vocablo (es como una simple distancia a la media). Los modelos de Markov ocultos (HMM) se utilizan para realizar el alineamiento de secuencias teniendo en cuenta esta variabilidad, ponderando adecuadamente la importancia relativa de cada tipo de 'desalineamiento'. Para ello se genera mediante algún tipo de aprendizaje (ajuste de parámetros) un conjunto de autómatas probabilísticos que generan en sus transiciones los códigos (de espectro) observados. El reconocimiento de vocablos consiste en encontrar el modelo que con mayor probabilidad ha emitido la secuencia observada. También existen algoritmos de programación dinámica para resolver este problema.

Ejercicios:

- Graba una secuencia de vocales en un fichero (p. ej., .wav). Calcula y representa su espectrograma.
- Intenta reconocer vocales simples (del mismo locutor, pronunciadas en el mismo tono, etc.).
- Prueba algún sistema de dictado comercial.
- Echa un vistazo a `http://www.speech.cs.cmu.edu/sphinx/`
- Si podemos acceder en tiempo real a la señal acústica, es interesante representar en un espacio bidimensional la posición de los dos formantes principales.
- Un ejercicio relacionado es un detector de la frecuencia fundamental del sonido para el reconocimiento (transcripción) de música.

Capítulo 3

Diseño de Clasificadores

Este capítulo está dedicado a la etapa de *clasificación* de los sistemas de reconocimiento de modelos. En ella, los vectores de propiedades extraídas de la información 'bruta' deben identificarse como miembros de un conjunto preestablecido de categorías elementales.

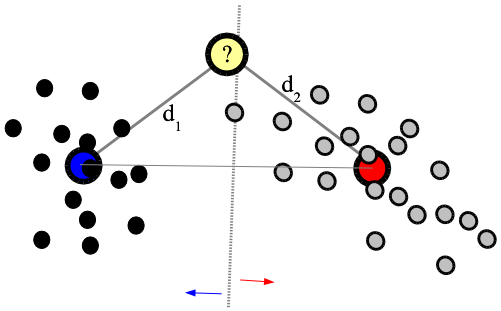
A primera vista, este problema admite un análisis teórico bastante general e independiente de la aplicación. De hecho, un volumen enorme de la literatura sobre *Machine Perception* se ha dedicado a estudiar técnicas de *Pattern Classification* (véase p. ej., el libro clásico de Duda y Hart [5] y el más reciente de Bishop [2]) dando lugar a numerosas soluciones alternativas. Pero, en la práctica, si las propiedades extraídas son suficientemente discriminantes, la etapa de clasificación puede resolverse perfectamente usando cualquier método sencillo como los que revisaremos en la sección siguiente. En caso contrario, (cuando hay ambigüedad), el uso de clasificadores más elaborados puede contribuir a *no empeorar* demasiado los resultados.

Normalmente, el diseño de clasificadores tiene que recurrir a 'máquinas' (algoritmos) de aprendizaje 'inductivo' (cuyos parámetros se ajustan a partir de ejemplos resueltos; se trata simplemente de una forma especial de aproximación de funciones). Estas máquinas tienen numerosas aplicaciones, incluso fuera del ámbito de la percepción artificial. En este capítulo explicaremos los principios teóricos de la clasificación y justificaremos la validez de algunas máquinas muy sencillas. Posteriormente introduciremos técnicas más avanzadas como los 'modelos conexionistas' (redes neuronales artificiales) y las 'máquinas de kernel' (entre las que destacan las 'maquinas de vectores de soporte'). .

Antes de comenzar debemos insistir en que la etapa de clasificación solo es capaz de producir una percepción artificial de 'bajo nivel'. Para organizar la información sensorial dentro de estructuras de un nivel de abstracción cada vez mayor son necesarios procesos de *interpretación* como los que explicaremos en el apartado 6.3.

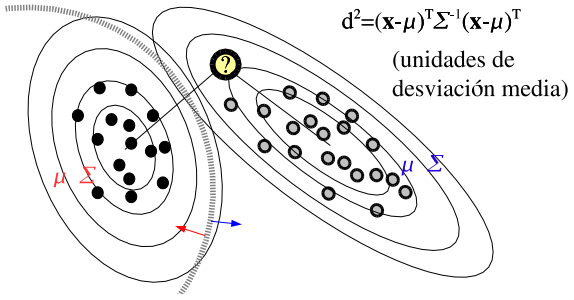
3.1. Clasificadores sencillos

Cualquier clasificador puede expresarse, directa o indirectamente, en función de algún tipo de medida de 'distancia' o 'similitud' entre el vector de propiedades y la 'nube de puntos' que describe cada clase. Sin profundizar en consideraciones teóricas, el sentido común sugiere algunas medidas de similitud razonables. Por ejemplo, podemos definir un clasificador basado en *mínima distancia a la media* de cada clase. En este caso, la nube de puntos queda descrita únicamente mediante su 'localización promedio':



Clasificación por mínima distancia a la media

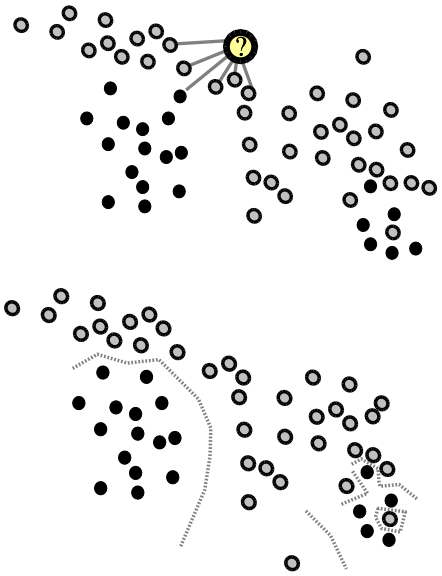
Si deseamos mayor precisión, podemos tener en cuenta la dispersión de cada nube, *normalizando* las distancias a las medias de cada clase (dividiendo por las desviaciones, o usando la distancia de Mahalanobis (Sección C.2)). Estos clasificadores funcionan bien cuando las nubes no están muy solapadas:



Clasificación por mínima distancia de Mahalanobis

Observa que aunque la muestra desconocida está más cerca de la media de la población de la izquierda, la distancia de Mahalanobis (en unidades de desviación) a la clase de la derecha es mucho menor.

Un método todavía más sencillo y general es el de los *vecinos más próximos*, según el cual la clase de una observación se determina en función de la de los ejemplos de entrenamiento más próximos, típicamente por votación y exigiendo una mayoría suficiente. Este método no necesita extraer ninguna propiedad estadística de las nubes de puntos de cada clase y tiene una gran flexibilidad para establecer la frontera de clasificación en el espacio de propiedades. Como contrapartida, puede requerir un tiempo de respuesta largo si el conjunto de ejemplos es muy numeroso:



Clasificación por el método de los vecinos más próximos

3.2. Evaluación

Cualquiera que sea el método de diseño utilizado, el clasificador finalmente construido deberá evaluarse objetivamente.

El indicador de calidad más importante de un clasificador es la *Probabilidad de Error*. Esta probabilidad debe estimarse mediante un conjunto de *ejemplos de validación* tomados en las mismas condiciones de utilización real del sistema. Estos ejemplos deben seleccionarse al azar en un primer momento y guardarse 'bajo llave'. No deben utilizarse en ninguna etapa de diseño o ajuste de parámetros del clasificador.

Esto es especialmente importante si, como es habitual, el clasificador se entrena mediante aprendizaje 'supervisado': los ejemplos de aprendizaje y de validación deben ser totalmente independientes. Por supuesto, si evaluamos con los mismos ejemplos de entrenamiento (*resustitución*) obtendremos resultados injustificadamente optimistas.

> Cuando el conjunto de ejemplos etiquetados es pequeño la división en dos subconjuntos independientes provoca que bien el ajuste, bien la validación o bien ambos procesos no dispongan de un tamaño de muestra estadísticamente significativo. Es posible aprovechar mejor los ejemplos disponibles mediante la técnica de *leave-one-out*: se realiza el ajuste varias veces, quitando una muestra cada vez, evaluando el clasificador sobre la muestra eliminada y promediando los resultados. Se utilizan todos los ejemplos disponibles en aprendizaje y en validación y, a pesar de todo, cada conjunto de entrenamiento y validación es independiente.

También podemos hacernos una idea aproximada de la precisión de un estimador mediante *bootstrap*, una técnica basada en generar muestras artificiales a partir de las disponibles.

Si las propiedades son discretas es importante tener en cuenta el *off-training-set error*: la probabilidad de error sobre ejemplos *no utilizados* en el aprendizaje. En dominios continuos las observaciones no pueden repetirse exactamente. Pero en el caso discreto podríamos reducir el error sobre *el dominio completo* mediante pura memorización. La medición del *off-training-set error* descuenta el aprendizaje memorístico y mide la auténtica generalización sobre casos no vistos.

Es conveniente desglosar la probabilidad de error en forma de *matriz de confusión*, que contiene la frecuencia de los diferentes tipos de error. Un ejemplo hipotético podría ser el siguiente:

clase real \ predicción	c1	c2	c3	c4	R (rechazo)
c1	**7**	1	0	1	2
c2	0	**12**	0	0	1
c3	2	3	**5**	1	0
c4	0	1	0	**11**	1

Cuando se estiman probabilidades es necesario proporcionar también alguna medida de la precisión de la estimación, p. ej., mediante un intervalo de confianza.

El clasificador obtenido depende de los ejemplos de aprendizaje y de otros factores. Una vez construido tiene una probabilidad de error concreta, aunque desconocida. Para estimarla necesitamos una muestra aleatoria de ejemplos de validación; cuanto mayor sea su número la estimación será más precisa. Como ejercicio, intenta calcular un intervalo que contenga con 'confianza' del 99 % la probabilidad de un evento que ha ocurrido en 75 ocasiones al realizar 100 experimentos.

Otros indicadores de la calidad de un clasificador son el *tiempo de ejecución* (debe ser pequeño si el número de observaciones a clasificar es elevado, p. ej., caracteres, trozos de habla, etc.) y el *tiempo de aprendizaje* (menos importante, ya que el ajuste del clasificador es un proceso *off-line*, aunque es conveniente que no se dispare).

3.2.1. Ejemplos

A continuación se muestran las matrices de confusión conseguidas con estos métodos de clasificación sencillos sobre el conjunto MNIST de dígitos manuscritos. El

órden de las clases es de '0' a '9' en filas (clase real) y columnas (clase predicha). Trabajaremos con los 5000 primeros dígitos del conjunto `trainimages-idx3-ubyte`. De ellos, los 4000 primeros se utilizarán para diseño de los clasificadores y los 1000 últimos para la evaluación. Todas las imágenes se representarán mediante las 20 componentes principales de la muestra de diseño. (No hemos implementado la opción de rechazo.)

El método de mínima distancia a la media consigue una tasa de aciertos del 80.5 % con la siguiente distribución de decisiones:

$$\begin{bmatrix} 93 & 0 & 1 & 1 & 0 & 1 & 3 & 0 & 1 & 0 \\ 0 & 112 & 1 & 0 & 0 & 4 & 0 & 0 & 2 & 0 \\ 0 & 8 & 69 & 1 & 3 & 1 & 4 & 1 & 2 & 2 \\ 0 & 2 & 2 & 79 & 0 & 7 & 1 & 2 & 5 & 3 \\ 0 & 1 & 1 & 0 & 88 & 0 & 0 & 1 & 1 & 13 \\ 6 & 5 & 0 & 15 & 4 & 46 & 1 & 1 & 1 & 5 \\ 1 & 4 & 5 & 0 & 2 & 2 & 81 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 & 6 & 0 & 0 & 105 & 0 & 4 \\ 1 & 5 & 6 & 4 & 2 & 5 & 1 & 0 & 59 & 6 \\ 1 & 0 & 2 & 1 & 13 & 0 & 0 & 8 & 0 & 73 \end{bmatrix}$$

La clasificación por mínima distancia de Mahalanobis consigue una tasa de aciertos notablemente superior, del 93.5 %, con el siguiente desglose:

$$\begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 107 & 3 & 1 & 0 & 0 & 1 & 1 & 6 & 0 \\ 0 & 0 & 87 & 2 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 92 & 0 & 4 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 101 & 0 & 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 & 0 & 80 & 0 & 0 & 3 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 93 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 104 & 0 & 12 \\ 0 & 0 & 7 & 1 & 0 & 0 & 0 & 0 & 81 & 0 \\ 1 & 0 & 0 & 1 & 4 & 0 & 0 & 1 & 1 & 90 \end{bmatrix}$$

Las entradas no diagonales con un valor elevado nos indican las parejas de dígitos difíciles de distinguir: 1-9, 3-5, 8-2, 7-9, etc.

Finalmente, la clasificación por el vecino más próximo consigue una tasa de aciertos del 82.0 % con el siguiente desglose.

$$\begin{bmatrix} 16 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 28 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 11 & 2 & 1 & 3 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 17 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 21 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 & 1 & 13 & 1 & 1 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 17 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 13 & 0 & 4 \\ 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 14 & 0 \\ 0 & 0 & 0 & 1 & 5 & 0 & 0 & 2 & 0 & 14 \end{bmatrix}$$

Debido a la lentitud de este método, para este experimento hemos utilizado sólo los 200 primeros ejemplos de los conjuntos de diseño y evaluación definidos anteriormente.

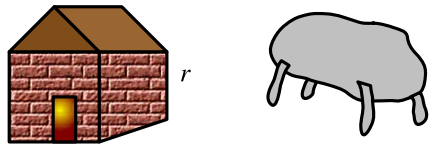
Estos resultados no son casuales. La distancia de Mahalanobis es una alternativa muy prometedora para atacar muchos problemas de clasificación: es eficiente y suele conseguir buenas tasas de aciertos. Para mejorar sus resultados suele ser necesario recurrir a técnicas mucho más avanzadas a un coste que no siempre está justificado.

Ejercicios:

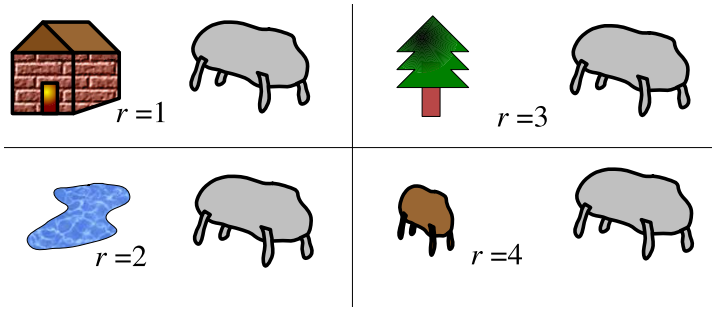
- Probar estos clasificadores sencillos en alguno de los ejercicios propuestos en el capítulo anterior.

3.3. El Clasificador Óptimo

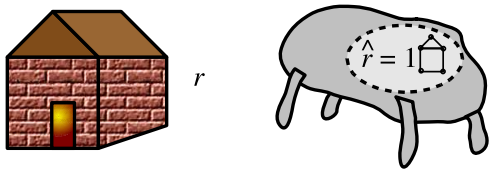
En esta sección estudiaremos el problema de la percepción desde un punto de vista un poco más formal. La siguiente figura muestra un sistema autónomo inmerso en un entorno cuyo estado denotaremos mediante r, indicando que es el estado 'real' del mundo externo:



Por supuesto, una descripción detallada del entorno tendría una complejidad inmensa. Por claridad, por ahora supondremos simplemente que r podrá tomar o bien un conjunto continuo de valores o bien un conjunto discreto y finito de posibilidades o clases excluyentes:



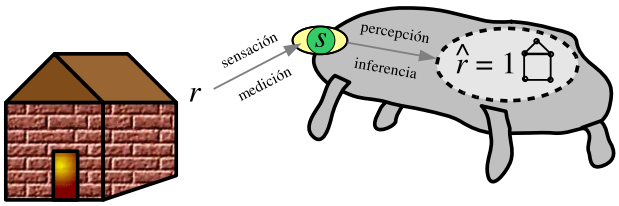
Nuestro objetivo es que el sistema consiga información sobre el estado r en que se encuentra su entorno. Esta información se denotará como $\hat{r}$ para indicar que es una estimación o predicción del estado real. Es el modelo o representación de la realidad elaborada por el sistema:



Lógicamente, deseamos que ese modelo sea correcto, es decir, que $\hat{r} = r$ (o, si el estado es una variable continua, que la diferencia $|\hat{r} - r|$ sea pequeña.)

Para que la estimación $\hat{r}$ no sea una mera adivinación de la realidad, el sistema debe medir algunas magnitudes de su entorno utilizando sensores. El resultado de

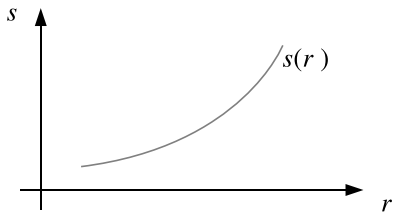
estas mediciones se denotará mediante s (estímulos). La estimación de $\hat{r}$ se calculará fundamentalmente a partir de esas sensaciones recibidas:



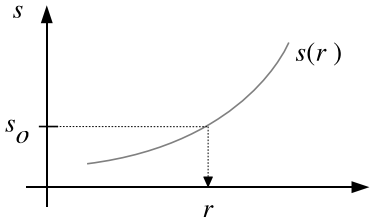
Aunque será conveniente tener en cuenta también toda la información adicional disponible sobre el entorno como, p. ej., la frecuencia de aparición de cada posible estado real e, incluso, las transiciones posibles entre estados.

3.3.1. Modelo inicial

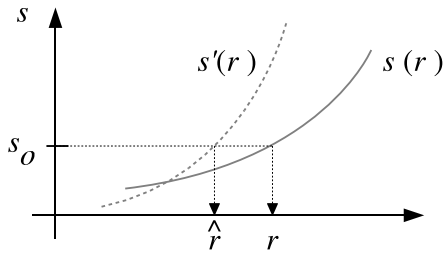
La idea clave es quen no tenemos 'acceso directo' a la realidad. Solo disponemos de ciertas mediciones realizadas sobre ella, que a veces pueden producir resultados poco informativos. El caso ideal es el de un sensor cuya respuesta está bien definida para cualquier valor de sus entradas, de acuerdo con una cierta función:



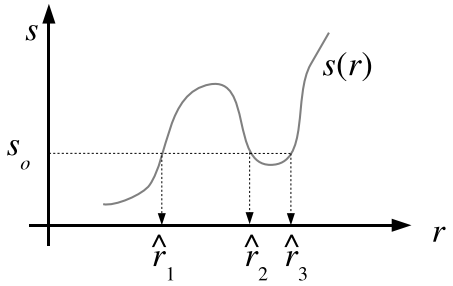
Cuando observamos un cierto estímulo s_o no hay ningún problema para deducir el estado real que lo ha provocado. Sólo tenemos que invertir la función que describe la respuesta del sensor:



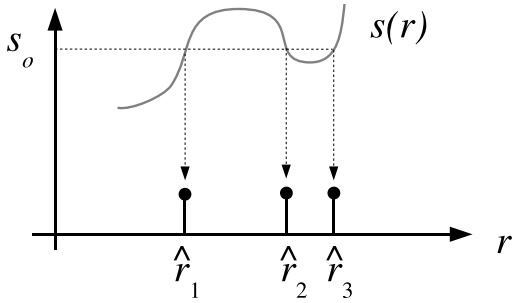
Por supuesto, es muy posible que dispongamos de un modelo imperfecto $s'(r)$ del funcionamiento del sensor. En este caso la estimación obtenida $\hat{r}$ se alejará un poco del estado real r:



Incluso si nuestro modelo es correcto, un problema que puede aparecer es que el sensor tenga una respuesta 'no monótona', respondiendo igual frente a situaciones distintas. Esta ambigüedad conduce a un estimador de la realidad compuesto de varias posibilidades:

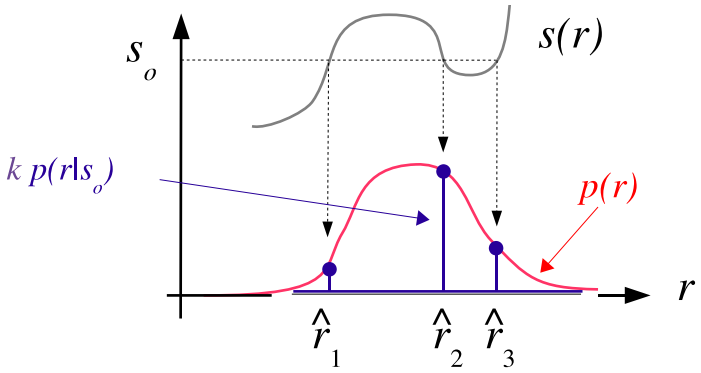


El conjunto de alternativas se puede representar mediante su 'función indicadora', que vale uno en los elementos del conjunto y cero en el resto:



3.3.2. Modelo del mundo

Los estados reales no siempre ocurren con la misma frecuencia. P. ej., en el idioma español es más probable encontrar una 'e' o una 's' que una 'w' o una 'k'. Si conocemos la probabilidad *a priori* $p(r)$ de aparición de cada una de las situaciones, podemos distinguir mejor entre las posibles alternativas calculando la probabilidad de que cada una de ellas haya producido el estímulo observado (probabilidad *a posteriori*):



Los elementos de la función indicadora anterior quedan multiplicados por las probabilidades *a priori* de cada alternativa, modificando sus alturas relativas. Hemos *restringido* el juego completo de probabilidades de la realidad a los elementos que son consistentes con la observación.

> Hay que tener en cuenta el detalle técnico de que esos valores no son directamente las probabilidades *a posteriori* propiamente dichas de las hipótesis, ya que no están normalizados (no suman 1). La constante de normalización (llamada k en la figura anterior) se consigue simplemente dividiendo por la suma de todos ellos. En cualquier caso, la distribución *relativa* es correcta.

Las probabilidades *a priori* constituyen nuestro *modelo del mundo*. Ciertamente es un modelo muy simplificado, ya que asume que los diferentes estados ocurren independientemente unos de otros. Por ahora mantendremos esta suposición y más adelante estudiaremos cómo se pueden modelar las transiciones entre estados.

El resultado final del proceso de 'percepción' es esta distribución *a posteriori* completa $p(r|s_o)$. Contiene toda la información que puede inferirse de la observación s_o a partir únicamente de la función de respuesta del sensor $s(r)$ y las probabilidades *a priori* de los estados reales $p(r)$.

3.3.3. Predicción

En muchos casos estamos obligados a arriesgarnos a dar una predicción concreta. Entonces debemos computar un *estimador* o *predictor* de la variable aleatoria a

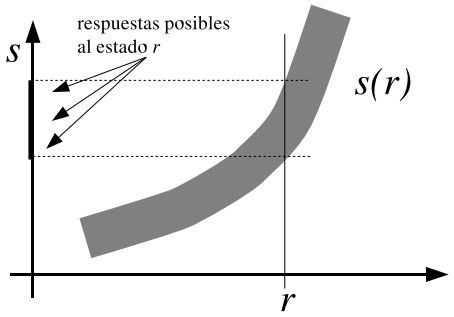
partir de su distribución *a posteriori*. Éste depende del criterio de coste que deseamos minimizar (ver Sec. C.4). En problemas de clasificación (r es una variable discreta) si deseamos minimizar el número total de decisiones erróneas el estimador óptimo es la moda: el valor más probable. Esta regla de decisión se denota como MAP (*maximum a posteriori*). Más adelante veremos que en determinados problemas en los que unos errores son más graves que otros es preferible minimizar el *coste* total de los errores.

Si la variable r es continua no tiene sentido intentar acertar exactamente su valor, sino acercarnos lo más posible a él. El estimador que minimiza el error *cuadrático* medio de las predicciones es el valor *medio* de la densidad *a posteriori*. Para minimizar el error *absoluto* medio se utiliza la *mediana* de la distribución, un estimador más robusto frente a fallos de medida. Pero cuando la distribución *a posteriori* posee mucha dispersión o es *multimodal*, los estimadores de ese tipo no son adecuados.

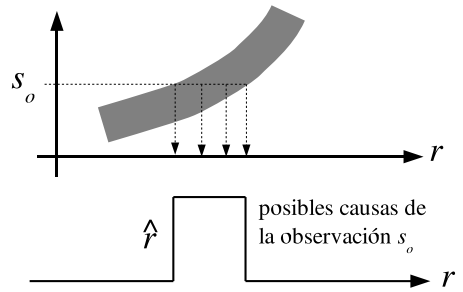
A veces merece la pena quedarnos con toda la distribución de probabilidades y retrasar el cómputo del estimador. Imagina una aplicación de OCR. Si nuestro objetivo fuera clasificar letras independientes, deberíamos arriesgar una decisión para cada mancha de tinta más o menos aislada. Sin embargo, no tiene sentido arriesgar una predicción si no hay un candidato claramente ganador. Es más razonable intentar reconocer palabras completas. Para ello deberíamos combinar las distribuciones de probabilidad de secuencias de manchas consecutivas, para encontrar la *palabra completa* más probable de las existentes en un diccionario. El reconocimiento de caracteres individuales es percepción de bajo nivel, el de palabras es de mayor nivel, y así sucesivamente hasta llegar hasta frases o párrafos completos, lo que requeriría auténtica percepción de alto nivel. A medida que aumenta la complejidad de los objetos que deseamos percibir hay que integrar las detecciones elementales en modelos cada vez más elaborados. En la actualidad somos capaces de ascender muy pocos niveles de la 'escala perceptual' artificial.

3.3.4. Incertidumbre

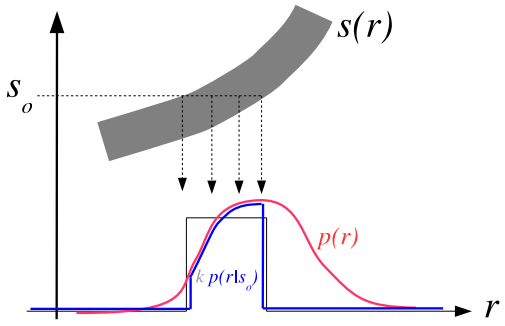
Continuando con nuestro análisis de la sensación y la percepción, en la práctica existe una fuente de ambigüedad mucho más importante que la provocada por la no monotonicidad. Debido a múltiples circunstancias, la respuesta de un sensor casi nunca es 'determinista': fijado un estado real, los resultados de diferentes mediciones oscilan alrededor de unos valores más o menos cercanos. El modelo del sensor es una especie de función con 'anchura vertical', indicando que para cada posible valor de r hay un cierto rango de posibles respuestas s:



En consecuencia, cuando recibimos un estímulo s_o existen uno (o varios) intervalos en los que podría estar el estado real:



De nuevo, si disponemos de la probabilidad *a priori* $p(r)$ de cada estado real podemos restringirla a los valores consistentes con la observación:



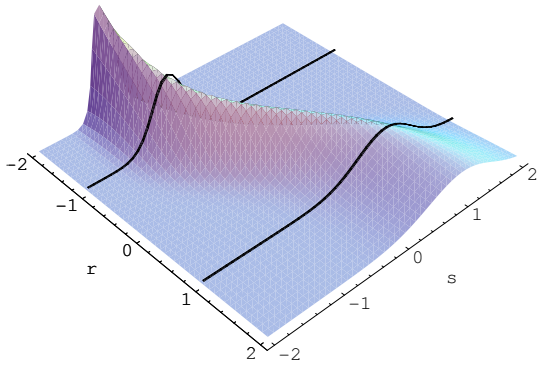
Esta distribución restringida es esencialmente (una vez normalizada) la probabilidad *a posteriori* $p(r|s_o)$ de los estados reales dada la observación. Con ella podemos computar el predictor $\hat{r}$ que minimice el criterio de coste deseado.

3.3.5. Modelo probabilístico

La incertidumbre en las respuestas del sensor frente a cada estado real no se caracteriza muy bien por un simple intervalo de valores posibles. Es mejor modelarla mediante la distribución de probabilidad de los valores de s para cada posible estado r, que denotamos como $p(s|r)$. Este juego de distribuciones de probabilidad 'condicionadas' es el *modelo del sensor*.

Intuitivamente, debemos pensar en una colección de densidades de probabilidad de s, una para cada valor de r. Dependiendo de la naturaleza del problema, cada una tendrá una forma (localización, dispersión, etc.) en principio distinta. Otra forma de expresarlo podría ser $p_r(s)$, pero la notación de 'probabilidad condicionada' $p(s|r)$ es matemáticamente más apropiada para las ideas que mostraremos más adelante.

La figura siguiente muestra un ejemplo sintético en el que las medidas s tienden a crecer con el valor de una magnitud continua r (de forma ligeramente no lineal), y a la vez, tienen mayor dispersión a medida que el valor de r es mayor. (Por ejemplo, un hipotético sensor de temperatura podría medir con mayor precisión las temperaturas bajas que las altas; a medida que sube la temperatura las medidas tienen más ruido.) Se ha resaltado la distribución de medidas para dos valores concretos de r (cortes 'verticales' de la superficie):

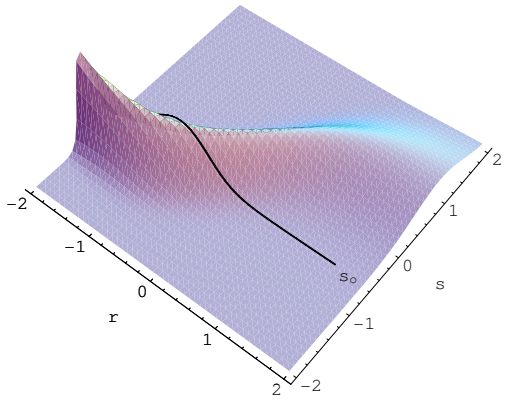


Modelo del sensor: $p(s|r)$

En este ejemplo la distribución de medidas para cada valor de r es simplemente una campana de Gauss, pero en principio podría tener cualquier forma y variar de unos cortes a otros. Observa que los cortes de mayor dispersión son más bajos y los de menor dispersión son más altos. Esto se debe a que todos ellos son densidades de probabilidad propiamente dichas, normalizadas con area unidad.

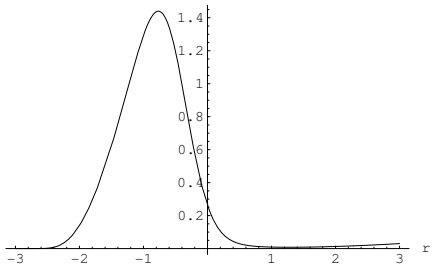
Ahora el conjunto de hipótesis consistentes con una cierta medida observada, p. ej., $s_o = -0.6$ ya no es uno o varios intervalos discretos (como en el caso de la

función $s(r)$ anterior con 'anchura vertical'), sino una función con altura variable correspondiente al corte 'horizontal' de la superficie definida por el modelo del sensor:



Función de verosimilitud $p(s_0|r)$

Esa función se denomina *verosimilitud* de la observación. Se representa como $l_{s_o}(r) = p(s_o|r)$ y *no* es una densidad de probabilidad, ni tiene ese significado aunque la normalicemos. Su significado es el 'apoyo puramente experimental' o evidencia que recibe cada una de las posibles hipótesis candidatas r por el resultado de la medida concreta s_o. En este ejemplo ilustrativo tiene forma parecida a una campana:



Función de verosimilitud $p(-0.6|r)$

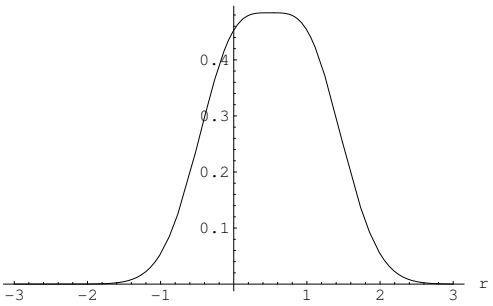
pero en otros casos puede ocurrir que, p. ej., la verosimilitud sea grande en todos los valores de r y ni siquiera se pueda normalizar. Si el sensor tiene mucha incertidumbre entonces una observación apoyará con la misma fuerza a muchas hipótesis distintas.

Observa que el modelo del sensor (y por tanto la verosimilitud de las observaciones) no tiene para nada en cuenta la probabilidad *a priori* de los estados reales. Puede que un cierto estado r no ocurra casi nunca, lo cual se reflejará en $p(r)$, pero aún así tendrá su modelo de medida (corte vertical) $p(s|r)$ correspondiente, como todos los demás estados, para describir las observaciones que se producirían si ese estado

ocurriera. El modelo del mundo y el modelo del sensor son completamente independientes uno del otro.

La probabilidad *a priori* juega su papel en el cálculo de la probabilidad *a posteriori* de las hipótesis: la verosimilitud por sí sola no es suficiente: p. ej., una observación puede apoyar mucho a dos posibles hipótesis, pero si una de ellas es muy poco probable es lógico que nuestra decisión final sea la otra.

Continuando con el ejemplo anterior, supongamos que la probabilidad *a priori* de r tiene la siguiente distribución parecida a una campana un poco achatada, en la que se observa que el estado real se encontrará casi seguro entre -1 y $+2$:



Modelo del mundo $p(r)$

¿Cómo se combina la función de verosimilitud anterior con esta probabilidad *a priori* para obtener la probabilidad *a posteriori* de las hipótesis? El procedimiento es similar al del ejemplo anterior de una respuesta del sensor con 'anchura vertical' donde la verosimilitud era simplemente una función indicadora (0-1) que restringía la probabilidad *a priori*. En el caso general se utiliza la

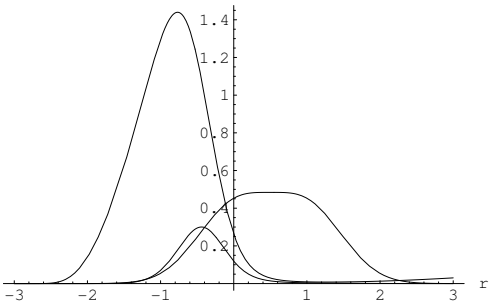
3.3.6. Regla de Bayes

La probabilidad *a posteriori* es simplemente el producto normalizado de la función de verosimilitud y de la probabilidad *a priori* de las hipótesis:

$$p(r|s_o) = \frac{p(s_o|r)p(r)}{p(s_o)}$$

Es inmediato demostrar esto usando la *regla de Bayes*. Sirve para 'cambiar el sentido' de una probabilidad condicionada, basándose en la forma equivalente de expresar la probabilidad de la conjunción de dos sucesos: $p(A, B) = p(A)p(B|A) = p(B)p(B|A)$. Se utiliza cuando nos interesa una probabilidad condicionada pero la contraria puede deducirse más fácilmente de las condiciones del problema.

A continuación se muestran las funciones de verosimilitud (izquierda), la probabilidad *a priori* (derecha) y el producto de ambas (en medio). Inspeccionando esta última vemos que el estado real estará probablemente alrededor de $r \simeq -0.5$:

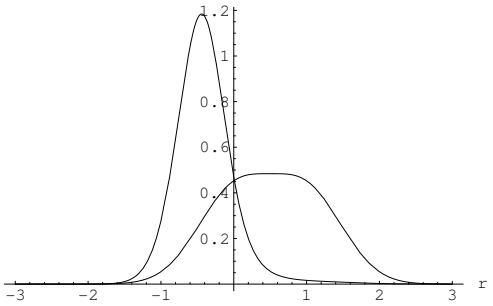


$p(s_o|r)$ (izqda.), $p(r)$ (dcha.) y $p(s_o|r)p(r)$ (centro)

El factor de normalización depende solo de $p(s_o) = \int_{-\infty}^{+\infty} p(s_o|r)p(r)dr$. Su significado está relacionado con la frecuencia relativa de aparición de la medida s_o aglutinando todos los estados reales r que la pueden producir. Si tiene un valor 'muy pequeño' es que algo va mal, la observación es inconsistente con nuestro modelo del mundo y del sensor: según ellos no hay ningún estado real con probabilidad apreciable de ocurrir que pueda causar esa medida.

Como el factor de normalización es común para todas las hipótesis r no es imprescindible para hacernos una idea de qué estados reales son ahora más o menos probables, pero sí lo es para obtener una distribucion de probabilidad propiamente dicha con la que podamos continuar trabajando de forma matemáticamente consistente.

La figura siguiente muestra la evolución de la incertidumbre acerca de r:



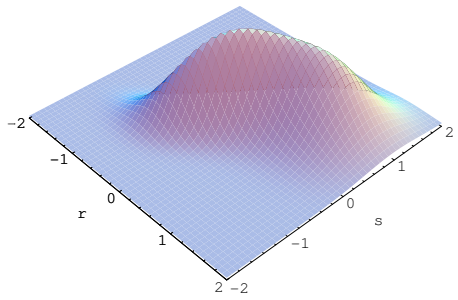
Evolución $p(r) \rightarrow p(r|s_o)$

Antes de mirar la lectura de nuestro sensor lo único que podemos afirmar es lo que nos dice la probabilidad *a priori* (función de la derecha). Despues de consultar el sensor y ver que marca $s = -0.6$ actualizamos la distribución de r, que se convierte en la probabilidad *a posteriori* (función de la izquierda, el producto $p(s_o|r)p(r)$ anterior ya normalizado). La observación disminuye la incertidumbre del modelo del mundo.

El proceso podría continuarse con nuevas medidas *independientes*, donde el papel de la probabilidad *a priori* lo jugaría la probabilidad *a posteriori* de cada etapa anterior.

3.3.7. Modelo conjunto

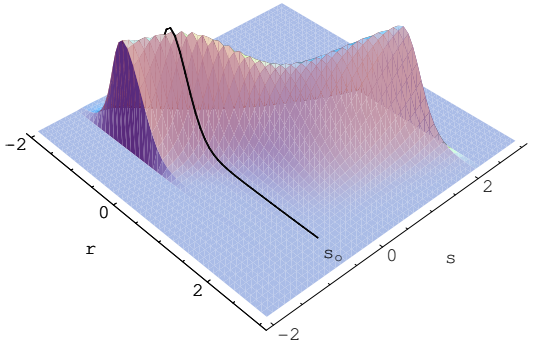
Es conveniente dar una interpretación del producto $p(s|r)p(r)$ anterior, sin normalizar, para todos los posibles valores de s y r. Cada vez que registramos una medida s_o el proceso de inferencia requiere el producto del corte $p(s_o|r)$ de toda la superficie $p(s|r)$ y la distribución $p(r)$. Podemos precalcular todos los posibles productos fabricando una nueva superficie $p(s|r)p(r) = p(s, r)$ que no es más que la probabilidad de que ocurran conjuntamente el estado real r y la observación s. La probabilidad *a posteriori* de r dado s_o es simplemente el corte de esa superficie por s_o, una vez normalizado. La superficie $p(s, r)$ es el modelo *conjunto* del sensor y del mundo:



Modelo conjunto sensor - mundo $p(s, r)$

3.3.8. Modelo inverso

Si también precalculamos la normalización de todos los cortes horizontales, conseguimos el juego completo de probabilidades *a posteriori*, lo que constituye el *modelo inverso* $p(r|s)$ del sensor:



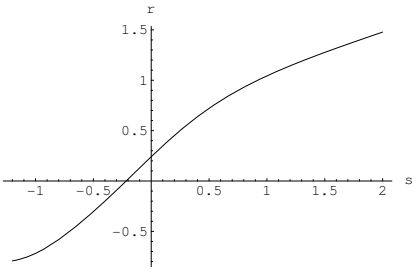
Modelo inverso $p(r|s)$

El corte de esa superficie por cada observación s es ya directamente la probabilidad *a posteriori* $p(r|s)$. La figura anterior muestra la distribución correspondiente a la medida $s_o = -0.6$ del ejemplo anterior. (El corte abrupto de la superficie se debe a que sólo hemos representado la zona donde $p(s)$ es apreciable. Más allá deberíamos sospechar de la consistencia del modelo.)

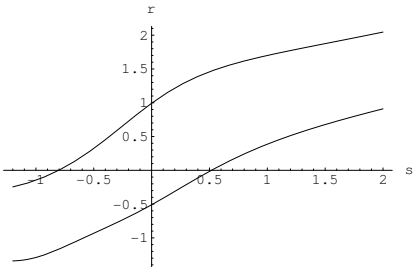
Curiosamente, si un sensor tiene incertidumbre o ambigüedad su modelo inverso requiere un modelo del mundo.

La información necesaria para inferir una variable a partir de otra se encuentra en la densidad conjunta de ambas (ver Sección C.4). A partir de ella podemos obtener cualquier densidad condicionada mediante simple normalización del corte deseado. Los cortes verticales normalizados son el modelo directo del sensor $p(s|r)$ y los cortes horizontales normalizados son el modelo inverso $p(r|s)$. En problemas de percepción como los que nos ocupan es natural expresar la densidad conjunta como producto de los dos ingredientes fundamentales que, en principio, es más fácil conocer: el modelo del sensor y el del mundo.

Finalmente, podríamos precalcular también un predictor $\hat{r}(s)$ para cada posible observación. Si deseamos minimizar el error cuadrático medio calculamos la media de cada distribución *a posteriori*, lo que se conoce como *curva de regresión*. También se puede calcular un intervalo de confianza que incluya, p. ej., 1,5 desviaciones típicas alrededor de la media:



Curva de regresión



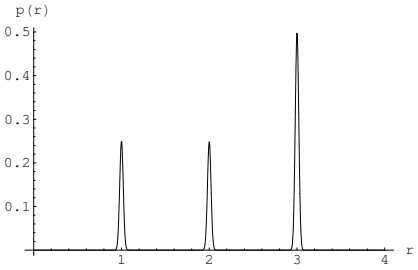
Intervalo de confianza

Este ejemplo es relativamente sencillo porque tanto el estado real r como la observación s son variables continuas unidimensionales. En general, ambas magnitudes podrían ser vectores con componentes continuas o discretas, lo que requeriría más esfuerzo computacional para realizar la inferencia y, sobre todo, una tremenda dificultad para obtener modelos probabilísticos válidos.

3.3.9. Clasificación

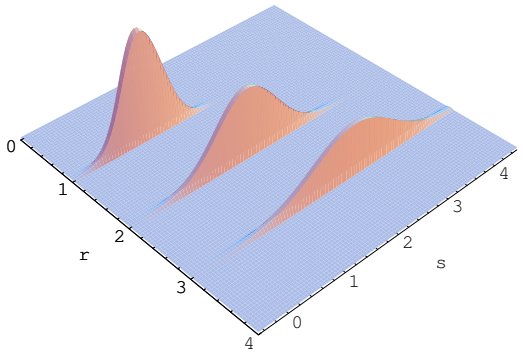
Una situación muy común es la clasificación de algún aspecto del entorno dentro de un número finito de categorías excluyentes. En este caso r es una variable que

sólo puede tomar un número finito de valores y s es un vector de propiedades, normalmente con componenentes continuas. Vamos a intentar representar gráficamente esta situación. En primer lugar analizaremós el caso de un vector de propiedades con una única componente y un problema con, p. ej., tres clases $r = \{c_1, c_2, c_3\}$. El modelo del mundo es por tanto un juego de tres valores de probabilidad $p(r) = \{p(c_1), p(c_2), p(c_3)\}$ que abreviamos como $p(r) = \{p_1, p_2, p_3\}$. Supongamos que $p(r) = \{0.25, 0.25, 0.50\}$:



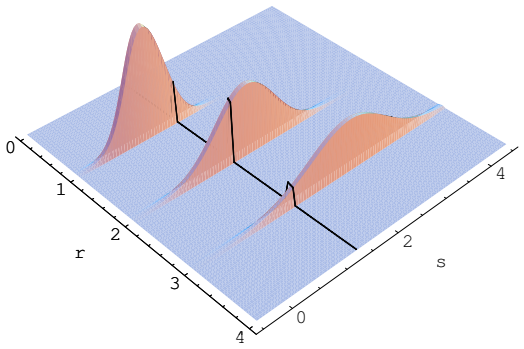
Modelo del mundo (clasificación)

Análogamente, el modelo del sensor $p(s|r)$ ya no es un juego continuo de densidades de probabilidad que forman una superficie, sino sólo tres densidades unidimensionales correspondientes a los únicos valores posibles de r: $p(s|r_1)$, $p(s|r_2)$ y $p(s|r_3)$. P. ej., , podríamos tener un sensor con la siguiente distribución de respuestas:



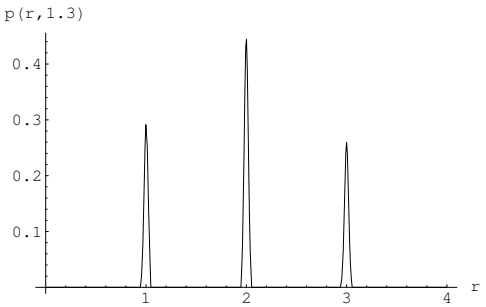
Modelo del sensor (clasificación)

La función de verosimilitud de una observación, p. ej., $s_o = 1.3$ es el corte horizontal del modelo del sensor. Al tratarse de un problema de clasificación se reduce a un conjunto discreto de valores asociados a cada categoría (en este ejemplo, la observación apoya más a r_2 y luego a r_1 y a r_3):



Verosimilitud de la observación $s_o = 1.3$

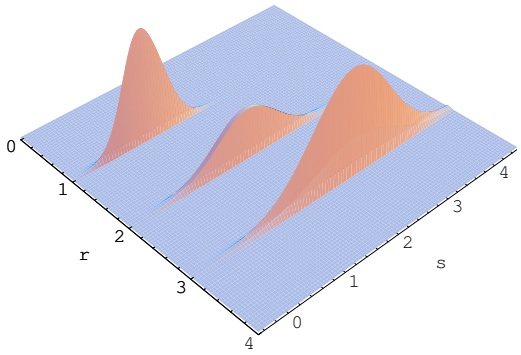
La probabilidad *a posteriori* se calcula multiplicando los valores de verosimilitud por las correspondientes probabilidades *a priori* del modelo del mundo y normalizando:



Probabilidades *a posteriori* $p(r|1.3)$

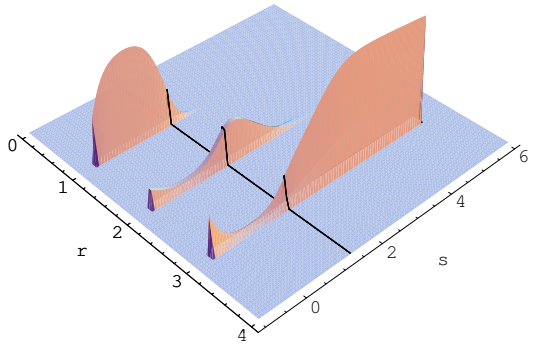
Antes de medir (*a priori*) el estado más probable era r_3. El resultado de la medida hace evolucionar las probabilidades y ahora (*a posteriori*) el estado más probable es r_2. Como ya indicamos anteriormente, este proceso de actualización Bayesiana puede repetirse con observaciones sucesivas (siempre que sean independientes) reduciendo progresivamente la incertidumbre hasta un grado aceptable. En cada etapa las probabilidades *a priori* que se utilizan son las probabilidades *a posteriori* obtenidas en la etapa anterior.

Igual que en el caso continuo, podemos construir el modelo conjunto, en el que las densidades condicionadas a cada clase quedan multiplicadas por las probabilidades *a priori*:



Modelo conjunto (clasificación)

y normalizar cada corte horizontal para conseguir el modelo inverso (se muestran las probabilidades *a posteriori* correspondientes a la observación anterior, $p(r|1.3)$):



Modelo inverso (clasificación)

Observa que, en este ejemplo, cuando la observación disminuye las tres clases tienden a ser equiprobables, y cuando aumenta la probabilidad se decanta claramente por r_3. Si los valores observados son muy extremos tanto en un sentido como en otro $p(s)$ se hace muy pequeño y, aunque aparentemente existen unas ciertas probabilidades *a posteriori* de cada clase, deberíamos rechazar la decisión porque la medida sería inconsistente con nuestros modelos.

3.3.10. Test de Bayes

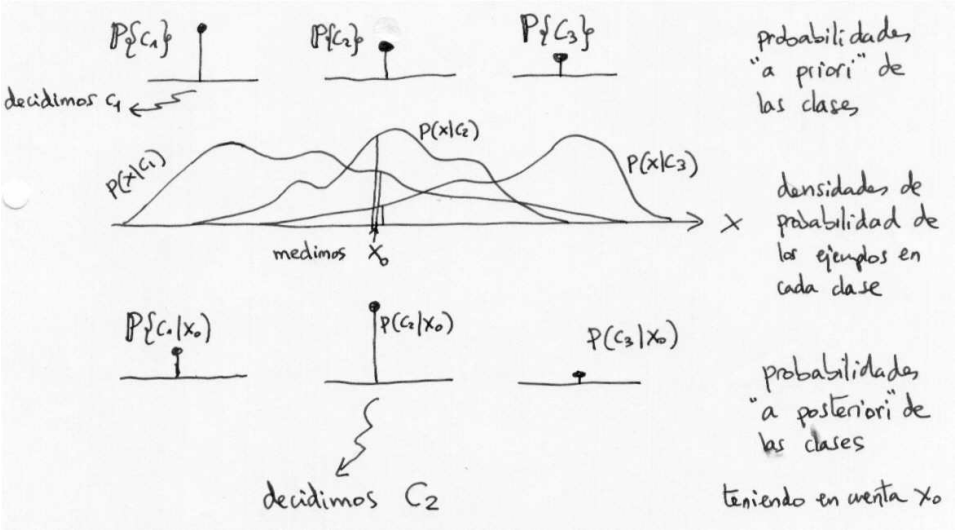
En resumen, y cambiando la notación para recordar que la magnitud de interés r es el índice de una clase $c \in \{1, \ldots, C\}$ y que el estímulo observado s es un vector

de propiedades $\boldsymbol{x}$, la regla de clasificación óptima MAP (de **m**áxima probabilidad ***a posteriori***) puede escribirse[1] como:

$$\hat{c} = \underset{c}{argmax}\{p(c|\boldsymbol{x}_o)\} = \underset{c}{argmax}\{p(\boldsymbol{x}_o|c)p(c)\} = \underset{c}{argmax}\, p(c, \boldsymbol{x}_o)$$

Por tanto, el clasificador óptimo (conocido también como *Test de Bayes*) se expresa de forma muy simple a partir de los modelos del sensor y del mundo. Intuitivamente, se decide la clase que con mayor frecuencia produce el vector de propiedades observado, lo que está de acuerdo con el sentido común, pero, como hemos visto en la discusión anterior, es necesario tener en cuenta el 'sesgo' producido por las probabilidades *a priori* .

Aun a riesgo de resultar pesados, la siguiente figura ilustra de nuevo la evolución de las probabilidades de los estados reales tras realizar una la observación:



Actualización bayesiana de las probabilidades.

Ningún otro método de clasificación, basado únicamente en la observación de las mismas propiedades $\boldsymbol{x}$, cometerá en promedio menos errores.

> Cuando sólo hay dos clases, la regla de decisión óptima se puede expresar en forma de un umbral sobre el ratio de verosimilitudes $l(x)$ (*likelihood ratio*):
>
> $$\frac{p(\boldsymbol{x}|c_1)}{p(\boldsymbol{x}|c_2)} \equiv l(\boldsymbol{x}) \underset{c_2}{\overset{c_1}{\gtrless}} h \equiv \frac{P(c_2)}{P(c_1)}$$

[1] La notación *argmax* sirve para referirnos al valor del argumento que maximiza una función. El factor de normalización $1/p(\boldsymbol{x}_o)$ es común en todos los elementos y por tanto no afecta al valor de c que maximiza la probabilidad *a posteriori*.

(la notación anterior significa que si se cumple la condición $>$ se decide la clase c_1 y si se cumple la condición $<$ se decide la clase c_2. La regla óptima también se puede expresar como un umbral sobre la *log-likelihood*:

$$-\ln l(\boldsymbol{x}) \underset{c_2}{\overset{c_1}{\gtrless}} -\ln h$$

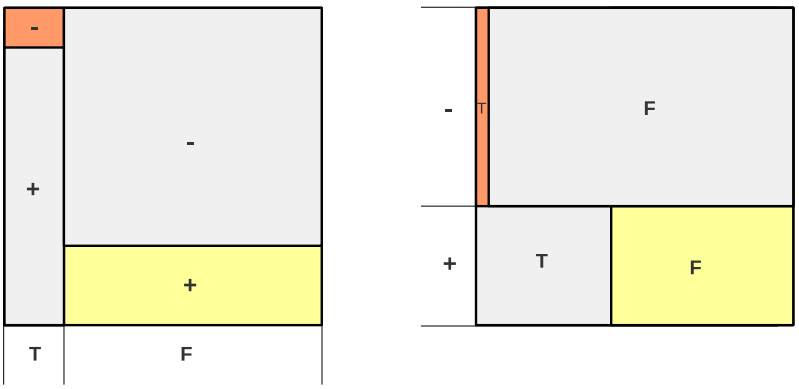
lo cual es útil para simplificar expresiones con densidades de tipo exponencial (p. ej., gaussianas).

3.3.11. Ejemplo

La situación más sencilla para ilustrar el proceso de inferencia bayesiana ocurre cuando tanto el estado del mundo r como la magnitud observable s admiten únicamente dos valores. Por ejemplo, $r \in \{T, F\}$ (el mundo se encuentra en un cierto estado o no) y $s \in \{+, -\}$ (s es el resultado de un *test* que trata de determinar el estado r y produce un resultado positivo o negativo).

La calidad del test se mide por su tasa de aciertos, pero es muy importante tener en cuenta que esa medida hay que desglosarla en la 'sensibilidad' $p(+|T)$ y la 'selectividad' $p(-|F)$. Son necesarios ambos, ninguno por separado caracteriza completamente el test. De nada sirve una gran sensibilidad (responder positivamente cuando ocurren el fenómeno) si el test no tiene selectividad y también responde positivamente cuando no ocurre. Alternativamente podríamos caracterizar el sensor mediante las probabilidades de un falso positivo $p(+|F)$ y de un falso negativo $p(-|V)$. En cualquier caso, el test queda descrito por las 4 probabilidades $p(+|T)$, $p(-|T)$, $p(+|F)$, y $p(-|F)$.

La siguiente figura muestra una interpretación gráfica de la densidad conjunta y sus dos factorizaciones:



Teorema de Bayes

Supongamos que nuestro test tiene un 99 % de sensibilidad y un 95 % de selectividad. Si efectuamos el test y obtenemos $s = +$. ¿Qué podemos decir de r?

¿Es T ó F? Lo cierto es que si desconocemos las probabilidades *a priori* del fenómeno no podemos afirmar mucho... Supongamos que la 'prevalencia' general del fenómeno es $p(T) = 1/1000$. Antes de efectuar el test la probabilidad de que ocurra es uno entre mil. Un sencillo cálculo muestra que si el test da positivo la probabilidad evoluciona de $p(T) = 0.1\,\%$ a $p(T|+) \simeq 2\,\%$. El test es muy preciso y cuando da positivo incrementa mucho la probilidad del fenómeno que trata de medir, pero debido al valor de las probabilidades *a priori* aún así sigue siendo mucho más probable que no ocurra. El test da posivo en situaciones T, acertando, y también en situaciones F, equivocándose, y en este caso estas últimas son más numerosas. Si el resultado de un segundo test independiente también fuera positivo ya sí se haría más probable *a posteriori* el estado T.

Observa la evolución del ratio de verosimilitudes. Antes de hacer ningún test F es casi 1000 veces más probable que T. Tras el primer resultado positivo:

$$\frac{\frac{1}{1000}}{\frac{999}{1000}}\frac{\frac{99}{100}}{\frac{5}{100}} = \frac{11}{555} \simeq 1/50$$

es decir, ahora F es sólo unas 50 veces más probable que T. Tras un segundo positivo tendríamos:

$$\frac{11}{555}\frac{99}{5} = \frac{363}{925} \simeq 1/2.5$$

Todavía el estado T es unas 2 veces y media más probable que F ($p(T|++) \simeq 72\,\%$).

Un tercer positivo sí haría más probable el estado F.

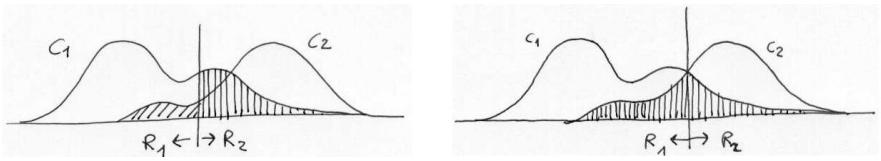
3.3.12. Error de Bayes

En la Sección 1.6 introdujimos el concepto de *error intrínseco*, relacionado con el 'solapamiento' de las nubes de puntos en el espacio de propiedades. Ahora estamos en condiciones de formalizar un poco mejor esta idea. Si las propiedades $\boldsymbol{x}$ son muy ambiguas (obtienen valores parecidos en clases diferentes) se cometerán inevitablemente muchos errores de clasificación, incluso aunque usemos el método óptimo que acabamos de describir. El error intrínseco EB (*Error de Bayes*) es simplemente la probabilidad de error que conseguiría el método de clasificación óptimo. Cualquier otro método, incluyendo los algoritmos 'prácticos' que usaremos en situaciones reales, tendrán una probabilidad de error no inferior a EB, que es algo así como el ideal de precisión que nos gustaría conseguir.

La probabilidad de error de cualquier método de clasificación (que, en definitiva, se reducirá a unas regiones y fronteras de clasificación más o menos bien elegidas) es la proporción de observaciones de cada clase que caen en la región de decisión equivocada). Las nubes de puntos quedan descritas matemáticamente por las densidades de probabilidad $p(\boldsymbol{x}|c)$ del modelo del sensor. El conjunto de valores s en los

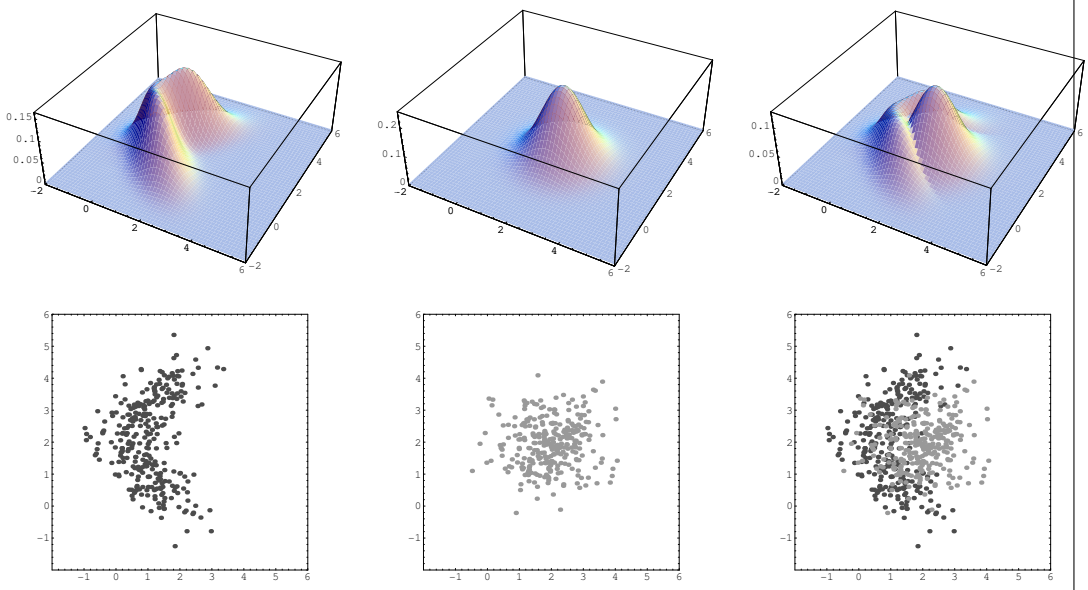
que $p(c_k|\boldsymbol{x})p(c_k)$ es mayor que los demás es la región de decisión óptima de la clase k-ésima y donde se 'cruzan' se encuentra la frontera de decisión óptima.

La figura siguiente ilustra gráficamente la probabilidad de error de una regla de clasificación no muy buena (izquierda) y el error de Bayes de ese mismo problema (derecha).



Observa que hay una cantidad de error inevitable (el mínimo de las dos funciones) y que un clasificador que no se sitúe en la frontera óptima cometerá un error adicional.

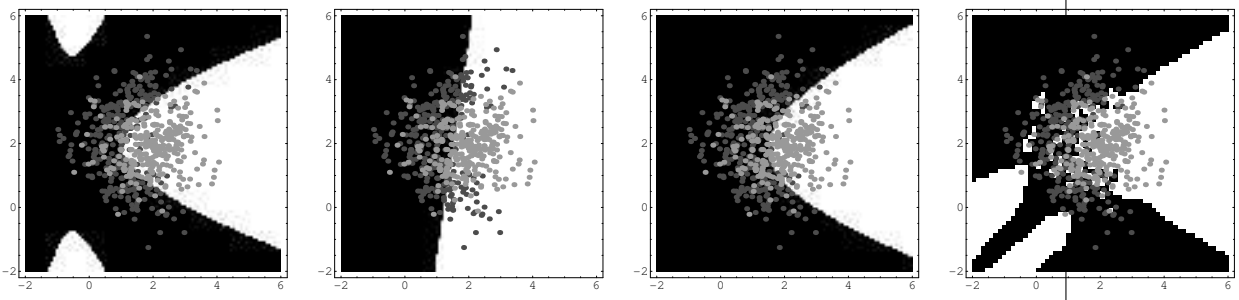
Cuando el vector de propiedades es bidimensional las regiones de decisión son regiones planas y la fronteras entre ellas son líneas. En la figura siguiente se muestran dos densidades de probabilidad que generan unos objetos bidimensionales (muestras con dos propiedades que se representan como puntos en el plano):



Modelo de un sensor que mide dos propiedades: $p(x_1, x_2|c_1)$ (izquierda), $p(x_1, x_2|c_2)$ (centro), y ambas (derecha). Debajo de cada una se incluye un conjunto de observaciones obtenido de cada modelo.

Las regiones y fronteras de decisión de la regla MAP para este problema sintético, suponiendo que las clases c_1 y c_2 son equiprobables, se ilustran a continuación y se

comparan con las obtenidas por los métodos de clasificación sencillos propuestos al principio de este capítulo:



De izquierda a derecha, fronteras de decisión del método óptimo (probabilidad de aciertos de 0.79), mínima distancia a la media (0.72), mínima distancia de Mahalanobis (0.76) y vecino más próximo (0.74).

Este problema tiene mucha ambigüedad (el método óptimo comete más del 20 % de fallos). Curiosamente, la distancia de Mahalanobis consigue un resultado muy bueno a pesar de que una de las clases tiene una distribución curvada. El método del vecino más próximo produce una frontera bastante flexible pero generaliza peor al estar basado en las muestras directamente observadas, sin obtener una representación compacta del problema.

Lógicamente, es deseable tener una idea de la magnitud de EB para saber lo alejado que está del óptimo teórico el clasificador que hemos diseñado. Desafortunadamente, el EB de un problema concreto es difícil de calcular incluso si conociéramos perfectamente (lo que casi nunca ocurre) el modelo del sensor; calcularlo con precisión a partir únicamente de información muestral es una tarea extremadamente dificultosa que casi nunca merece la pena intentar.

> El error de clasificación (por simplicidad consideramos solo dos clases) de un cierto método que define regiones de decisión R_0 y R_1 se puede expresar formalmente como:
>
> $$PE = p(1)\int_{R_0} p(\boldsymbol{x}|1)d\boldsymbol{x} + p(0)\int_{R_1} p(\boldsymbol{x}|0)d\boldsymbol{x}$$
>
> Este valor teórico es difícil de calcular analíticamente incluso conociendo $p(\boldsymbol{x}|c)$. Normalmente recurrimos a estimaciones estadísticas como las explicadas en la Sección 3.2. Para el Error de Bayes existe una expresión más concisa (aunque sin excesiva utilidad práctica). Las regiones de decisión del clasificador óptimo R_0 y R_1 son las zonas donde los integrandos son uno mayor que el otro, por lo que:
>
> $$EB = \int \min_c \{p(c)p(\boldsymbol{x}|c)\}d\boldsymbol{x}$$

La tasa de error de un clasificador debe compararse con 1 - $max_c\{p(c)\}$, la probabilidad de error del mejor método 'ciego' que, sin mirar el vector de propiedades, decide la clase más probable. (Como ejemplo, un clasificador que consigue un 92 % de aciertos no es realmente muy espectacular si en ese problema una de las clases aparece el 90 % de las veces.)

Se han propuesto aproximaciones (típicamente cotas superiores) del EB basadas en suposiciones sobre las densidades condicionadas. A veces pueden resultar útiles como indicadores de la complejidad del problema o de la calidad del vector de propiedades.

Recordemos que si el EB de un problema de clasificación es muy grande la única alternativa es cambiar las propiedades $\boldsymbol{x}$; pero si la ambigüedad está en la información original (p. ej., número cero / letra 'O') la única posibilidad de deshacerla es en una etapa posterior de interpretación.

3.3.13. Ponderación de los errores.

El análisis probabilístico precedente, junto con conceptos elementales de la Teoría de la Decisión (Sección C.4, pág. 235), nos permite también resolver la situación en la que unos errores son más graves que otros. Si tratamos de distinguir bombas de manzanas o en el diagnóstico de una enfermedad un falso positivo puede ser menos grave que un falso negativo. Mejor que el simple número de errores de clasificación, deseamos minimizar el 'coste' total que nos suponen los errores producidos.

La función de coste de este ejemplo podría ser:

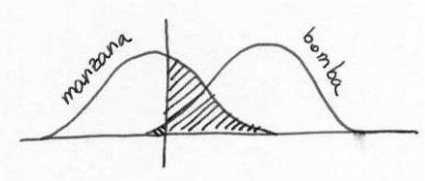
$$L = \begin{bmatrix} 0 & 1000 \\ 1 & 0 \end{bmatrix}$$

indicando que es 1000 veces más costoso confundir una bomba con una manzana que a la inversa.

El riesgo *a posteriori* de la decisión c_i es $R_i(x) = \sum_i L_{ij} P(c_j|x)$. Debemos tomar la decisión de mínimo riesgo. Cuando hay dos clases, la decisión óptima se expresa de forma concisa como:

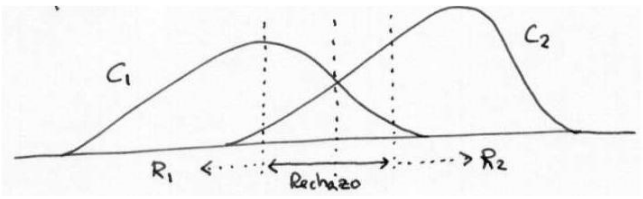
$$\frac{p(x|c_1)}{p(x|c_2)} \underset{c_2}{\overset{c_1}{\gtrless}} \frac{L_{12} - L_{22}}{L_{21} - L_{11}} \frac{P(c_2)}{P(c_1)}$$

esto es, el coste de los errores sólo afecta al umbral del test de Bayes expresado en términos del ratio de verosimilitudes. La frontera de decisión se desplaza hacia dentro de la región menos 'grave', de manera que ante observaciones ambiguas se toma la decisión menos comprometedora. Se cometerá un numero mayor de errores, pero en su desglose aparecerá una proporción menor de los errores más costosos. Si la tabla de costes L es simétrica la decisión óptima minimiza la probabilidad de error.



3.3.14. Rechazo.

Si la observación disponible no reduce suficientemente la incertidumbre sobre la clase real –o en términos de riesgo (Sección C.4), si el coste medio de la decisión óptima es aun así demasiado alto– puede ser recomendable *rechazar* la decisión (decidir que no se quiere decidir). Las fronteras se 'ensanchan' creando un 'banda de seguridad' sobre las zonas de solapamiento de las nubes de cada clase. Cuando el coste es 0-1 (probabilidad de error) rechazamos si la probabilidad *a posteriori* de la clase ganadora no es muy próxima a 1. Esta estrategia reduce la probabilidad de error del clasificador todo lo que se desee, pero por supuesto a costa de no tomar decisiones, lo que reduce también la tasa de aciertos global.



3.3.15. Combinación de Información.

Si las componentes del vector de propiedades son *condicionalmente independientes* el clasificador óptimo se puede expresar como una especie de votación ponderada, en la que cada propiedad contribuye aisladamente a la decisión con una cierta cantidad a favor o en contra (que depende de lo discriminante que sea). Esta idea se puede aplicar también para acumular evidencia obtenida por observaciones sucesivas (e independientes) de una misma propiedad.

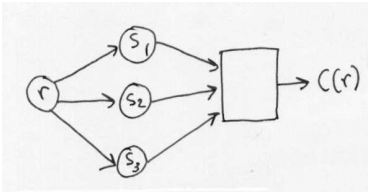
Cuando hay independencia condicional las densidades condicionadas pueden expresarse como un producto de densidades marginales:

$$p(\boldsymbol{x}|c) = p(x_1|c)p(x_2|c)\dots p(x_n|c)$$

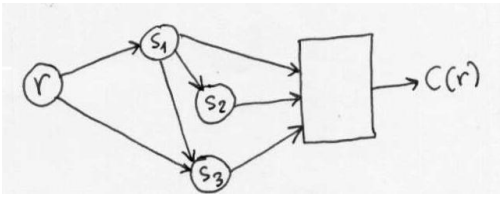
por tanto en su versión de *log-likelihood* el cociente de verosimilitud global se convierte en la suma de los cocientes de cada componenente del vector de propiedades:

$$\ln l(\boldsymbol{x}) = \ln \frac{p(\boldsymbol{x}|1)}{p(\boldsymbol{x}|2)} = \ln \frac{p(x_1|1)}{p(x_1|2)} + \ln \frac{p(x_2|1)}{p(x_2|2)} + \ldots + \ln \frac{p(x_n|1)}{p(x_n|2)} - \ln \frac{p(2)}{p(1)} =$$
$$= l(x_1) + l(x_2) + \ldots + l(x_n) + K$$

Esto sugiere que cuando las fuentes de información sobre un cierto hecho son independientes entre sí podemos combinarlas siguiendo una regla muy simple. P. ej., si varios 'jueces' aproximadamente igual de fiables observan *directamente* un hecho, una simple votación es adecuada para combinar sus decisiones individuales.



En caso contrario la combinación correcta de todos los resultados requiere inevitablemente tener en cuenta la densidad conjunta. Si varias fuentes de información han tenido un 'intermediario' común deben reducir su 'peso' en la decisión final.



La decisión óptima probabilística bajo la suposición de atributos condicionalmente independientes se conoce como *naive Bayes*.

Ejercicios:

- Analiza la probabilidad de que al hacer n experimentos independientes, cada uno con probabilidad de éxito p, obtengamos una mayoría de éxitos. (Un experimento con probabilidad p se puede interpretar como un 'juez', un 'votante' o un clasificador, cuya probabilidad de tomar la decisión correcta es p.).

3.4. Estimación de densidades

Desde el punto de vista teórico anteriormente expuesto, toda la información necesaria para efectuar inferencias probabilísticas se encuentra en la densidad conjunta de las variables involucradas, que nos permite calcular cualquier densidad condicionada

mediante normalización de los cortes correspondientes. En problemas de percepción el modelo conjunto se expresa de forma natural como el producto del modelo del sensor y el modelo del mundo: $p(s, r) = p(s|r)p(r)$, o, en el caso de clasificación a partir de vectores de propiedades: $p(\boldsymbol{x}, c) = p(\boldsymbol{x}|c)p(c)$.

Por tanto, cuando nos enfrentamos a una aplicación real, el sentido común podría sugerir la idea de obtener de alguna manera estos modelos probabilísticos e 'insertarlos' en la regla de decisión óptima.

Una posibilidad sería intentar deducir el modelo del sensor y el del mundo a partir de conocimientos teóricos sobre el problema en cuestión. Sin embargo, en la mayoría de los casos relacionados con la percepción artificial, este enfoque no es viable. P. ej., no parece fácil encontrar mediante consideraciones teóricas una descripción matemática precisa de, p. ej., la distribución de tinta en las imágenes de letras manuscritas en sobres de cartas enviadas por correo.

Una alternativa podría consistir en recoger a un número suficientemente grande de ejemplos de vectores de propiedades, etiquetados con la clase a que pertenecen, para construir algún tipo de estimación empírica, estadística o aproximación a las distribuciones de probabilidad involucradas.

> En principio, esto podría hacerse de dos maneras. Cuando observamos 'desde fuera' un cierto sistema mundo-sensor (las variables r y s) podemos registrar las parejas de ocurrencias (s, r). Y mediante las ténicas que explicaremos más adelante, podemos construir una aproximación a la densidad conjunta $p(s, r)$.
>
> Alternativamente, podemos 'manipular' o controlar un poco el mundo y fijar (o filtrar) cada estado posible r_i para registrar la distribución de respuestas frente a ese estado y conseguir las correspondientes densidades $p(s|r_i)$. De esta forma modelaríamos el sensor. Luego necesitaríamos estimar el modelo del mundo a partir de la frecuencia de aparición de cada posible estado r_i. Este segundo método es más natural para abordar problemas de clasificación.

La estimación de las probabilidades *a priori* es una tarea comparativamente sencilla. Podemos usar directamente la frecuencia observada de aparición de cada tipo de objeto en nuestro problema. Por suerte, cuando las propiedades son suficientemente discriminantes, la influencia de las probabilidades *a priori* es relativamente pequeña y en general no cambiarán la decisión tomada por el clasificador. Sólo tienen influencia decisiva cuando la medida $\boldsymbol{x}_o$ es muy ambigua, en cuyo caso lo mas razonable es rechazar la decisión.

Por el contrario, las densidades condicionadas que constituyen el modelo del sensor –esenciales para construir el clasificador óptimo– son en general mucho más difíciles de estimar, especialmente cuando las observaciones son multidimensionales. Se trata de un problema mal condicionado (*ill posed*), que requiere explícita o implícitamente algún tipo de 'regularización' o suavizado que no es sencillo deducir simplemente de los datos disponibles.

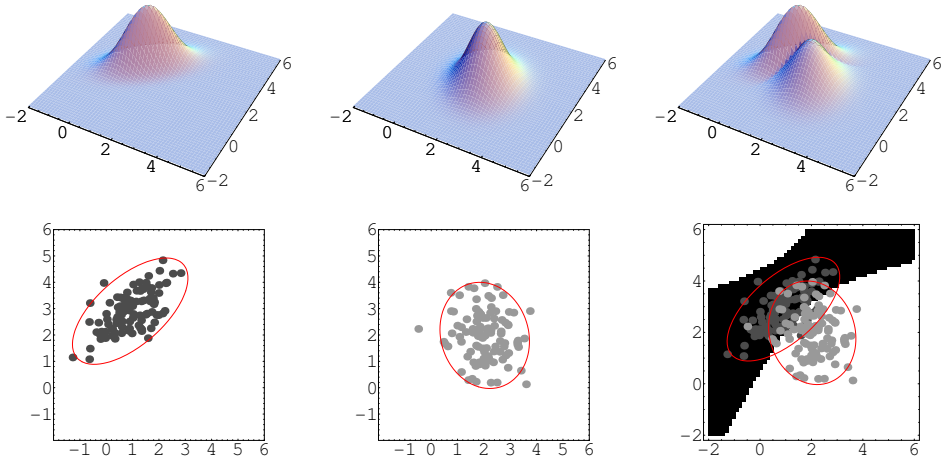
A continuación vamos a comentar algunos métodos prácticos de estimación de densidades de probabilidad que se han utilizado frecuentemente para fabricar clasificadores secillos que, en algún sentido, pueden considerarse como 'aproximaciones' más o menos ajustadas al clasificador óptimo. Pero es muy importante insistir en que este enfoque no es el ideal. Estimar densidades de probabilidad para –mediante el test de Bayes– trazar la frontera de clasificación es dar 'un rodeo' innecesario: nos enfrentamos a un problema más complejo como herramienta auxiliar para resolver uno más simple. En realidad, de acuerdo con la teoría matemática de la generalización inductiva desarrollada por Vapnik [27] (que estudiaremos en el capítulo siguiente), ajustar directamente una función de decisión (situar una buena frontera) es más 'fácil' que estimar densidades de probabilidad completas sobre todo el espacio de propiedades.

3.4.1. Métodos Paramétricos

La manera más simple de estimar una densidad de probabilidad es suponer que pertenece a una cierta 'familia' y estimar sus parámetros. (La validez de esa suposición puede comprobarse mediante tests estadísticos que permiten analizar si una muestra es consistente o no con un cierto modelo.)

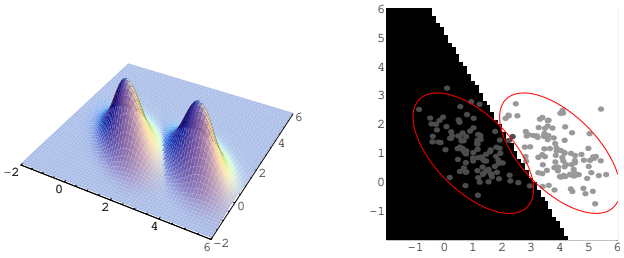
Un modelo típico es la densidad gaussiana multidimensional. Es matemáticamente manejable y describe bien las mediciones de una magnitud real contaminada con ruido que depende de muchos factores independientes (una situación física usual).

Los únicos parámetros que hay que estimar son los valores medios y las matrices de covarianza del vector de propiedades en cada clase. Al insertar los modelos estimados en el test de Bayes se obtienen fronteras cuadráticas relacionadas con la distancia de Mahalanobis a cada clase:

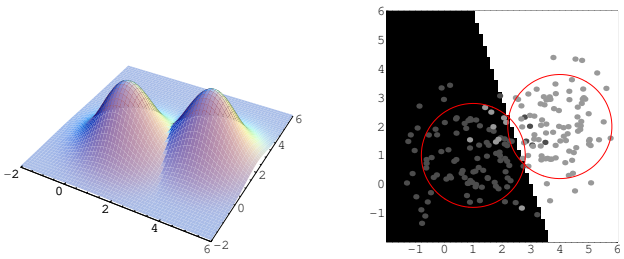


Modelos gaussianos generales

Si da la casualidad de que las matrices de covarianza de las clases son iguales, entonces se cancelan los términos cuadráticos y las fronteras se hacen lineales (hiperplanos):



Si además las nubes son de tamaño esférico, del mismo tamaño, y las clases son equiprobables, el clasificador óptimo se reduce al método extremadamente sencillo de mínima distancia a las medias de cada clase:



Para comprobar que el clasificador óptimo para clases gaussianas produce fronteras cuadráticas expresamos el test de Bayes en forma de *log-likelihood*. Si $p(\boldsymbol{x}|c_i) = N(\boldsymbol{x}, \boldsymbol{\mu}_i, \Sigma_i)$, tenemos que

$$-\ln l(\boldsymbol{x}) = -\ln \frac{p(\boldsymbol{x}|c_1)}{p(\boldsymbol{x}|c_2)} = -\ln \frac{N(\boldsymbol{x}, \boldsymbol{\mu}_1, \Sigma_1)}{N(\boldsymbol{x}, \boldsymbol{\mu}_2, \Sigma_2)} \underset{c_2}{\overset{c_1}{\gtrless}} -\ln \frac{P(c_2)}{P(c_1)}$$

Cancelando exponenciales y logaritmos podemos escribir:

$$\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\Sigma_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\Sigma_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) + \frac{1}{2}\ln\frac{|\Sigma_1|}{|\Sigma_2|} \underset{c_2}{\overset{c_1}{\gtrless}} \ln\frac{P(c_1)}{P(c_2)}$$

que no es más comparar las distancias de Mahalanobis a cada densidad:

$$d_1^2(\boldsymbol{x}) - d_2^2(\boldsymbol{x}) + cte. \underset{c_2}{\overset{c_1}{\gtrless}} 0$$

Cuando las dos clases tienen una matriz de covarianza común Σ_1=Σ_2=Σ el término cuadrático se cancela y la regla de decisión es lineal:

$$A\boldsymbol{x} \underset{c_2}{\overset{c_1}{\gtrless}} b$$

donde los componentes de $\boldsymbol{x}$ se ponderan con $A = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T\Sigma^{-1}$ y el umbral de decisión es $b = -\frac{1}{2}(\boldsymbol{\mu}_1^T\Sigma\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T\Sigma\boldsymbol{\mu}_2) + \ln P(c_1) - \ln P(c_2)$. Finalmente, si además las clases son esféricas ($\Sigma = I$) y equiprobables ($P(c_1) = P(c_2)$) el clasificador óptimo gaussiano se reduce a la regla de mínima distancia a la media: los pesos se reducen a $A = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$ y el umbral $b = \frac{1}{2}(\boldsymbol{\mu}_2^T\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\boldsymbol{\mu}_1)$, dando lugar a

$$||\boldsymbol{x} - \boldsymbol{\mu}_2|| \underset{c_2}{\overset{c_1}{\gtrless}} ||\boldsymbol{x} - \boldsymbol{\mu}_1||$$

Estos resultados indican que, independientemente de la magnitud del error intrínseco del problema de clasificación, en ciertos casos la solución óptima se reduce a un algoritmo elemental. Por supuesto, las simplificaciones anteriores nunca se verificarán exactamente en la práctica. Lo importante es que, en algunos casos (p. ej., cuando las propiedades son muy discriminantes), son aproximaciones aceptables que no tiene sentido intentar mejorar con métodos de clasificación más complejos.

→ Estimación de los parámetros $\boldsymbol{\mu}$ y Σ.

→ Estimación incremental.

3.4.2. Métodos No Paramétricos

Si no tenemos ninguna información sobre el tipo de densidad de probabilidad, podemos usar métodos *no paramétricos*, que pueden 'adaptarse' a la forma geométrica de cualquier nube, pero en contrapartida necesitan un número mucho mayor de muestras para conseguir resultados aceptables.

Un método muy sencillo de este tipo es la estimación frecuencial directa: aproximamos la densidad de probabilidad en un punto a partir de la frecuencia relativa de aparición de muestras en un región pequeña alrededor de ese punto. En espacios de dimensión muy reducida (2 ó 3) podríamos usar directamente un histograma como aproximación a la densidad de probabilidad.

Método de vecinos

Una idea mejor es utilizar una región de tamaño variable, p. ej., con forma esférica, que contiene un número k de muestras. Curiosamente, insertando este tipo de

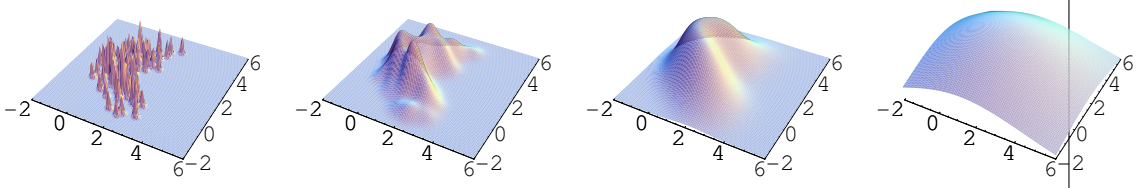
estimación frecuencial en el test de Bayes obtenemos el método de clasificación de los k vecinos más próximos (el tamaño de la región y los números de muestras se cancelan). De nuevo, un método de clasificación muy sencillo es una buena aproximación al método óptimo.

→ justificar

En condiciones razonables, esta estimación frecuencial converge a la densidad real. Es fácil demostrar que, cuando hay 'infinitas' muestras, la clasificación basada en el vecino más próximo ($k = 1$) tiene un error que es no más del doble del error de Bayes. Pero si la muestra es finita no se puede garantizar mucho sobre el error de este método.

Método de Parzen

Otro método no paramétrico muy utilizado es el de Parzen. La idea es situar una función de densidad sencilla (*kernel* rectangular, triangular, gaussiano, etc.) encima de cada muestra y promediar el valor de todas ellas. Lógicamente, la regiones con mucha densidad de puntos contribuyen con mayor altura a la densidad estimada. La 'anchura' del *kernel* es un parámetro de suavizado que debemos ajustar para que la estimación no se pegue excesivamente a los datos, generando un conjunto de picos independientes, ni tampoco 'emborrone' completamente la distribución de puntos:



Estimaciones de Parzen con anchura de ventana creciente

Se puede demostrar que el método de Parzen es una estimación empírica de una versión suavizada de la densidad real. Converge a ella si la anchura del *kernel* se va haciendo progresivamente más pequeña (pero no demasiado rápido) al aumentar el número de muestras. En la práctica la muestra es finita, por lo que el nivel de suavizado debe estimarse heurísticamente mediante, p. ej., técnicas de *cross-validation*. A veces hay que usar un grado de suavizado variable en cada punto o región, lo que complica todavía más el método.

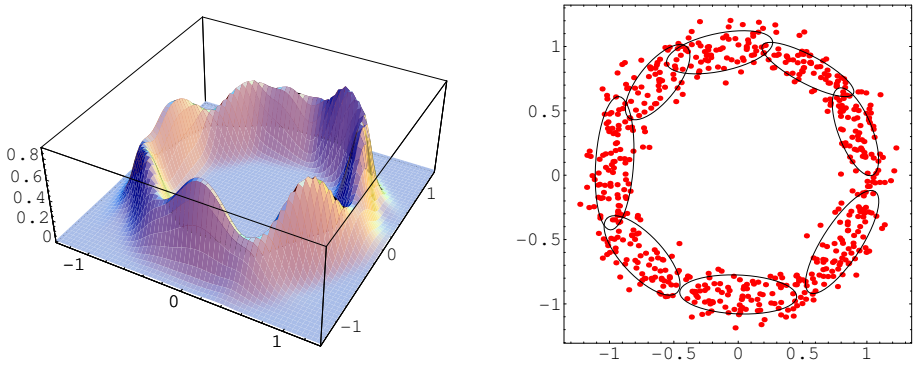
→ justificar

En general los estimadores no paramétricos producen clasificadores que no son 'concisos' (usan toda la base de datos de ejemplos disponibles sin extraer de ella ninguna

información relevante) y que en consecuencia poseen un tiempo de ejecución que puede ser inaceptablemente largo. (Se han propuesto técnicas para reducir el tamaño de la base de ejemplos, p. ej., eliminando muestras que no afectan a la regla de clasificación.)

3.4.3. Modelos del mezcla

Una solución intermedia entre los métodos paramétricos (que sitúan una sola densidad simple encima de todas las muestras) y los no paramétricos (que sitúan una densidad simple encima de todas y cada una de las muestras) es utilizar métodos 'semiparamétricos', como los basados en modelos de mezcla. La idea es expresar la densidad como una suma de un número pequeño de densidades simples cuyo tamaño y posición se ajusta para situarse encima de la nube de puntos adaptándose lo mejor posible a los detalles de la distribución.



Modelo de mezcla de gaussianas

Los parámetros de un modelo de mezcla se estiman eficientemente mediante el notable algoritmo EM (*Expectation-Maximization*, [17]). Se trata de un método general para estimar modelos probabilísticos en los que los datos tienen información 'incompleta'. P. ej., en el caso de un modelo de mezcla ignoramos qué componente ha 'generado' cada muestra. Si dispusiéramos de esa información sería inmediato estimar los parámetros de cada componente individual usando las muestras que le corresponden. El método funciona iterando sucesivamente el paso E, en el que se calcula el valor esperado de la información incompleta dados los parámetros estimados en la iteración anterior, y el paso M, en el que se reestiman los parámetros del modelo a partir de la muestra y de la estimación de la información incompleta obtenidas en el paso E anterior. El algoritmo no tiene parámetros ajustables y en condiciones normales converge rápidamente a un máximo local de la función de verosimilitud del modelo.

Un modelo de mezcla tiene la forma siguiente:

$$p(\boldsymbol{x}) = \sum_{c=1}^{L} \mathrm{P}\{c\} p(\boldsymbol{x}|c)$$

Podemos interpretarla diciendo que la naturaleza elige una componente c con probabilidad $\mathrm{P}\{c\}$ y genera una observación $\boldsymbol{x}$ con la distribución de esa componente, $p(\boldsymbol{x}|c)$. Nosotros vemos las observaciones de todas las componentes 'aglutinadas', no sabemos qué componente ha generado cada observación.

Si las componentes son gaussianas el modelo toma la forma

$$p(\boldsymbol{x}) = \sum_{c=1}^{L} \pi_c \, \mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}_c, \Sigma_c)$$

donde las 'proporciones' de cada componente se abrevian como $\pi_c = \mathrm{P}\{c\}$ y las densidades de cada componente son gaussianas con una cierta media y matriz de covarianza: $p(\boldsymbol{x}|c) = \mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}_c, \Sigma_c)$.

A partir de un conjunto de muestras $\{\boldsymbol{x}_i\}_{i=1}^n$ deseamos estimar los parámetros $\{\pi_c, \boldsymbol{\mu}_c, \Sigma_c\}_{c=1}^L$ que maximizan la verosimilitud $J = \prod_i p(\boldsymbol{x}_i)$. Este producto de sumas es una función de coste matemáticamente complicada de optimizar directamente (tomar logaritmos no ayuda). Pero observa que si conociéramos la componente que ha generado cada observación, podríamos estimar de manera independiente los parámetros de cada gaussiana. El algoritmo EM estima la procedencia de cada muestra y recalcula los parámetros iterativamente (realiza un ascenso de gradiente (Sección D.3) sobre J). Intuitivamente, podemos entender fácilmente su funcionamiento pensando que los parámetros son los valores esperados de ciertas funciones $g(x)$ condicionados a cada componente. P. ej., para la media de cada componente $\mu_c = \mathrm{E}\{x|c\}$ tendríamos $g(x) = x$, para la matriz de covarianza tendríamos $g(x) = (x-\mu)(x-\mu)^T$ y para la proporción $g(x) = 1$. Usando el teorema de Bayes podemos expresar valores esperados condicionados como no condicionados:

$$\mathrm{E}\{g(x)|c\} = \frac{\mathrm{E}\{g(x)\,\mathrm{P}\{c|x\}\}}{\mathrm{P}\{c\}}$$

Esto se ve claramente si hacemos explícitas las densidades de los valores esperados:

$$\mathrm{E}_{p(x|c)}\{g(x)|c\} = \int g(x) p(x|c) dx = \int g(x) \frac{\mathrm{P}\{c|x\}p(x)}{\mathrm{P}\{c\}} dx = \frac{\mathrm{E}_{p(x)}\{g(x)\,\mathrm{P}\{c|x\}\}}{\mathrm{P}\{c\}}$$

Los valores esperados no condicionados se pueden estimar con la muestra $\{x_i\}$ observada:

$$\mathrm{E}\{g(x)\,\mathrm{P}\{c|x\}\} \simeq \frac{1}{n} \sum_{i=1}^{n} g(x_i)\,\mathrm{P}\{c|x_i\}$$

Aparecen unos coeficientes $\mathrm{P}\{c|x_i\}$ (la probabilidad de que x_i haya sido generado por la componente c) que son el peso de cada muestra para calcular los parámetros de cada componente. En principio son desconocidos pero podemos estimarlos iterativamente.

El algoritmo EM para estimar una mezcla de gaussianas es el siguiente. Primero inicializamos los parámetros$\{\pi_c, \boldsymbol{\mu}_c, \Sigma_c\}_{c=1}^{L}$ aleatoriamente. Después repetimos los pasos E y M hasta que la verosimilitud del modelo J deje de aumentar:

- Paso E: calculamos los coeficientes $q(c,i) = \mathrm{P}\{c|x_i\}$ con los parámetros actuales. (Es una tabla de L valores para cada muestra x_i.). Usamos las densidades de cada componente y sus proporciones:

$$q'(c,i) \leftarrow \pi_c \, \mathcal{N}(\boldsymbol{x}_i, \boldsymbol{\mu}_c, \Sigma_c)$$

  y normalizamos cada juego de probabilidades:

$$q(c,i) \leftarrow \frac{q'(c,i)}{\sum_{s=1}^{L} q'(s,i)}$$

- Paso M: recalculamos los parámetros con los nuevos coeficientes de 'pertenencia' $q(c,i)$. En primer lugar las proporciones, que son simplemente los valores esperados de la 'etiqueta' de cada clase $(1|c)$:

$$\pi_c \leftarrow \frac{1}{n} \sum_{i=1}^{n} q(c,i)$$

  y después la media y matriz de covarianza de cada componente:

$$\boldsymbol{\mu}_c \leftarrow \frac{1}{n\pi_c} \sum_{i=1}^{n} q(c,i) \; \boldsymbol{x}_i$$

$$\Sigma_c \leftarrow \frac{1}{n\pi_c} \sum_{i=1}^{n} q(c,i) \; (\boldsymbol{x}_i - \boldsymbol{\mu}_c)(\boldsymbol{x}_i - \boldsymbol{\mu}_c)^T$$

Algunas características de EM son:

- Todas las muestras se utilizan para recalcular todas las componentes (a diferencia de otros métodos de agrupamiento más simples, como K-medias.)
- No requiere ningún parámetro de convergencia, a diferencia de otros métodos de optimización como el *backpropagation* (Sección 4.6).
- Consigue máximos locales.

- Para calcular automáticamente el mejor número de componentes se establece un compromiso entre la complejidad del modelo y la calidad del ajuste basado en *mínima longitud de descripción*:

$$Q = -L + \frac{1}{2} P \log n$$

donde L es la *log likelihood*, P es el número de parámetros y n el número de muestras.

Aprendizaje no supervisado. Los modelos de mezcla están muy relacionados con las técnicas de agrupamiento automático (*cluster analysis*), que tratan de encontrar cierta estructura en una población. El planteamiento general está basado en optimización combinatoria: se trata de encontrar una partición de los datos que consiga un buen compromiso entre la similitud intragrupo y la diferencia intergrupo.

Se han propuesto ciertas heurísticas para conseguir algoritmos eficientes. P. ej., :

- K-medias: se establecen semillas aleatorias y se calculan los más próximos a cada una. Las semillas se reestiman y se repite el proceso hasta la convergencia.
- Grafos de expansión y corte de arcos atípicos.
- Agrupamiento jerárquico (*dendogramas*), por división o unión.
- Detección de valles de probabilidad (enfoque no paramétrico).
- Técnicas de IA simbólica (formación de conceptos, etc.).

Por supuesto, los modelos de mezcla son una alternativa prometedora en este contexto. Pero hay que tener en cuenta que las componentes encontradas por el algoritmo EM son simplemente bloques constructivos de un buen modelo global de la población, y pueden no tener ningún significado especial.

3.5. Aprendizaje Bayesiano

→ pendiente

3.6. Selección de Modelos

→ pendiente

Capítulo 4

Máquinas de Aprendizaje

Nothing is more practical than a good theory.

–Vapnik

4.1. Introducción

Siempre ha existido interés por conocer la capacidad y las limitaciones de las máquinas para realizar tareas. Por ejemplo, la *computabilidad* estudia los problemas de decisión que tienen solución algorítmica; la *complejidad* estudia los problemas que pueden resolverse mediante algoritmos eficientes, etc. En este contexto podemos incluir también la *teoría computacional del aprendizaje*, que estudia los problemas cuya solución puede 'aprenderse' automáticamente por la experiencia. Desde los primeros tiempos de la informática se intentó modelar parte de las capacidades cognitivas de alto nivel con las técnicas clásicas de la inteligencia artificial. Y, a la vez, se estudiaron máquinas que trataban de reconocer situaciones simples mediante algoritmos de corrección de error. ¿Es posible fabricar máquinas que se 'autoorganicen' y puedan programarse automáticamente?

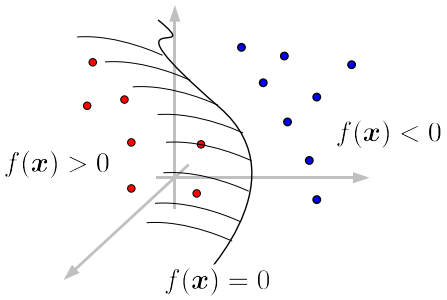
En el capítulo anterior hemos explicado varios métodos de diseño de clasificadores basados en la estimación explícita de los ingredientes del clasificador óptimo bayesiano a partir de las muestras disponibles. Algunos (p. ej., la distancia de Mahalanobis) son muy sencillos y funcionan sorprendentemente bien aunque no se cumplan exactamente las suposiciones realizadas. Otros, como los métodos de vecinos más próximos o de Parzen, en teoría son más generales pero poseen un mayor coste computacional. En todos los casos se intentaba obtener aproximaciones a la densidades condicionadas del modelo del sensor sobre *todo* el espacio de propiedades. Sin embargo, lo que realmente es importante para nosotros es la *frontera de decisión*, que depende de esas densidades

pero sólo en las regiones donde tienen valores parecidos. La forma concreta del modelo del sensor en regiones alejadas de la frontera no influye en la clasificación. Por tanto, en este capítulo estudiaremos un enfoque más directo del aprendizaje, tratando de encontrar directamente una buena frontera de decisión.

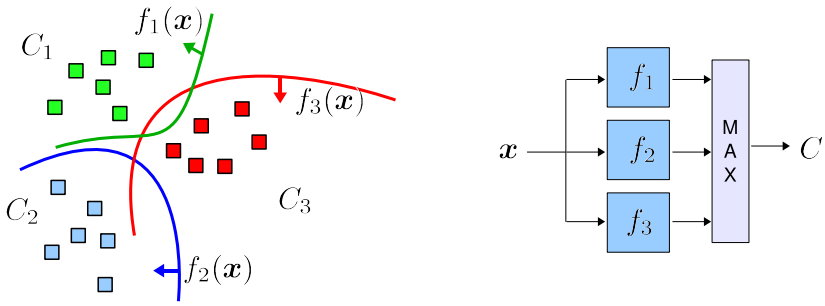
Problema de clasificación. Suponemos que un cierto sistema emite pares $(\boldsymbol{x}, c)$ de acuerdo con una cierta distribución de probabilidad $F(\boldsymbol{x}, c)$, donde $\boldsymbol{x} \in \mathbb{R}^n$ es el vector de propiedades y $c \in \{+1, -1\}$ indica la clase (positiva o negativa) a la que realmente pertenece. Esta distribución es desconocida, pero disponemos de una muestra de entrenamiento $S_m = \{(\boldsymbol{x}_i, c_i)\}_{i=1}^{m}$ y otra de validación $T_{m'} = \{(\boldsymbol{x}_i, c_i)\}_{i=1}^{m'}$.

Nuestro objetivo es predecir la clase c de futuros vectores de propiedades $\boldsymbol{x}$ procedentes de esa misma distribución.

Función de decisión. Una *función de decisión* $f : \mathbb{R}^n \to \mathbb{R}$ es un procedimiento matemático/algorítmico que trata de distinguir esos dos tipos de objetos. Si $f(\boldsymbol{x}) > 0$ el objeto $\boldsymbol{x}$ se clasifica como 'positivo' ($c = +1$) y si $f(\boldsymbol{x}) < 0$ se clasifica como 'negativo' ($c = -1$). Los puntos que cumplen $f(\boldsymbol{x}) = 0$ constituyen la hipersuperficie frontera entre las dos regiones de decisión.



En la práctica necesitamos clasificadores multiclase. Pueden construirse fácilmente a partir de varias funciones de decisión *binarias* que, por ejemplo, se entrenan para distinguir cada clase del resto de ellas. En la etapa de clasificación se elige la que produce una respuesta más 'fuerte':



Máquina de aprendizaje. Dado cualquier problema de clasificación, definido por los ejemplos de entrenamiento y validación S_m y $T_{m'}$, es necesario especificar el tipo de función de decisión que vamos a usar. Debemos elegir una familia o colección $\mathcal{F}$ de funciones, con la esperanza de que alguna de ellas resuelva el problema (y que podamos encontrarla sin demasiado esfuerzo computacional).

Cada elemento $f_W \in \mathcal{F}$ se describe mediante un conjunto de parámetros ajustables que permiten situar la frontera en diferentes posiciones y formas. Por ejemplo, las fronteras verticales en $\mathbb{R}^2$ se especifican simplemente mediante un número que indica la posición en la que la frontera corta el eje x_1; las fronteras con forma de círculo quedan definidas por su centro y su radio, etc. Algunas clases de frontera se han hecho muy famosas: los hiperplanos, las redes neuronales artificiales, los árboles de decisión, etc.

Una vez elegido el tipo de frontera habrá que realizar algún tipo de ajuste u optimización para encontrar una 'buena' solución de ese tipo. Una máquina de aprendizaje es una familia de funciones $\mathcal{F}$ junto con un método para elegir una de ellas a partir de los ejemplos de entrenamiento S_m.

Principio de inducción. Pero ¿qué es exactamente una 'buena' frontera? Obviamente, cualquiera que tenga una probabilidad de error suficientemente pequeña para las necesidades de nuestra aplicación. La probabilidad de error es el promedio de fallos sobre los *infinitos* objetos $\boldsymbol{x}$ que se le pueden presentar a la máquina en su utilización y que, en general, no habrán sido vistos durante la etapa de diseño. Dada una frontera con parámetros W, su probabilidad de error es[1]:

$$E(W) = \int_{\mathbb{R}^n} [\![f_W(\boldsymbol{x}) \neq c]\!] dF(\boldsymbol{x}, c)$$

Por supuesto, el error real $E(W)$ es desconocido: ni conocemos el proceso natural $F(\boldsymbol{x}, c)$ ni podemos trabajar con las infinitas muestras que pueden surgir de él[2]. Sólo podemos trabajar con estimaciones empíricas, a partir de muestras. En concreto, si de alguna manera nos decidimos por una solución f_{W^*} que parece prometedora, su error real se puede estimar mediante el error observado sobre la muestra de validación T'_m:

$$E(W^*) \simeq \frac{1}{m'} \sum_{(\boldsymbol{x},c)\in T_m} [\![f_{W^*}(\boldsymbol{x}) \neq c]\!]$$

Si m' es grande y $T_{m'}$ no se ha utilizado en el diseño de la máquina (para elegir W^*), entonces la frecuencia de error sobre $T_{m'}$ convergerá a la probabilidad de error

[1] La notación $[\![cond]\!]$ es la función indicadora del conjunto de objetos que cumplen la condición *cond* (vale 1 si la condición es cierta y 0 si es falsa).

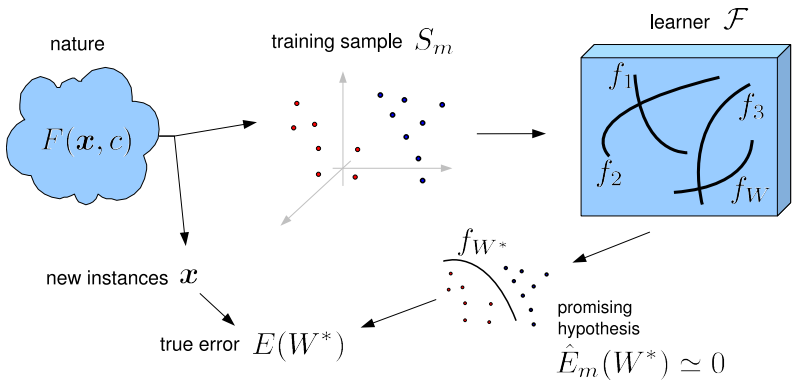
[2] Como vimos en el capítulo anterior, desde un punto de vista teórico la frontera de mínimo error es la que surge al aplicar el test de Bayes. Pero se apoya en las distribuciones de probabilidad $p(\boldsymbol{x}|c)$, cuya estimación es mucho más difícil que encontrar directamente una frontera aceptable.

al aumentar el tamaño de la muestra. Hasta aquí no hay ningún problema: una vez elegida la frontera W^* podemos comprobar su funcionamiento con los ejemplos de validación.

La verdadera dificultad es la elección de la 'mejor' frontera W^* dentro del conjunto $\mathcal{F}$. Imaginemos por un momento que mediante un supercomputador 'infinitamente potente' calculamos el error empírico sobre S_m de *todas* las fronteras que hay en $\mathcal{F}$. Para cada configuración W obtenemos una expresión similar a la del error de validación anterior, pero sobre la muestra de aprendizaje:

$$\hat{E}_m(W) = \frac{1}{m}\sum_{i=1}^{m}[\![f_W(\boldsymbol{x}_i) \neq c_i]\!]$$

En estas condiciones parece razonable seleccionar la frontera W^* que cometa menos errores sobre la muestra de entrenamiento. Porque si ni siquiera resuelve bien los ejemplos disponibles poco podemos esperar de nuestra máquina en casos futuros. Por supuesto, en la práctica no tenemos ese hipotético supercomputador para probar por fuerza bruta todos los elementos $f_W \in \mathcal{F}$, sino que usamos algoritmos de optimización que encuentran de forma más o menos inteligente soluciones 'prometedoras'.



Esta es la idea, aparentemente razonable, en la que se basó inicialmente el aprendizaje automático: la *minimización del error empírico* (o principio de inducción ingenuo): si una solución separa bien una muestra de entrenamiento suficientemente grande, seguramente separará bien la mayoría de las muestras no observadas. La investigación se orientó hacia tipos de máquina $\mathcal{F}$ cada vez más flexibles, capaces de conseguir de manera eficiente errores empíricos pequeños. Sin embargo, como veremos en la sección 4.7, el principio de inducción ingenuo no siempre es correcto.

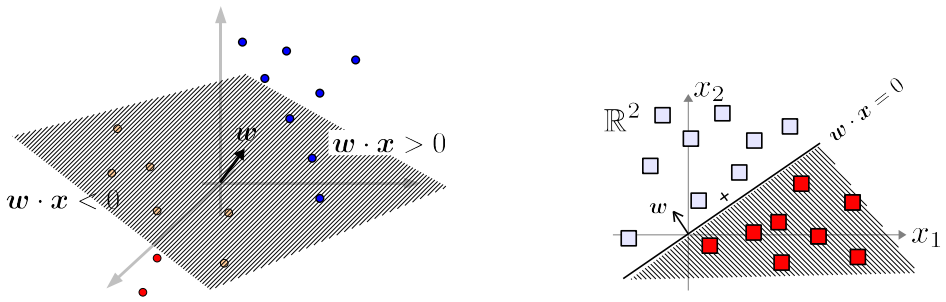
Pero antes de estudiar la teoría de la generalización revisaremos brevemente algunas máquinas de aprendizaje típicas.

4.2. La máquina lineal

En primer lugar estudiaremos una de las máquinas de clasificacion más simples: la máquina lineal.

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

Dado un conjunto de coeficientes o pesos, cada propiedad x_i se multiplica por su correspondiente coeficiente w_i y se suman todos los términos. Intuitivamente, la magnitud de cada peso indica la importancia relativa del atributo correspondiente, y su signo determina si contribuye a favor o en contra de la decisión positiva $c = +1$. Geométricamente (ver apéndice A.1) la superficie $f_{\boldsymbol{w}}(\boldsymbol{x}) = 0$ es un hiperplano perpendicular a $\boldsymbol{w}$ en el espacio de propiedades que separa los semiespacios donde $f_{\boldsymbol{w}}(\boldsymbol{x}) > 0$ y $f_{\boldsymbol{w}}(\boldsymbol{x}) < 0$.

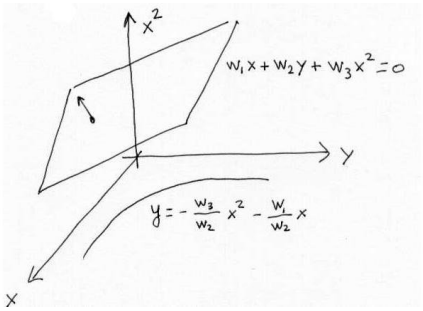


En lugar de usar un término independiente w_0 explícito, que es necesario para que la frontera pueda separarse del origen de coordenadas, a veces es preferible conseguir el mismo efecto añadiendo a todos los vectores de propiedades una constante (p. ej., 1), incluyendo así a w_0 dentro de $\boldsymbol{w}$:

$$\boldsymbol{w} \cdot \boldsymbol{x} + w_0 = (w_0, w_1, w_2, \ldots, w_n) \cdot (1, x_1, x_2, \ldots, x_n)$$

Las máquinas lineales (los parámetros ajustables w_i solo intervienen como factores multiplicativos de los atributos de los objetos) admiten algoritmos de ajuste muy sencillos. Por el contrario, las máquinas no lineales pueden requerir algoritmos de ajuste considerablemente más complicados.

Un truco para conseguir fronteras "curvadas", más flexibles que los hiperplanos, consiste en añadir ciertas propiedades nuevas, predeterminadas (p. ej., potencias y productos de los atributos originales), de manera que los parámetros ajustables w_k sólo afecten multiplicativamente a estas nuevas propiedades. Este tipo de *máquina lineal generalizada* sigue admitiendo un tratamiento esencialmente lineal, aunque las fronteras obtenidas, vistas desde el espacio original, sean no lineales. Sin embargo, tras la invención de las máquinas de *kernel* (sec. 4.7) este tipo de expansión explícita de propiedades carece de sentido.



Máquina lineal generalizada

Siguiendo el principio de inducción ingenuo, trataremos de encontrar la frontera que minimiza el error empírico $\hat{E}_m(\boldsymbol{w})$. ¿Cómo calculamos los coeficientes $\boldsymbol{w}$ de la máquina para conseguir el *menor número de fallos* sobre los ejemplos de entrenamiento? Planteado así, el problema es difícil a pesar de la sencillez de este tipo de máquina. Debe cumplirse que $\text{sign}(\boldsymbol{w} \cdot \boldsymbol{x}_i) = c_i$, o, de forma equivalente, se trata de satisfacer el mayor número posible de restricciones lineales del tipo:

$$c_i \boldsymbol{x}_i \cdot \boldsymbol{w} > 0, \quad i = 1 \dots m$$

Esto se parece a un problema típico de investigación operativa. Si se pueden cumplir todas las desigualdades, un 'programa lineal' de tipo *simplex* encontrará sin ningún problema una solución, pero en caso contrario la situación se complica porque tenemos que atribuir costes a las violaciones de restricción, etc.

Desafortunadamente, el error empírico es una función *discontinua* (pequeños cambios en los elementos de $\boldsymbol{w}$ no cambia casi nunca el número de ejemplos mal clasificados), por lo que su minimización es esencialmente un problema combinatorio. Una alternativa mucho más viable es minimizar alguna función de coste que sea matemáticamente más manejable y que esté lo más relacionada posible con la tasa de error.

Por ejemplo, el *análisis discriminante*, uno de los primeros métodos empíricos de clasificación, utiliza como función objetivo el criterio de Fisher de separabilidad estadística, cuya solución óptima depende sólo de los vectores medios y las matrices de covarianza de cada clase. Está muy relacionado con las propiedades mas discriminantes que estudiamos en la sección 2.2.2.

4.3. Clasificación por Mínimos Cuadrados

Quizá la función de coste más sencilla de manejar matemáticamente sea el error cuadrático medio (MSE). En lugar de exigir que la frontera deje a un lado y a otro los vectores de propiedades de cada clase (lo que produce el conjunto de desigualdades anterior), podemos intentar que la función discriminante tome exactamente el valor $c \in \{-1, +1\}$ que indica su clase:

$$\boldsymbol{x}_i \cdot \boldsymbol{w} = c_i$$

Esta imposición es más fuerte de lo necesario, ya que no solo pide que el punto esté en el lado correcto del hiperplano frontera, sino también a una distancia fija. Pero si conseguimos cumplirla el problema quedaría resuelto. Las restricciones impuestas a $\boldsymbol{w}$ por todos los ejemplos de S_m pueden expresarse de manera compacta como un sistema de ecuaciones lineales:

$$\mathtt{X}\boldsymbol{w} = \boldsymbol{C}$$

donde $\mathtt{X}$ es una matriz $m \times n$ que contiene por filas los vectores de propiedades x_i y $\boldsymbol{C}$ es un vector columna que contiene las etiquetas correspondientes c_i.

Si tenemos un número m de ejemplos linealmente independientes (restricciones) menor o igual que el número n de incógnitas (la dimensión de los vectores de propiedades) el sistema anterior siempre tendrá solución, incluso si las etiquetas de clase son aleatorias, o 'absurdas'. Hay grados de libertad suficientes para permitir que el hiperplano separe esa cantidad de ejemplos situados arbitrariamente en el espacio de propiedades. No existe ninguna garantía de que la frontera obtenida se apoye en propiedades relevantes. Puede ser engañosa debido a un exceso de interpolación.

Si por el contrario el número de ejemplos es mayor que la dimensión del espacio, el sistema de ecuaciones anterior queda sobredeterminado. En principio no puede tener solución exacta. Sólo si la distribución de los ejemplos tiene una gran regularidad (las nubes de puntos no están demasiado entremezcladas), será posible verificar todas o la mayoría de las ecuaciones. Por tanto, si con $m \gg n$ se obtiene un hiperplano que separa muy bien los ejemplos, es razonable confiar en que hemos capturado correctamente la esencia de la frontera. Si las etiquetas fueran arbitrarias, sin ninguna relación con los vectores de propiedades, sería extraordinariamente improbable que pudiéramos satisfacer tantas ecuaciones con tan pocos grados de libertad.

Por tanto, nuestro objetivo es resolver un sistema sobredeterminado de ecuaciones lineales (véase el apéndice D.2.1). El procedimiento usual es minimizar una función de error respecto a los parámetros ajustables. La más sencilla es el error cuadrático medio:

$$E_{RMS}(\boldsymbol{w}) = \frac{1}{2}||\mathtt{X}\boldsymbol{w} - \boldsymbol{C}||^2$$

cuyo mínimo $\boldsymbol{w}^*$ se obtiene de forma inmediata como:

$$\boldsymbol{w}^* = \mathtt{X}^+\boldsymbol{C}$$

donde la matriz $\mathtt{X}^+ \equiv (\mathtt{X}^\mathsf{T}\mathtt{X})^{-1}\mathtt{X}^\mathsf{T}$ es la *pseudoinversa* de $\mathtt{X}$.

Una máquina lineal ajustada mediante este método puede resolver satisfactoriamente muchos problemas de clasificación. Y también sirve para aproximar funciones continuas, donde los valores deseados c_i pueden tomar cualquier valor: se trata del

método clásico de ajuste de funciones por 'mínimos cuadrados'. Como curiosidad, éste es uno de los algoritmos de aprendizaje más simples de programar: por ejemplo, en Matlab/Octave se reduce a algo tan simple como `w = pinv(x)*c`. Aunque, por supuesto, esta simplicidad es engañosa porque el cálculo de la pseudoinversa es un problema numérico bastante complejo.

→ Least squares classification como aproximación a las probabilidades a posteriori.

Ejercicios:

- Probar un clasificador lineal basado en la pseudoinversa sobre alguno de los problemas del capítulo 2.

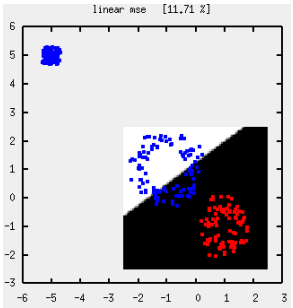
→ penalización del tamaño de los pesos: regularización. Ver el efecto en MNIST p.ej. con dos clases, y obtener la curva de Eapren - Etest

4.4. Análisis Bayesiano de la regresión lineal

→ pendiente

4.5. Máquinas lineales con saturación

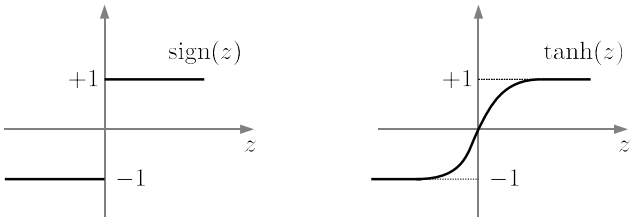
Desafortunadamente, la máquina anterior tiene un inconveniente. Al exigir el valor concreto c a la función discriminante $f_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$ (en lugar de la desigualdad que indica el lado deseado de la frontera), los puntos muy bien clasificados (alejados de la frontera por el 'lado bueno') también producen coste, equiparable al de los que se sitúan a la misma distancia en el lado equivocado. En distribuciones de ejemplos como la siguiente la frontera de mínimo coste cuadrático comete muchos errores, a pesar de que las clases son linealmente separables:



Para remediar esto, vamos a intentar cumplir mejor la condición que realmente deseamos: una decisión concreta $+1$ ó -1 dependiendo del lado de la frontera en que cae el ejemplo:

$$\text{sign}(\boldsymbol{w} \cdot \boldsymbol{x}_i) = c_i$$

El problema es que la discontinuidad de la función signo es matemáticamente inmanejable. Pero podemos aproximarla mediante alguna función que se comporte cualitativamente igual pero que sea 'suave' (continua y derivable). Una elección típica para aproximar el escalón de la función signo es la tangente hiperbólica:



Por tanto, intentaremos encontrar la frontera que minimice la siguiente función de coste:

$$E_S(\boldsymbol{w}) = \frac{1}{2} \sum_i |g(\boldsymbol{w} \cdot \boldsymbol{x}_i) - c_i|^2$$

donde $g(x) = \tanh(x)$. Al cambiar el salto abrupto de la función signo por su equivalente suave la función de coste global E_S también se vuelve suave: cambia gradualmente al modificar ligeramente los coeficientes $\boldsymbol{w}$. Aunque ya no existe una fórmula cerrada que nos proporcione la configuración óptima $\boldsymbol{w}^*$ a partir de S_m, podemos utilizar métodos de optimización iterativa muy sencillos, basados en 'descenso de gradiente' (ver apéndice D.3). Dada una solución $\boldsymbol{w}$, la corregimos mediante la sencilla regla:

$$\Delta \boldsymbol{w} = -\alpha \nabla E_S(\boldsymbol{w})$$

Cada componente de $\Delta \boldsymbol{w}$ es:

$$-\alpha \frac{\partial E_S}{\partial w_k} = \sum_i \underbrace{-\alpha[\overbrace{g(\boldsymbol{w} \cdot \boldsymbol{x}_i)}^{o_i} - c_i] g'(\boldsymbol{w} \cdot \boldsymbol{x}_i)}_{\delta_i} x_{i,k}$$

donde definimos $o_i = g(\boldsymbol{w} \cdot \boldsymbol{x}_i)$ como la salida de la máquina para el ejemplo $\boldsymbol{x}_i$, y δ_i es la contribución común de todas las componentes de $\boldsymbol{x}_i$:

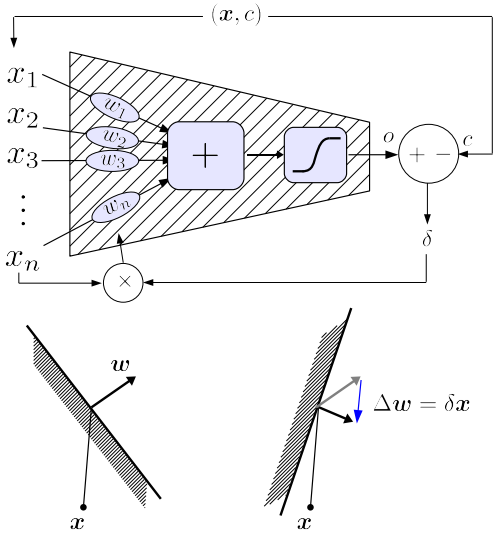
$$\delta_i = -\alpha(o_i - c_i)(1 - o_i^2)$$

Hemos aprovechado la coincidencia matemática de que la derivada de la tangente hiperbólica se puede expresar de forma más sencilla en términos de la propia función que de la variable: si $g(z) = \tanh z$ entonces $g'(z) = \operatorname{sech}^2 z = 1 - \tanh^2 z = 1 - g^2(z)$.

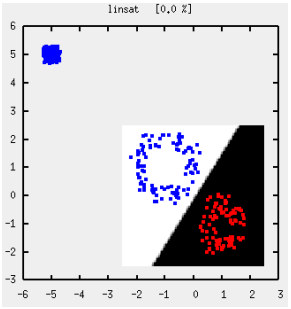
En resumen, la corrección global producida por todos los ejemplos de entrenamiento se puede expresar como una simple combinación lineal de todos ellos:

$$\Delta \boldsymbol{w} = \sum_i \delta_i \boldsymbol{x}_i$$

En la práctica se suele aplicar la corrección de cada ejemplo de forma independiente: cada vez que la máquina recibe un ejemplo etiquetado $\boldsymbol{x}$ hace una corrección de sus parámetros $\boldsymbol{w}$ proporcional a ese mismo ejemplo, con un peso δ que depende del error cometido (y que tiene en cuenta la saturación g). Este algoritmo tiene una gran elegancia y sencillez; puede considerarse como la máquina de aprendizaje (iterativa) más simple posible. Además, tiene una curiosa interpretación geométrica: cada ejemplo 'tira' de la frontera intentando quedar bien clasificado:



Este tipo de máquina lineal con saturación ha recibido diferentes nombres: 'neurona formal', *linear threshold unit*, etc. En la siguiente figura se muestra la frontera obtenida sobre el problema de clasificación anterior:

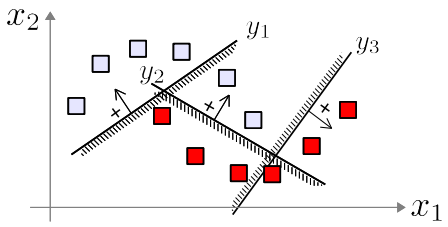


A pesar de la mejora que supone la saturación en la respuesta, lo cierto es que en muchos casos una frontera lineal es demasiado rígida. Muchas distribuciones requiren fronteras no lineales (hipersuperficies más o menos curvadas). En principio esto podría conseguirse automáticamente con las técnicas anteriores usando funciones lineales generalizadas, en las que las propiedades originales se amplían, p. ej., con potencias. Sin embargo una expansión explícita es intratable en espacios de dimensión elevada. En la sección siguiente comentaremos muy brevemente una alternativa mucho más atractiva.

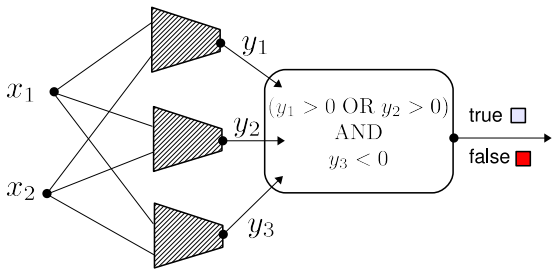
→ Interpretación de sigmoidal en función de verosimilitudes, logit, softmax.

4.6. Máquinas Neuronales

El elemento de proceso lineal con saturación que acabamos de explicar tiene un verdadero interés cuando se utiliza como elemento constructivo de máquinas más potentes. Observa la siguiente distribución no linealmente separable de ejemplos en $\mathbb{R}^2$:



Hemos dibujado encima 3 fronteras lineales que, aunque por separado no pueden resolver el problema, "particionan" el espacio de entrada en varias regiones que adecuadamente combinadas podrían distinguir perfectamente las dos clases. Por ejemplo, podríamos conectar esos 3 discriminantes lineales a una regla de decisión muy sencilla como la siguiente:

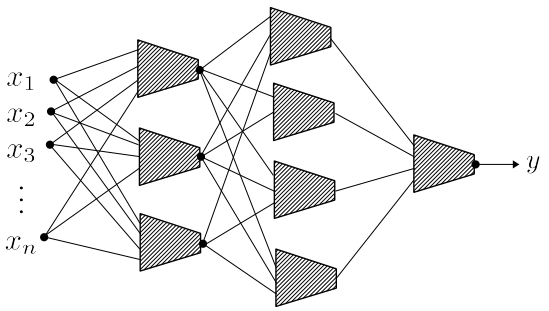


Mediante un procedimiento análogo se podrían fabricar clasificadores con fronteras no lineales de cualquier complejidad. En este ejemplo hemos situado "a mano" los discriminantes elementales y hemos escrito una lógica de decisión apropiada para ellos. Hemos podido hacerlo porque la distribución tiene solo 2 atributos y visualmente podemos captar inmediatamente su estructura. Pero, por supuesto, en los problemas reales nos enfrentamos a distribuciones multidimensionales cuya estructura es imposible representar gráficamente y, sobre todo, estamos interesados en máquinas de clasificación cuyas fronteras se ajustan automáticamente mediante aprendizaje.

¿Es posible diseñar un algoritmo que sitúe de forma razonable las fronteras elementales auxiliares y a la vez construya la lógica necesaria para combinarlas? La respuesta es afirmativa y el algoritmo que lo consigue es muy simple e interesante.

4.6.1. El *perceptrón* multicapa

Un primer paso es sustituir la lógica de decisión que combina las fronteras elementales por un conjunto adicional de unidades lineales, creando una estructura en red llamada *perceptrón multicapa*:



El esquema anterior muestra una red con dos capas intermedias ('ocultas') y una tercera capa 'de salida', que en este caso contiene una sola unidad, suficiente para problemas de clasificación binaria. En los problemas multiclase la capa de salida tiene más elementos, p. ej., uno por clase, aunque también se puede utilizar cualquier otro sistema de codificación. Podemos imaginar redes con cualquier número de capas intermedias, aunque en muchos casos se utiliza solo una.

Se puede demostrar que este tipo de red, con un número adecuado de nodos y capas puede aproximar con la precisión deseada cualquier función razonable.

> Las puertas lógicas OR, AND y NOT, suficientes en último término para combinar las fronteras lineales elementales, puede implementarse sin ningún problema mediante máquinas lineales saturadas eligiendo adecuadamente sus coeficientes. Este argumento también demuestra que las máquinas neuronales generales (grafos de procesamiento de conectividad arbitraria que pueden contener ciclos) son Turing-completas. Sin embargo, en la práctica no suele ser posible encontrar un significado concreto a las operaciones realizadas por los elementos de una red. De hecho, la característica esencial de este tipo de máquinas es que realizan 'procesamiento paralelo distribuído', donde la tarea se reparte de forma más o menos redundante entre todas las unidades.

4.6.2. El algoritmo *backprop*

Una de las ideas revolucionarias de la computación neuronal es el ajuste global de los pesos de todos los nodos de la red directamente mediante descenso de gradiente. A primera vista esto parece una locura, ya que no parece fácil que una secuencia de correcciones locales sea capaz de resolver el problema 'huevo-gallina' que se plantea: las unidades encargadas de la combinación solo pueden ajustarse cuando las discriminaciones elementales estén bastante bien situadas, pero, a la vez, las discriminaciones elementales deberán situarse teniendo en cuenta una lógica que pueda combinarlas. Sorprendentemente, una sencilla optimización local puede conseguir casi siempre soluciones correctas si la arquitectura de la red es suficientemente rica.

> En ocasiones puede ocurrir que para llegar a una buena solución hay que atravesar regiones más costosas en el espacio de parámetros, por lo que la optimización puede atascarse en mínimos locales. Pero esto ocurre con mayor frecuencia en problemas artificiales, elegidos típicamente para poner a prueba las máquinas de aprendizaje, que en problemas naturales con un número relativamente alto de atributos. En estos casos el elevado número de parámetros, y su simetría, dan lugar a muchas soluciones alternativas, alguna de las cuales caerá probablemente cerca de cualquier posición inicial de la optimización.

Una red multicapa implementa la composición de varias transformaciones vectoriales $\mathbb{R}^p \rightarrow \mathbb{R}^q$. Dado un vector de entrada $\boldsymbol{x}$ y otro de salida deseada $\boldsymbol{d}$ (que normalmente codifica la clase de $\boldsymbol{x}$), una red de, p. ej. 3 capas, produce un resultado

$$\boldsymbol{r} = \boldsymbol{y}_3(\boldsymbol{y}_2(\boldsymbol{y}_1(\boldsymbol{x})))$$

donde cada etapa tiene la estructura:

$$\boldsymbol{y}_c(\boldsymbol{z}) = g(\mathtt{W}_c\boldsymbol{z})$$

$\mathtt{W}_c$ es una matriz que tiene tantas filas como el número de elementos en la capa c, y tantas columnas como entradas tiene esa capa (que es el número de elementos de la capa anterior, o la dimensión de $\boldsymbol{x}$ si es la primera capa). Las filas de esa matriz contienen los coeficientes $\boldsymbol{w}$ de cada uno de los elementos de esa capa. La función g calcula la tangente hiperbólica de todos sus argumentos para conseguir la necesaria saturación suave (sin ella la transformación global sería un producto de matrices que se reduciría a una única capa efectiva, quedando una máquina lineal). Así pues, una red neuronal de c capas queda definida mediante las c matrices de conexión correspondientes a cada capa, que denotaremos por $\mathbb{W}$.

> Un detalle técnico importante es la implementación de los términos independientes necesarios en cada elemento de proceso (el umbral de saturación) para permitir fronteras que no pasen obligatoriamente por el origen de coordenadas. Esto puede conseguirse añadiendo una componente constante a todos los vectores de entrada, como vimos en la sección 4.2. En principio, parecería necesario añadir también componentes constantes en las capas intermedias, para que todos los elementos de la red puedan adaptar su umbral de saturación. Sin embargo, basta con añadir una única componente constante al vector de atributos. Si es realmente necesario la red puede ajustar los pesos para transmitir la constante de la entrada hacia las capas posteriores.

La función de coste es el error cuadrático acumulado sobre todos los ejemplos de entrenamiento. La contribución del ejemplo $(\boldsymbol{x}, \boldsymbol{d})$ es:

$$E(\mathbb{W}) = \frac{1}{2}||\boldsymbol{y}_3(\boldsymbol{y}_2(\boldsymbol{y}_1(\boldsymbol{x}))) - \boldsymbol{d}||^2$$

y la corrección correspondiente será:

$$\Delta\mathbb{W} = -\alpha\nabla E(\mathbb{W})$$

que es un conjunto de matrices con las misma estructura que $\mathbb{W}$ que hay que sumar a la red para mejorar un poco su comportamiento sobre este ejemplo.

Pero ¿cómo calculamos el gradiente $\nabla E(\mathbb{W})$ de una función tan compleja como la realizada por este tipo de red? El cómputo de cada componente del gradiente:

$$\frac{\partial E}{\partial w_{k,i,j}}$$

donde $w_{k,i,j}$ es el peso de conexión entre el elemento i de la capa k con el elemento j de la capa $k-1$, es muy ineficiente si se hace de manera ingenua.

Afortunadamente, en los años 80 se inventó un método particularmente elegante y eficiente para calcular el gradiente, llamado *backprop* (retropropagación del error). El algoritmo explota el hecho de que muchas de esas derivadas parciales comparten cálculos, debido a la composición de transformaciones realizada por la red. En primer lugar vamos a expresar la dependencia del error respecto a cada peso $w_{k,i,j}$ como la

combinación de la influencia de ese peso en la salida de su nodo, con la influencia de ese nodo en el error global:

$$\frac{\partial E}{\partial w_{k,i,j}} = \frac{\partial E}{\partial y_{k,i}} \frac{\partial y_{k,i}}{\partial w_{k,i,j}}$$

El primer factor, que definimos $\delta_{k,i}$ y calcularemos más adelante, mide la influencia en el error del elemento i de la capa k:

$$\frac{\partial E}{\partial y_{k,i}} \equiv \delta_{k,i}$$

El segundo factor mide la influencia de un peso en su nodo, y puede calcularse directamente, ya que no es más que la entrada de ese peso ajustada con la derivada de la función de saturación (un resultado análogo al obtenido en el apartado 4.5):

$$\frac{\partial y_{k,i}}{\partial w_{k,i,j}} = y_{k-1,j}(1 - y_{k-1,j}^2)$$

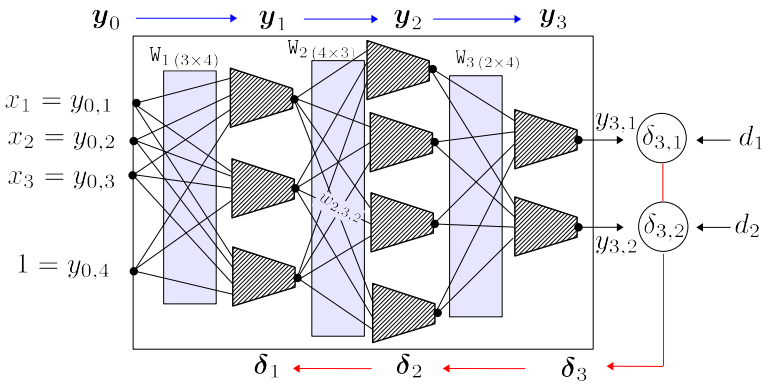
Dada una asociación entrada/salida deseada $(\boldsymbol{x}, \boldsymbol{d})$, cada elemento de proceso tendrá una salida $y_{k,i}$ y también una 'discrepancia' $\delta_{k,i}$. Si es positiva, significa que ese elemento debería 'cambiar su estado' para que la red funcionara mejor, y si es negativa, debería 'reforzar' su respuesta. A partir de los valores $\delta_{k,i}$, la corrección de pesos es inmediata con la expresión anterior. Denotemos mediante el vector $\boldsymbol{\delta}_k$ a todos los $\delta_{k,i}$ de la capa k. El algoritmo *backprop* calcula los $\boldsymbol{\delta}_k$ de forma análoga al funcionamiento normal de la red pero en dirección contraria: una vez calculadas las salidas $\boldsymbol{y}_k$ para el ejemplo $\boldsymbol{x}$ (funcionamiento 'hacia delante'), con la salida deseada se calculan los $\boldsymbol{\delta}_c$ de la última capa (con una expresión idéntica a la que vimos en la sección 4.5 para una única unidad lineal saturada):

$$\boldsymbol{\delta}_c = (\boldsymbol{y}_c - \boldsymbol{d}) \otimes (1 - \boldsymbol{y}_c^2)$$

(el símbolo $\otimes$ significa multiplicación elemento a elemento). A partir de ellos se calculan sucesivamente ('hacia atrás'), los deltas de las capas $c-1$, $c-2$,..., 1, con una expresión 'dual' a la de la activación de la red:

$$\boldsymbol{\delta}_{k-1} = (1 - \boldsymbol{y}_{k-1}^2) \otimes \mathtt{W}_k^\mathsf{T} \boldsymbol{\delta}_k$$

Esta expresíon surge de forma natural al expresar la influencia de un cierto delta sobre el error global $E(\mathbb{W})$ en términos de su influencia sobre toda la capa siguiente. A continuación se muestra un esquema de los flujos de información del algoritmo *backprop*:



El algoritmo *backprop* se basa en una versión general de la regla de la cadena. Recordemos que la derivada respecto a x de una función $f(x) = g(h_1(x), h_2(x), \ldots)$ que se puede expresar como una combinación g de varias funciones más simples h_k se puede expresar como:

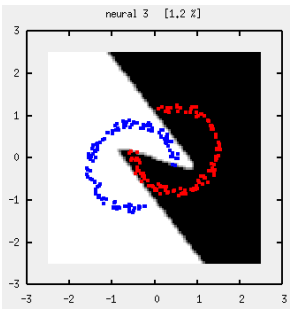
$$\frac{df}{dx} = \frac{dg(h_1(x), h_2(x), \ldots)}{dx} = \sum_k \frac{\partial g}{\partial h_k} \frac{dh_k}{dx}$$

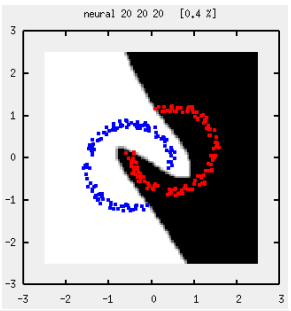
P. ej., la función $f(x) = \sin(e^x \cos x)$ puede expresarse de esa manera con $g(h_1, h_2) = \sin(h_1 h_2)$, $h_1(x) = e^x$ y $h_2(x) = cos(x)$. Su derivada coincide con la que obtendríamos aplicando la regla de la cadena ordinaria:

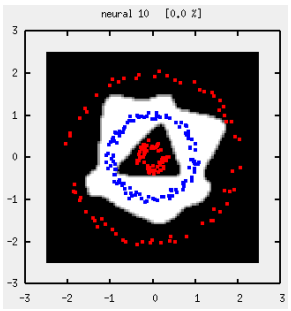
$$\frac{df}{dx} = \cos(h_1 h_2) h_2 e^x + \cos(h_1 h_2) h_1 (-\sin x) = \cos(e^x \cos(x))(e^x \cos(x) - e^x \sin(x))$$

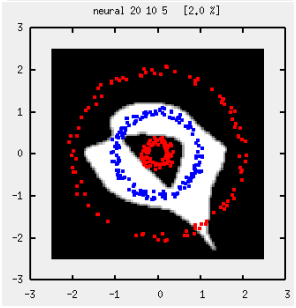
4.6.3. Ejemplos

Para hacernos una idea de lo que podemos esperar del algoritmo *backprop* observa la solución obtenida en dos ejemplos de juguete sobre $\mathbb{R}^2$ con diferentes estructuras de red (el número de elementos en las capas intermedias se indica en el título de cada gráfico). En todos los casos el tiempo de aprendizaje es de unos pocos segundos.









Finalmente veamos una aplicación más realista. A continuación se muestra la matriz de confusión al clasificar los dígitos manuscritos de la base de datos MNIST ($28 \times 28 = 784$ atributos) mediante una red con dos capas ocultas de 30 y 20 elementos y una capa de salida con 10 elementos que codifica posicionalmente las clases:

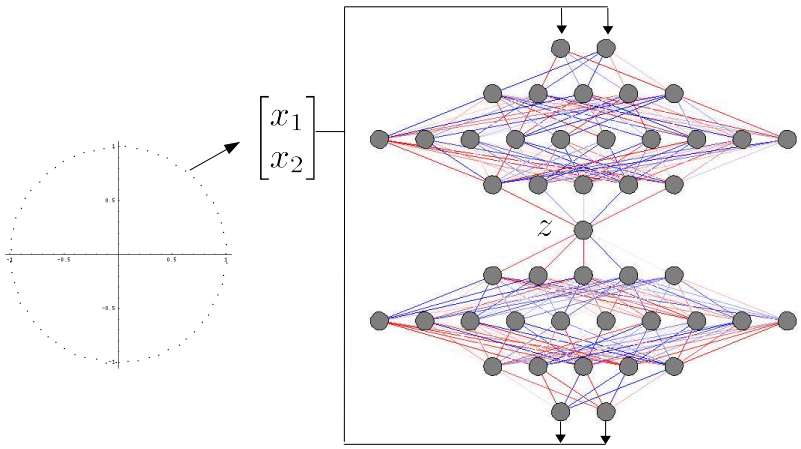
```
 97   0   0   0   0   0   1   0   0   2
  0 114   3   0   0   1   1   0   0   0
  1   2  79   1   0   0   4   0   4   0
  0   2   1  89   0   3   0   4   1   1
  0   1   0   0 100   0   0   0   0   4
  4   0   0   4   1  70   3   0   1   1
  2   0   3   0   0   3  85   0   1   2
  1   0   0   0   2   0   0 109   0   5
  2   0   5   3   6   1   0   1  69   2
  0   0   0   1   3   0   1   2   0  91
```

El error sobre los 1000 ejemplos de evaluación es del 9,7 %. Se ha producido sobreaprendizaje, ya el error sobre los 4000 ejemplos de entrenamiento es solo de 1,15 %. El tiempo de aprendizaje fue de una ó dos horas. Se realizaron 50 pasadas por los 4000 ejemplos en orden aleatorio. El único procesamiento sobre las imágenes originales fue el escalado de las entradas, de (0,255) a (-1.0,1.0) para adaptarlas al intervalo no saturado de los elementos de proceso.

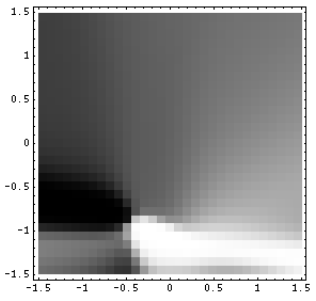
4.6.4. Extracción de propiedades no lineales

Acabamos de ver que una red multicapa puede aproximar cualquier transformación continua $\mathbb{R}^p \rightarrow \mathbb{R}^q$ entre espacios vectoriales. Es interesante considerar la transformación identidad, $\mathbb{R}^n \rightarrow \mathbb{R}^n$, que transforma cada vector $\boldsymbol{x}$ en sí mismo (la entrada $\boldsymbol{x}$ y la salida deseada son el mismo vector, $\boldsymbol{d} = \boldsymbol{x}$). Evidentemente, esta transformación aparentemente trivial puede conseguirse con una red sin elementos intermedios,

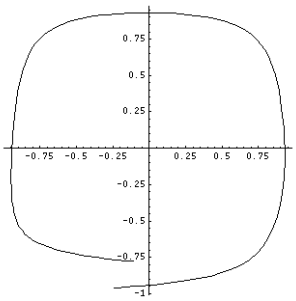
donde cada elemento de salida se conecta con su correspondiente entrada con un peso=1 y los demás pesos son cero. Sin embargo, consideremos la siguiente estructura de red con forma de cuello de botella, que vamos a entrenar para que reproduzca vectores de entrada de dos componentes que se distribuyen uniformemente en un círculo:



Observa que la información tiene que atravesar una etapa (etiquetada como z) con menos dimensiones que el espacio de entrada, para luego expandirse otra vez al espacio original. Si es posible realizar el ajuste entonces las dos mitades de la red implementan una pareja de funciones $f : \mathbb{R}^2 \to \mathbb{R}^1$ (codificación) y $g : \mathbb{R}^1 \to \mathbb{R}^2$ (descodificación), tales que $z = f(\boldsymbol{x})$ y $\boldsymbol{x} = g(z)$ para los puntos de esa distribución. El algoritmo *backprop* resuelve este ejemplo concreto en pocos segundos (la distribución es muy simple y en realidad podría resolverse con una red más pequeña). Si representamos la curva $\boldsymbol{x} = g(z)$ para valores de $z \in (-1, 1)$ (el rango de función de saturación), vemos la parametrización aproximada del círculo que ha sido 'aprendida' por la red:



$z = f(\boldsymbol{x})$



$\boldsymbol{x} = g(z)$

4.6.5. Comentarios

Desde el principio de la informática hubo dos enfoques complementarios en el estudio de la inteligencia artificial. Uno de ellos se orientó hacia la resolución lógica de problemas abstractos y otro hacia el modelado del *hardware* de procesamiento de información de los seres vivos. Éstos poseen un sistema nervioso basado en un número inmensamente grande de elementos de proceso ($\simeq 10^{11}$) relativamente lentos ($\simeq 1\text{ms}$), muy densamente interconectados ($\simeq 10^4$). No se almacena explícitamente el conocimiento, sino que está distribuido sobre las 'conexiones' entre los elementos de proceso. No hace falta una programación explícita de las tareas, sino que se produce algún tipo de aprendizaje y autoorganización más o menos automático. El procesamiento suele ser local, asíncrono y masivamente paralelo. Y, lo que es más importante, trabaja con datos masivos, ruidosos, sin una segmentación explícita, y con una sorprendente tolerancia a fallos (diariamente mueren miles de neuronas). Aparentemente, la estrategia de procesamiento se basa en la exploración simultánea de hipótesis alternativas con múltiples restricciones, de modo que solo sobreviven las que son consistentes con la información sensorial.

Este tipo de procesamiento paralelo distribuido contrasta de forma muy marcada con el tipo computación disponible en la actualidad, caracterizada por un número muy pequeño de procesadores extraordinariamente rápidos ($\simeq 1\text{ns}$) pero que ejecutan tareas secuenciales sobre información exacta, sin ninguna tolerancia a fallos. Las tareas deben ser programadas explícitamente, y un error en un solo bit hace fracasar una computación de complejidad arbitrariamente grande. Claramente, este estilo de programación no es el más adecuado para resolver problemas de percepción.

La computación neuronal [11, 1, 14] tuvo un desarrollo muy importante a finales del siglo XX, cuando se inventó el algoritmo *backprop* de entrenamiento para las redes multicapa. (Mucho antes ya se habían utilizado máquinas lineales para resolver problemas de reconocimiento sencillos, pero sus limitaciones para resolver problemas más complejos, no linealmente separables, produjeron una temporal falta de interés en la comunidad investigadora.) Además de las redes multicapa existen muchos otros tipos de máquinas neuronales orientadas a diferentes tareas. Por ejemplo, hay memorias asociativas, capaces de almacenar un conjunto de vectores y recuperar el más parecido a partir de una configuración ruidosa o incompleta; redes recurrentes, capaces de procesar la información teniendo en cuenta un estado, de forma similar a los autómatas, etc. En el campo de la robótica se han aplicado a la coordinación sensor-motor y para la percepción de formas visuales se han inventado redes especiales que obtienen resultados espectaculares en bancos de prueba como MNIST.

Durante un tiempo todo se intentaba resolver mediante algún tipo de máquina neuronal.

Ejercicios:

- Implementa el algoritmo *backprop*. Comprueba que funciona con distribuciones 2D de juguete y luego intenta aplicarlo a un problema real.

4.7. Máquinas de Vectores de Soporte

Don't try to solve a problem by solving
a harder one as an intermediate step.

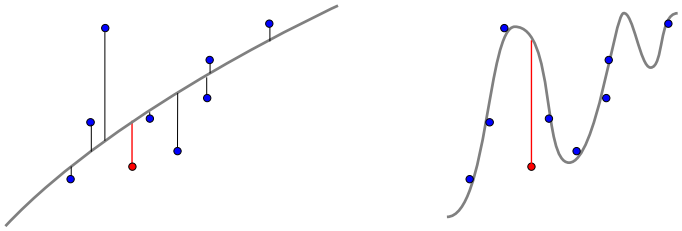
–Vapnik

El gran entusiasmo suscitado por la computación neuronal estaba plenamente justificado, ya que permite resolver de forma muy sencilla problemas de clasificación complejos, de naturaleza no lineal.

Sin embargo, cada vez se hizo más evidente que las máquinas de gran 'poder expresivo' (árboles de clasificación, cierto tipo de redes, etc.) a veces fracasan en su capacidad de *generalización*: casi siempre son capaces de resolver los ejemplos de entrenamiento, pero a veces lo hacen basándose en casualidades irrelevantes y con ejemplos nuevos funcionan mal.

Para intentar paliar este problema se propusieron técnicas heurísticas (P. ej.: detención temprana del aprendizaje, asignación de peso a la magnitud los coeficientes, etc.), pero no siempre tenían éxito. Finalmente, se observó que a veces las fronteras sencillas (p. ej., lineales) producen mejores resultados que otros métodos con mayor poder expresivo, capaces de sintetizar regiones mucho más complicadas en el espacio de propiedades. Un modelo más general o 'flexible' puede adaptarse mejor a cualquier tipo de distribución, pero al mismo tiempo corre el riesgo de producir una excesiva interpolación o 'sobreajuste' (*overfitting*).

Esta situación es análoga al problema del ajuste de un polinomio a un conjunto de medidas, ilustrado en la figura siguiente:



Si los datos son ruidosos es mejor utilizar un polinomio de orden bajo (izquierda), aunque no pase exactamente encima de todos ellos. Si elegimos un orden mayor podemos modelar perfectamente los puntos observados (derecha), pero se cometerá un mayor error sobre datos futuros (en rojo).

En este apartado comentaremos los resultados más importantes de la teoría de la generalización desarrollada por V. Vapnik [27, 28], quien analizó desde un punto de vista matemáticamente riguroso el problema del aprendizaje estadístico y encontró las condiciones que debe cumplir un algoritmo de aprendizaje para que sus resultados sean fiables. Como veremos, la clave se encuentra en la *capacidad* de la máquina.

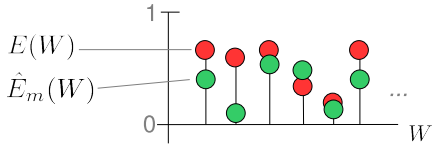
4.7.1. Consistencia

La cuestión fundamental es la siguiente: Dado un cierto tipo $\mathcal{F}$ de función de decisión con parámetros ajustables f_W, ¿tiene algo que ver el error observado $\hat{E}_m(W)$ con el error real $E(W)$?

Si la respuesta es negativa debemos desechar ese tipo de máquina: no es fiable, no 'generaliza'. Su comportamiento en casos concretos (los utilizados para el diseño) 'extrapola' incorrectamente los casos no observados. Peor aún, es engañosa: puede ocurrir que $\hat{E}_m(W)$ sea muy pequeño pero $E(W)$ sea muy grande, por mucho que aumente el número de ejemplos m.

En cambio, si tenemos una familia $\mathcal{F}$ para la que el error empírico de cualquier frontera W ofrece una estimación más o menos precisa de su error real, entonces sí podemos fiarnos de la máquina: la apariencia se corresponderá aproximadamente con la realidad. Y en este caso, si además alguna frontera W^* se comporta muy bien sobre los ejemplos de entrenamiento, podemos admitirla y tener confianza en que su funcionamiento con futuros ejemplos desconocidos será aproximadamente igual de bueno. Nuestro objetivo es encontrar la configuración de parámetros W que define una frontera de decisión que separe lo mejor posible las muestras disponibles, pero utilizando un modelo de máquina (familia de fronteras) $\mathcal{F}$ que sea de tipo 'fiable'.

Imaginemos que con nuestro supercomputador hipotético calculamos el error empírico $\hat{E}_m(W)$ de todas las fronteras f_W de una clase $\mathcal{F}$ sobre la muestra de entrenamiento. Cada una de ellos es una estimación frecuencial de tamaño m de las verdaderas probabilidades $E(W)$.



Por la ley de los grandes números, todas las frecuencias convergen individualmente a sus probabilidades:

$$\hat{E}_m(W) \to E(W)$$

Incluso podemos calcular para cada una un margen de confianza (ver C.1):

$$\mathrm{P}\{|\hat{E}_m(W) - E(W)| > \epsilon\} = \delta < \frac{1}{4m\epsilon^2}$$

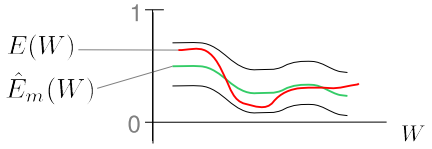
Ahora bien, aunque cada estimación converge a su valor real con probabilidad superior a δ, a medida que aumenta el número de elementos en la familia $\mathcal{F}$ se va haciendo cada vez más probable que *alguna* de ellas no cumpla la condición anterior. Por ejemplo, si $\mathcal{F}$ contiene un número finito t de elementos (hipótesis), la probabilidad de que todas ellas cumplan la condición anterior (suponiendo el caso peor, en que son 'independientes') es δ^t, que tiende muy rápidamente a cero en función de t.

> Por ejemplo, con 250.000 muestras y $\epsilon = 0.01$ tenemos $\delta >= 99\,\%$. Si nuestra máquina tiene sólo $t = 100$ hipótesis W independientes, entonces la probabilidad de que todas ellas verifiquen $|\hat{E}_m(W) - E(W)| > \epsilon$ es $0.99^{100} = 0.37$. La confianza se reduce mucho, hay una probablidad que puede llegar a ser del $63\,\%$ de que algún error empírico difiera del error real en una cantidad mayor que ϵ.

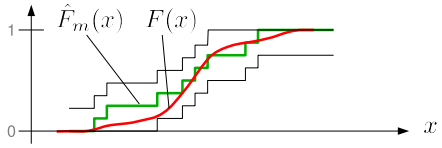
En cualquier caso, si $\mathcal{F}$ es finita, en principio se puede calcular un número de ejemplos (absurdamente grande) que garantice una probabilidad razonable de que el error aparente de todas las hipótesis se parece a su error real. Al menos en teoría, las máquinas finitas son consistentes y pueden generalizar.

Sin embargo, en la práctica se utilizan máquinas de aprendizaje que contienen *infinitas* hipótesis, para las cuales el análisis anterior nos dice que *seguro* aparecerá alguna hipótesis no fiable. Claramente, es necesario averiguar cuántas hipótesis de $\mathcal{F}$ son realmente 'independientes': muchas fronteras son muy parecidas, así que no tiene mucho sentido tenerlas en cuenta con el mismo peso en la estimación anterior.

Necesitamos determinar bajo qué condiciones se produce una *convergencia uniforme* de las frecuencias a sus probabilidades en una colección infinita de sucesos (la clase de fronteras $\mathcal{F}$).



> Un ejemplo de conjunto infinito de sucesos donde se produce este tipo de convergencia uniforme es la distribución empírica, que converge uniformemente a la distribución real. Es la base del conocido test no paramétrico de Kolmogorov-Smirnov.



4.7.2. Capacidad

Una forma muy conveniente de cuantificar el poder expresivo de una cierta clase de hipótesis $\mathcal{F}$ es su *dimensión de Vapnik-Chervonenkis*, $d_{VC}(\mathcal{F})$, que se define como el máximo número de vectores en posición general que pueden ser correctamente separados por elementos de $\mathcal{F}$ con cualquier asignación de etiquetas. Puede interpretarse como su capacidad puramente memorística.

→ ejemplos de dVC de varias clases

> La capacidad (d_{VC}) de las fronteras lineales (hiperplanos) es esencialmente n, la dimensión del espacio de propiedades: Hay n incógnitas o grados de libertad, que siempre pueden satisfacer $m \leq n$ restricciones de igualdad. Sin embargo, solo deberíamos exigir desigualdades indicando el lado del hiperplano donde debe estar cada ejemplo. Un estudio detallado de la probabilidad de que una configuración de m ejemplos con etiquetas arbitrarias sea linealmente separable en $\mathbb{R}^n$ muestra que cuando n es grande es casi seguro que $2m$ ejemplos arbitrarios serán linealmente separables. La capacidad de una máquina lineal es en realidad *dos veces* la dimensión del espacio. De nuevo observamos que para generalizar necesitamos $m >> n$.

El resultado fundamental de la teoría de Vapnik es que la convergencia uniforme, necesaria para que el aprendizaje sea consistente, solo require que $d_{VC}(\mathcal{F}) < \infty$. La máquina debe tener capacidad limitada. Si es capaz de almacenar cualquier número de ejemplos, con etiquetas arbitrarias, entonces no podemos confiar en que generalice.

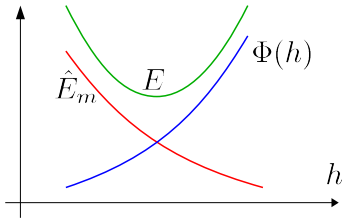
Incluso se ha obtenido la velocidad de convergencia de los errores empíricos a los errores reales. Para cualquier colección de hipótesis con capacidad $h = d_{VC}(\mathcal{F})$ y m ejemplos tomados de *cualquier* distribución $F(\boldsymbol{x}, c)$, entonces con confianza δ podemos acotar simultáneamente el error real de todas las hipótesis W de $\mathcal{F}$ mediante una expresión del tipo:

$$E(W) < \hat{E}_m(W) + \Phi(h, m, \delta)$$

donde

$$\Phi(h, m, \delta) = \sqrt{\frac{h \ln\left(\frac{2m}{h} + 1\right) + \ln\left(\frac{4}{\delta}\right)}{m}}$$

$\Phi(h, m, \delta)$ es la máxima diferencia entre el error real y el aparente que podemos esperar (con ese nivel de confianza). Como es válida para cualquier clase de hipótesis (con capacidad h) y cualquier problema de clasificación, es comprensible que sea una cota 'conservadora', no suele usarse como tal en la prática. Lo importante es su comportamiento cualitativo: a medida que aumenta la capacidad relativa h/m de la máquina el error empírico puede hacerse tan pequeño como queramos. Pero a la vez, la cota del error aumenta:



Se hace necesario algún tipo de *control de capacidad*. En el caso frecuente de colecciones de hipótesis que podemos ordenar de menor a mayor capacidad, en lugar de minimizar el error empírico $\hat{E}_m$, debemos minimizar la cota del error real $\hat{E}_m + \Phi(h/m)$. Este es el principio de inducción mejorado, conocido como *structural risk minimization*.

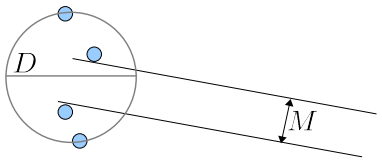
La idea sería obtener la solución W_h^* óptima para cada subfamilia $\mathcal{F}_h \subset \mathcal{F}$ de capacidad h creciente, y quedarnos con la que nos garantiza un menor error real de acuerdo con la cota anterior. Sin embargo, aunque esta idea es correcta desde un punto de vista teórico, no resulta muy práctica. Necesitamos una forma más directa de controlar la capacidad.

4.7.3. Margen

Durante un tiempo se intentó conseguir máquinas de aprendizaje con un poder expresivo cada vez mayor. Pero esto ya no parece tan deseable, una vez que comprendemos que debemos controlar su capacidad. ¿Existen máquinas de aprendizaje no lineales y a la vez de baja capacidad? A primera vista parece que estamos pidiendo demasiado...

Curiosamente, incluso un simple hiperplano puede ser excesivamente flexible si el espacio de propiedades es de dimensión elevada. P. ej., , si nuestros vectores de propiedades son directamente los píxeles de una imagen, el número de ejemplos disponible será casi siempre insuficiente para sobredeterminar una frontera. Por tanto, incluso en el caso más simple, lineal, necesitamos algún método de control de capacidad.

Esto puede conseguirse mediante la exigencia de un *margen* de separación. Vamos a imaginar que nuestra máquina lineal contiene fronteras hiperplano de diferentes grosores. Cuanto más ancha sea la frontera menos capacidad tendrá: debido a su anchura no puede separar tan fácilmente unos vectores de otros.



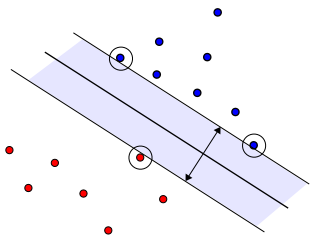
La capacidad de una máquina lineal con margen M es:

$$h \leq \left\lceil \frac{D^2}{M^2} \right\rceil + 1$$

donde D es la mayor distancia entre vectores del conjunto de entrenamiento. Este resultado es muy importante: la capacidad ¡no depende de la dimensión del espacio de propiedades! El hecho de exigir un gran margen de separación consigue limitar la capacidad de la máquina lineal, independientemente del número de atributos de los ejemplos, que incluso podrían llegar a ser potencialmente infinitos.

4.7.4. Hiperplano de máxima separación

Para aplicar correctamente esta idea, deberíamos elegir un margen suficientemente grande y encontrar el hiperplano que, con ese margen, separa correctamente el mayor número de ejemplos. Pero en la práctica resulta mucho más sencillo buscar el hiperplano con mayor margen que separa todos los ejemplos. Podemos pensar en que se encuentra una solución y luego se va ensanchando poco a poco; cada vez que el hiperplano choca con algún ejemplo sigue creciendo apoyado en él, hasta que no puede ensancharse más:



El hiperplano de máxima separación se consigue resolviendo un problema típico de la investigación operativa, la programación cuadrática (convexa) con restricciónes lineales (sec. D.8). La solución óptima se puede expresar en función de los vectores en los que finalmente se apoyó la frontera. Son los *vectores de soporte*.

> Deseamos que una máquina lineal del tipo $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$ separe todos los ejemplos $(\boldsymbol{x}_i, c_i)$, lo que se puede expresar como un conjunto de desigualdades $c_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$. El margen de separación M de una solución $(\boldsymbol{w}, b)$ es la distancia entre las perpendiculares a $\boldsymbol{w}$ que tocan un vector de soporte de la clase -1 (que cumple $\boldsymbol{w} \cdot \boldsymbol{x} = -1 - b$) y otro de la clase $+1$ (que cumple $\boldsymbol{w} \cdot \boldsymbol{x} = -1 - b$). Recordando (ver sec. A.1) que la proyección de $\boldsymbol{x}$ sobre $\boldsymbol{w}$ está a la distancia $\boldsymbol{x} \cdot \boldsymbol{w}/||\boldsymbol{w}||$, el margen depende (inversamente) del tamaño de $\boldsymbol{w}$:
>
> $$M = \frac{\boldsymbol{x}_+ \cdot \boldsymbol{w}}{||\boldsymbol{w}||} - \frac{\boldsymbol{x}_- \cdot \boldsymbol{w}}{||\boldsymbol{w}||} = \frac{1-b}{||\boldsymbol{w}||} - \frac{-1-b}{||\boldsymbol{w}||} = \frac{2}{||\boldsymbol{w}||}$$
>
> Por tanto, el hiperplano de máximo margen se puede conseguir minimizando la función objetivo cuadrática $1/2||\boldsymbol{w}||^2 = 1/2\boldsymbol{w} \cdot \boldsymbol{w}$ sujeta a las desigualdades lineales $c_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$.

Normalmente se forma un lagrangiano (sec. D.6) y de él se deduce una forma dual más conveniente sin $\boldsymbol{w}$ ni b. Es un problema estándar de optimización cuadrática con restricciones lineales, pero en la práctica se utiliza software especial, como comentaremos en la sec. 4.7.6.

→ explicar mejor

Para que esta idea sea útil en situaciones reales, hay que modificarla para admitir distribuciones no linealmente separables. Lo que se hace es añadir a la función objetivo un coste para posibles ejemplos mal clasificados. Esto supone una modificación menor en el algoritmo de optimización cuadrática, y en la práctica se observa que las soluciones obtenidas son muy robustas frente al valor concreto de coste elegido.

4.7.5. El 'truco' de *kernel*

Hace un momento nos preguntábamos si es posible conseguir máquinas no lineales de baja capacidad. Una respuesta afirmativa la ofrecen las *máquinas de vectores de soporte* [4, 18]. Se basan en la unión de dos ideas extraordinarias: el hiperplano de máxima separación y el 'truco de kernel'.

Hemos visto que la capacidad de una máquina lineal sólo depende del margen de separación, y no de la dimensión del espacio de atributos. Esto parece indicar que podríamos transformar nuestros vectores originales $\boldsymbol{x} \in \mathbb{R}^n$ mediante una expansión fija de propiedades de cualquier tipo $\phi : \mathbb{R}^n \to \mathbb{R}^s$ (donde $\boldsymbol{y} = \phi(\boldsymbol{x}) \in \mathbb{R}^s$ puede tener muchísimas más componentes que el vector original $\boldsymbol{x}$), y aplicar un hiperplano de máxima separación sobre las muestras transformadas. Si el margen conseguido fuera suficientemente grande, entonces podríamos confiar en que la solución sería fiable. Pero, por supuesto, como ya hemos comentado al hablar de las máquinas lineales generalizadas, el cálculo explícito de cualquier expansión de atributos ϕ mínimamente interesante es intratable.

Sin embargo, es posible trabajar *implícitamente* con expansiones de atributos, sin tener que calcular sus componentes explícitamente. La clave está en que prácticamente cualquier algoritmo que procesa vectores puede expresarse en último término como productos escalares de unos vectores con otros. Por ejemplo, la distancia entre dos vectores (útilizada p. ej., en varios algoritmos de clasificación) se puede escribir como $||\boldsymbol{x} - \boldsymbol{y}||^2 = (\boldsymbol{x} - \boldsymbol{y}) \cdot (\boldsymbol{x} - \boldsymbol{y}) = \boldsymbol{x} \cdot \boldsymbol{x} + \boldsymbol{y} \cdot \boldsymbol{y} - 2\boldsymbol{x} \cdot \boldsymbol{y}$.

Si transformarmos nuestros vectores mediante $\boldsymbol{y} = \phi(\boldsymbol{x})$, entonces en cada ocasión en que un determinado algoritmo calcule un producto escalar $\boldsymbol{x}_a \cdot \boldsymbol{x}_b$, podemos sustituirlo por $\boldsymbol{y}_a \cdot \boldsymbol{y}_b = \phi(\boldsymbol{x}_a) \cdot \phi(\boldsymbol{x}_b)$ y entonces ese algoritmo estará trabajando en el espacio de propiedades definido por ϕ, en lugar del espacio original. La clave está en que podemos calcular el producto escalar $\phi(\boldsymbol{x}_a) \cdot \phi(\boldsymbol{x}_b)$ de forma increíblemente simple si lo hacemos directamente, en función de las componentes de los vectores $\boldsymbol{x}$,

en lugar de calcularlo explícitamente en términos de las numerosísimas coordenadas del espacio de $\boldsymbol{y}$.

> Veamos esto en detalle. Consideremos la siguiente expansión $\phi : \mathbb{R}^2 \to \mathbb{R}^6$, que a partir de puntos del plano obtiene unas cuantas propiedades de que son productos y potencias de coordenadas hasta grado dos (el coeficiente $\sqrt{2}$ se entenderá en un instante):
>
> $$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix}$$
>
> El producto escalar de las expansiones de $\boldsymbol{x}_a = [a_1, a_2]^\mathsf{T}$ y $\boldsymbol{x}_b = [b_1, b_2]^\mathsf{T}$ es un número en el que se acumulan los 6 productos de las nuevas coordenadas:
>
> $$\phi(\boldsymbol{x}_a) \cdot \phi(\boldsymbol{x}_b) = 1 + 2a_1b_1 + 2a_2b_2 + a_1^2b_1^2 + a_2^2b_2^2 + 2a_1a_2b_1b_2$$
>
> Curiosamente, ese número se puede calcular de forma directa de la siguiente forma:
>
> $$(\boldsymbol{x}_a \cdot \boldsymbol{x}_b + 1)^2 = 1 + (a_1b_1 + a_2b_2)^2 + 2(a_1b_1 + a_2b_2)$$
>
> en función del producto escalar de los vectores originales mediante una sencilla operación que solo require 2 productos (en lugar de los 6 anteriores) la suma de 1 y elevar al cuadrado. Por supuesto podemos elegir cualquier otro exponente: si calculamos $(\boldsymbol{x}_a \cdot \boldsymbol{x}_b + 1)^5$ obtendremos con el mismo esfuerzo computacional el producto escalar de una expansión (implícita) de 21 dimensiones basada en productos y potencias de los atributos originales hasta orden 5.

El truco de kernel consiste en sustituir en el algoritmo que nos interese todos los productos escalares de vectores $\boldsymbol{x}_a \cdot \boldsymbol{x}_b$ por una función no lineal simétrica $k(\boldsymbol{x}_a, \boldsymbol{x}_b)$ que calcula el producto escalar en un espacio de propiedades implícito, normalmente de muy elevada dimensión. La dificultad de hacer correctamente esta sustitución depende del algoritmo concreto; a veces hay que reescribirlo de forma un poco diferente para que todas las operaciones queden expresadas como productos escalares. Además, lo usual es inventar funciones de kernel que sean fáciles de calcular, sin preocuparnos demasiado de las propiedades concretas que se inducen, ya que al generarse un número tan grande será muy fácil para los algoritmos elegir la combinación adecuada en cada situación. Dos funciones de kernel típicas son el *polinomial* de orden p, ya visto en el ejemplo anterior:

$$k(\boldsymbol{x}_a, \boldsymbol{x}_b) = (\boldsymbol{x}_a \cdot \boldsymbol{x}_b + 1)^p$$

y el kernel *gaussiano* de anchura σ:

$$k(\boldsymbol{x}_a, \boldsymbol{x}_b) = \exp\left(\frac{\boldsymbol{x}_a - \boldsymbol{x}_b}{\sigma}\right)^2$$

que genera un espacio de propiedades implícito de *dimensión infinita* (!). (Esto es así porque la exponencial se puede expresar como una serie de infinitas potencias). Valores razonables para los parámetros k ó σ de las funciones de kernel dependen realmente de cada problema, y suelen elegirse mediante prueba y error. Otra función de kernel que también se podría utilizar está basada en tangente hiperbólica, que da lugar a máquinas muy similares a las redes neuronales que estudiamos en la sección anterior.

Como ejemplo, vamos a aplicar el truco de kernel a la máquina lineal $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$ basada en la pseudoinversa estudiada en la sec. 4.2. En el espacio original deseábamos cumplir la igualdad

$$\mathtt{X}\boldsymbol{w} = \boldsymbol{C}$$

que en el espacio de propiedades se convierte en

$$\Phi\boldsymbol{w}' = \boldsymbol{C}$$

donde Φ es una matriz cuyas filas son las expansiones $\phi(\boldsymbol{x}_i)^\mathsf{T}$ de cada ejemplo de entrenamiento (de dimensión potencialmente infinta) y $\boldsymbol{w}'$ es un vector de pesos de esa misma dimension enorme. La idea esencial consiste en expresar este vector $\boldsymbol{w}'$ como una combinación lineal de los ejemplos expandidos[3]:

$$\boldsymbol{w}' = \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i) = \Phi^\mathsf{T}\boldsymbol{\alpha}$$

Introduciendo esta expresión para $\boldsymbol{w}'$ en el sistema de ecuaciones obtenemos la siguiente condición:

$$\Phi\Phi^\mathsf{T}\boldsymbol{\alpha} = \boldsymbol{C}$$

cuya solución de mínimos error cuadrático para $\boldsymbol{\alpha}$ puede expresarse inmediatamente, como ya sabemos, en términos de la pseudoinversa:

[3] Aquí surge un detalle técnico: $\boldsymbol{w}'$ tiene la misma dimensión (enorme) que los ejemplos expandidos pero el número de ejemplos en la combinación lineal será normalmente mucho menor. Por tanto, estamos restringiendo la solución a un subespacio (la envolvente lineal de los ejemplos expandidos) aparentemente muy restringido dentro del espacio de soluciones posibles. Pero esto no supone ningún problema, ya que si existe una solución para $\Phi\boldsymbol{w}' = \boldsymbol{C}$ también existirá aunque $\boldsymbol{w}'$ se restrinja al subespacio generado por los ejemplos. La misma diversidad que aparece en los ejemplos se usa para fabricar la solución.

$$\boldsymbol{\alpha} = (\boldsymbol{\Phi}\boldsymbol{\Phi}^\mathsf{T})^+ \boldsymbol{C}$$

La matriz $\boldsymbol{\Phi}\boldsymbol{\Phi}^\mathsf{T}$ es de tamaño 'razonable' ($n \times n$, siendo n el número de ejemplos), a pesar de que se construye como el producto de dos matrices con una de sus dimensiones (la que se contrae en el producto) que es inmensamente grande. La clave está en que sus elementos son simplemente los productos escalares de unos ejemplos con otros, que pueden calcularse de forma inmediata con la función de kernel:

$$\boldsymbol{\Phi}\boldsymbol{\Phi}^\mathsf{T} = \{\phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j)\}_{i=1,j=1}^{n} = \{k(\boldsymbol{x}_i, \boldsymbol{x}_j)\}_{i=1,j=1}^{n}$$

Una vez que tenemos $\boldsymbol{\alpha}$ podemos construir la máquina de clasificación de futuros vectores $\boldsymbol{t}$:

$$f(\boldsymbol{t}) = \boldsymbol{w}' \cdot \phi(\boldsymbol{t}) = \left(\sum_{i}^{n} \alpha_i \phi(\boldsymbol{x}_i) \right) \cdot \phi(\boldsymbol{t}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{t})$$

El inconveniente de la mayoría de los algoritmos 'kernelizados' es que la solución se basa en algún tipo de expresión en la que aparecen todos los ejemplos de entrenamiento. Por el contrario, cuando se aplica el truco de kernel al hiperplano de máxima separación, la solución queda en función únicamente de los vectores de soporte (es *sparse*), dando lugar a un clasificador muy eficiente.

4.7.6. La máquina de vectores de soporte (SVM)

La máquina de vectores de soporte se refiere normalmente a un hiperplano de máxima separación en el espacio de propiedades inducido por una función de kernel. Tiene muchas ventajas:

- Es no lineal: puede resolver problemas de cualquier complejidad.
- Es de máximo margen: se controla la capacidad, consiguiendo generalización.
- El algoritmo de aprendizaje es directo: la programación cuadrática involucrada tiene una solución bien definida que puede encontrarse eficientemente mediante un algoritmo especialmente diseñado para este problema. No tiene mínimos locales. No se basa en procesos de optimización iterativa como los utilizados en las redes neuronales, que pueden encontrar o no, tras un tiempo indeterminado alguna solución más o menos aceptable.
- La solución es concisa: tiene la forma de una combinación lineal de la función de kernel únicamente sobre los vectores de soporte, lo que hace que el clasificador obtenido sea computacionalmente muy eficiente.

- Dependiendo del tipo de kernel elegido, la estructura de la máquina obtenida equivale a otros modelos ya conocidos, pero en este caso su ajuste se basa en un algoritmo eficiente y cerrado, y que además obtiene una solución con más esperanzas de generalización.

Por estas razones no es de extrañar que sea una de las técnicas más utilizadas en la actualidad. En la página `www.kernelmachines.org` hay mucha más información, tutoriales y software.

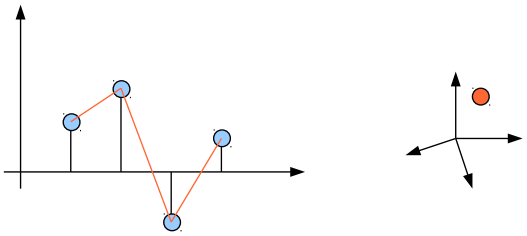
Ejercicios:

- Descarga alguna implementación de la SVM (*svmlight*, SOM, etc.) y pruébala con alguno de los problemas de clasificación que han surgido en las prácticas.

4.8. Procesos Gaussianos

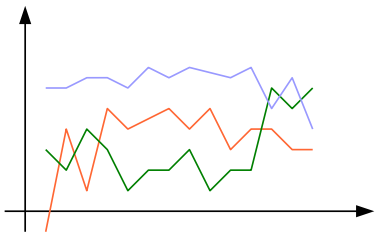
Los procesos gaussianos son una tecnología alternativa para diseñar máquinas de kernel, basadas en un sencillo y atractivo enfoque probabilístico. Aquí simplemente motivaremos el tema con una introducción intuitiva, en la línea del excelente tutorial [6]. Una explicación más detallada puede consultarse en el libro [23] (disponible online).

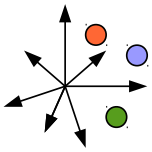
Normalmente estamos visualizando las poblaciones aleatorias como nubes de puntos en un espacio multidimensional. Este modelo mental es casi siempre el más adecuado. Sin embargo, podríamos también representar un objeto $\boldsymbol{y} = \{y_1, y_2, \ldots y_n\}$ como una colección de n puntos del plano, con coordenadas (i, y_i). Esto se parece mucho a una función $y(x)$ muestreada en las posiciones $x = 1, 2, \ldots n$:



Supongamos ahora que la dimensión del espacio va aumentando más y más, de modo que los objetos pueden considerarse como funciones más y más densamente muestreadas [4]:

[4] Esta representación sólo permite mostrar unos pocos puntos del espacio simultáneamente, ya que rápidamente el gráfico se vuelve muy confuso.





En el límite, podemos pensar en un número infinito de coordenadas, distribuidas de forma continua en el eje horizontal, cuyos valores definen una función. Esta función podría ser de cualquier tipo, sin ninguna relación entre los valores en posiciones cercanas, dando lugar a una gráfica parecida a la del ruido aleatorio. Si por el contrario los valores próximos tienen alguna dependencia estadística la función tenderá a ser suave. Esta es la situación natural en los problemas de regresión.

Las funciones pueden considerarse puntos de un espacio de dimensión infinita, sobre el que podemos establecer una densidad de probabilidad y aplicar las reglas de inferencia elementales: si observamos un subconjunto de variables, ¿cuál es la distribución de probabilidad de otras variables no observadas? Se trata de un problema de interpolación tradicional, replanteado como inferencia Bayesiana.

Para resolverlo necesitamos un modelo conjunto de todas las variables. Sorprendentemente, un modelo gaussiano de dimensión infinita (proceso gaussiano) es muy útil para este propósito. La media puede ser la función cero, para no preferir arbitrariamente valores positivos o negativos. Y la matriz de covarianza es una matriz infinita que contiene el coeficiente de covarianza σ_{ab} de cada pareja de coordenadas x_a, x_b. Cuando deseamos modelar funciones suaves debería tener un valor grande para coordenadas próximas, y tender a cero a medida que se alejan entre sí. Por ejemplo:

$$\sigma_{ab} = k(x_a, x_b) = \sigma^2 e^{-\frac{(x_a - x_b)^2}{2l^2}} + \eta\delta_{ab}$$

donde l es un parámetro que controla el grado de suavidad de las funciones. La diagonal, expresada con la delta de Kronecker, contiene el ruido η de las observaciones.

Evidentemente, no es posible trabajar con un modelo gaussiano infinito. Pero en la práctica tenemos un número finito de observaciones de la función y deseamos predecir su valor en otra colección finita de posiciones. Por tanto, podemos marginalizar las infinitas variables no utilizadas para conseguir una gaussiana de dimensión finita, sobre la que podemos aplicar directamente los resultados de la Sección C.5 para obtener estimadores, con localización y dispersión, en las posiciones de interés.

Para ello sólo tenemos que construir los tres bloques necesarios de la matriz de covarianza usando la función $k(x_a, x_b)$ anterior sobre las coordenadas x_k de las muestras observadas y de las posiciones x_k^* que deseamos predecir. Con ellos se obtiene fácilmente la distribución de los valores desconocidos y_k^* condicionados a las observaciones y_k usando la expresión C.1.

- Es posible muestrear prior y posterior
- Bayesian estimation of free parameters
- Classification
- Relación con máquinas de kernel
- Relación con inferencia bayesiana de los coefs. de una máquina lineal.
- Sparse machines (relevance vectors).

4.9. Boosting

→ "Metaalgoritmos": p. ej. arboles de clasificación.

4.9.1. Random Forests

4.9.2. AdaBoost

→ `http://www.cs.princeton.edu/~schapire/boost.html`

- Stumps.
- adaboost
- ejemplos (face detection)

Parte II

Visión por Computador

Guión de clase

C0

- Planteamiento
- Bibliografía
- Entornos de desarrollo
- Transparencias divulgativas
- Demos

C1

- Image digitalizada vs geometría
- representación homogénea puntos y rectas del plano
- Coords homogéneas, rayos, puntos en el infinito
- unión, intersección, dot, cross, expressions, geom interpretation.
- Ejer "horiz" from imag rectangle, linf
- Grupos de transformaciones en el plano, interpret, represent como matriz homog
- Ejer comprobar composición
- Propiedades invariantes (I), "tipos de matriz", cross-ratio
- Ejer "farolas" (predict position of equispaced points in image)

C2

- Cross ratio de 5 rectas
- Transformación de rectas. Covarianza y contravarianza.
- Objetos invariantes en cada tipo de transformación. Desp: p infs., rot centro, afin linf
- Ejer pensar en obj invariante a similar.
- Ejer comprobar transformación de rectas.
- Cónicas: incidencia, tangencia, (pole-polar), cónica dual, transformación.
- comprobar transformación de cónicas.

C3

- 3D points, planes, lines (dof?), plano inf, analogía con 2D
- Transfs 3D: desp, rots, esc, invars, etc.
- Proyección P3 to P2, pinhole model, matriz transf posición normal, camara oscura, lente
- expresión no homogénea, matriz P (proyectar = ver un punto en el rayo que es la rep homog)
- moverla, cámara calibrada C=PRT, 6 dof, parámetros extrínsecos o pose.
- pixelización: interpretación de elementos de K, Kf
- ejer: calibración a ojo
- radial distortion
- KPRT, forma usual K [R|t]
- f es relativa. Coordenadas de imagen: raw, calibradas, normalizadas: f equiv. to field of view.
- Ejer: proyectar cubo con cam sintética (R=I)
- Ejer: generar matriz de cámara desde un C mirando a un p

C4

- Homografía de un plano (ej. z=0).
- Rectificación mediante puntos correspondientes.
- Solución a sistema homogéneo sobredeterminado
- Ejer: Deducir ecuaciones para H. Usar producto kronecker.
- Ejer: estimar cónica
- Ejer: Estimar homografía con correspondencias y rectificar plano
- Ejer: Estimar camara mediante correspondencias y dibujar algo virtual.
- Ejer: Invariante de 5 puntos (p.ej. reconoc. de poligons).
- Invariante proy de 1 cónica no, de 2 sí. Verlo mediante construcción geométrica, y también mediante propiedades de $C_1 C_2^{-1}$. Esa cosa y su imagen son matrices "similares" (la misma transformación en base distinta, por lo que se preserva rank, traza, eigenvalues). Mientras que $C' = H^{-\mathsf{T}} C H^{-1}$ no tiene invariantes, $C_1' {C_2'}^{-1} = H^{-\mathsf{T}} C_1 {C_2}^{-1} H^{\mathsf{T}}$, tiene estructura $P^{-1}AP$.
- estimación de H a partir de correspondencias de cónicas.

C5

- Homografías interesantes relacionadas con cámara: Interimagen planar, del infinito, rotación conjugada
- Ejer: panorámica (de plano, o de rotación)
- Anatomía de cámara: columnas (comprobar en ejer de estimación de M)
- Reprojección de puntos y rectas
- Anatomía de cámara: filas
- Extracción de KRC: factorización RQ. Centro=nullspace. Fix con K ok.
- Ejer: Extraer KRC de cámara sintética y estimada
- Error geométrico vs algebraico. Ejemplo con estim. de una cónica.
- Aproximación afín.

C6

- Rectificación de planos con otra info (sin correspondencias)
- Afín: poner el horizonte en su sitio
- Ejer: probarlo
- f from right angle in horizon, construcción geométrica
- Medición de ángulos en imagen: $cos\alpha$ invariante a homografías.
- Invariante métrico: puntos circulares o su cónica dual C^*_∞.
- Interpretación con diag(f,f,1).
- Ejer (optional): rectif métrica mediante puntos circulares (intersección de círculos)
- Ejer (optional): rectif métrica mediante ángulos rectos
- Recuperación de cámara a partir de homografía de rectificación (varios métodos (equiv. a calib planar Zhang)).
- linear pose (Fiori)
- Ejer: Realidad aumentada en un plano.

C6b

- Analogía con ángulo de rectas en el plano: Ángulos de planos 3D. Comportamiento al transformar. Como cuádrica: Cuádrica dual absoluta. Invariante métrico 3D.
- Plano del infinito. Formación de puntos de fuga y líneas de fuga. Ángulo de rayos/puntos de fuga. Se mete K de una cierta forma.
- Puntos circulares de varios planos en el espacio. ¿dónde van?
- La cónica absoluta. Dual a la cuádrica. Simple eq. en pinf. Es invariante métrico. Con Hinf vemos que su imagen w sólo depende de K. El ángulo de puntos de fuga anterior usa w.
- Ortogonalidad de vanishing points and van. pt and ln. Calibración como homografías y ortho constraints on ω. Deducir métodos de calibración sencillos.
- Construcción del pp

- Ejer Rectificación calibrada from vertical vanishing point. Completar H más simple?
- Ejer alturas relativas.

C7

- Geometría estéreo
- Rayos de un punto. Línea base. Epipolos. Líneas epipolares. A partir de cámaras llegamos a F. Propiedades, grados de libertad. Restricción epipolar. Rectificación.
- Ejer: comprobar restricción epipolar con cámaras sintéticas.
- Computación de F con correspondencias. Normalización.
- Ejer: deducir ecuaciones. Comprobar en caso sintético.
- Ambigüedad projectiva.
- Extracción de cámaras compatibles. Necesidad de calibración.

C8

- Matriz esencial, relación con F, estructura interna. propiedades.
- Ejer: comprobación en caso sintético.
- Extracción de cámaras. Selección del par correcto.
- Ejer: comprobación en caso sintético.
- Triangulación lineal.
- Deducir ecuaciones e implementación.
- Autocalibración simple: Extracción de f from F. Closed form, optimization.
- Ejer: reconstrucción estéreo real.

C9

- SfM, loop closing, Bundle adjustment, Newton, pprox. al Hessian, SBA

C10

- restricción trifocal, cuadrifocal
- intro a tensores
- intro a Grassman, etc.

C11 . . .

- Extracción de propiedades geométricas interesantes: invariantes, identificadas, etc. : puntos, bordes, bordes rectos, contornos, etc.
- Operaciones locales de proceso de imagen. Operaciones lineales. Convolución vs big matrix. Separabilidad.
- Paso bajo, paso alto, mediana.
- image warping
- Filtrado gaussiano: cascading, Fourier, scale space
- Gradiente. Dibujarlo. Máscaras. Módulo y angulo. Laplaciano, LoG, Canny.
- Hough.
- Ejer: horiz of plane from live rectangle, live cross ratio of (5 lines of) pentagon.
- Points: Harris, Hessian
- Lucas-Kanade
- SIFT. Detector y Descriptor. Matching consistencia geométrica. Bag of words (interpretación).
- Ferns, Fast, etc.
- MSER
- LBP, SWT
- . . .

Capítulo 5

Visión por Computador I

In discussing the sense of sight, we have to realize that (outside of a gallery of modern art!) one does not see random spots of color or spots of light. When we look at an object we see a man or a thing; in other words, the brain interprets what we see. How it does that, no one knows, and it does it, of course, at a very high level.

–Feynman

La evolución biológica ha producido organismos capaces de extraer un conocimiento muy preciso de la 'estructura espacial' del mundo externo, en tiempo real, a partir de secuencias de imágenes. Desde el punto de vista del procesamiento de la información este problema presenta una complejidad extraordinaria. La visión artificial supone un desafío para la tecnología y la ciencia actual: ni siquiera las máquinas más potentes pueden acercarse a la capacidad de los sistemas visuales naturales. A pesar de esto, en los últimos años se han producido avances muy significativos tanto en los recursos de cómputo disponibles (GPU) como en nuevos enfoques teóricos (p.ej. propiedades locales invariantes) que hacen posible resolver satisfactoriamente muchos problemas de visión artificial en condiciones de trabajo cada vez más realistas.

La bibliografía sobre visión artificial es inmensa. Este capítulo contiene solamente un guión o esquema de algunas ideas importantes. La referencia esencial sobre geometría visual es [10] y un compendio muy actualizado de la visión artificial en general es [25].

5.1. Planteamiento

En un sistema de visión artificial deseamos obtener a partir de los estímulos (elementos de imagen) una representación de la realidad externa que refleje ciertas propiedades

que nos interesan: típicamente el reconocimiento de objetos, o clases de objetos, o algún modelo de la 'estructura espacial' del espacio circundante.

La dificultad de esta tarea depende esencialmente del grado de control que poseemos sobre las 'condiciones de trabajo': cualquier aplicación de visión artificial puede simplificarse hasta hacerse trivial restringiendo suficientemente el tipo de escenas admitidas. Por tanto, nuestro objetivo es conseguir sistemas capaces de operar en ambientes cada vez más abiertos, menos estructurados. Por ejemplo, es posible diseñar sistemas de guiado visual de robots en el interior de edificios, capaces de 'comprender', en cierta medida, la situación del robot respecto al entorno, p.ej. determinando su posición relativa respecto a paredes, habitaciones o pasillos, etc. Claramente, la navegación en exteriores presenta mucha mayor complejidad.

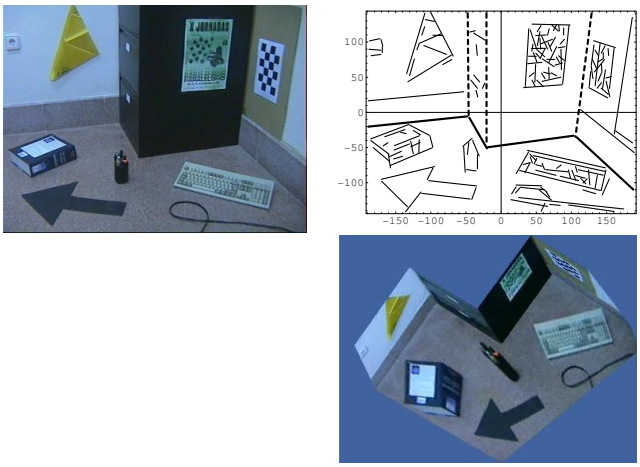
En alguna ocasión se ha definido la visión artificial como el problema inverso de los gráficos por computador: obtener el modelo tridimensional que ha generado una o varias imágenes. Esto ya se ha resuelto: aprovechando resultados teóricos de la geometría proyectiva se pueden obtener modelos 3D del entorno a partir de múltiples imágenes, incluso sin conocer las posiciones desde las que se han tomado ni los parámetros (zoom, etc.) de las cámaras. Se trata de uno de los avances más importantes (si no el que más) de la visión artificial [10].

Sin embargo, una representación perceptual no puede ser una simple 'copia' de la realidad. Debemos evitar la regresión infinita de 'homúnculos' que 'miran' esas representaciones. Es imprescindible extraer ciertas propiedades útiles para el objetivo del sistema.

En ocasiones estas propiedades pueden ser bastante simples: muchas aplicaciones en ambientes controlados pueden resolverse considerando que las imágenes son simplemente puntos en un espacio de dimensión muy elevada, sin tener en cuenta sus 'relaciones espaciales'. Ejemplos de esto son la comparación directa de imágenes o los métodos basados en la extracción de componentes principales (véase el cap. 10.4 de Trucco y Verri [26]). Curiosamente, utilizando este tipo de descripciones estadísticas se obtendrán exactamente los mismos resultados aunque los pixels de la imagen estén permutados arbitrariamente, lo que demuestra que en cierto sentido desperdician el 'aspecto visual' de la información disponible…

En situaciones menos simplificadas la percepción visual deberá apoyarse en procesos de interpretación de 'alto nivel' como los descritos en la sección 1.3. El resultado será un conjunto de propiedades y relaciones *abstractas*, organizadas en una representación *estructurada*.

> Un ejemplo de lo que nos gustaría conseguir se muestra a continuación (véase la tesis doctoral del profesor López-de-Teruel [16, 7]):



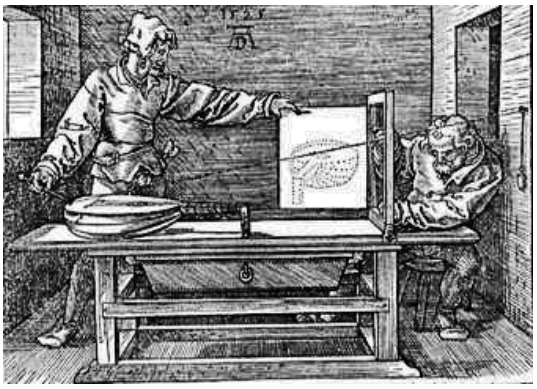
La figura de la izquierda es una imagen de la esquina de una habitación. En el centro se muestran propiedades interesantes de la imagen (segmentos de recta). Tras un proceso de interpretación, algunos se identifican (líneas gruesas) como intersecciones de 'grandes' planos, que, en un edificio, sabemos que son horizontales (suelo) o verticales (paredes). La figura de la derecha muestra la representación obtenida (un modelo 3D basado en planos), que podemos manipular de acuerdo con el objetivo del sistema (en este caso simplemente lo observamos desde otro punto de vista).

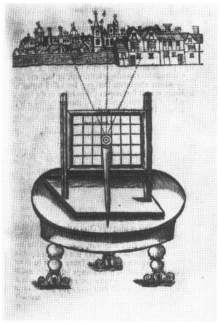
En la interpretación se combina la secuencia de imágenes recibida en cada momento y el modelo actual, para mantenerlo actualizado. En la representación 3D se han 'mapeado' las texturas que aparecen en la imagen original bajo la suposición planar, por lo que los objetos 'con volumen' aparecen deformados. (Obsérvese que esta deformación será inconsistente en varias imágenes, lo que permite detectar objetos sólidos en la escena.)

5.2. Formación de las Imágenes

Una *imagen* se forma en una superficie cuando a cada punto de ella llega luz de un solo punto del mundo. En este apartado estudiaremos las leyes geométricas que rigen la formación de la imágenes.

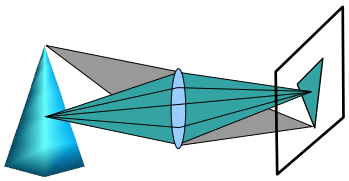
Modelo *Pinhole*. Hacemos pasar la luz por un agujero muy pequeño, forzando a que a cada punto de un plano solo pueda llegar luz de un punto del mundo (un 'rayo'). (Recordar la 'cámara oscura'.)



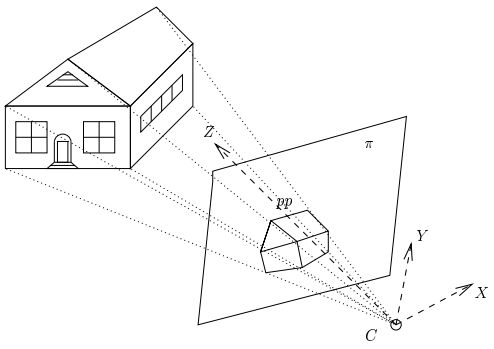


d. John Bate:
Perspectiva.
Grabado sobre madera perteneciente a
Mystères de la Nature et de l'Art, Londres, 1635.

Lente. Produce el efecto de un agujero 'más grande', para que entre más luz. Conseguimos que todo un haz de rayos (no solo uno) que parten un un punto del mundo vayan a parar a un único punto de la imagen. (Surge la necesidad del enfoque, el concepto de profundidad de campo, etc.)



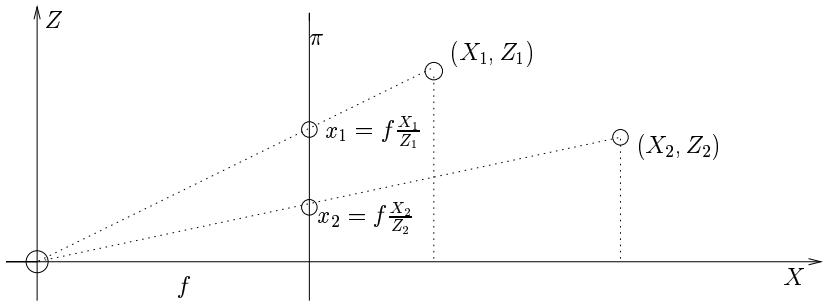
Transformación de Perspectiva. Una cámara queda completamente determinada por el centro de proyección C y el plano de imagen π. Los puntos 3D dan lugar a 'rayos' hasta C, y la imagen es la intersección de estos rayos con el plano π, que está a una distancia f. La dirección en la que mira la cámara es la recta normal al plano que pasa por el centro de proyeccion. El *punto principal* (*pp*) es la intersección de esa recta con el plano de imagen.



Las ecuaciones de la transformación de perspectiva son muy sencillas cuando el centro de proyección es el origen y la cámara apunta en la dirección del eje Z:

$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z}$$

(La coordenada $z = f$ constante en toda la imagen se omite.) Esta transformación se puede interpretar intuitivamente como un simple escalado de las coordenadas, pero con un factor inversamente proporcional a la distancia al punto en *dirección perpendicular* al plano de imagen.



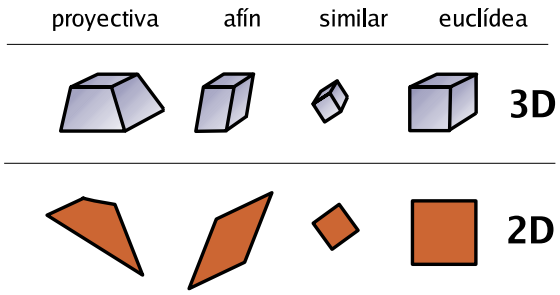
La transformación de perspectiva tiene el inconveniente de que es *no lineal*. Afortunadamente disponemos de herramientas matemáticas (geometría proyectiva, coordenadas homogéneas, etc.) que nos permitirán estudiar la formación de las imágenes mediante técnicas esencialmente lineales.

Perspectiva Débil. Existen otros modelos de formación de imágenes más sencillos. P. ej., en la proyección *ortográfica* nos olvidamos de la coordenada Z, 'aplastando' toda la escena. En general estos modelos producen imágenes que no son 'correctas' (nuestro ojo no las ve así en realidad). No obstante, en ciertas condiciones son perfectamente utilizables: Por ejemplo, si un objeto tiene una profundidad pequeña respecto a la distancia a la cámara se puede usar la transformación de *perspectiva débil*, en la que se supone que todos los puntos de un objeto se encuentran a la misma distancia 'media' $\bar{Z}$. En ese caso la transformación es un simple escalado de coordenadas (con factor de escala $f/\bar{Z}$). Esta simplificación nos permite calcular fácilmente, p.ej., la orientación de un círculo respecto al plano de la cámara.

Geometría proyectiva 2D. Es imprescindible estudiar el capítulo 1 del libro de Hartley y Zisserman [10]. Representación de puntos, rectas y cónicas. Incidencia, intersección. Interpretación como rayos / planos por el origen. Puntos del infinito / direcciones. Transformaciones proyectivas $2D \leftrightarrow 2D$ (*homografías* o *colineaciones*). Acción sobre puntos: $x' = Hx$, rectas: $l' = H^{-T}l$ y cónicas: $C' = H^{-T}CH^{-1}$. Grupos de transformaciones. Cualquier transformación proyectiva puede descomponerse en 3 partes: similar, afín y proyectiva (p.22):

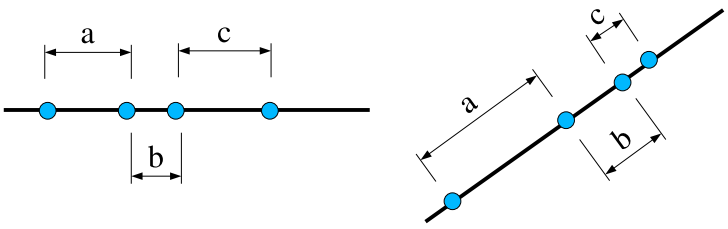
$$H = H_s H_A H_p = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ \mathbf{v}^T & v \end{bmatrix} = \begin{bmatrix} A & t \\ \mathbf{v}^T & v \end{bmatrix}$$

Transformaciones e Invariantes. Para entender bien la naturaleza de la transformación de perspectiva la estudiaremos en el contexto de otras ya conocidas que no afectan al 'aspecto visual' de las figuras. En la Sección 2.3 vimos que es muy sencillo reconocer formas formas planas, independientemente de su posición, su tamaño y su orientación (usando, p.ej., propiedades frecuenciales del contorno). Estas transformaciones, que se llaman 'de similitud', preservan distancias relativas y ángulos.



Las transformaciones 'afines' son más generales: permiten escalas diferentes en cada eje y que los ejes de coordenadas dejen de estar perpendiculares. Aunque ciertas figuras ahora se confunden (círculos y elipses; cuadrados, rectángulos y rombos) la transformación es lineal y mantiene muchas propiedades del aspecto de una figura. Pueden construirse invariantes capaces de reconocer formas sometidas a este tipo de transformación. Preserva paralelismo y áreas relativas.

La transformación de perspectiva es todavía más general, y produce deformaciones en los objetos que pueden llegar a ser muy extremas. Ya ni siquiera se preserva el paralelismo y prácticamente lo único que se mantiene es la 'colinealidad' (las rectas siguen siendo rectas), y una propiedad de 4 puntos sobre una recta llamada ***cross-ratio***:

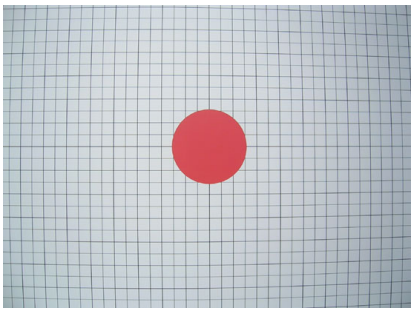


$$\text{cr} = \frac{a}{a+b} \cdot \frac{c}{b+c}$$

Curiosamente, las cónicas también siguen siendo cónicas pero pueden cambiar de 'tipo' (p.ej.: círculo → elipse, elipse → hipérbola, etc., dependiendo de las circunstancias).

Horizonte. Una particularidad fundamental de la transformación de perspectiva es la aparición del horizonte: la imagen de puntos de un plano infinitamente alejados. La geometría proyectiva permite manejar el infinito sin ningún problema, con coordenadas homogéneas. Por ejemplo, más adelante veremos que podemos 'rectificar' un plano visto en perspectiva devolviendo el horizonte al su posición 'original'.

Distorsiones. Los sistemas sistemas ópticos reales suelen presentar desviaciones respecto al modelo proyectivo ideal. Por ejemplo, frecuentemente aparece una deformación radial, que hace que las líneas rectas se vean curvadas. Es más grave en imágenes con gran campo de visión, como la imagen izquierda siguiente. En la imagen derecha también se aprecia una ligera distorsión radial en las rectas verticales de la puerta:



La distorsión radial puede compensarse fácilmente con una tabla de acceso a los pixels. El modelo de compensación más utilizado es el siguiente:

$$x' = (x-o_x)(1+k((x-o_x)^2+(y-o_y)^2)) \qquad y' = (y-o_y)(1+k((x-o_x)^2+(y-o_y)^2))$$

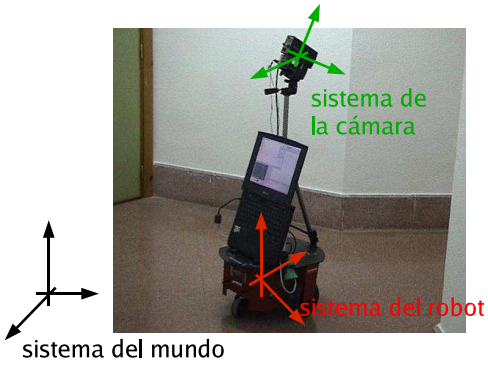
Sus parámetros son el centro de distorsión (o_x, o_y) y el coeficiente de compensación k. Para estimarlos lo más sencillo es minimizar mediante el método de Newton (ver Sec. D.4) la curvatura de un conjunto de rectas conocidas en la imagen.

Dependiendo de la aplicación, si la cámara presenta poca distorsión radial (p. ej., si sólo se aprecia en los bordes de la imagen) puede no ser necesario corregirla.

5.3. Modelo lineal de cámara

Parámetros Extrínsecos. La posición y orientación de la cámara en el espacio queda definida por las coordenadas del centro de proyección $C = (c_x, c_y, c_z)$ y la matriz

de rotación R que hace girar los ejes en el espacio para que la cámara apunte en la dirección deseada.



La matriz R tiene 9 coeficientes pero sólo 3 grados de libertad, que son los ángulos de giro en un cierto convenio. (Hay varias formas de parametrizar una matriz de rotación: ángulos de Euler, *quaterniones*, mediante un vector que codifica la dirección y el ángulo, etc.) En definitiva, la posición de la cámara en el espacio queda determinada por 6 parámetros, que llamamos 'externos' o 'extrínsecos'.

Coordenadas Homogéneas. Las coordenadas homogéneas de un vector (x, y, z) son el conjunto (rayo) de puntos $\lambda(x, y, z, 1)$. Las coordenadas ordinarias de un vector homogéneo (x, y, z, w) son $(x/w, y/w, z/w)$. El 'truco' de añadir una coordenada constante permite expresar de forma lineal transformaciones de perspectiva o desplazamiento (los escalados y rotaciones ya lo son):

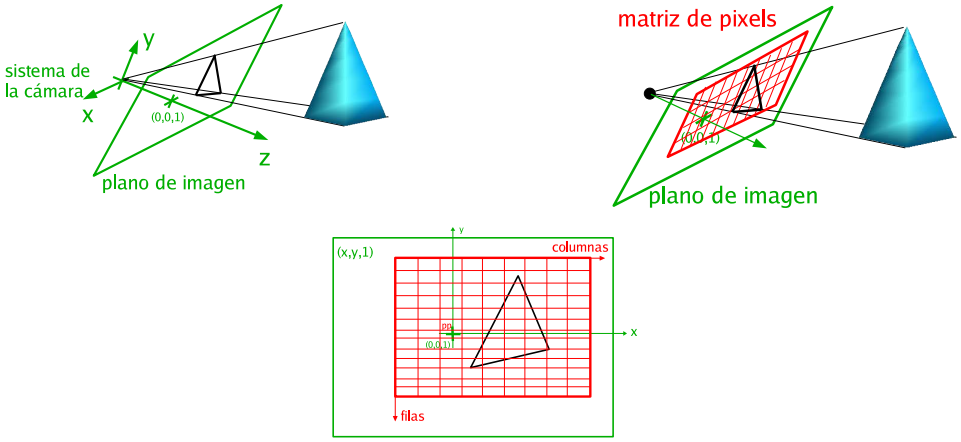
$$T = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} & R_{3\times 3} & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

La transformación de perspectiva P consiste simplemente en 'quitar' la última fila para que la última coordenada $W = Z$, con lo que la división se hace sólo cuando se pasa de coordenadas homogéneas a coordenadas ordinarias.

En realidad, las coordenadas homogéneas son una representación de los elementos del espacio proyectivo, que nos permite manejar convenientemente las transformaciones de perspectiva teniendo en cuenta de manera natural, entre otras cosas, puntos infinitamente alejados (direcciones), que en la imagen aparecen en posiciones finitas (horizonte).

Parámetros Intrínsecos. Los parámetros propiamente dichos de una cámara, también llamados 'internos' o 'intrínsecos', son los que relacionan la transformación entre

el plano de imagen 'ideal' en el espacio, y la malla o matriz de elementos sensibles a la luz (pixel) que 'ponemos encima' para digitalizar la imagen. Permiten calcular la correspondencia desde puntos del mundo hasta los pixels de imagen, y, a la inversa, desde los pixels de imagen hasta conjuntos de de puntos (rayos) del mundo.



Un modelo bastante general y matemáticamente sencillo de la relación entre el plano de imagen (x, y) y la matriz de pixels (p, q) es la siguiente transformación afín:

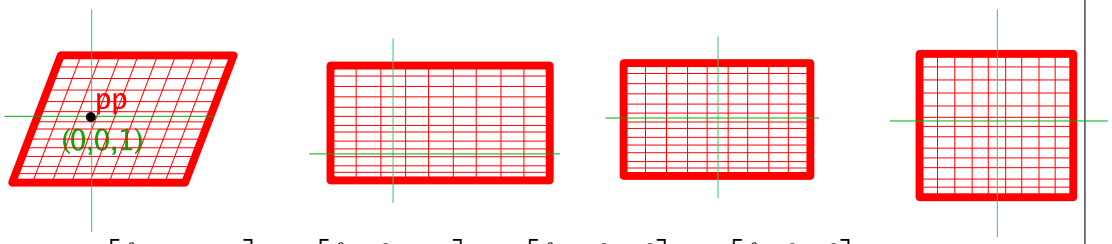
$$\begin{bmatrix} p \\ q \\ 1 \end{bmatrix} = K \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad K = \begin{bmatrix} f & s & o_x \\ 0 & fr & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

En realidad resulta equivalente tener pixels grandes y el plano de imagen muy alejado que tener pixels pequeños y el plano de imagen muy cerca del centro de proyección. El parámetro fundamental del modelo *pinhole* es la *distancia efectiva* f (la distancia real entre el centro de proyección y el plano de imagen, dividida por el tamaño del pixel). Indica el tamaño en pixels de un objeto de tamaño unidad a distancia unidad, o, alternativamente, la distancia en pixels correspondiente a rayos separados 45 grados.

Además de la distancia efectiva f, la transformación K incluye otros parámetros internos: el *aspect ratio* r de la imagen (para tener en cuenta la posibilidad de pixels rectangulares), el *skew* s o inclinación de los ejes (que se puede producir si la matriz de píxels no es paralela al plano de imagen) y la posición (o_x, o_y) del punto principal.

Normalmente los pixels son cuadrados, por lo que $r = 1$ y $s = 0$. Por otra parte, si el campo de visión de la cámara no es excesivamente amplio (menor que, digamos, 40 ó 50 grados) y siempre que la aplicación no exija precisiones muy elevadas, es perfectamente razonable trabajar bajo la suposición de que el punto principal está en el centro del array de pixels. (De hecho, es prácticamente imposible estimar el pp, a partir de medidas reales, y por tanto imprecisas, con precisión mejor que 10 ó 15

grados.) La figura siguiente muestra mallas de pixels correspondientes a modelos de cámara cada vez más sencillos:

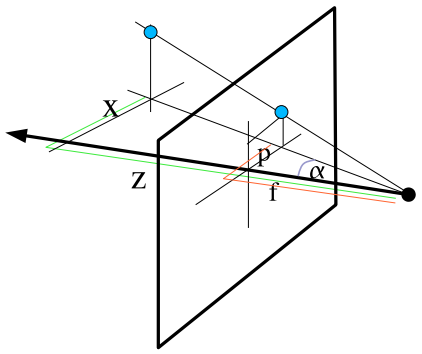


$$\begin{bmatrix} f & s & o_x \\ 0 & fr & o_y \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} f & 0 & o_x \\ 0 & fr & o_y \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

En conclusión, en muchos casos prácticos sólo hay un parámetro interno relevante de la cámara: la focal efectiva f. En este caso, trasladando el origen de los pixels al centro de la imagen, la matriz de calibración es simplemente:

$$K_f = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Cámara como medidor de ángulos. Acabamos de ver que la matriz K es la transformación que relaciona los pixels de imagen con los rayos –y sus ángulos correspondientes– en el espacio.



$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} p \\ q \\ 1 \end{bmatrix}$$

$$\frac{x}{z} = \frac{p}{f} = \tan\alpha$$

En una primera aproximación, para pixels próximos al punto principal (intersección del eje óptico con el plano de imagen), el ángulo entre los rayos definidos por dos pixels es simplemente la distancia entre ellos divida por f. Esto nos permite hacernos una idea del orden de magnitud de f a partir de la imagen de un objeto con tamaño angular conocido.

Matriz de cámara. La transformación 'global' de puntos del mundo a pixels de imagen, teniendo en cuenta la posición, orientación y parámetros internos de la cámara, se obtiene de la acción combinada de las transformaciones elementales T, R, P y K. El uso de coordenadas homogéneas permite expresarlas todas de forma lineal y, por tanto, la composición de transformaciones se 'implementa' directamente como un producto de matrices.

En la práctica la matriz de cámara se expresa concisamente como:

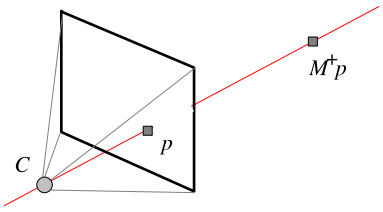
$$\boxed{M = K[R|t]}$$

donde R es la matriz de rotación no homogénea, $t = -RC$ es un un vector columna que depende del centro de proyección y K es la matriz de calibración. (La notación $[A|B]$ se usa para describir matrices 'por bloques'.) M es una matriz 3×4 homogénea con 11 grados de libertad (la escala es irrelevante): hasta 5 parámetros intrínsecos y 6 extrínsecos.

Esta descomposición de la matriz de cámara es muy importante y se utilizará más adelante en muchas ocasiones.

Centro de la cámara. Puede extraerse fácilmente de la matriz de cámara porque (en coordenadas homogéneas) es su espacio nulo: $\mathtt{M}\boldsymbol{C} = \mathbf{0}$.

Rayo dado un punto. Es la recta que contiene todos puntos del espacio que se proyectan sobre ese punto de la imagen. Puede expresarse de forma paramétrica como la recta que une el centro de proyección y un punto cualquiera del rayo:



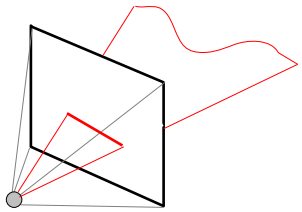
$$\boldsymbol{X}_{\boldsymbol{p}}(\lambda) = \lambda \boldsymbol{C} + \mathtt{M}^{+}\boldsymbol{p}$$

Se utiliza la *pseudoinversa* $\mathtt{M}^{+}$ de $\mathtt{M}$, que en este caso verifica (ver Sec. A.4) $\mathtt{M}\mathtt{M}^{+} = \mathtt{I}$. El centro de proyección $\boldsymbol{C}$ es el espacio nulo de $\mathtt{M}$, por lo que claramente $\mathtt{M}\boldsymbol{X}_{\boldsymbol{p}}(\lambda) = \boldsymbol{p}$.

> En coordenadas ordinarias la recta que une dos puntos $\boldsymbol{a}$ y $\boldsymbol{b}$ se expresa de forma paramétrica como $\boldsymbol{x}(\lambda) = \boldsymbol{a} + \lambda(\boldsymbol{b} - \boldsymbol{a})$. En coordenadas homogéneas la familia $\mu\boldsymbol{x}(\lambda) = \mu\boldsymbol{a} + \mu\lambda(\boldsymbol{b} - \boldsymbol{a})$ representa los mismos puntos, por lo que la recta que une $\boldsymbol{a}$ y $\boldsymbol{b}$ también puede expresarse como $\boldsymbol{x}(\eta, \nu) = \eta\boldsymbol{a} + \nu\boldsymbol{b}$. (Hay dos parámetros, η y ν, que dependen de los anteriores μ y λ, pero un solo grado de libertad debido al factor de escala común irrelevante de la representación homogénea.) En otras palabras, los puntos $\boldsymbol{a}$ y $\boldsymbol{b}$ del espacio proyectivo son

subespacios vectoriales de dimensión 1 y la recta que los une es la envolvente lineal de esos dos subespacios. Si excluimos uno de los puntos de la parametrización, digamos $\boldsymbol{a}$, podemos expresar la recta que los une con un solo parámetro ($\lambda = \frac{\eta}{\nu}$) como $\boldsymbol{x}(\lambda) = \boldsymbol{\lambda} a + \boldsymbol{b}$.

Plano dada una recta. El plano $\boldsymbol{\pi}$ del espacio que se proyecta sobre una recta de la imagen $\boldsymbol{l}$ es simplemente $\boldsymbol{\pi} = \mathtt{M}^\mathsf{T}\boldsymbol{l}$.



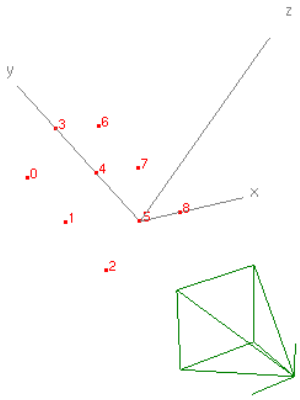
$$\boldsymbol{\pi} = \mathtt{M}^\mathsf{T}\boldsymbol{l}$$

Los puntos del espacio que están en ese plano cumplen $\boldsymbol{\pi}^\mathsf{T}\boldsymbol{X} = 0$ y la imagen de cualquier $\boldsymbol{X}$ es $\boldsymbol{x} = \mathtt{M}\boldsymbol{X}$. Los puntos de imagen que están en la recta cumplen $\boldsymbol{l}^\mathsf{T}\boldsymbol{x} = 0$, por tanto $\boldsymbol{l}^\mathsf{T}\boldsymbol{x} = \underbrace{\boldsymbol{l}^\mathsf{T}\mathtt{M}}_{\boldsymbol{\pi}^\mathsf{T}}\boldsymbol{X} = 0$.

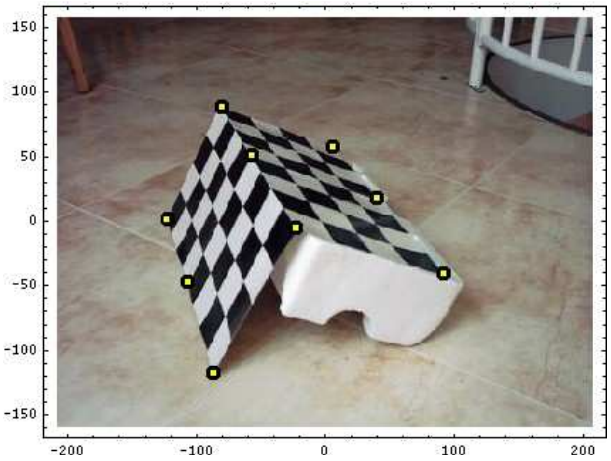
5.4. Calibración

A partir de correspondencias entre posiciones de imagen (pixels) y puntos 3D con coordenadas conocidas es posible estimar la matriz completa M de la cámara. Para ello habrá que resolver un sistema de ecuaciones homogéneo por mínimos cuadrados. Cada correspondencia da lugar a dos ecuaciones o restricciones independientes sobre los elementos de M, por lo que necesitamos al menos 5 puntos 'y medio' para estimar sus 11 grados de libertad. Por supuesto, es mejor utilizar más puntos de calibración para sobredeterminar el sistema y mejorar robustez de la solución frente a los inevitables errores de medida.

La imagen siguiente contiene una plantilla de calibración (un tablero de ajedrez de $8 \times 53mm$ de lado, doblado en ángulo recto) que define un sistema de coordenadas en el que podemos identificar puntos 3D.

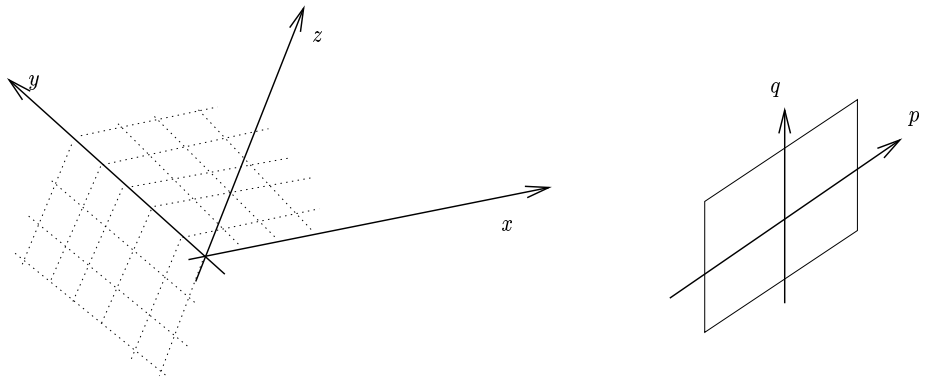


De arriba a abajo y de izquierda a derecha hemos hemos marcado 9 correspondencias:



$$\begin{pmatrix} x & y & z \\ 0 & 2 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & -1 \\ 0 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} p & q \\ -123 & 2 \\ -107 & -47 \\ -87 & -117 \\ -81 & 89 \\ -57 & 51 \\ -23 & -5 \\ 5 & 58 \\ 40 & 18 \\ 91 & -40 \end{pmatrix}$$

Las coordenadas (x, y, z) están en unidades simplificadas ($4 \times 53mm$) y los pixels (p, q) se miden desde el centro de la imagen. El convenio de ejes elegido es el siguiente:



Cada correspondencia $(x, y, z) \rightarrow (p, q)$ impone la restricción $M(x, y, z, 1) = \lambda(p, q, 1)$ sobre los elementos de M, lo que implica que $(p, q, 1) \times M(x, y, z, 1) = (0, 0, 0)$. (La igualdad homogénea significa en realidad paralelismo de vectores.) Aunque de las tres ecuaciones sólo dos de ellas –cualesquiera– son independientes, es recomendable usar las tres para obtener un sistema de ecuaciones mejor condicionado numéricamente. Los coeficientes de las ecuaciones suministradas por cada correspondencia son:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & -x & -y & -z & -1 & qx & qy & qz & q \\ x & y & z & 1 & 0 & 0 & 0 & 0 & -px & -py & -pz & -p \\ -qx & -qy & -qz & -q & px & py & pz & p & 0 & 0 & 0 & 0 \end{bmatrix}$$

Las correspondencias del ejemplo anterior dan lugar al sistema siguiente (se muestran sólo las ecuaciones correspondientes al primer, segundo y último punto):

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -2 & 1 & -1 & 0 & 4 & -2 & 2 \\ 0 & 2 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 246 & -123 & 123 \\ 0 & -4 & 2 & -2 & 0 & -246 & 123 & -123 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 0 & -47 & 47 & -47 \\ 0 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 107 & -107 & 107 \\ 0 & 47 & -47 & 47 & 0 & -107 & 107 & -107 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & -40 & 0 & 0 & -40 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -91 & 0 & 0 & -91 \\ 40 & 0 & 0 & 40 & 91 & 0 & 0 & 91 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_{12} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

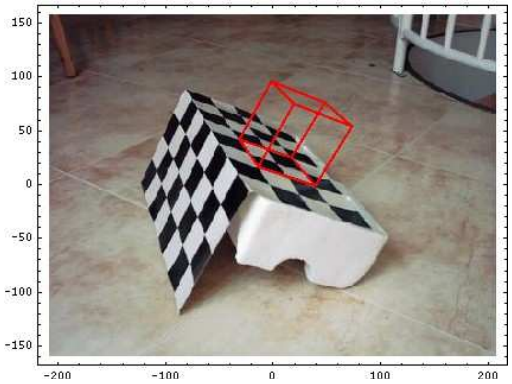
Optimización lineal: mínimos cuadrados. Recordemos que para resolver un sistema de ecuaciones no homogéneo por mínimos cuadrados $Ax = b$ se utiliza la *pseudoinversa*: $x = A^{+}b$. Cuando el sistema es homogéneo ($b = 0$) este procedimiento no funciona (obtiene la solución trivial $x = 0$).

Una solución no trivial de $Ax = 0$ es el vector propio de menor valor propio de la matriz $A^T A$. Alternativamente podemos utilizar la descomposición en valores singulares de A; la solución es el vector fila derecho de menor valor singular. (Ver Apéndice A.6 de Trucco y Verri o la sección A.4). (Otra posibilidad, menos recomendable, es fijar a 1 el valor de un coeficiente que confiemos no sea cero (en nuestro caso, p.ej., m_{12}) y resolver entonces un sistema no homogéneo, con 11 coeficientes libres, mediante el método ordinario de la pseudoinversa.) Para mejorar la estabilidad se recomienda normalizar los puntos.

La solución al sistema del ejemplo anterior, reescalada para que el elemento m_{12} sea 1 y organizada como matriz 3×4, da lugar a la siguiente matriz de cámara:

$$M = \begin{pmatrix} 123.088 & -46.516 & 72.955 & -23.079 \\ -39.263 & 66.110 & 124.601 & -4.405 \\ 0.097 & 0.217 & -0.107 & 1 \end{pmatrix}$$

Para comprobar que es correcta, podemos proyectar los puntos de referencia sobre la imagen, lo cual nos permite cuantificar el error de la transformación. Mejor aún, es posible representar 'objetos virtuales' sobre la imagen, p.ej., un cubo unitario:



Identificación de parámetros. Una vez obtenida la matriz de la cámara, es posible extraer de ella los parámetros intrínsecos y extrínsecos. Aunque hay otros procedimientos (ver Trucco y Verri, capítulo 6) sin ninguna duda la mejor alternativa es usar la factorización RQ, que descompone una matriz directamente en la forma deseada.

La matriz de cámara anterior puede factorizarse como `M = KRT`, donde :

$$K = \begin{bmatrix} 569.96 & 24.83 & -86.78 \\ 0 & 560.13 & -40.18 \\ 0 & 0 & 1.00 \end{bmatrix}, \quad R = \begin{bmatrix} -0.90 & 0.21 & -0.39 \\ 0.24 & -0.51 & -0.82 \\ -0.37 & -0.83 & 0.41 \end{bmatrix}, \quad T = \begin{bmatrix} 1 & 0 & 0 & 1.75 \\ 0 & 1 & 0 & 3.23 \\ 0 & 0 & 1 & -1.20 \end{bmatrix}$$

La matriz de intrínsecos K indica que la focal efectiva es aproximadamente 565. De la traslación `T` $= [$`I`$| -$ `C`$]$ deducimos, p. ej., que el centro de la cámara estaba aproximadamente a 82 cm del origen de coordenadas. (Aunque se pueden interpretar los elementos de `R` y `T` en términos de ángulos y desplazamientos respecto a los ejes de la plantilla, en muchas aplicaciones esto no suele ser necesario).

Optimización no lineal. Usando algún entorno de cálculo científico es posible programar el método de optimización iterativa de Newton o similar, para ajustar los parámetros de modelos no lineales. Esto puede ser recomendable cuando conocemos con precisión ciertos parámetros (p.ej. los intrínsecos, o una K simplificada) y solo deseamos estimar los demás. (p.ej. la posición y orientación de la cámara o de un objeto).

Enfoque no calibrado. En general, la estimación conjunta de todos los parámetros, intrínsecos y extrínsecos no es excesivamente robusta; en realidad están bastante acoplados.

Muchas veces es aconsejable utilizar técnicas de reconstrucción no calibrada, capaces de fabricar un modelo 3D casi correcto, a falta de una transformación proyectiva (ver Sec. 7.3.1) que pone las cosas (los ángulos) en su sitio. Esa transformación puede obtenerse a partir de conocimiento de la escena o mediante *autocalibración* (ver

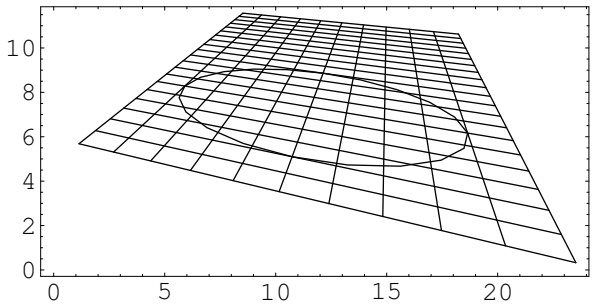
Sec. 7.4): en lugar de utilizar correspondencias entre puntos de imagen y posiciones 3D conocidas, es posible extraer los parámetros de calibración a partir solamente de puntos correspondientes en varias imágenes, independientemente de su posición real en el espacio (siempre que sea fija).

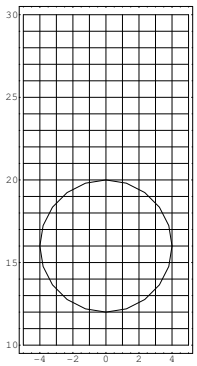
5.5. Rectificación de planos

La correspondencia entre puntos 3D arbitrarios y puntos de imagen (2D) no es invertible: a un punto de la imagen le corresponde todo un rayo de posibles puntos del mundo. Para determinar la profundidad necesitamos más información, como p.ej. una imagen desde otro punto de vista e intersectar los rayos (visión estéreo).

Sin embargo, si los puntos observados pertenecen a un plano conocido, es posible obtener la profundidad intersectando el rayo óptico y el plano. Por tanto, una vista en perspectiva de un plano sí puede 'deshacerse' (decimos que lo *rectificamos*), ya que la correspondencia entre planos es biyectiva ('uno a uno'). Para deshacer el efecto de una transformación homogénea H se usa la inversa ordinaria H^{-1}.

Las figuras siguientes muestran un plano sintético visto en perspectiva y una versión perfectamente rectificada del mismo:





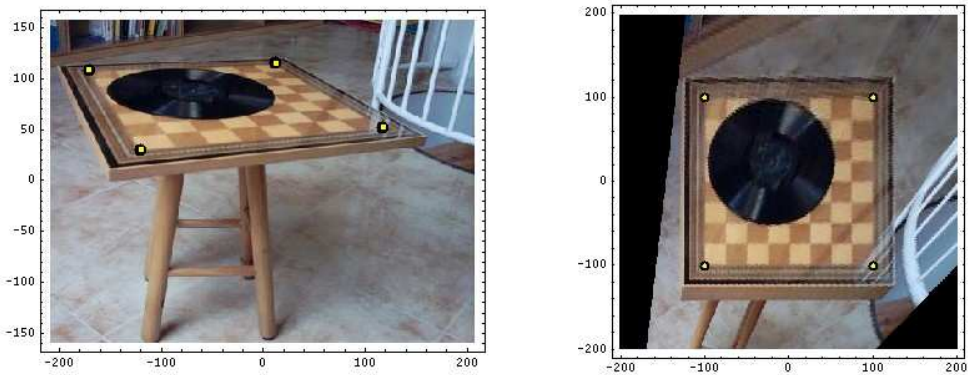
En principio, podría parecer que para rectificar la imagen de un plano necesitamos conocer la posición C, orientación R y parámetros internos K de la cámara respecto al plano, y con ellos deducir la transformación H y su inversa. (Más adelante estudiaremos la estructura de la homografía entre un plano y la imagen de ese plano visto por una camára.) En realidad no es necesaria tanta información: la homografía entre dos planos queda completamente determinada por su acción sobre solo cuatro puntos (o cuatro rectas), por lo que podemos estimarla directamente a partir de cuatro (o más) puntos de la imagen cuyas coordenadas reales son conocidas.

El procedimiento a seguir es exactamente el mismo que el de la estimación de la matriz de cámara mediante una plantilla de calibración. En este caso estimamos los 9 elementos de una matriz H homogénea 3×3 (con 8 grados de libertad). Cada

correspondencia $(x, y) \leftrightarrow (p, q)$ impone 2 restricciones sobre los elementos de H que, de nuevo, es conveniente expresar como $(p, q, 1) \times H(x, y, 1) = (0, 0, 0)$, dando lugar a 3 ecuaciones homogéneas. La matriz de coeficientes de cada correspondencia es:

$$\begin{bmatrix} 0 & 0 & 0 & -x & -y & -1 & qx & qy & q \\ x & y & 1 & 0 & 0 & 0 & -px & -py & -p \\ -qx & -qy & -q & px & py & p & 0 & 0 & 0 \end{bmatrix}$$

Como ejemplo vamos a rectificar un tablero de ajedrez visto en perspectiva:



En el plano rectificado las casillas del tablero se ven cuadradas y el disco circular. Lógicamente, los objetos que no pertenecen al plano aparecen distorsionados. Las correspondencias utilizadas y la transformación de rectificación obtenida han sido las siguientes:

$$\begin{pmatrix} x & y \\ -171 & 109 \\ -120 & 31 \\ 117 & 53 \\ 11 & 115 \end{pmatrix} \rightarrow \begin{pmatrix} x' & y' \\ -100 & 100 \\ -100 & -100 \\ 100 & -100 \\ 100 & 100 \end{pmatrix}, \qquad H = \begin{bmatrix} 0.63868 & 0.77290 & -39.73059 \\ -0.11082 & 1.94177 & -165.90578 \\ -0.00034 & -0.00378 & 1 \end{bmatrix}$$

Este método de rectificación es sencillo y solo requiere las coordenadas reales de (al menos) cuatro puntos observados. Curiosamente, la geometría proyectiva permite obtener también información útil sobre el plano real a partir, incluso, de menos información:

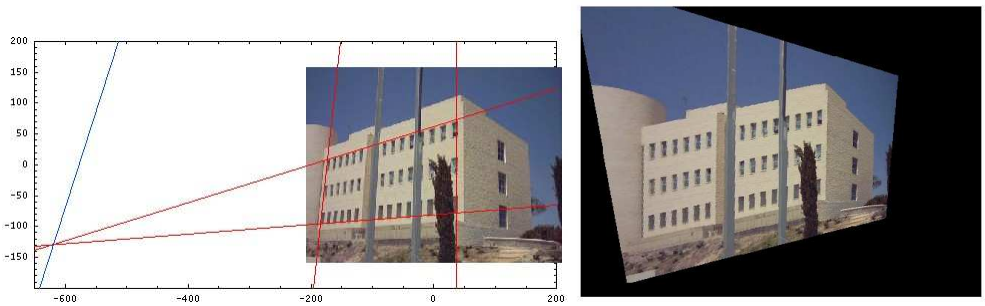
Rectificación afín: horizonte. Es fácil comprobar que si una transformación es solo afín, sin componente proyectivo, la línea del infinito $l_\infty = (0, 0, 1)$ del plano original no cambia: si imaginamos un plano cuadriculado, cada cuadro se transforma un un

rombo arbitrario, pero no hay un efecto 'horizonte'. (En una transformación proyectiva general l_∞ sí se transforma en una línea finita: el horizonte del plano.) Se puede demostrar fácilmente que cualquier transformación que devuelva el horizonte, con coordenadas $l'_\infty = (l_1, l_2, l_3)$, a su posición inicial 'infinita' $(0, 0, 1)$, producirá un plano que se encuentra solo a falta de una transformación afín del verdadero plano original. Sería una especie de 'rectificación incompleta', pero que en ciertos casos puede ser útil.

Esto se consigue, p. ej., con la siguiente homografía de rectificación:

$$H_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{bmatrix}$$

Si el horizonte no es claramente visible en la imagen podemos identificarlo aprovechando la intersección de rectas que sabemos que en realidad son paralelas :



(En este ejemplo la intersección de las rectas verticales se sale de la imagen y no se muestra en la figura). El horizonte del plano de la fachada, en azul, es la recta que une las intersecciones de los dos juegos de paralelas. La imagen resultante no ha recuperado los ángulos reales pero sí el paralelismo.

En conclusión, una rectificación afín (a falta de una transformación afín) solo requiere identificar el horizonte del plano. Para algunas aplicaciones esto puede ser suficiente, ya que propiedades como el paralelismo y los cocientes de áreas pueden aportar bastante información sobre la escena.

Rectificación similar: ángulos. Acabamos de ver que toda transformación afín deja invariante la línea del infinito. Análogamente, toda transformación similar deja invariante el conjunto de puntos que verifican $x^T \mathcal{C} x = 0$, donde

$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

(Algo parecido a una cónica pero con coordenadas complejas, sin existencia en el plano 'real'; podemos considerarlo simplemente como un instrumento matemático útil.)

Esos puntos sólo sufren el componente afín y proyectivo de una transformación H, que, usando la notación anterior, se convierte en:

$$\mathcal{C}' = \begin{bmatrix} KK^T & KK^T\mathbf{v} \\ \mathbf{v}^T KK^T & \mathbf{v}^T KK^T\mathbf{v} \end{bmatrix}$$

Si conseguimos identificar los coeficientes de $\mathcal{C}'$, mediante la expresión anterior podemos deducir el componente afín (K) y proyectivo ($\mathbf{v}$) de la transformación H, de modo que podríamos hacer una rectificación 'similar' del plano (a falta de una transformación similar): un modelo a escala (y más o menos girado). En la práctica esto se hace directamente expresando la descomposición SVD de $\mathcal{C}'$ como $U\mathcal{C}U^T$, y entonces $H_s = U$ es la transformación de rectificación deseada.

La cuestión clave, por supuesto, es identificar la imagen de de una cónica compleja (!). (En principio, la rectificación afín es más fácil porque el horizonte tiene coordenadas 'reales' en la imagen; puede ser directamente visible o deducirse a partir de pistas como p.ej. la intersección de rectas que sabemos que son paralelas.) La rectificación similar requiere dar un pequeño rodeo para identificar $\mathcal{C}'$.

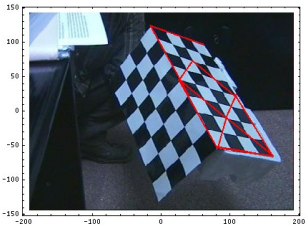
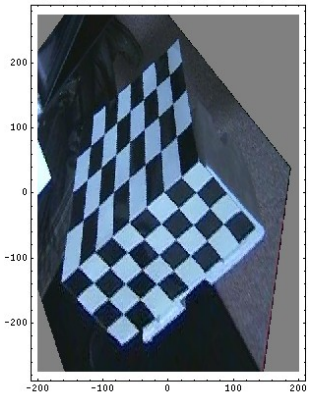
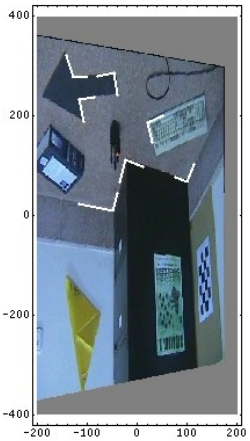
Imaginemos que deseo calcular el ángulo real que forman dos rectas que veo en la imagen de un plano. Una solución obvia consiste en rectificar el plano (al menos hasta una versión similar), y calcular el ángulo entre las dos rectas rectificadas. ¿Podríamos calcular el ángulo real sin rectificar el plano? Sí, si conocemos $\mathcal{C}'$. La clave está en que el ángulo θ entre dos rectas l y m se puede obtener mediante una expresión invariante frente a transformaciones proyectivas:

$$cos\theta = \frac{l^T\mathcal{C}m}{\sqrt{(l^T\mathcal{C}l)(m^T\mathcal{C}m)}} = \frac{l'^T\mathcal{C}'m'}{\sqrt{(l'^T\mathcal{C}'l')(m'^T\mathcal{C}'m')}}$$

Por tanto, si conocemos $\mathcal{C}'$ podemos calcular los ángulos reales a partir de los ángulos en la imagen.

Ahora bien, lo importante es que la expresión anterior también nos sirve para deducir $\mathcal{C}'$ si conocemos los ángulos reales que forman ciertas rectas de la imagen. Lo más sencillo es buscar rectas perpendiculares, que dan lugar a restricciones simples del tipo $l'^T\mathcal{C}'m' = 0$ sobre los elementos de $\mathcal{C}'$. Si desconocemos el horizonte, $\mathcal{C}'$ tiene 5 grados de libertad, por lo que necesitamos identificar al menos 5 ángulos rectos.

Como ejemplo, se muestran varios ángulos rectos identificados en un plano de la plantilla y la rectificación similar obtenida, y la rectificación del suelo de de la esquina mostrada en la página 116:

Este procedimiento se simplifica bastante si identificamos previamente el horizonte y trabajamos sobre una rectificación afín. Entonces la matriz $\mathcal{C}'$ se queda solo con 2 grados de libertad:

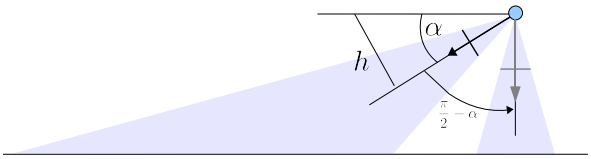
$$\mathcal{C}' = \begin{bmatrix} KK^T & 0 \\ 0 & 0 \end{bmatrix}$$

(el bloque KK^T es simétrico 2×2 y homogéneo) y puede estimarse a partir de sólo dos ángulos rectos. En este caso el componente afín K de la transformación sufrida por el plano se obtiene mediante la descomposición de *Cholesky* (ver Sec. A.5.2) de KK^T.

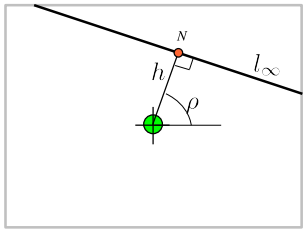
En resumen, podemos conseguir una rectificación afín si conocemos el horizonte del plano, y una rectificación similar si descubrimos 5 ángulos rectos o bien el el horizonte y dos angulos rectos.

Rectificación calibrada. Los resultados anteriores son completamente generales. Sin embargo, en la práctica suele conocerse cierta información sobre los parámetros de la cámara que pueden explotarse para conseguir una rectificación similar a partir de menos primitivas en la imagen.

Por ejemplo, si conocemos completamente la matriz de calibración podemos rectificar un plano a partir únicamente de su inclinación α respecto a la dirección del rayo principal de la cámara. La homografía de rectificación es una rotación (sintética) de la cámara para situarla frontoparalela al plano:



El ángulo α se deduce directamente de la altura del horizonte. Sean (h, ρ) las coordenadas polares del horizonte en la "imagen real" (corrigiendo previamente los pixels observados con la transformación $\mathtt{K}^{-1}$):



La siguiente rotación rectifica el plano:

$$\mathtt{H} = \mathtt{K}\mathtt{R}_2(\pi/2 - \arctan h)\mathtt{R}_3(-\rho)\mathtt{K}^{-1}$$

donde

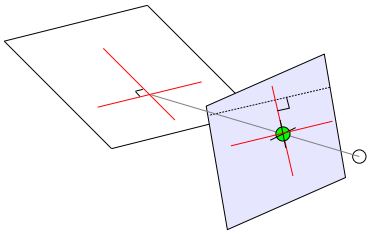
$$R_2(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \qquad R_3(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La rotación $\mathtt{R}_3$ pone el horizonte "horizontal" y a continuación $\mathtt{R}_2$ lo "sube hasta el infinito" al moverse para apuntar hacia el "suelo".

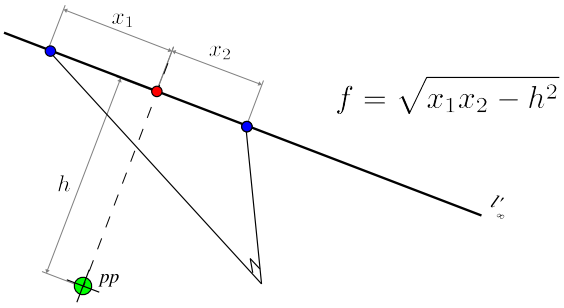
Ten en cuenta que los puntos rectificados pueden caer bastante lejos del origen de coordenadas de la imagen destino (se mueven en dirección contraria a la rotación sintética de la cámara). Suele ser conveniente desplazar la imagen resultante para que quede centrada, y escalarla al tamaño deseado.

Esta forma de rectificar es particularmente útil en la situación que veremos a continuación.

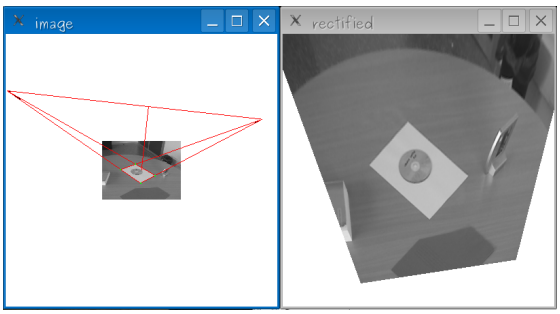
Rectificación simplificada. Si utilizamos una cámara razonable de tipo $\mathtt{K}_f$ es posible rectificar un plano a partir únicamente del horizonte y la imagen de un ángulo recto. Esto es debido a que el horizonte "induce" automáticamente la vista del ángulo recto adicional necesario para la rectificación similar, que no depende del parámetro de calibración desconocido f:



Además f puede deducirse de forma sencilla a partir de los puntos de fuga del ángulo recto, como se muestra en la siguiente construcción:



Esto significa que a partir de la vista de un rectángulo cuyos lados tienen longitudes relativas desconocidas podemos rectificar el plano y descubrir sus verdaderas proporciones. Por ejemplo, es posible distinguir si un trozo de papel tiene tamaño A4 o "letter" a partir de una única imagen del mismo:

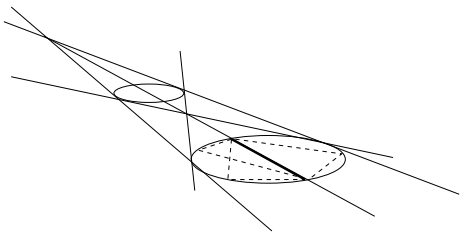


Este procedimiento se vuelve inestable si alguna de las intersecciones x_1 ó x_2 se acerca mucho al punto central N del horizonte.

Rectificación mediante círculos. Si la cámara está calibrada también es posible rectificar un plano a partir de la imagen (elíptica) de un círculo. De nuevo la idea es hacer una rotación sintética de la cámara para que mire perpendicularmente al plano, pero en este caso no se utiliza el horizonte sino la estructura de la elipse para deducir, a partir de su descomposición SVD, la rotación que la transforma en un círculo.

Como ejemplo se muestra la rectificación de un plano estimando una homografía a partir de los vértices del libro azul y se compara con la rectificación mediante la rotación que reconstruye la forma redonda del CD.

Si la cámara no está calibrada pero conocemos el centro del círculo podemos determinar un diámetro y con él un conjunto de ángulos rectos para rectificar el plano con el método basado en la imagen de $\mathcal{C}'$. Si desconocemos el centro pero tenemos otro círculo de cualquier tamaño en el mismo plano también podemos deducir un diámetro:



5.6. Homografías derivadas de una matriz de cámara

Homografía de un plano 3D. Una matriz de cámara $\mathtt{M} = \mathtt{K}[\mathtt{R}|\boldsymbol{t}]$ 'contiene' las homografías entre cualquier plano del mundo y la imagen. Por ejemplo, dado el plano Z=0, la homografía inducida tiene la expresión:

$$\mathtt{H} = \mathtt{K}[\hat{\mathtt{R}}|\boldsymbol{t}] = \mathtt{K}[\boldsymbol{r}_1|\boldsymbol{r}_2|\boldsymbol{t}]$$

donde K es la matriz de intrínsecos, $\hat{\mathtt{R}}$ es una matriz de rotación sin la tercera columna (que se hace 'invisible' porque todos los puntos del plano tienen la coordenada $Z = 0$) y $\boldsymbol{t} = -\mathtt{R}\boldsymbol{C}$ es un vector arbitrario. $\hat{\mathtt{R}}$ consiste en dos vectores columna perpendiculares y de longitud unidad.

Homografía de una rotación pura. Es fácil comprobar que la relación entre dos imágenes tomadas desde el mismo lugar (centro de proyección común, el resto de parámetros pueden variar) están relacionadas mediante una homografía plana. Supongamos que estamos en el origen y tenemos dos cámaras $\mathtt{M}_1 = \mathtt{K}[\mathtt{I}|0]$ y $\mathtt{M}_2 = \mathtt{K}[\mathtt{R}|\mathbf{0}]$. Un punto del mundo $\boldsymbol{X} = (X, Y, Z, W)^\mathsf{T}$ se ve respectivamente como $\boldsymbol{x} = \mathtt{K}(X, Y, Z)^\mathsf{T}$ y $\boldsymbol{x}' = \mathtt{KR}(X, Y, Z)^\mathsf{T}$, por lo que $\boldsymbol{x}' = \mathtt{KRK}^{-1}\boldsymbol{x}$.

Ambas vistas están relacionadas por una homografía con la estructura $\mathtt{H} = \mathtt{KRK}^{-1}$. (Los autovalores de H coinciden con los de R, por lo que de la fase de los autovalores de H en principio se puede estimar el ángulo de rotación (!), pero el cálculo es inestable para ángulos pequeños).

Mosaicos. El resultado anterior permite la fabricación sencilla de mosaicos, transformando varias vistas de una escena tomadas desde el mismo punto (pudiendo variar la orientación, zoom, etc.) a un sistema de referencia común. Como ejemplo, vamos a fundir en una panorámica tres imágenes del campus:

Hemos utilizado cuatro puntos comunes para transformar las imágenes de los extremos al sistema de referencia de la imagen central, en un espacio ampliado. La combinación de colores en cada punto se ha realizado tomando el valor máximo de cada componente R, G y B.

Homografía entre dos imágenes inducida por un plano. (Pendiente)

Ejercicios:

- Orientación a partir de la vista elíptica de un círculo.
- Calibración aproximada (p. ej., con imagen de la luna).
- Calibración lineal con plantilla 3D.
- Rectificación de planos.
- Mosaico de imágenes.

Capítulo 6

Procesamiento de Imágenes

El *procesamiento digital de imágenes* es una herramienta fundamental de la visión por computador. Uno de sus objetivos es mejorar, modificar o restaurar imágenes para uso humano. En nuestro caso estamos más interesados en automatizar, en la medida de lo posible, la detección de propiedades de una imagen que sean útiles para la interpretación de escenas o su reconstrucción tridimensional. Este campo es inmenso, por lo que en esta sección incluiremos únicamente un esquema de los temas más importantes y algunos ejemplos. Una buena referencia es el libro de Gonzalez y Woods [9]. Antes de continuar es conveniente recordar los conceptos básicos de representación frecuencial (Sección B).

6.1. Introducción al procesamiento digital de imágenes

Imágenes digitales. Elementos de imagen, discretización, representación del color, conectividad de pixels, etc.

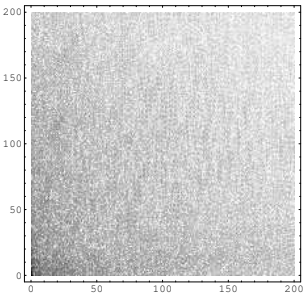
Operaciones con pixels. Operaciones punto a punto. Modificación de histograma. Operaciones pixel/entorno. Caso lineal: máscara de convolución. Morfología matemática.

Filtrado frecuencial y espacial. Representación frecuencial. Relación entre el dominio frecuencial y espacial. Filtrado inverso.

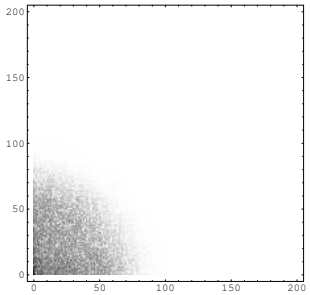
Eliminación de ruido. Media, filtrado gaussiano, propiedades, espacio de escala, filtro de mediana.

6.1.1. Ejemplos

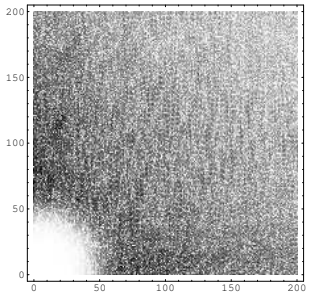
Filtrado frecuencial. En primer lugar, tomamos una imagen y la pasamos a blanco y negro. Al lado mostramos su distribución de frecuencias (DCT):

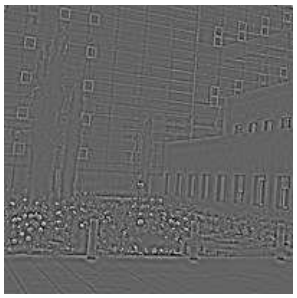


Vamos a someterla a un filtro de paso bajo. Esto lo podemos hacer tanto en el dominio espacial, utilizando una máscara de suavizado (como veremos más adelante) o alterando su distribución de frecuencias. Vamos a eliminar las frecuencias altas en todas las direcciones y a reconstruir la imagen con la transformada inversa:

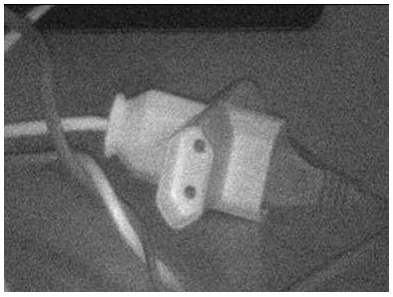


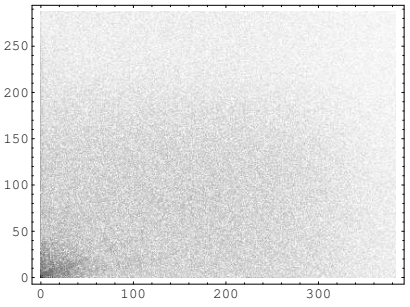
Ahora vamos a someterla a un filtro de paso alto, eliminando las frecuencias más bajas:



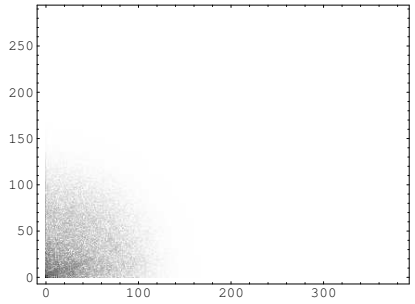


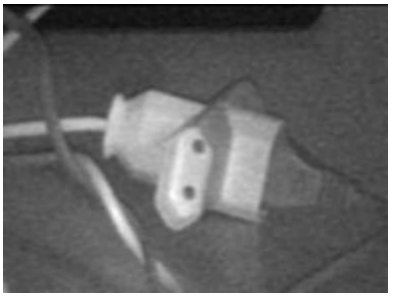
Reducción de ruido. La siguiente imagen se ha tomado en condiciones de muy baja iluminación:



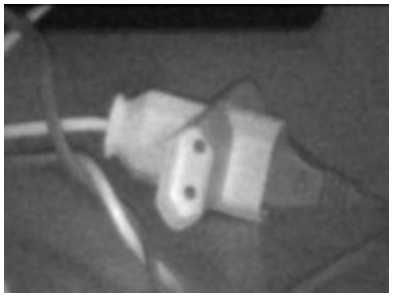


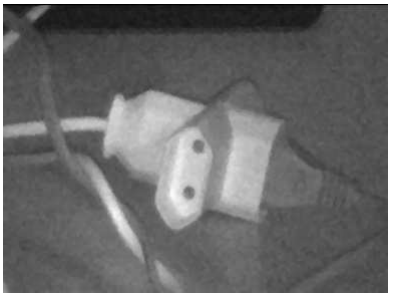
Para eliminar el ruido vamos intentar varias estrategias. En primer lugar, vamos a hacer un filtrado paso bajo similar al anterior:





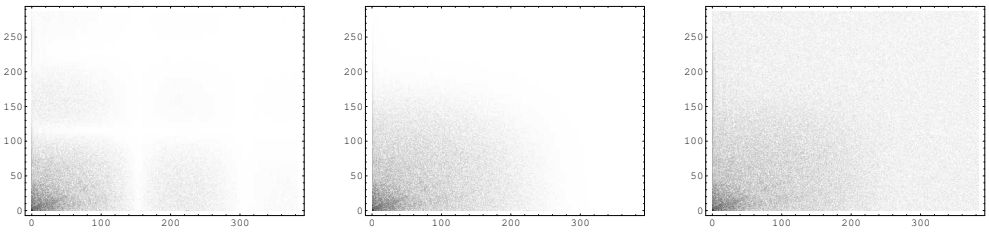
Veamos ahora el resultado de una máscara gaussiana de radio 2 (un entorno de 25 píxeles), aplicada 3 veces consecutivas, y a su lado un filtro de mediana de radio 1 (entorno de 9 píxeles) aplicada 4 veces:





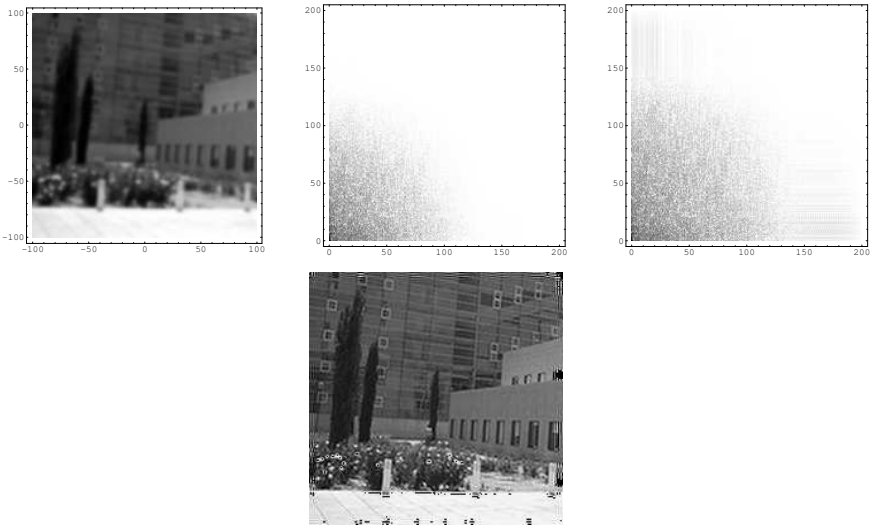
A diferencia del suavizado gaussiano, el filtro de mediana, no lineal, elimina mejor el ruido sin suavizar los detalles 'reales'.

Es interesante comparar el efecto sobre el espectro de frecuencias que tiene un filtrado de media, uno gaussiano, y uno de mediana:



Curiosamente, el filtro de media, parecido al gaussiano, no elimina todas las altas frecuencias (debido a los 'lóbulos' de su transformada de Fourier). El filtro de mediana es no lineal y por tanto no es fácil describirlo en términos de función de transferencia de frecuencias. Respeta los detalles y por tanto mantiene altas y bajas frecuencias.

Filtrado inverso. Viendo los ejemplos anteriores, surge la duda de si es posible restaurar una degradación sencilla de la imagen aplicando el filtro inverso. Como se comenta en la Sección B.13, en principio esto es posible si la degradación es moderada y completamente conocida. Por ejemplo, si sabemos que se ha producido un suavizado gaussiano (como evidencia el espectro de la imagen) podemos utilizar la inversa de su función de transferencia (bloqueando de alguna manera las divisiones por valores próximos a cero) para restaurar la imagen:



Desafortunadamente, resultados tan buenos como el anterior son difíciles de conseguir cuando la degradación sufrida por la imagen es desconocida.

Ejercicios:

- Encontrar componentes conexas.

- Filtrado de imágenes con diferentes máscaras de convolución.
- Suavizado gaussiano.
- Filtro de Mediana.

6.2. Propiedades de Bajo y Medio nivel

Salvo casos muy especiales no podemos trabajar con las imágenes en bloque, consideradas como 'un todo'. Necesitamos obtener automáticamente propiedades más o menos llamativas que nos permitan interpretar o reconstruir la estructura tridimensional de una escena. Las propiedades más simples son los elementos de borde: los pixels en donde el color de la imagen cambia significativamente. Entre ellos, los puntos calientes o 'esquinas' son especialmente importantes para establecer correspondencias en varias imágenes. Las líneas rectas también son muy útiles por su propiedad de invarianza proyectiva, igual que las cónicas. En muchas ocasiones debemos capturar regiones de la imagen de forma arbitraria que tienen alguna propiedad común, como la textura o el color. Una referencia excelente es el libro de Trucco y Verri. El esquema que seguiremos es el siguiente:

Detección de bordes. Filtros de paso alto. Estimación del gradiente. Necesidad de regularización (suavizado).

Operador de **Marr** (interés histórico). Buscamos extremos del módulo del gradiente mediante cruces por cero de la derivada segunda (laplaciano). Un suavizado previo gaussiano se puede hacer automáticamente: en lugar de calcular $\nabla^2(G * I)$ calculamos $(\nabla^2 G) * I$. El laplaciano de la gaussiana es la función *sombrero mexicano*. (En la sección D.1 se revisan los operadores de derivación multivariable).

Operador de **Canny**. Es el método más práctico para obtener bordes. La idea es buscar máximos locales de la *magnitud* de gradiente en la *dirección* del gradiente. (Mejor que usar el laplaciano, que mezcla las dos dimensiones y por tanto pierde información de la orientación del borde.) Los puntos seleccionados (*edgels*) se pueden enganchar para formar contornos conectados. Se utiliza un umbralizado con 'histéresis': se aceptan los candidatos que superen un umbral máximo y todos aquellos que superen un umbral mínimo y estén conectados a un punto aceptado.

Detección de esquinas. Un buen método (**Harris**) para detectar puntos 'importantes' en la imagen (esquinas o *corners*) está basado en el menor autovalor de la matriz de covarianza de la distribución local del gradiente.

Detección de líneas rectas y segmentos. Un método muy utilizado para encontrar estructuras parametrizadas (rectas, círculos, etc.) en las imágenes es la **Transformada de Hough**. La idea es discretizar el espacio de parámetros con un array de contadores.

Y cada punto de borde 'vota' a todos los modelos que pasan por él. Esto es viable cuando el número de parámetros es pequeño, típicamente rectas, con sólo dos parámetros, preferiblemente expresadas en forma polar.

En el caso de rectas, se pueden estimar sus direcciones mediante la dirección del gradiente en cada punto, lo que permite generar un sólo voto en lugar de una trayectoria. No hacen falta contadores, ni discretizar el espacio de parámetros, pero como contrapartida es necesario un análisis de agrupamientos de los modelos candidatos.

En realidad interesa más encontrar segmentos concretos, con extremos, que rectas infinitas. Una posibilidad es 'llevar la cuenta' de los puntos que han generado cada recta candidata, y analizar la distribución de proyecciones sobre ella.

Ajuste de cónicas. El método más sencillo para estimar la cónica $x^T C x = 0$ que pasa por un conjunto de puntos es resolver un sistema lineal homogéneo sobredeterminado (Sección D.2.1). Un caso muy frecuente es estimar la ecuación de una elipse que es la vista en perspectiva de un círculo. Aunque este tipo de estimación lineal no puede forzar el tipo de cónica (elipse, círculo, hipérbola o parábola) del resultado, en la mayoría de los casos, y normalizando los datos, el resultado es satisfactorio. Si los puntos están mal repartidos por la elipse (p.ej. solo vemos un trozo) o hay mucho ruido es necesario usar técnicas iterativas basadas en distancia geométrica.

Los puntos, las líneas rectas o los segmentos son propiedades adecuadas para la reconstrucción tridimensional de la escena mediante geometría proyectiva. La sección siguiente presentará una breve introducción de este enfoque de la visión artificial. Pero antes vamos a comentar brevemente otros métodos típicos para obtener representaciones de 'medio nivel' de la imagen.

Segmentación de regiones por color o nivel de gris. En situaciones controladas (p. ej., ambientes industriales, cintas transportadoras, etc.) puede ocurrir que el color (o nivel de gris) del 'fondo' sea diferente del de los objetos de interés, lo que permitiría separarlos mediante una simple unbralización. Un método interesante para obtener automáticamente los umbrales de decisión está basado en caracterizar el histograma de la imagen mediante un modelo de mezcla usando el algoritmo EM (Sección 3.4.3). Usándolo en modo *on-line* el umbral puede adaptarse dinámicamente a los cambios de iluminación.

La extracción de contornos no es complicada: una idea es moverse hasta un punto de borde y desde allí recorrer el borde con una máquina de estados que indica, dependiendo de la dirección del movimiento, en qué pixels hay que comprobar si hay figura o fondo para continuar el camino.

Representación de contornos. Las manchas de color (o textura) homogénea se representan de manera concisa mediante contornos (secuencias de puntos). Aunque se han propuesto métodos más económicos (códigos de diferencias) o representaciones

alternativas con determinadas propiedades de invarianza (coordenadas polares desde el centro, pendiente en función del ángulo, etc.), la representación como *polilínea* permite obtener cualquier propiedad geométrica de la figura sin excesivo coste computacional.

A veces interesa reducir el tamaño de un contorno, eliminando puntos alineados, y convirtíendolo en una polilínea con una forma aproximadamente igual a la del contorno original. El algoritmo **IPE** (*Iterative Point Elimination*) resuelve este problema. La idea es calcular el área de error del triángulo (perdido o ganado) al eliminar cada uno de los puntos del contorno. Se elimina el punto que genere menos error, se recalculan sus dos vecinos, y se repite el proceso. Cuando la eliminación de cualquier punto produzca un triángulo de error inaceptable detenemos el proceso.

Contornos deformables. Son una alternativa para obtener contornos de trozos interesantes de la imagen. La idea es modelar una especie de goma elástica –en realidad una polilínea– cada uno de cuyos 'nudos' tiende a moverse hacia configuraciones de baja 'energía'. La energía se define de manera que a) el contorno final tienda a reducirse y a mantener los nudos regularmente espaciados, b) tenga en general poca curvatura (excepto en verdaderos 'picos' del contorno, donde entonces la curvatura no se penaliza), y c) tienda a pegarse a zonas de la imagen con valor elevado de la magnitud del gradiente. Un método 'voraz' que prueba la energía de cada nudo en un pequeño entorno puede servir ajustar el contorno desde una posición cercana al trozo de imagen que nos interesa. (La energía producida por la imagen podría adaptarse a las necesidades de cada aplicación concreta.)

Textura. La textura es una propiedad difícil de definir. Tiene que ver con propiedades repetitivas de la imagen. Estas propiedades pueden ser muy abstractas, por lo que es difícil encontrar métodos generales de tratamiento de texturas: (p. ej., analizando la variación de la excentricidad de elipses se puede, en principio, deducir la inclinación de un plano lleno de círculos; análizando la distribución de ángulos se puede determinar la inclinación de un plano en el que se han dejado caer un monton de alfileres, etc.) Un enfoque clásico ha sido de tipo 'gramatical', donde se supone que las texturas están compuestas por primitivas o *textones*, que se distribuyen por la imagen siguiendo reglas de un cierto lenguaje formal. Este enfoque no es muy viable en la práctica.

Una alternativa es intentar encontrar propiedades estadísticas de la imagen que den alguna idea de las configuraciones frecuentes e infrecuentes de unos pixels en relación con otros. Una idea inicial muy simple es analizar propiedades del histograma de las regiones de interés de la imagen: p. ej., la varianza nos da idea de la 'rugosidad' de la imagen, y otras propiedades como la asimetría o el afilamiento del histograma podrían tal vez ser útiles. Pero estas medidas pierden toda la información de las relaciones espaciales de los pixels. Es mejor intentar analizar la densidad conjunta de cada pixel p y su vecino q en una posición relativa dada, mediante lo que se conoce

como matriz de *co-ocurrencia*. Dada una cierta textura, las matrices de co-ocurrencia de algunas posiciones relativas contendrán valores significativos.

Otra posibilidad interesante es realizar un análisis frecuencial de la imagen. A partir de la transformada de Fourier 2D, expresada en coordenadas polares, podemos acumular para cada dirección toda la potencia espectral y obtener una especie de signatura unidimensional, invariante a rotación, que proporcionará cierta información útil sobre la textura (véase Gonzalez y Woods).

"X **a partir de** Y". Se han propuesto muchos métodos para inferir propiedad de interés del entorno (forma, orientación, estructura, etc.) a partir de determinados aspectos de las imágenes (sombreado, estéreo, movimiento, textura, dibujos de líneas, etc.) La mayoría de ellos tienen un interés fundamentalmente teórico, ya que están basados en suposiciones y simplificaciones que no siempre se verifican en la práctica.

Ejercicios:

- Representar una imagen mediante una lista de secuencias de *edgels* conectados usando el algoritmo de Canny.
- Calcular puntos calientes (esquinas).
- Encontrar líneas rectas mediante la transformada de Hough (se puede aprovechar la orientación local de los *edgels*). Encontrar segmentos acotados.
- Experimentar con contornos deformables (*snakes*).

6.3. Reconocimiento e Interpretación

Una vez extraídas de la imagen un conjunto prometedor de propiedades geométricas, el siguiente paso es interpretarlas dentro de un modelo coherente. Esta etapa depende muchísimo de la aplicación concreta, y es lo que distingue realmente a la visión por computador del procesamiento de imágenes. Veamos algunas ideas.

Eigendescomposición

Alineamiento

Invariantes proyectivos

Interpretación de dibujos de línea. Graph Matching

Capítulo 7

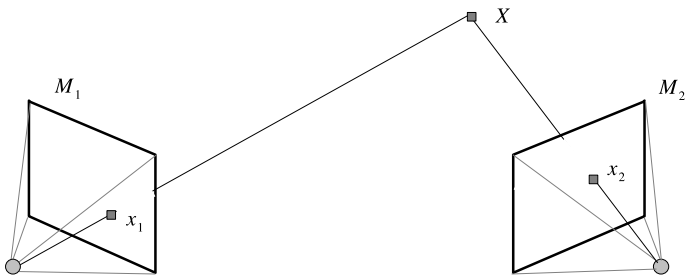
Reconstrucción 3D

Nuestro objetivo es obtener un modelo tridimensional de una escena a partir de dos o más imágenes. La idea ingenua inicial sería utilizar cámaras completamente calibradas (mediante p.ej. una plantilla) y posteriormente hacer triangulación de los rayos ópticos de puntos correspondientes en varias imágenes. Por supuesto, este procedimiento es perfectamente válido. Sin embargo, podemos aplicar de nuevo la geometría proyectiva para atacar el problema de manera más elegante. Descubriremos que las propias imágenes contienen implícitamente información sobre la posición relativa y los parámetros de calibración de las cámaras.

Este apartado contiene solamente un breve guión de los conceptos básicos. La referencia fundamental es el libro de Hartley y Zisserman [10]. También es muy recomendable el tutorial de Pollefeys [21].

7.1. Triangulación

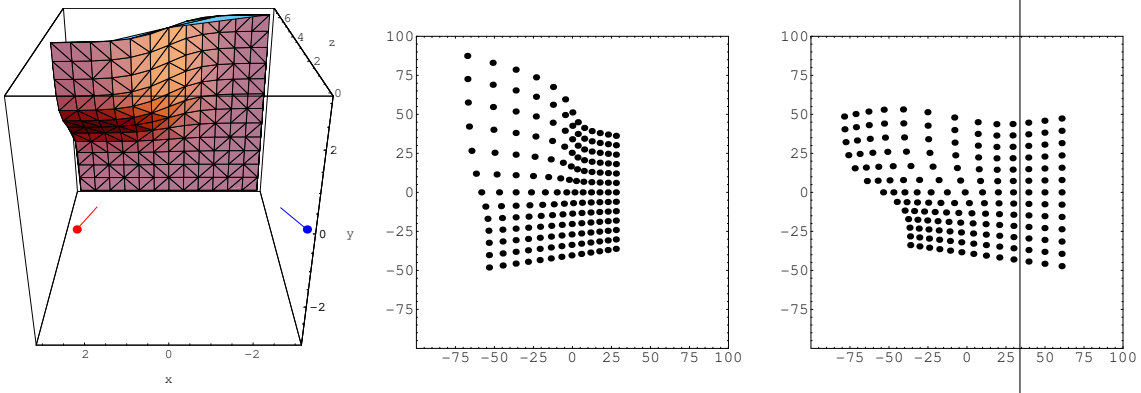
Comenzaremos estudiando la geometría de dos vistas. Consideremos dos cámaras situadas en cualquier posición y orientación observando una escena. Fijadas y conocidas las dos matrices de cámara globales M_1 y M_2 (ver Sec. 5.3), un punto 3D $\boldsymbol{X}$ determina perfectamente las imágenes izquierda $\boldsymbol{x}_1$ y derecha $\boldsymbol{x}_2$:



$$\boldsymbol{x}_1 = \mathtt{M}_1 \boldsymbol{X}$$

$$\boldsymbol{x}_2 = \mathtt{M}_2 \boldsymbol{X}$$

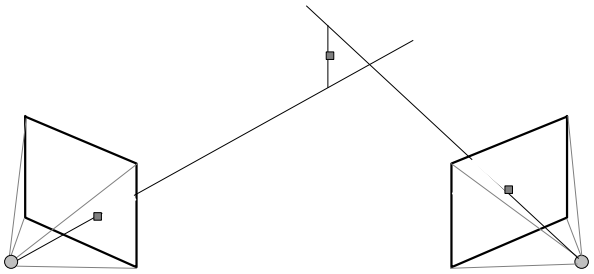
La figura siguiente muestra una escena 3D sintética observada por dos cámaras. A la derecha tenemos las imágenes tomadas por las cámaras, en las que sólo mostramos los puntos esquina (*corners*) detectados:



Para comprobaciones posteriores, las matrices de cámara utilizadas son las siguientes (están normalizadas de manera que la submatriz 3×3 izquierda tenga determinante unidad):

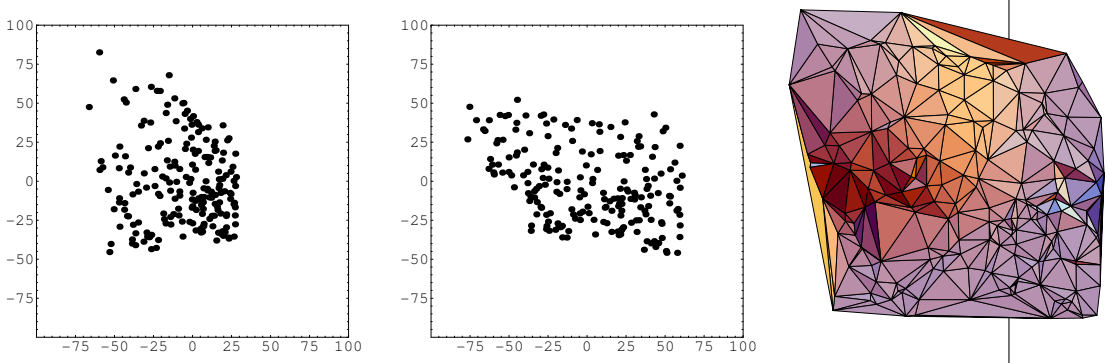
$$\mathtt{M}_1 = \begin{bmatrix} 4.36 & 0. & 1.59 & -8.72 \\ 0. & 4.64 & 0. & 0. \\ -0.02 & 0. & 0.04 & 0.03 \end{bmatrix} \qquad \mathtt{M}_2 = \begin{bmatrix} 5.01 & 0. & -2.34 & 15.03 \\ 0. & 4.25 & 0. & 0. \\ 0.02 & 0. & 0.04 & 0.05 \end{bmatrix}$$

Recíprocamente, conocidas las cámaras, las imágenes izquierda y derecha determinan perfectamente el punto 3D que las ha producido. Hay que tener en cuenta que debido a los inevitables errores de medida y a discretización de la malla de pixels, los rayos correspondientes a cada cámara no se cortarán exactamente en el espacio, por lo que nos conformaremos con el punto 3D que está más cerca de ambos rayos:



Normalmente los puntos que detectaremos no se encontrarán organizados en una malla regular, sino repartidos más o menos aleatoriamente sobre la escena.

Y los errores de localización de los puntos de interés producirán errores en la reconstrucción. Para visualizar un poco mejor la superficie reconstruida es conveniente unir los puntos 3D mediante una estructura de polígonos basada en el grafo de *Delaunay* (sobre la que se podría incluso 'mapear' la textura observada):

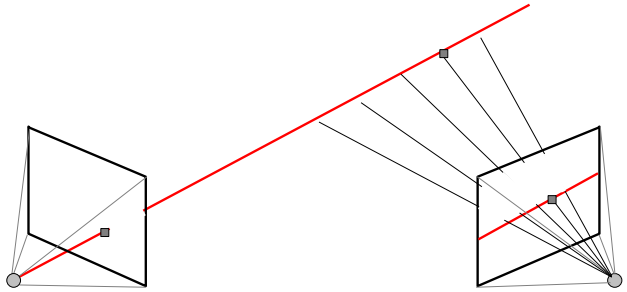


Un método sencillo de triangulación consiste en resolver el sistema lineal homogéneo sobredeterminado $\boldsymbol{x}_1 = \mathtt{M}_1\boldsymbol{X}$ y $\boldsymbol{x}_2 = \mathtt{M}_2\boldsymbol{X}$, donde las incógnitas son las 4 coordenadas homogéneas de cada punto 3D $\boldsymbol{X}$.

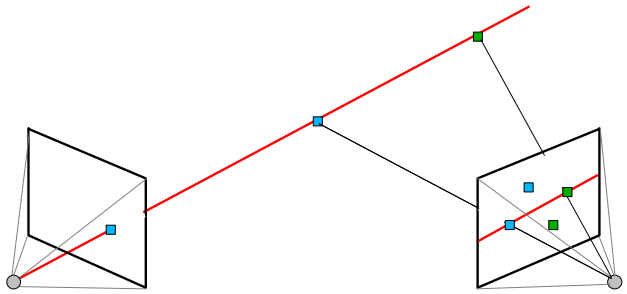
7.2. Restricción *Epipolar*

Para hallar la posición 3D de ciertos puntos de una imagen necesitamos verlos en otra y estar seguros de que se corresponden los unos con los otros. En general, este problema es bastante complicado, a menos que el entorno de cada punto tenga una distribución de colores característica y única en la imagen. La *restricción epipolar* resulta de gran ayuda para resolver este problema.

Supongamos que la cámara izquierda ve un punto del mundo 3D. En realidad ese punto de la imagen puede haber sido el resultado de la proyección de cualquier otro punto del espacio que se encuentre en el mismo rayo óptico (ver Sec. 5.3). Imaginemos que ese rayo es 'visible'. Su imagen en la cámara derecha es una línea recta; la vista lateral del rayo permite determinar la 'profundidad' de cada punto:



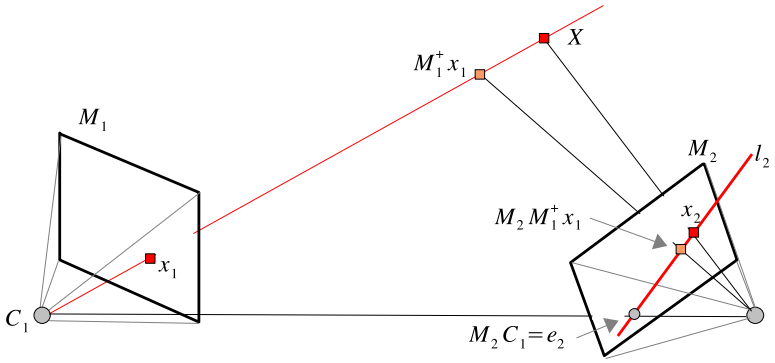
Por tanto, conocidas las cámaras $\mathtt{M}_1$ y $\mathtt{M}_2$, dada la imagen izquierda $\boldsymbol{x}_1$ de un punto 3D $\boldsymbol{X}$, la imagen derecha $\boldsymbol{x}_2$ no puede estar en cualquier sitio. Debe encontrarse necesariamente a lo largo de una línea recta, llamada *recta epipolar*, que es la imagen del rayo óptico. Podemos descartar correspondencias candidatas que se encuentren fuera de la recta epipolar correspondiente:



Aún así, existe ambigüedad, que podría reducirse con imágenes adicionales u otros métodos.

7.2.1. Matriz Fundamental.

¿Cuál es la recta epipolar $\boldsymbol{l}_2$ correspondiente al punto $\boldsymbol{x}_1$? Es muy fácil obtenerla porque pasa por la imagen de dos puntos conocidos. Uno de ellos es el centro de proyección $\boldsymbol{C}_1$ de la primera cámara (el espacio nulo de $\boldsymbol{M}_1$, cumple $\mathtt{M}_1\boldsymbol{C}_1 = \boldsymbol{0}$). Su imagen, $\boldsymbol{e}_2 = \mathtt{M}_2\boldsymbol{C}_1$, se llama *epipolo*. Otro punto de la recta epipolar es la imagen de cualquier punto del rayo óptico, p. ej., $\mathtt{M}_1^+\boldsymbol{x}_1$ (con $\lambda = 0$, ver Sección 5.3). Su imagen es $\mathtt{M}_2\mathtt{M}_1^+\boldsymbol{x}_1$.



Gracias a la geometría proyectiva y a las coordenadas homogéneas, la recta $\boldsymbol{l}_2$ que pasa por esos dos puntos se consigue de forma inmediata con su producto vectorial (ver Sec. 5.2):

$$\boldsymbol{l}_2 = \boldsymbol{e}_2 \times \mathtt{M}_2\mathtt{M}_1^+ \boldsymbol{x}_1$$

El punto $\boldsymbol{x}_2$ correspondiente a $\boldsymbol{x}_1$ tiene que estar sobre su recta epipolar y por tanto debe cumplir $\boldsymbol{x}_2^\top \boldsymbol{l}_2 = 0$. Ahora vamos a expresar esta condición de manera más compacta:

$$\boldsymbol{x}_2^\top \boldsymbol{l}_2 = \boldsymbol{x}_2^\top \underbrace{\boldsymbol{e}_2 \times \mathtt{M}_2\mathtt{M}_1^+}_{\mathtt{F}} \boldsymbol{x}_1 = 0$$

F es la *matriz fundamental*. Genera las líneas epipolares correspondientes a cualquier punto de la imagen:

$$\boxed{\boldsymbol{x}_2^\top \ \mathtt{F} \ \boldsymbol{x}_1 = 0}$$

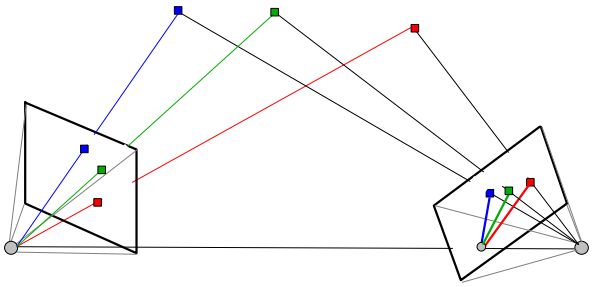
La matriz fundamental sólo depende de la configuración de las cámaras y no de los puntos 3D de la escena. Puede escribirse como:

$$\mathtt{F} = [\boldsymbol{e}_2]_\times \mathtt{M}_2\mathtt{M}_1^+$$

donde la notación $[\boldsymbol{v}]_\times$ denota una matriz antisimétrica que sirve para calcular productos vectoriales en forma de producto matricial: $[\boldsymbol{v}]_\times \boldsymbol{x} = \boldsymbol{v} \times \boldsymbol{x}$. En concreto:

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix}_\times = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}$$

El epipolo se puede extraer directamente de la propia F. Como todos los rayos salen del centro de proyección, todas las líneas epipolares,'salen' del epipolo. El punto $\boldsymbol{x}_2 = \boldsymbol{e}_2$ está en todas las líneas epipolares:



Para todo $\boldsymbol{x}_1$ se cumplirá $\boldsymbol{e}_2^\mathsf{T}\mathtt{F}\boldsymbol{x}_1 = 0$, o sea $\boldsymbol{e}_2^\mathsf{T}\mathtt{F} = \mathbf{0}^\mathsf{T}$. Por tanto el epipolo $\boldsymbol{e}_2$ es el espacio nulo de $\mathtt{F}^\mathsf{T}$.

La restricción epipolar es simétrica; todos los resultados que hemos dado para la imagen derecha son válidos para la imagen izquierda cambiando $\mathtt{F}$ por $\mathtt{F}^\mathsf{T}$.

La matriz fundamental de la configuración de cámaras del ejemplo sintético anterior, calculada con las matrices de cámara 'reales' $\mathtt{M}_1$ y $\mathtt{M}_2$ es:

$$\mathtt{F} = \begin{bmatrix} 0. & -0.00359 & 0. \\ -0.00377 & 0. & -1.03684 \\ 0. & 1. & 0. \end{bmatrix}$$

$\rightarrow$ epipolo y dibujarlo

7.2.2. Matriz Esencial

Veamos qué forma tiene la matriz fundamental de dos cámaras cuyos parámetros internos y externos son conocidos (ver secs. 5.3 y 5.4). Para simplificar la notación trabajaremos en el sistema de coordenadas de la cámara izquierda, de manera que $\mathtt{R}$ y $\boldsymbol{C}$ son la orientación y posición relativa de $\mathtt{M}_2$ respecto de $\mathtt{M}_1$:

$$\mathtt{M}_1 = \mathtt{K}_1[\mathtt{I}|\mathbf{0}] \qquad\qquad \mathtt{M}_2 = \mathtt{K}_2[\mathtt{R}|-\mathtt{R}\boldsymbol{C}_2] = [\mathtt{K}_2\mathtt{R}|\mathtt{K}_2\boldsymbol{t}]$$

El centro de $\mathtt{M}_1$ es $(0,0,0,1)^\mathsf{T}$, por tanto el epipolo es $\boldsymbol{e}_2 = \mathtt{K}_2\boldsymbol{t}$ (donde $\boldsymbol{t} = -\mathtt{R}\boldsymbol{C}$). Es fácil comprobar que $\mathtt{M}_1^+ = [\mathtt{I}|\mathbf{0}]^\mathsf{T}\mathtt{K}_1^{-1}$ por lo que $\mathtt{M}_2\mathtt{M}_1^+ = \mathtt{K}_2\mathtt{R}\mathtt{K}_1^{-1}$. Entonces:

$$\mathtt{F} = [\mathtt{K}_2\boldsymbol{t}]_\times \mathtt{K}_2\mathtt{R}\mathtt{K}_1^{-1}$$

Esta expresión (que necesitaremos más adelante) se puede arreglar un poco si somos capaces de conmutar el operador $[\cdot]_\times$ con una matriz no singular $\mathtt{A}$. La forma de hacerlo es la siguiente: $[\boldsymbol{v}]_\times\mathtt{A} = \mathtt{A}^{-\mathsf{T}}[\mathtt{A}^{-1}\boldsymbol{v}]_\times$. Usando esta propiedad $\mathtt{F} = \mathtt{K}_2^{-\mathsf{T}}[\mathtt{K}_2^{-1}\mathtt{K}_2\boldsymbol{t}]_\times\mathtt{R}\mathtt{K}_1^{-1} = \mathtt{K}_2^{-\mathsf{T}}\,[\boldsymbol{t}]_\times\mathtt{R}\,\mathtt{K}_1^{-1} = \mathtt{K}_2^{-\mathsf{T}}\mathtt{E}\,\mathtt{K}_1^{-1}$.

La matriz $\mathtt{E} = [\boldsymbol{t}]_\times \mathtt{R}$ se llama *matriz esencial* de la configuración de cámaras. Es la restricción epipolar de puntos del plano de imagen real, en coordenadas 'normalizadas', a los que se ha deshecho la transformación $\mathtt{K}$ sobre la la malla de pixels. Si $\boldsymbol{x}_1^n = \mathtt{K}_1^{-1}\boldsymbol{x}_1$ y $\boldsymbol{x}_2^n = \mathtt{K}_2^{-1}\boldsymbol{x}_2$ entonces:

$$\boxed{\boldsymbol{x_2^n}^\mathsf{T}\ \mathtt{E}\ \boldsymbol{x}_1^n = 0}$$

> Se llega a la misma conclusión si trabajamos inicialmente con coordenadas normalizadas. En este caso las cámaras son $\mathtt{M}_1 = [\mathtt{I}|\mathbf{0}]$ y $\mathtt{M}_2 = [\mathtt{R}|\boldsymbol{t}]$, el epipolo es $\boldsymbol{t}$ y $\mathtt{M}_2\mathtt{M}_1^+ = \mathtt{R}$ y la matriz fundamental en coordenadas normalizadas se reduce a $[\boldsymbol{t}]_\times\mathtt{R}$.

La matriz esencial $\mathtt{E}$ solo depende de la colocación relativa ($\mathtt{R}$ y $\boldsymbol{t} = -\mathtt{R}\boldsymbol{C}$) de las cámaras.

$\rightarrow$ mat E del ejemplo

7.3. Reconstrucción 3D

La verdadera importancia de la restricción epipolar no se debe a su capacidad de descartar algunas falsas correspondencias de puntos dadas dos cámaras perfectamente conocidas. En realidad se debe a que puede utilizarse en 'sentido contrario' para restringir la posición de las cámaras a partir de un conjunto de puntos correspondientes en las dos imágenes. El hecho de que el mundo sea 'rígido' impone restricciones a las imágenes que permiten deducir (casi completamente) desde donde, y con qué tipo de cámara, se han tomado las imágenes. En principio esto permite hacer triangulación y obtener los puntos 3D.

Observa que un punto 3D tiene 3 grados de libertad. Una pareja de puntos correspondientes en dos imágenes tiene cuatro grados de libertad. Este grado de libertad adicional es el que impone una restricción a las cámaras. Si tuviéramos 3 cámaras, cada punto 3D estaría sobredeterminado por 6 grados de libertad (2 en cada imagen), y nos sobrarían 3 grados de libertad para imponer restricciones, en este caso sobre las 3 matrices de cámara. Y así sucesivamente…

7.3.1. Reconstrucción Proyectiva

Sin embargo, queda una fuente de ambigüedad. De acuerdo con el *teorema fundamental de la reconstrucción proyectiva*, a partir simplemente de puntos correspondientes tomados de dos (o más) cámaras situadas en cualquier posición y orientación, y con cualesquiera parámetros internos, podemos recuperar su estructura tridimensional *salvo una transformación proyectiva 3D*. Por ejemplo, en el caso de dos vistas, a partir de un conjunto de parejas de puntos correspondientes $\boldsymbol{x}_1 \leftrightarrow \boldsymbol{x}_2$ podemos

estimar directamente la matriz fundamental $\mathtt{F}$ (esencialmente resolviendo un sistema lineal homogéneo, aunque con algún matiz adicional que explicaremos enseguida). El problema es que aunque $\mathtt{F}$ queda completamente determinada por la pareja de cámaras $\mathtt{M}_1$ y $\mathtt{M}_2$, los puntos observados son consistentes también con otras cámaras $\mathtt{P}_1$ y $\mathtt{P}_2$ que ven una escena transformada:

$$\boldsymbol{x}_1 = \mathtt{M}_1\boldsymbol{X} = \underbrace{\mathtt{M}_1\mathtt{D}}_{\mathtt{P}_1}\underbrace{\mathtt{D}^{-1}\boldsymbol{X}}_{\boldsymbol{X}'} = \mathtt{P}_1\boldsymbol{X}' \qquad \boldsymbol{x}_2 = \mathtt{M}_2\boldsymbol{X} = \underbrace{\mathtt{M}_2\mathtt{D}}_{\mathtt{P}_2}\underbrace{\mathtt{D}^{-1}\boldsymbol{X}}_{\boldsymbol{X}'} = \mathtt{P}_2\boldsymbol{X}'$$

Hay infinitas parejas de cámaras $\mathtt{P}_1$ y $\mathtt{P}_2$ consistentes con una matriz fundamental dada $\mathtt{F}$; aunque encontremos una de ellas (lo que es relativamente sencillo) quedaría por determinar la transformación proyectiva tridimensional $\mathtt{D}$ (representada por una matriz homogénea 4×4, con 15 grados de libertad) que han sufrido, compensándose mutuamente, las cámaras originales y todos los puntos de escena real.

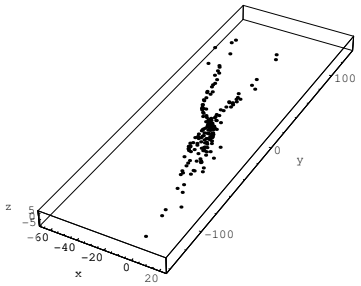
→ ¿figura?

Por supuesto, en la práctica las cámaras son desconocidas, por lo que tendremos que estimar $\mathtt{F}$ a partir únicamente de parejas de puntos correspondientes en las dos imágenes (suponiendo que fuéramos capaces de detectarlas).

> El algoritmo de estimación de $\mathtt{F}$ a partir de correspondencias $\boldsymbol{x}_1 \leftrightarrow \boldsymbol{x}_2$ se basa en resolver un sistema de ecuaciones lineales homogéneo en los 9 elementos de $\mathtt{F}$ (8 ratios independientes), donde cada correspondencia $\boldsymbol{x}_2^\intercal \mathtt{F} \boldsymbol{x}_1$ impone una restricción. Con 8 correspondencias podríamos estimar $\mathtt{F}$. En realidad esta matriz tiene solo 7 grados de libertad, ya que por construcción debe tener rango 2 (es el producto de $[\boldsymbol{e}_2]_\times$, que tiene rango 2, y otra matriz de rango 3). Es importante imponer esta restricción (ver Sec. A.4) para que $\mathtt{F}$ corresponda realmente a una configuración de dos cámaras (de modo que todas las líneas epipolares realmente coincidan en el epipolo). También es necesario normalizar las coordenadas de los puntos de imagen para que el sistema lineal sea más estable.
>
> → explicar normalización
>
> Para que el ejemplo sea realista estimamos la matriz fundamental con los puntos ligeramentes contaminados con ruido. Obtenemos una aproximación que parece bastante aceptable:
>
> $$\hat{\mathtt{F}} = \begin{bmatrix} -0.00002 & -0.00363 & -0.00161 \\ -0.00368 & 0.00003 & -1.03586 \\ 0.00351 & 1. & -0.01328 \end{bmatrix}$$
>
> De ella podemos extraer una pareja de cámaras consistente:
>
> $$\mathtt{P}_1 = \begin{bmatrix} 1. & 0. & 0. & 0. \\ 0. & 1. & 0. & 0. \\ 0. & 0. & 1. & 0. \end{bmatrix} \qquad \mathtt{P}_2 = \begin{bmatrix} -0.00005 & 0.01136 & -0.02678 & 275.51221 \\ 0.02486 & 7.08364 & -0.09402 & -0.44200 \\ 0.02605 & -0.00017 & 7.33757 & 1.00000 \end{bmatrix}$$

→ explicar cómo

Con estas cámaras podemos hacer triangulación y conseguir puntos 3D. Desafortunadamente, en este caso la deformación proyectiva es tan extrema que difícilmente podemos ver ninguna propiedad de la escena. Ni siquiera merece la pena construir la superficie poligonal.



→ ransac para outliers

Esta ambigüedad no se puede eliminar aunque aumentemos el número de vistas de la escena, si se han tomado con cámaras completamente arbitrarias. Pero en la práctica siempre se tiene alguna información, aunque sea mínima, sobre los parámetros internos de las cámaras, lo que en principio permite aplicar técnicas de autocalibración (ver Sec. 7.4).

7.3.2. Reconstrucción Calibrada

Si conocemos completamente las matrices de cámara $\mathtt{M}_1$ y $\mathtt{M}_2$, a partir de puntos correspondientes en las dos imágenes $\boldsymbol{x}_1 \leftrightarrow \boldsymbol{x}_2$ es inmediato reconstruir las posiciones 3D reales mediante triangulación (ver Sec. 7.1).

Supongamos que solo conocemos las matrices de calibración interna $\mathtt{K}_1$ y $\mathtt{K}_2$ de las dos cámaras, pero no su localización espacial. A partir de puntos correspondientes normalizados podemos estimar la matriz esencial $\mathtt{E}$ de la configuración de cámaras, a partir de la cual es posible deducir la rotación $\mathtt{R}$ y la (dirección de la) traslacion $\boldsymbol{t}$ de una cámara respecto a la otra y reconstruir un modelo a escala de la escena, que por lo demás es correcto.

→ R y t a partir de E

→ ambigüedad de escala

→ grados de libertad y arreglo de E

Continuando con el ejemplo sintético anterior, si por algún procedimiento alternativo (calibración con una plantilla, etc.) conseguimos los parámetros internos de las cámaras:

$$\mathtt{K}_1 = \begin{bmatrix} 100. & 0. & 0. \\ 0. & 100. & 0. \\ 0. & 0. & 1. \end{bmatrix} \qquad \mathtt{K}_2 = \begin{bmatrix} 130. & 0. & 0. \\ 0. & 100. & 0. \\ 0. & 0. & 1. \end{bmatrix}$$

entonces podemos estimar la matriz esencial:

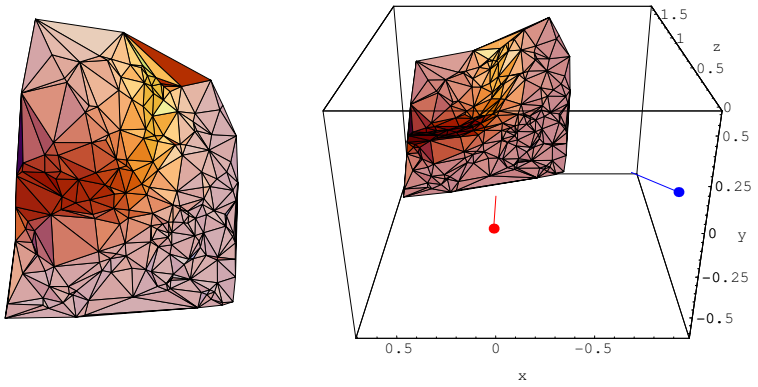
$$\hat{\mathtt{E}} = \mathtt{K}_2^{\mathsf{T}}\, \hat{\mathtt{F}}\, \mathtt{K}_1 = \begin{bmatrix} -3.01383 & -876.84446 & -3.58301 \\ -702.87350 & -5.61589 & -1945.09730 \\ 4.04444 & 1878.58316 & 1. \end{bmatrix}$$

Una verdadera matriz esencial, de la forma $[\boldsymbol{t}]_\times \mathtt{R}$, es de rango 2 y tiene los dos valores singulares distintos de cero iguales. Nuestra estimación se aproxima bastante a esta condición, ya que sus valores singulares son $\{2076.06, 2065.29\}$.

El par de cámaras consistente con esta matriz esencial son (de las cuatro opciones mostramos las que reconstruyen por delante):

→ mostrarlas

Haciendo triangulación con ellas conseguimos la siguiente reconstrucción 3D, que es bastante aceptable:

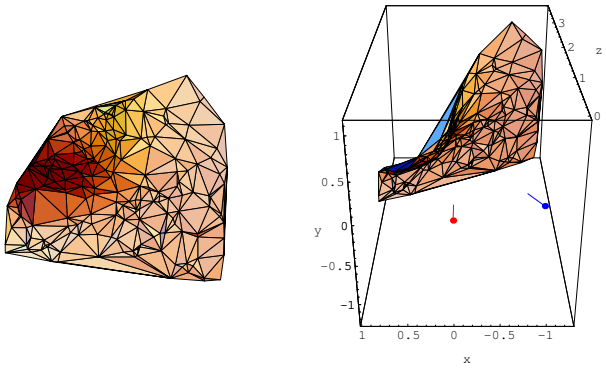


Observa que la superficie de profundidad se expresa desde el punto de vista de la cámara izquierda, que se sitúa en el origen y apuntando en la dirección del eje z, por lo que la escena se muestra inclinada respecto al sistema de referencia mostrado en la sec. 7.1). La escala se expresa en unidades de separación entre las dos cámaras ($||\boldsymbol{t}|| = 1$), que en este ejemplo suponemos desconocida. Se han utilizado los parámetros intrínsecos correctos por lo que la reconstrucción es correcta salvo escala.

Si hubiéramos utilizado unas matrices de cámara incorrectas, p. ej.:

$$\mathtt{K}_1' = \begin{bmatrix} 80. & 0. & 0. \\ 0. & 80. & 0. \\ 0. & 0. & 1. \end{bmatrix} \qquad \mathtt{K}_2' = \begin{bmatrix} 80. & 0. & 0. \\ 0. & 140. & 0. \\ 0. & 0. & 1. \end{bmatrix}$$

no conseguiríamos eliminar toda la deformación proyectiva de la escena (la correspondencia entre ángulos y pixels es errónea), aunque por lo menos conseguimos hacernos una idea de la estructura espacial del nuestro entorno cercano:



Los valores singulares de la matriz esencial estimada con las matrices de calibración incorrecta son ahora muy distintos: $\{2834.66, 1563.62\}$, presagiando que la reconstrucción tendrá una deformación proyectiva importante.

En la siguiente página web de Zhang hay una pareja de imágenes y puntos correspondientes para evaluar los algoritmos de reconstrucción 3D:

`http://www-sop.inria.fr/robotvis/personnel/zzhang/SFM-Ex/SFM-Ex.html`

→ nuestros resultados

→ recomendaciones

7.4. Autocalibración

Una forma de eliminar la ambigüedad proyectiva de una reconstrucción no calibrada consiste en encontrar algunos puntos en la reconstrucción cuyas coordenadas reales conocemos y con ellos calcular la transformación `D` (ver Sec. 7.3.1). También se puede hacer una reconstrucción afín detectando el plano del infinito (intersección de tres juegos de paralelas) y llevándolo a su posición canónica, de forma análoga a la rectificación afín de planos (ver Sec. 5.5), etc.

Curiosamente, en muchos casos la deformación proyectiva se puede reducir automáticamente hasta una transformación de similitud: se consigue un modelo correcto, salvo la escala, que es lo máximo que puede conseguirse sin utilizar información 'extravisual'.

Autocalibrar significa encontrar automáticamente las matrices $\mathtt{K}_i$ de parámetros intrínsecos de cada cámara, a partir únicamente de primitivas (puntos, rectas, cónicas, etc.) correspondientes en varias imágenes, sin ninguna información de la estructura tridimensional de la escena. Con ellas, directa o indirectamente, conseguimos una reconstrucción similar. La escala real podría finalmente determinarse a partir distancia real entre dos cámaras (línea base) o entre dos puntos de la escena.

Las matrices de calibración se pueden extraer de diferentes objetos geométricos: matrices de cámara completas, homografías entre planos de la escena y la imagen, homografías de rotaciones puras, matrices fundamentales, etc. En casi todos los casos, y aunque pueden darse interpretaciones geométricas más o menos intuitivas (muchas veces involucran objetos geométricos con coordenadas complejas invisibles en la imagen), lo que en realidad siempre se aprovecha es la existencia de matrices de rotación más o menos profundamente 'enterradas' en el objeto geométrico de que se trate, que podemos cancelar de manera más o menos ingeniosa.

La extracción de las matrices de calibración suele hacerse indirectamente identificando primero alguna de las dos matrices simétricas siguientes, según convenga:

$$\boldsymbol{\omega} = \mathtt{K}^{-\mathsf{T}}\mathtt{K}^{-1} \qquad\qquad \boldsymbol{\omega}^* \equiv \boldsymbol{\omega}^{-1} = \mathtt{K}\mathtt{K}^{\mathsf{T}}$$

Una vez estimada alguna de ellas se extrae $\mathtt{K}$ ó $\mathtt{K}^{-1}$ mediante la factorización de *Cholesky* (ver Sec. A.5.2).

$\rightarrow$ interpretación de omega

En general, $\boldsymbol{\omega}$ es una matriz simétrica (con 5 grados de libertad) cuyos coeficientes dependen de los parámetros intrínsecos incluidos en $\mathtt{K}$ (ver Sec. 5.3). La forma concreta de los elementos de $\boldsymbol{\omega}$ en función de los elementos de $\mathtt{K}$ es irrelevante para los métodos lineales de estimación; lo importante es la existencia de ceros o de elementos repetidos.

> En la práctica, el parámetro de *skew* es cero (los pixels son rectangulares, no romboidales), lo que reduce los grados de libertad de $\boldsymbol{\omega}$ a 4:
>
> $$\mathtt{K} = \begin{bmatrix} f & 0 & o_x \\ 0 & fr & o_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \rightarrow \qquad \boldsymbol{\omega} = \begin{bmatrix} a & 0 & c \\ 0 & b & d \\ c & d & 1 \end{bmatrix}$$
>
> Si conocemos el punto principal y trabajamos con pixels 'centrados' tenemos dos grados de libertad:
>
> $$\mathtt{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & fr & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \rightarrow \qquad \boldsymbol{\omega} = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Y si finalmente conocemos el *aspect ratio* y lo corregimos la calibración se reduce a un solo parámetro:

$$\mathtt{K} = \mathtt{K}_f = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \rightarrow \qquad \boldsymbol{\omega} = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(En los dos últimos casos la 'estructura' de $\boldsymbol{\omega}^*$ es similar a la de $\boldsymbol{\omega}$.)

Veamos algunos ejemplos.

7.4.1. Calibración 'mediante planos'

A causa de la estructura general de una matriz de cámara $\mathtt{M} = \mathtt{K}[\mathtt{R}|\boldsymbol{t}]$ (ver Sec. 5.3), si encontramos una cierta matriz triangular superior $\mathtt{K}^*$ tal que al multiplicarla por $\mathtt{M}$ da lugar a una (sub)matriz ortogonal ($\mathtt{K}^*\mathtt{M} = \lambda[\mathtt{R}|\boldsymbol{t}]$), entonces $\mathtt{K}^*$ es proporcional a $\mathtt{K}^{-1}$. En principio esto permitiría calcular $\mathtt{K}$ a partir de una matriz de cámara completa $\mathtt{M}$, ya que la matriz ortogonal λR aporta suficientes restricciones. Pero, por supuesto, en este caso se puede usar directamente la descomposición QR para extraer $\mathtt{K}$.

Curiosamente, esa restricción de ortogonalidad se puede explotar para extraer $\mathtt{K}$ no solo de matrices de cámara completas, sino también de homografías entre planos de la escena y sus imágenes en la cámara; estas transformaciones contienen internamente dos columnas de la matriz de rotación $\mathtt{R}$ (ver Sec. 5.6).

Una forma de abordar el problema es la siguiente. Supongamos que conocemos la matriz $\boldsymbol{\omega} = \mathtt{K}^{-\mathsf{T}}\mathtt{K}^{-1}$. Entonces para cualquier homografía $\mathtt{H} = \mathtt{K}[\hat{\mathtt{R}}|\boldsymbol{t}]$ entre un plano 3D y la imagen, la expresión $\mathtt{H}^{\mathsf{T}}\boldsymbol{\omega}\mathtt{H}$ 'elimina las $\mathtt{K}$' y por ortogonalidad de las dos columnas de $\hat{\mathtt{R}}$ obtendremos una matriz con la siguiente estructura:

$$\mathtt{H}^{\mathsf{T}}\boldsymbol{\omega}\mathtt{H} = \begin{bmatrix} \lambda & 0 & ? \\ 0 & \lambda & ? \\ ? & ? & ? \end{bmatrix}$$

que proporciona 3 restricciones lineales sobre los elementos de $\boldsymbol{\omega}$ (no hay ninguna restricción sobre los elementos marcados como '?'). Puede ser necesario disponer de homografías de varios planos, dependiendo del número de elementos desconocidos de $\mathtt{K}$. Por ejemplo, una $\mathtt{K}_f$ simplificada como la explicada en la Sección 5.3 se podría estimar con una única homografía. En general, podemos fijar los parámetros que nos interesen, y trabajar con todas las homografías disponibles para plantear sistemas sobredeterminados, obteniendo soluciones más robustas.

En realidad este procedimiento *no* es de autocalibración (basado únicamente en puntos correspondientes en varias imágenes), ya que se apoya en la suposición 'extravisual' de que los puntos observados yacen en un plano 3D, y se utilizan sus coordenadas reales (con $Z = 0$).

Ejemplo. A partir de la (inversa de la) homografía de rectificación del tablero de ajedrez de la Sección 5.5 podemos extraer una estimación de matriz de cámara simplificada $\mathtt{K}_f$ con $f = 540.293$.

→ cálculo detallado

7.4.2. Autocalibración mediante rotaciones

La estructura $\mathtt{H} = \mathtt{K}\mathtt{R}\mathtt{K}^{-1}$ de las homografías procedentes de rotaciones puras (ver Sec. 5.6) sugiere un método de autocalibración para encontrar $\mathtt{K}$. Cada homografía de rotación $\mathtt{H}$ da lugar a la siguiente restricción sobre la matriz $\boldsymbol{\omega} = \mathtt{K}^{-\mathsf{T}}\mathtt{K}^{-1}$:

$$\mathtt{H}^{\mathsf{T}}\boldsymbol{\omega}\mathtt{H} = \mathtt{K}^{-\mathsf{T}}\mathtt{R}^{\mathsf{T}}\mathtt{K}^{\mathsf{T}}\ \mathtt{K}^{-\mathsf{T}}\mathtt{K}^{-1}\ \mathtt{K}\mathtt{R}\mathtt{K}^{-1} = \mathtt{K}^{-\mathsf{T}}\mathtt{K}^{-1} = \boldsymbol{\omega}$$

Aunque existe un factor de escala desconocido por la homogeneidad de $\mathtt{H}$, si normalizamos las homografías de rotación para que tengan determinante unidad conseguimos que el factor de escala de los dos términos de la igualdad sea el mismo, dando lugar a un sistema lineal homogéneo en los elementos de $\boldsymbol{\omega}$:

$$\mathtt{H}^{\mathsf{T}}\boldsymbol{\omega}\mathtt{H} = \boldsymbol{\omega}$$

Una matriz de calibración simplificada $\mathtt{K}_f$ se puede obtener de una única homografía de rotación que relaciona dos vistas de una escena tomadas desde el mismo lugar.

→ ejemplo gpu/match, expresión sencilla para f

Si hay más parámetros desconocidos de $\mathtt{K}$ podríamos necesitar más rotaciones, que además deberán realizarse alrededor de ejes de giro diferentes. P. ej., para determinar el *aspect ratio* de la cámara es imprescindible que haya un giro alrededor del eje óptico de la cámara (algo que no es muy usual en algunos sistemas automáticos de movimiento, que sólo admiten *pan* y *tilt*).

7.4.3. Autocalibración a partir de matrices fundamentales

La estructura de la matriz fundamental también permite imponer condiciones sobre la matriz de calibración $\mathtt{K}$. En la Sección 7.2.2 mostramos que

$$\mathtt{F} = [\boldsymbol{e}_2]_\times \mathtt{K}\mathtt{R}\mathtt{K}^{-1}$$

(ahora tenemos la matriz fundamental de dos imágenes tomadas con la misma cámara, $\mathtt{K}_1 = \mathtt{K}_2 = \mathtt{K}$). En este caso, para cancelar la rotación $\mathtt{R}$ utilizaremos $\boldsymbol{\omega}^* = \mathtt{K}\mathtt{K}^{\mathsf{T}}$:

$$\mathtt{F}\boldsymbol{\omega}^*\mathtt{F}^{\mathsf{T}} = [\boldsymbol{e}_2]_\times \mathtt{K}\mathtt{R}\mathtt{K}^{-1}\mathtt{K}\mathtt{K}^{\mathsf{T}}\mathtt{K}^{-\mathsf{T}}\mathtt{R}^{\mathsf{T}}\mathtt{K}^{\mathsf{T}}[\boldsymbol{e}_2]_\times^{\mathsf{T}} = [\boldsymbol{e}_2]_\times \boldsymbol{\omega}^* [\boldsymbol{e}_2]_\times^{\mathsf{T}}$$

En la Sección 7.2.1 vimos que el epipolo $\boldsymbol{e}_2$ es el espacio nulo de $\mathtt{F}^\mathsf{T}$. Agrupando en un factor de escala común λ la homogeneidad de $\mathtt{F}$ y la antisimetría de $[\boldsymbol{e}_2]_\times$ podemos escribir:

$$\mathtt{F}\boldsymbol{\omega}^*\mathtt{F}^\mathsf{T} = \lambda[\boldsymbol{e}_2]_\times\boldsymbol{\omega}^*[\boldsymbol{e}_2]_\times$$

Se trata de las ecuaciones de *Kruppa*, que a veces se justifican mediante consideraciones geométricas más o menos intuitivas. Es un conjunto de ecuaciones cuadráticas en los elementos de $\boldsymbol{\omega}^*$. A diferencia de los métodos anteriores, éste no posee una solución lineal.

→ solución simplificada con SVD

→ ejemplo con la F estimada anterior

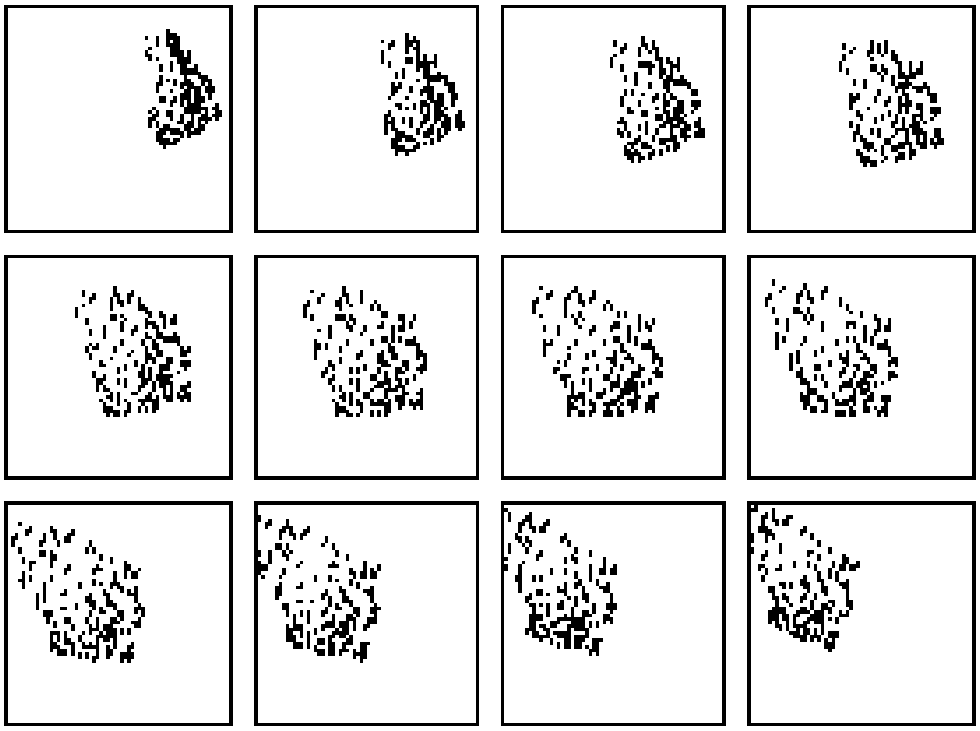
7.4.4. Autocalibración 'mediante cámaras'

Supongamos que tenemos una reconstrucción con deformación proyectiva de la escena (ver Sec. 7.3.1), compuesta por varias cámaras $\mathtt{P}_i$, consistentes con las imágenes $\boldsymbol{x}_i$ y unos puntos 3D $\boldsymbol{X}'$. La idea es buscar una transformación proyectiva tridimensional $\mathtt{D}^*$ (representada mediante una matriz homogénea 4×4, con 15 grados de libertad) que compense el efecto de $\mathtt{D}$ (idealmente sería $\mathtt{D}^{-1}$), corrigiendo las cámaras para que tengan unos parámetros intrínsecos $\mathtt{K}_i$ consistentes con nuestra información (p. ej., pixel cuadrado, punto principal = $\mathbf{0}$, etc.). En consecuencia:

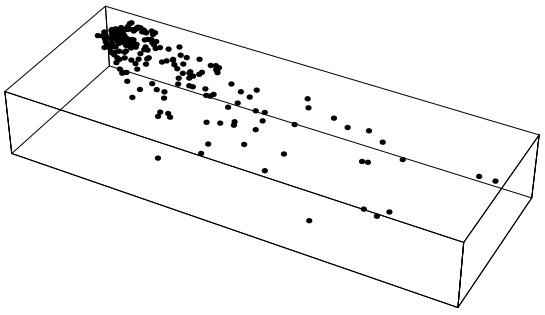
$$\mathtt{P}_i\mathtt{D}^* = \lambda_i\mathtt{M}_i = \lambda_i\mathtt{K}_i[\mathtt{R}_i|\boldsymbol{t}_i]$$

→ Un procedimiento lineal para estimar $\mathtt{D}^*$ se basa en

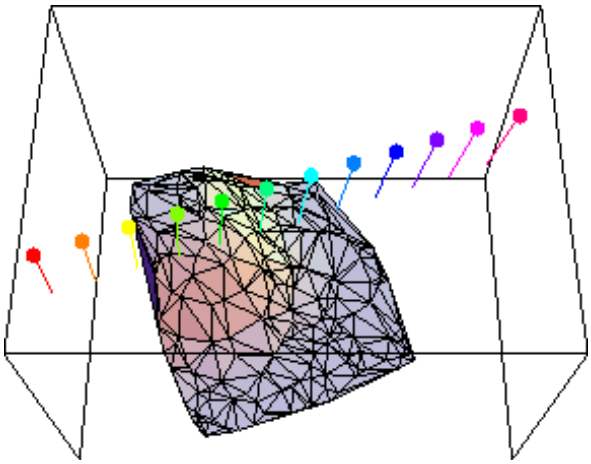
Imaginemos que sobrevolamos nuestra escena 3D sintética y tomamos varias imágenes desde distintas posiciones, ángulos, y zoom:



Si las imágenes no cambian bruscamente es posible hacer un seguimiento más o menos fiable de puntos correspondientes a lo largo de toda la secuencia. A partir de cualquier pareja de imágenes podemos calcular F y con ella una reconstrucción 3D proyectiva (aunque sería más robusto hacerlo con un método global como los que comentaremos más adelante):



A partir de esta reconstrucción proyectiva inicial podemos estimar el resto de las matrices de cámara $\mathtt{P}_i$ (ver Sec. 5.4) en un marco de referencia común (aunque deformado). Aplicando el método de autocalibración que acabamos de explicar conseguimos la transformación $\mathtt{D}^*$, que rectifica la escena, y su inversa, que corrige las cámaras, descubriendo sus parámetros intrínsecos y sus localizaciones y orientaciones relativas en el espacio:



7.4.5. Autocalibración 'mediante *odometría*'

Ver [16].

7.5. Ejemplo detallado

En esta sección resolveremos detalladamente todos los pasos de la reconstrucción estéreo con datos artificiales de comprobación (disponibles en la página web de material de la asignatura).

1. Partimos de puntos correspondientes en el sistema de coordenadas (columna,fila) de las imágenes originales, que tienen una resolución de 640 columnas $\times$ 640 filas. Los primeros puntos del archivo son:

c	f	c'	f'
217	423	203	409
217	406	205	385
217	389	206	362
217	371	208	340
...	...	...	...

2. Aunque es posible trabajar directamente con las coordenadas originales anteriores, es mucho mejor utilizar un sistema de referencia centrado en la imagen, con un valor máximo del orden de la unidad, y con el sentido del eje x hacia la izquierda. De esta forma conseguimos un sistema bien orientado, en el que la profundidad de los puntos será su coordenada en el eje z, y en el que la magnitud de las coordenadas es la adecuada para evitar inestabilidades numéricas en los cálculos posteriores. Para conseguirlo realizamos la transformación:

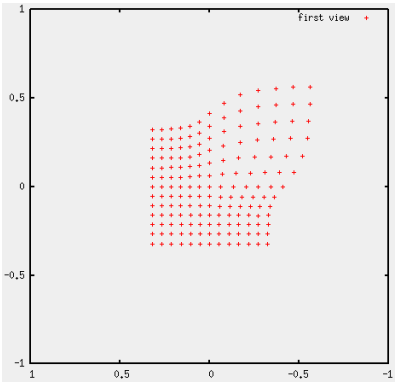
$$x = -\frac{c - c_m}{c_m}, \qquad y = -\frac{r - r_m}{c_m}$$

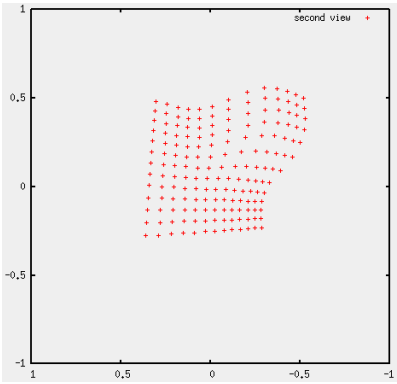
donde (c_m, r_m) es la posición del centro de la imagen, en nuestro ejemplo (320,320). Ten en cuenta que en imágenes reales el número de columnas y filas suele ser diferente, pero hay que dividir en ambos casos por el mismo factor c_m para mantener la relación de aspecto.

Los puntos correspondientes en este sistema de pixels "mejorado" son:

x	y	x'	y'
0.322	-0.322	0.366	-0.278
0.322	-0.269	0.359	-0.203
0.322	-0.216	0.356	-0.131
0.322	-0.159	0.350	-0.063
...	...	...	...

Hay que tener cuidado al representar gráficamente los puntos, ya que la dirección de los ejes es distinta de la tradicional tanto en el sistema (columna,fila) (donde la fila crece hacia abajo) como en el sistema de pixels que nos interesa (donde la x crece hacia la izquierda). En cualquier caso, lo que observamos en las imágenes es lo siguiente:





3. El siguiente paso es la estimación de la geometría "epipolar", que caracteriza la configuración de las cámaras en el espacio, en forma de una restricción para la pareja de imágenes (x, y) y (x', y') de cualquier punto del espacio:

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} \mathtt{F} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

La matriz fundamental F se puede estimar resolviendo un sistema lineal homogéneo (sobredeterminado) a partir simplemente de 8 ó más puntos correspondientes. (En nuestro caso tenemos muchos más, lo que puede contribuir a mejorar la estabilidad de la estimación.)

Cada correspondencia impone una restricción a los elementos de F:

$$\begin{bmatrix} x'x & x'y & x' & xy' & y'y & y' & x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathtt{F}_{11} \\ \mathtt{F}_{12} \\ \vdots \\ \mathtt{F}_{33} \end{bmatrix} = 0$$

El sistema con las ecuaciones generadas por todas las correspondencias (en nuestro ejemplo son 169) se resuelve mediante el procedimiento habitual (A.4). La solución, reorganizada en forma de matriz es la siguiente:

$$\mathtt{F} = \begin{bmatrix} 0.208 & 0.113 & -0.337 \\ 0.144 & -0.128 & -0.574 \\ 0.366 & 0.573 & 0.043 \end{bmatrix}$$

4. Por construcción geométrica (los rayos de puntos correspondientes se cortan en el punto 3D), toda matriz fundamental "ideal" tiene rango 2. Es conveniente corregir la F estimada para que cumpla exactamente esta condición. Para ello simplemente se reemplaza con un cero su menor valor singular y se vuelve a reconstruir. Los valores singulares de la matriz anterior son 0.727, 0.687 y 0.001, por lo que la corrección produce una matriz F muy parecida. En este ejemplo la estimación inicial ha sido bastante buena dado que los puntos utilizados son exactos y el único ruido proviene del redondeo a valores de pixel enteros.

5. A partir de la matriz F ya estamos en condiciones de obtener unas cámaras y una reconstrucción 3D que solo se diferencia de la real en una transformación proyectiva del espacio. Sin embargo, es preferible intentar una reconstrucción similar, por ejemplo mediante el procedimiento descrito a continuación.

6. Para conseguir una reconstrucción similar necesitamos las matrices de calibración de las cámaras. Si el único parámetro desconocido es la distancia focal efectiva f (lo que es razonable en cámaras que produzan imágenes 'naturales') entonces es sencillo descubrir ese valor f a partir de la matriz fundamental. Existen varias formas de conseguirlo, algunas de las cuales pueden expresarse como fórmulas cerradas más o menos complejas (p.ej., las fórmulas de Bougnoux o Sturm). Sin embargo, aquí veremos una muy sencilla de entender:

   La autocalibración estéreo puede expresarse como la búsqueda de unas matrices de calibración K y K′ de las cámaras que convierten a la matriz fundamental F

en una matriz esencial $\mathtt{E} = \mathtt{K'}^{\mathsf{T}}\mathtt{FK}$. Debido al análisis geométrico anterior, la matriz esencial $\mathtt{E} = [\boldsymbol{t}]_\times \mathtt{R}$, tiene los dos valores singulares distintos de cero iguales (las matrices de rotación tienen los 3 valores singulares iguales, y el factor $[\boldsymbol{t}]_\times$ simplemente elimina uno de ellos). Por tanto, un posible método de autocalibración consiste en buscar matrices $\mathtt{K}$ y $\mathtt{K}'$ que produzcan una matriz $\mathtt{K'}^{\mathsf{T}}\mathtt{FK}$ cuyos dos valores singulares mayores son iguales.

Para cuantificar la igualdad de los valores singulares $s_1 \geq s_2$ es conveniente usar la discrepancia normalizada:

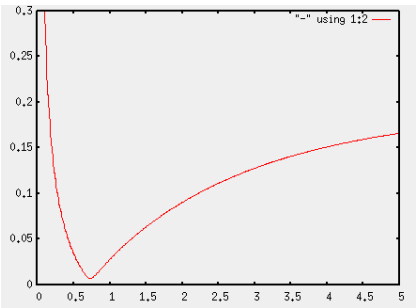
$$d = \frac{s_1 - s_2}{s_1 + s_2}$$

Y la búsqueda es muy sencilla si las matrices de calibración tienen la estructura

$$\mathtt{K}_f = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

porque dependen de un solo parámetro desconocido f. Si además las dos cámaras son iguales (y en su caso tienen el mismo *zoom*) entonces hay que descubrir un único parámetro f común, lo que puede hacerse simplemente recorriendo valores razonables, quedándonos con el que minimice la discrepancia normalizada anterior.

Si en nuestro ejemplo calculamos el valor de d para $f \in (0.1, 5)$ obtenemos la siguiente curva:



cuyo mínimo se encuentra aproximadamente alrededor de $f = 0.73$ y da lugar a la matriz esencial:

$$\mathtt{E} = \begin{bmatrix} 0.155 & 0.082 & -0.346 \\ 0.107 & -0.094 & -0.591 \\ 0.371 & 0.582 & 0.061 \end{bmatrix}$$

La discrepancia relativa que obtenemos en la $\mathtt{E}$ inducida por esta f es del orden de 7×10^{-3}, que es muy aceptable.

7. De la matriz esencial es inmediato extraer las dos cámaras. La primera está siempre en el origen de coordenadas, apuntando hacia el eje z y sin ninguna rotación:

$$\mathtt{M} = \mathtt{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

y para la segunda tenemos 4 posibilidades:

$$\mathtt{M}' = \mathtt{K}'[\mathtt{U}\mathtt{A}\mathtt{V}^\mathsf{T} \mid \pm \mathtt{U}_3]$$

donde $(\mathtt{U}, \_, \mathtt{V}) = svd(\mathtt{E})$, A puede ser W ó W^T, $\mathtt{U}_3$ es la tercera columna de $\mathtt{U}$, y

$$\mathtt{W} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

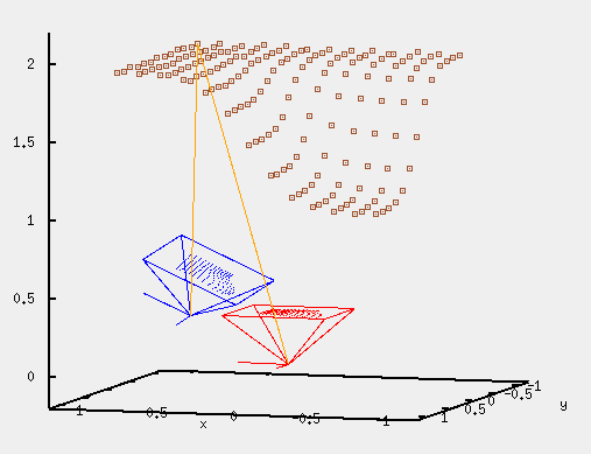
La $\mathtt{M}'$ correcta es la que reconstruye los puntos por delante de las dos cámaras. Se puede comprobar que en este ejemplo la pareja de cámaras correcta es la siguiente:

$$M = \begin{bmatrix} 0.73 & 0. & 0. & 0. \\ 0. & 0.73 & 0. & 0. \\ 0. & 0. & 1. & 0. \end{bmatrix} \qquad M' = \begin{bmatrix} 0.662 & -0.003 & 0.295 & -0.606 \\ 0.050 & 0.715 & -0.106 & 0.370 \\ -0.401 & 0.162 & 0.902 & 0.201 \end{bmatrix}$$

8. Finalmente hacemos la triangulación de los puntos correspondientes con las cámaras obtenidas. Los primeros puntos son:

X	Y	Z
0.857	-0.859	1.933
0.856	-0.714	1.928
0.860	-0.572	1.933
0.861	-0.426	1.939
...	...	...

La configuración geométrica de los puntos 3D y las dos cámaras se muestra en la siguiente figura:



Para representar las cámaras en el gráfico 3D podemos extraer los parámetros R y $\boldsymbol{C}$ de M' mediante el procedimiento explicado en la calibración mediante plantilla (los de M ya los sabemos).

Ejercicios:

- Comprueba que tus funciones obtienen los mismos resultados que los mostrados aquí (recuerda que las transformaciones homogéneas son iguales aunque difieran en un factor de proporcionalidad).
- Comprueba que la matriz fundamental cumple con gran precisión la restricción epipolar sobre los puntos correspondientes. Observa lo que ocurre si estimamos la F usando las coordenadas originales (columan, fila).
- Representa los epipolos y las líneas epipolares de algunos puntos.
- Repite este ejercicio con dos fotos de una escena real sencilla.
- Compara los resultados de diferentes métodos de calibración para obtener la f de una cámara.

→ Normalización/Desnormalización en general.

→ Comprobar que las cámaras extraídas producen F

→ Obtener la reconstrucción proyectiva

→ Detalles del algoritmo de triangulación

7.6. Más vistas

→ tres vistas reducen ambigüedad

→ problemas con 2 Fs

→ restricción trifocal

→ multiview, bundle, etc.

→ reconstrucción densa

Parte III

Herramientas Matemáticas

A mathematical investigation, difficult as it may be, and much space as it may occupy in the presentation of a physical theory, is never, and can never be, identical with the theory itself.

–von Mises

Apéndice A

Factorización de matrices

Existen muchas formas de expresar una transformación lineal (matriz) como composición (producto) de transformaciones más sencillas. Aquí recordaremos algunas de las más útiles. El libro de Golub y Van Loan [8] es una referencia fundamental sobre computación matricial.

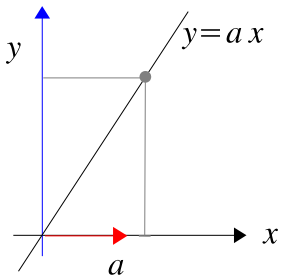
Antes de empezar conviene tener a mano un esquema gráfico del significado de la notación matricial.

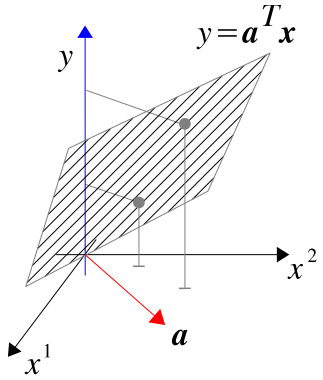
A.1. Transformaciones lineales

Uno de los tipos de función más simple que podemos imaginar es la función *lineal*. Dado un vector $\boldsymbol{x} = \{x^1, x^2, \ldots, x^n\} \in \mathbb{R}^n$, la función lineal $f_{\boldsymbol{a}} : \mathbb{R}^n \to \mathbb{R}$ se define como:

$$f_{\boldsymbol{a}}(\boldsymbol{x}) = \boldsymbol{a}^\mathsf{T}\boldsymbol{x} = a_1 x^1 + a_2 x^2 + \ldots + a_n x^n$$

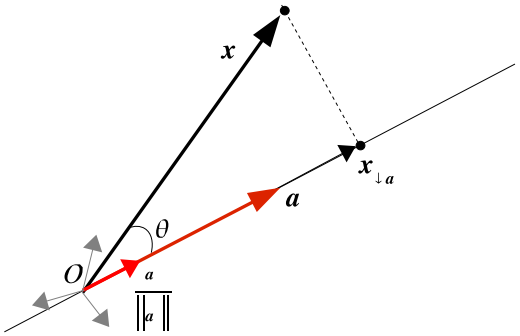
Está completamente especificada por el conjunto de coeficientes $\boldsymbol{a}$ que afectan a cada coordenada. Una función lineal sobre $\mathbb{R}^1$ es simplemente una recta con pendiente a. Sobre varias variables es un hiperplano que pasa por el origen:





Una función lineal tiene un ritmo de crecimiento (o decrecimiento) constante en todo el dominio: su *gradiente* es $\boldsymbol{a}$ (el vector de derivadas parciales respecto a cada coordenada, que indica la dirección de máximo crecimiento de la función).

El valor $\boldsymbol{a}^\mathsf{T}\boldsymbol{x}$ de la función tiene una interpretación geométrica relacionada con la proyección $\boldsymbol{x}_{\downarrow\boldsymbol{a}}$ de $\boldsymbol{x}$ sobre la dirección definida por $\boldsymbol{a}$:

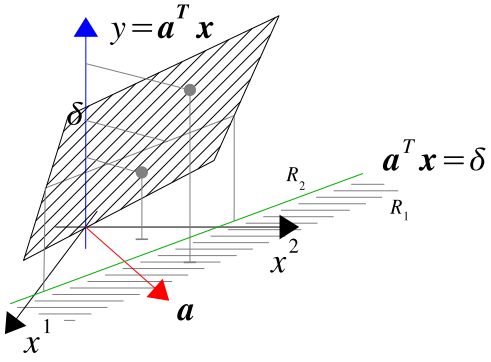


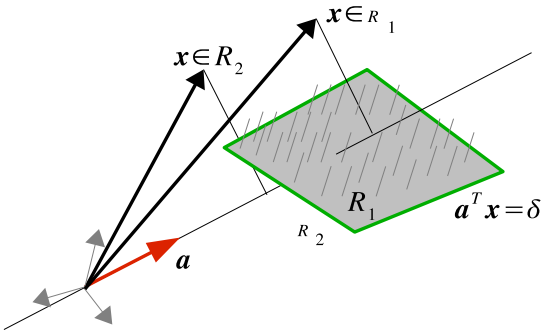
$$\boldsymbol{x}_{\downarrow\boldsymbol{a}} = \frac{\boldsymbol{a}^\mathsf{T}\boldsymbol{x}}{||\boldsymbol{a}||}\frac{\boldsymbol{a}}{||\boldsymbol{a}||}$$

$$||\boldsymbol{x}_{\downarrow\boldsymbol{a}}|| = \frac{\boldsymbol{a}^\mathsf{T}\boldsymbol{x}}{||\boldsymbol{a}||}$$

$$\cos\theta = \frac{\boldsymbol{a}^\mathsf{T}\boldsymbol{x}}{||\boldsymbol{a}||||\boldsymbol{x}||}$$

Las curvas de nivel de la función son hiperplanos. El conjunto de puntos $\boldsymbol{x}$ que cumplen $\boldsymbol{a}^\mathsf{T}\boldsymbol{x} = \delta$ es un hiperplano que divide el espacio original en dos semiespacios: $R_1 = \{x : \boldsymbol{a}^\mathsf{T}\boldsymbol{x} > \delta\}$ y $R_2 = \{x : \boldsymbol{a}^\mathsf{T}\boldsymbol{x} < \delta\}$. El hiperplano está situado a una distancia $\delta||\boldsymbol{a}||$ desde el origen:





A.2. Organización matricial

Distinguiremos entre *vectores* y *covectores*. Los vectores son los elementos $\boldsymbol{x} \in \mathbb{R}^n$ del espacio de trabajo. Se representan como columnas y sus elementos se indexan mediante superíndices:

$$\boldsymbol{x} = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^n \end{bmatrix}$$

Los covectores contienen los coeficientes de funciones lineales que obtienen escalares. Se representan como vectores fila (traspuestos) y sus elementos se indexan mediante subíndices:

$$\boldsymbol{a}^\mathsf{T} = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix}$$

La acción sobre un vector $\boldsymbol{x}$ de la función lineal $f_{\boldsymbol{a}} : \mathbb{R}^n \to \mathbb{R}$ es:

$$y = f_{\boldsymbol{a}}(\boldsymbol{x}) = \boldsymbol{a}^\mathsf{T}\boldsymbol{x} = \sum_{i=1}^{n} a_i x^i$$

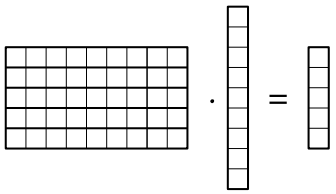
Una matriz $\mathtt{A}$ de tamaño $m \times n$ es una disposición rectangular de números organizada en m filas y n columnas. Sus elementos se indexan con un superíndice (fila) y un subíndice (columna):

$$\mathtt{A} = \{a^i_j\} = \begin{bmatrix} a^1_1 & a^1_2 & \dots & a^1_n \\ a^2_1 & a^2_2 & \dots & a^2_n \\ \vdots & \vdots & \ddots & \vdots \\ a^m_1 & a^m_2 & \dots & a^m_n \end{bmatrix}$$

Normalmente una matriz $\mathtt{A}$ contiene los coeficientes de una transformación lineal de $\mathbb{R}^n$ sobre $\mathbb{R}^m$. En este caso la interpretaremos como una 'columna de covectores':

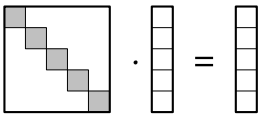
$$\mathtt{A} = \{\boldsymbol{a}^{\boldsymbol{i}\,\mathsf{T}}\} = \begin{bmatrix} \boldsymbol{a}^{1\,\mathsf{T}} \\ \boldsymbol{a}^{2\,\mathsf{T}} \\ \vdots \\ \boldsymbol{a}^{m\,\mathsf{T}} \end{bmatrix} = \left[\begin{array}{cccc} a_1^1 & a_2^1 & \dots & a_n^1 \\ \hline a_1^2 & a_2^2 & \dots & a_n^2 \\ \hline & & \vdots & \\ \hline a_1^m & a_2^m & \dots & a_n^m \end{array}\right]$$

En un producto matriz $\times$ vector las filas de la matriz son distintas funciones lineales que se aplican al vector para obtener las diferentes componentes del resultado: $y^i = \boldsymbol{a}^{\boldsymbol{i}\,\mathsf{T}}\boldsymbol{x}$. Cuando nos interese esta interpretación escribiremos la matriz sin trasponer:



$$\mathtt{A}\boldsymbol{x} = \boldsymbol{y}$$

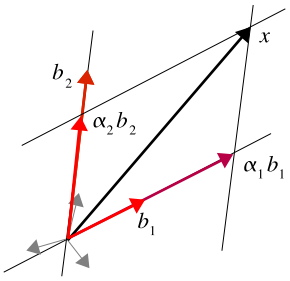
Un *operador diagonal* modifica cada coordenada por separado, multiplicándola por un coeficiente. Es el tipo de transformación lineal más simple posible:

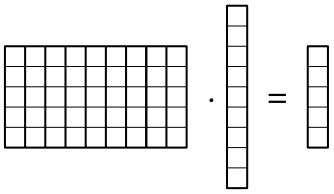


Alternativamente, una matriz se puede interpretar como una 'fila de vectores':

$$\mathtt{B}^{\mathsf{T}} = \{\boldsymbol{b_i}\} = \begin{bmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \dots & \boldsymbol{b}_n \end{bmatrix} = \left[\begin{array}{c|c|c|c} b_1^1 & b_2^1 & & b_n^1 \\ b_1^2 & b_2^2 & & b_n^2 \\ \vdots & \vdots & \dots & \vdots \\ b_1^m & b_2^m & & b_n^m \end{array}\right]$$

En este caso el producto matriz $\times$ vector se puede interpretar como una *combinación lineal* de los vectores $\boldsymbol{b}_i$ con unos coeficientes α_i. Cuando nos interese esta interpretación escribiremos la matriz traspuesta:



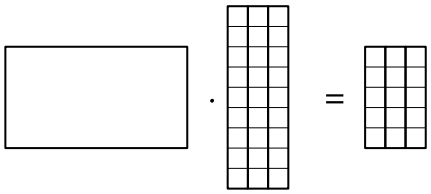


$$\mathtt{B}^\mathsf{T}\boldsymbol{\alpha} = \boldsymbol{x}$$

Por supuesto, ambas interpretaciones están íntimamente relacionadas: p. ej., las columnas de la matriz considerada como operador son las imágenes de la base.

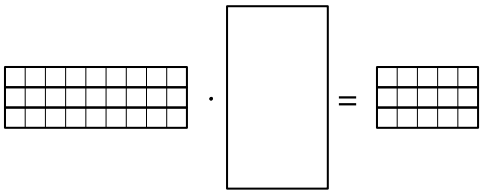
> Los covectores son duales a los vectores y cambian de 'forma contraria' a ellos al sufrir una transformación lineal. Al utilizar subíndices y superíndices los símbolos de sumatorio son redundantes. Siguiendo el convenio de Einstein los índices iguales a distinta altura indican implícitamente suma (contracción). P. ej., $\boldsymbol{y} = \mathtt{A}\boldsymbol{x}$ se escribiría como $y^j = a^j_k x^k$.

Una matriz $m \times n$ también puede representar una lista de m vectores de $\mathbb{R}^n$. Aunque el producto de matrices se interpreta normalmente como la *composición* de transformaciones lineales, también sirve para transformar 'de una vez' una colección de vectores almacenados en una matriz $\mathtt{X}$. Si los vectores están almacenados por columnas aplicamos el operador por la izquierda:



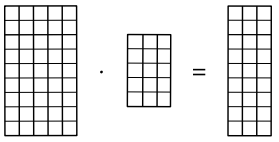
$$\mathtt{A}\mathtt{X}^\mathsf{T} = \mathtt{Y}^\mathsf{T}$$

y si lo están por filas aplicamos el operador traspuesto por la derecha:



$$\mathtt{X}\mathtt{A}^\mathsf{T} = \mathtt{Y}$$

> Análogamente, el producto de matrices permite expresar el cálculo de varias combinaciones lineales de un conjunto de vectores columna con diferentes juegos de coeficientes:



$$\mathtt{B}^\mathsf{T}\boldsymbol{\alpha}^\mathsf{T} = \mathtt{X}^\mathsf{T}$$

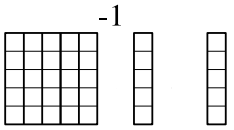
Todas las expresiones anteriores tienen una forma traspuesta cuyo significado o interpretación es inmediatamente deducible de la forma correspondiente sin trasponer.

Los elementos de un espacio vectorial se expresan mediante combinaciones lineales de los elementos de una base. Los coeficientes de la combinación lineal son las coordenadas del vector en esa base. P. ej., dada una base $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n\}$ organizada como las columnas de la matriz $\mathtt{B}^\mathsf{T}$, si escribimos $\boldsymbol{x} = \mathtt{B}^\mathsf{T}\boldsymbol{\alpha} = \sum_k \alpha_k \boldsymbol{b}_k$ queremos decir que $\boldsymbol{\alpha}$ son las coordenadas del vector $\boldsymbol{x}$ en la base $\mathtt{B}$.

Las matriz identidad $\mathtt{I}$ contiene la base 'canónica' de $\mathbb{R}^n$. Los componentes de un vector son directamente sus coordenadas en la base canónica: $\boldsymbol{x} = \mathtt{I}^\mathsf{T}\boldsymbol{x}$. Es inmediato deducir cuáles son sus coordenadas $\boldsymbol{\alpha}$ en cualquier otra base $\mathtt{B}$ (usamos la notación $\mathtt{B}^{-\mathsf{T}} \equiv (\mathtt{B}^\mathsf{T})^{-1}$):

$$\boldsymbol{x} = \mathtt{I}^\mathsf{T}\boldsymbol{x} = \mathtt{B}^\mathsf{T}\underbrace{\mathtt{B}^{-\mathsf{T}}\boldsymbol{x}}_{\boldsymbol{\alpha}} = \mathtt{B}^\mathsf{T}\boldsymbol{\alpha}$$

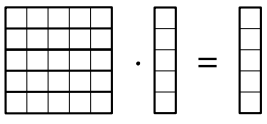
Por tanto, el *cambio de base* a una nueva base cuyos elementos son las filas de $\mathtt{B}$ se consigue con el operador $\mathtt{B}^{-\mathsf{T}}$:



$$\boldsymbol{\alpha} = \mathtt{B}^{-\mathsf{T}}\boldsymbol{x}$$

La recuperación o reconstrucción del vector en la base original es $\boldsymbol{x} = \mathtt{B}^\mathsf{T}\boldsymbol{\alpha}$.

Cuando la nueva base $\mathtt{V}$ es ortonormal ($\mathtt{V}^{-1} = \mathtt{V}^\mathsf{T}$) el cambio de base se reduce a $\boldsymbol{\alpha} = \mathtt{V}\boldsymbol{x}$.



$$\boldsymbol{\alpha} = \mathtt{V}\boldsymbol{x}$$

Esto justifica la conocida expansión de $\boldsymbol{x}$ en una base ortonormal $\mathtt{V} = \{v_i\}$ como $\boldsymbol{x} = \sum_i (\boldsymbol{v}_i^\mathsf{T}\boldsymbol{x})\boldsymbol{v}_i$:

$$\boldsymbol{x} = \mathtt{I}^\mathsf{T}\boldsymbol{x} = \overbrace{\mathtt{V}\underbrace{\mathtt{V}^\mathsf{T}}_{\boldsymbol{\alpha}}\boldsymbol{x}}^{\mathtt{I}^\mathsf{T}} = \mathtt{V}^\mathsf{T}\boldsymbol{\alpha}$$

La expresión de una *transformación* $\mathtt{A}$ *en una nueva base* $\mathtt{B}$ es:

$$\mathtt{A}' = \mathtt{B}^{-\mathsf{T}}\mathtt{A}\mathtt{B}^\mathsf{T}$$

> Si el cambio de base es $\boldsymbol{x}' = \mathtt{B}^{-\mathsf{T}}\boldsymbol{x}$ y $\boldsymbol{y}' = \mathtt{B}^{-\mathsf{T}}\boldsymbol{y}$, entonces la transformación $\boldsymbol{y} = \mathtt{A}\boldsymbol{x}$ puede escribirse como $\mathtt{B}^\mathsf{T}\boldsymbol{y}' = \mathtt{A}\mathtt{B}^\mathsf{T}\boldsymbol{x}'$, lo que implica que $\boldsymbol{y}' = \mathtt{B}^{-\mathsf{T}}\mathtt{A}\mathtt{B}^\mathsf{T}\boldsymbol{x}'$.

Si la base es ortonormal la expresión anterior se reduce a:

$$\mathtt{A}' = \mathtt{B}\mathtt{A}\mathtt{B}^\mathsf{T}$$

> Este tipo de operadores se llaman *conjugados*, y sirven para construir transformaciones más generales en función de otras más sencillas. Por ejemplo, si deseamos hacer una rotación respecto a un cierto eje de giro y solo sabemos hacerlo respecto al origen, podemos traer nuestro vector al origen, girarlo, y llevarlo de nuevo a su sitio. O si solo sabemos escalar el eje x pero deseamos escalar el eje y podemos rotar 90 grados, escalar el eje x y rotar -90 grados.

Una base interesante $\mathtt{B}$ es la que consigue que el operador $\mathtt{A}'$ tome forma diagonal $\mathtt{D}$. Se conseguiría mediante una factorización de $\mathtt{A}$ como:

$$\mathtt{A} = \mathtt{B}^\mathsf{T}\mathtt{D}\mathtt{B}^{-\mathsf{T}}$$

Si la base $\mathtt{V}$ que consigue la diagonalización es ortornormal tenemos una factorización muy simple:

$$\mathtt{A} = \mathtt{V}^\mathsf{T}\mathtt{D}\mathtt{V}$$

A.3. Valores y vectores propios (*eigensystem*)

Los vectores propios de un operador son elementos especiales sobre los que el operador actúa de manera muy sencilla, simplemente multiplicándolos por un escalar (que es el valor propio asociado). Cada uno de ellos cumple $\mathtt{A}\boldsymbol{v}^i = \lambda_i\boldsymbol{v}^i$ y todos ellos en conjunto verifican $\mathtt{V}\mathtt{A}^\mathsf{T} = \mathtt{\Lambda}\mathtt{V}$ (donde los valores propios λ_i son los elementos de la matriz diagonal $\mathtt{\Lambda}$ y ${\boldsymbol{v}^i}^\mathsf{T}$ son las filas de $\mathtt{V}$).

La 'eigendescomposición' es extraordinariamente útil. Además de desentrañar la esencia de una transformación lineal, entre otras muchas aplicaciones permite también analizar la dispersión de una población aleatoria (Sección C.2) y comprimir su

representación (Sección 2.2.1); las componentes frecuenciales son las funciones propias de los sistemas lineales invariantes a desplazamientos (Sección B); los vectores propios de transformaciones proyectivas (ver Sec. 5.2) son puntos fijos, etc.

Todos los entornos de cálculo científico proporcionan algoritmos eficientes para calcular el *eigensystem* de una matriz. Suelen estar basados en aplicar sucesivas rotaciones elementales a la matriz para hacer ceros en las posiciones deseadas.

Si el operador es simétrico ($\mathtt{A} = \mathtt{A}^\top$) los vectores propios forman una base ortonormal y los valores propios son reales. Los vectores propios son una base en la que el operador adquiere forma diagonal:

$$\mathtt{A} = \mathtt{V}^\top \Lambda \mathtt{V}$$

$\rightarrow$ justificarlo

Método de la potencia. Es un algoritmo muy sencillo para calcular los vectores y valores propios de una matriz simétrica semidefinida positiva $\mathtt{A}$. La idea es partir de un vector cualquiera $\boldsymbol{v}$ y repetir, hasta que converja, la operación:

$$\boldsymbol{v} \leftarrow \frac{\mathtt{A}\boldsymbol{v}}{||\mathtt{A}\boldsymbol{v}||}$$

El resultado es el vector propio de mayor autovalor. (La razón de que el método funcione es que el operador $\mathtt{A}$ produce imágenes en las que el vector propio dominante 'tira' más fuerte que los demás y va acercando progresivamente a $\boldsymbol{v}$ hasta alinearlo con él.

El valor propio correspondiente se calcularía haciendo:

$$\lambda = \boldsymbol{v}^\top \mathtt{A} \boldsymbol{v}$$

ya que, al ser $\mathtt{V}$ ortonormal, en $\boldsymbol{v}^\top \mathtt{A} \boldsymbol{v} = \boldsymbol{v}^\top \mathtt{V}^\top \Lambda \mathtt{V} \boldsymbol{v}$ el producto $\mathtt{V}\boldsymbol{v}$ sólo produce salida distinta de cero en la posición de su propio autovalor.

Si deseamos calcular el siguiente vector propio más dominante podemos repetir el procedimiento eliminando antes de A la contribución que ya hemos calculado:

$$\mathtt{A} \leftarrow \mathtt{A} - \lambda \boldsymbol{v}\boldsymbol{v}^\top$$

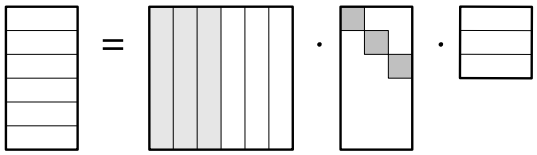
(La factorización $\mathtt{V}^\top \Lambda \mathtt{V}$ es una abreviatura de $\sum_i \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$. La ortonormalidad de $\mathtt{V}$ hace que podamos escribir $\mathtt{A}$ como una suma ponderada de *outer products* $\boldsymbol{v}_i \boldsymbol{v}_i^\top$ del mismo vector $\boldsymbol{v}_i$.)

A.4. Valores singulares

La descomposición en valores singulares permite analizar cualquier operador y descubrir los efectos que produce al actuar sobre los elementos de un espacio. Tiene incontables aplicaciones. Dada una matriz `A` rectangular, existen matrices ortonormales `U` y `V` y una matriz diagonal real `S` tales que:

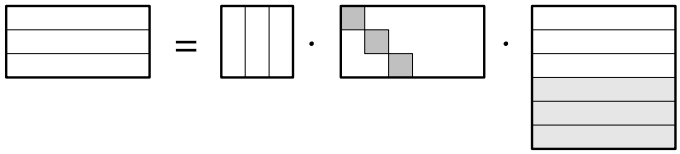
$$\mathtt{A} = \mathtt{U}^\mathsf{T}\mathtt{S}\mathtt{V}$$

El operador `A` equivale a un cambio a la base `V`, donde cada una de las nuevas coordenadas sufre un escalado independiente dado por cada valor singular, y finalmente el resultado es una combinación lineal de los elementos de `U` con los coeficientes obtenidos en el paso anterior (se vuelve a cambiar de base). Si `A` es rectangular 'vertical' la descomposición se puede representar como:



(Las filas sombreadas son una base del subespacio imagen (*range*) del operador `A`; el resto siempre estará afectado por coeficientes nulos)

Si `A` es rectangular 'horizontal' la SVD se representa como:



Espacio nulo. Las filas de `V` correspondientes a los valores singulares iguales cero (sombreadas en la figura anterior) son una base del espacio nulo (*nullspace*) de `A` (el conjunto de soluciones no triviales ($\boldsymbol{x} \neq \mathbf{0}$) al sistema homogéneo $\mathtt{A}\boldsymbol{x} = \mathbf{0}$). Las componentes de la entrada en esos elementos de la base se multiplican por valores singulares nulos y serán 'invisibles' a la salida.

Relación con los valores propios. Si interpretamos `A` no como un operador sino como un conjunto de vectores (por filas) procedentes de una población aleatoria centrada (con media $\mathbf{0}$), entonces `V` son los vectores propios de $\mathtt{A}^\mathsf{T}\mathtt{A}$, que es la matriz de covarianza de la observaciones:

$$\mathtt{A}^\mathsf{T}\mathtt{A} = \mathtt{V}^\mathsf{T}\underbrace{\mathtt{S}\overbrace{\mathtt{U}\mathtt{U}^\mathsf{T}}^{\mathtt{I}}\mathtt{S}}_{\Lambda}\mathtt{V}$$

Análogamente, U son los vectores propios de $\mathtt{A}\mathtt{A}^\mathsf{T}$, la matriz que contiene los productos escalares de todas las parejas de elementos de A (fundamental en las máquinas de *kernel*, Capítulo 4). Finalmente, los valores singulares al cuadrado son los valores propios de ambas matrices: $\mathtt{S}^2 = \Lambda$.

Rango y condición de una matriz. Decimos que una matriz está mal condicionada si tiene valores singulares de magnitudes muy diferentes. En este caso algunos algoritmos numéricos son inestables. P. ej., podemos encontrar matrices que, a pesar de tener un determinante 'muy grande', están mal condicionadas; en estos casos el cálculo de la inversa presenta graves problemas de precisión. El cociente entre el mayor y menor valor singular es el *número de condición* de la matriz; cuanto menor sea, mejor condicionada estará. Si es muy grande la matriz será esencialmente singular.

El número de valores singulares (suficientemente) mayores que cero es el rango (*rank*) del operador (la dimensión del subespacio imagen).

Mínimos cuadrados. La SVD permite resolver fácilmente sistemas de ecuaciones lineales sobredeterminados por mínimo error cuadrático, tanto homogéneos como no homogéneos. Para resolver un sistema no homogéneo sobredeterminado $\mathtt{A}\boldsymbol{x} = \boldsymbol{b}$ usamos la *pseudoinversa*, que estudiaremos en el punto siguiente.

Dado un sistema homogéneo $\mathtt{A}\boldsymbol{x} = \mathbf{0}$, las filas de V correspondientes a los menores valores singulares son vectores unitarios que minimizan $||\mathtt{A}\boldsymbol{x}||^2$. La solución ideal sería el espacio nulo, pero debido a errores numéricos u otras causas, los valores singulares que deberían ser cero no lo son exactamente. Aún así, las filas de V correspondientes a los menores valores singulares son las mejores aproximaciones a la solución en el sentido de mínimo error cuadrático.

Inversa y Pseudoinversa. Dado un operador A cuya SVD es $\mathtt{A} = \mathtt{U}^\mathsf{T}\mathtt{S}\mathtt{V}$, su inversa se puede calcular de forma inmediata con $\mathtt{A}^{-1} = \mathtt{V}^\mathsf{T}\mathtt{S}^{-1}\mathtt{U}$.

Si la matriz A es singular o rectangular algunos elementos de la diagonal de S serán cero y no existirá $\mathtt{S}^{-1}$ ni tampoco $\mathtt{A}^{-1}$. No obstante, es posible construir una matriz *pseudoinversa* que, hasta cierto punto, actúa como una inversa. La pseudoinversa $\mathtt{S}^+$ de una matriz diagonal S se consigue invirtiendo únicamente los elementos de la diagonal que son (suficientemente) distintos de cero. La pseudoinversa de una matriz general $\mathtt{A} = \mathtt{U}^\mathsf{T}\mathtt{S}\mathtt{V}$ es $\mathtt{A}^+ = \mathtt{V}^\mathsf{T}\mathtt{S}^+\mathtt{U}$. Si la matriz es cuadrada e invertible $\mathtt{A}^+ = \mathtt{A}^{-1}$.

Un operador A que es singular o rectangular 'horizontal' reduce la dimensión del espacio original (es 'de muchos a uno') y por tanto sus efectos no pueden deshacerse.

En este caso la pseudoinversa obtiene alguna de las preimágenes que cumplen perfectamente las condiciones exigidas (aunque hay otras soluciones): $\mathtt{A}\mathtt{A}^+ = \mathtt{I}$. Esto ocurre, p. ej., cuando deseamos calcular el rayo óptico inducido por un pixel de imagen (ver Sec. 5.3).

Si el operador $\mathtt{A}$ es rectangular 'vertical', entonces expande su entrada a un espacio de dimensión mayor que el original sin poder cubrirlo completamente; algunos vectores no tienen preimagen. Trabajando sobre el espacio completo, la pseudoinversa A^+ proyecta cada vector al más cercano de los que tienen preimagen, lo que corresponde a una solución de mínimo error cuadrático. Ahora $\mathtt{A}^+\mathtt{A} = \mathtt{I}$. Esto permite, p. ej., resolver un sistema lineal sobredeterminado como los que aparecen en el ajuste de máquinas lineales (ver Sec. 4.2): dada la ecuación $\mathtt{A}\boldsymbol{x} = \boldsymbol{b}$, el vector $\boldsymbol{x}^*$ que minimiza $||\mathtt{A}\boldsymbol{x} - \boldsymbol{b}||^2$ es $\boldsymbol{x}^* = \mathtt{A}^+\boldsymbol{b}$.

Otra forma de verlo es la siguiente. Un operador $\mathtt{A}$ de 'reducción dimensional' es una verdadera inversa (deshace perfectamente el efecto de) su pseudoinversa $\mathtt{A}^+$. Cuando el operador es de 'ampliación dimensional' ocurre lo contrario: la pseudoinversa $\mathtt{A}^+$ es una verdadera inversa del efecto producido por el operador, con la ventaja de que sobre el resto de puntos 'inaccesibles' mediante $\mathtt{A}$ hace un trabajo útil: proyecta al más próximo que tiene preimagen.

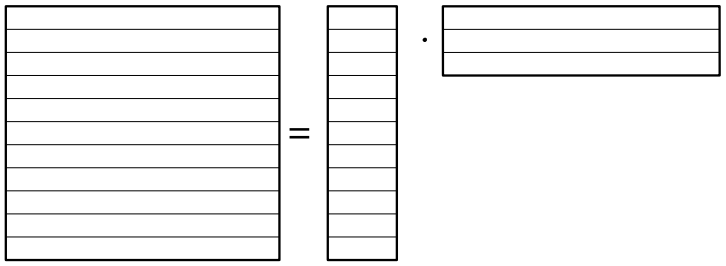
Forzar condiciones a un operador. En ocasiones una matriz debe cumplir condiciones tales como ser ortonormal (p. ej., una matriz de rotación), tener un cierto rango (p. ej., una matriz fundamental, Sección 7.2.1) o tener dos valores singulares iguales (p. ej., una matriz esencial, Sección 7.2.2).

Los métodos de optimización más sencillos (lineales) no son capaces de obtener un estimador que cumpla exactamente esa condición. Aun así, son computacionalmente muy eficientes, por lo que si los datos de trabajo no son excesivamente ruidosos, se puede aprovechar la estimación disponible y 'arreglarla' para que cumpla la condición deseada. Por ejemplo, si necesitamos una matriz de rotación, con la SVD 'abrimos' la matriz $\mathtt{A} = \mathtt{U}^\mathsf{T}\mathtt{S}\mathtt{V}$ y hacemos $\mathtt{S} = \mathtt{I}$. Si necesitamos una matriz esencial hacemos que s_1 y s_2 tomen, p. ej., la media de los valores que tuvieran inicialmente. Para arreglar una matriz fundamental hacemos cero el menor valor singular.

Componentes Principales. En muchas aplicaciones (p. ej., Sección 2.2.1) nos interesa representar una poblacion de objetos en una base de menor dimensión sin que se pierda excesiva información. En la Sección C.2 vimos que las direcciones principales de la población, las que tienen más varianza, son muy apropiadas para esta tarea. Corresponden con los vectores propios de la matriz de covarianza Σ con mayor valor propio. La descomposición en valores singulares nos permite un cálculo alternativo de las componentes principales.

Supongamos que tenemos una colección de m objetos de dimensión n, representados por filas en la matriz $\mathtt{X}$. Por simplicidad, los objetos están centrados en el origen

(tienen media **0**). Imaginemos que podemos expresar X como el producto de una matriz vertical Y y otra y otra horizontal V, ambas bastante 'estrechas':



$\mathtt{X} = \mathtt{YV}$

Esto significa que cada uno de los objetos originales $\boldsymbol{x}_i$ es en realidad una combinación lineal de los elementos de V con coeficientes $\boldsymbol{y}_i$. O, lo que es lo mismo, los vectores $\boldsymbol{x}$ de $\mathbb{R}^n$ pueden expresarse en una nueva base reducida V, en la que sólo m coordenadas (el vector $\boldsymbol{y}$) son distintas de cero.

A partir de la SVD de $\mathtt{X} = \mathtt{U}^\mathsf{T}\mathtt{SV}$ podemos obtener inmediatamente la base V (correspondiente a los valores singulares significativamente distintos de cero) y las nuevas coordenadas $\mathtt{Y} = \mathtt{U}^\mathsf{T}\mathtt{S}$ (descartamos filas y columnas de U, S y V que producen ceros para conseguir las matrices rectangulares deseadas).

A.5. Otras Descomposiciones

A.5.1. Descomposición RQ

Toda matriz cuadrada puede descomponerse como el producto de una matriz ortogonal Q y una matriz triangular superior (R) o inferior (L). Dependiendo del orden y del tipo de matriz triangular deseada hay cuatro diferentes factorizaciones: RQ, QR, LQ, QL. La descomposición RQ es especialmente interesante en visión por computador (Sección 5.4, pág. 126), ya que permite extraer automáticamente de una matriz de cámara los parámetros intrínsecos (relación entre píxels y ángulos en el espacio) y extrínsecos (orientación y posición).

Aunque algunos entornos de cálculo científico solo disponen de una de las formas posibles es sencillo construir cualquier otra mediante manipulaciones sencillas.

A.5.2. Descomposición de Cholesky

Toda matriz simétrica se puede expresar como el producto $\mathtt{C} = \mathtt{K}^\mathsf{T}\mathtt{K}$, donde K es triangular superior. Esta factorización se utiliza p. ej., en visión por computador para extraer los parámetros de calibración de una cámara a partir de la imagen de la cónica absoluta $\mathcal{C}'$ (Sección 5.5, pág. 133).

Apéndice B

Representación Frecuencial

El análisis frecuencial (también llamado análisis *armónico*) es una herramienta fundamental en muchos campos científicos y tecnológicos. Por supuesto, es imposible mencionar todas sus aplicaciones: sin la transformada de Fourier o sus derivados no tendríamos sonido *mp3*, fotos *jpeg* ni videos *mpeg*; tampoco radio, televisión, teléfono móvil, etc.

B.1. Espacio de funciones

Los elementos de un espacio vectorial se representan mediante sus coordenadas en una base. Las funciones continuas también son elementos de un espacio vectorial, aunque de dimensión infinita:

$$f(x) = \boldsymbol{F}^{\mathsf{T}}\mathtt{H}(x) = \sum_{k=0}^{\infty} F_k H_k(x)$$

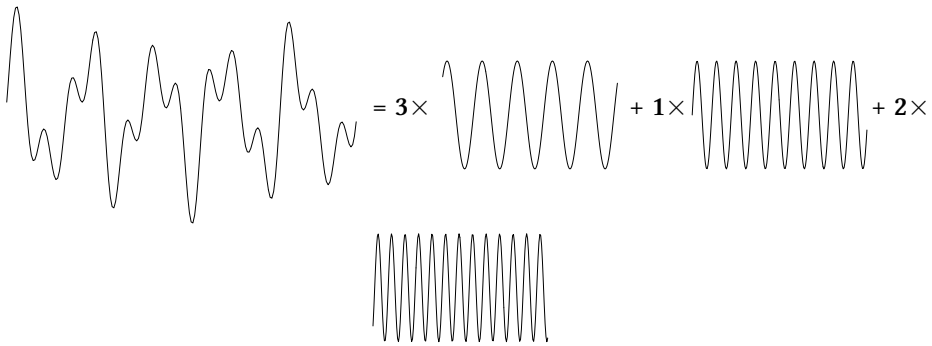
El vector $\boldsymbol{F} = \{F_k\}_{k=0}^{\infty}$ contiene las (infinitas) coordenadas de f en la base $\mathtt{H}(x) = \{H_k(x)\}_{k=0}^{\infty}$.

> Por ejemplo la función $f(x) = \exp(-x^2)$ se puede expresar en una base de potencias como $f(x) = 1 - x^2 + \frac{1}{2}x^4 - \frac{1}{6}x^6 + \ldots$ Por tanto, en la base de potencias $H_k(x) = x^k$ sus coordenadas son $\boldsymbol{F}^{\mathsf{T}} = [1, 0, -1, 0, \frac{1}{2}, 0, -\frac{1}{6}, \ldots]$. Incluso podemos escribir la expansión en forma matricial:
>
> $$exp(-x^2) = [1, 0, -1, 0, \frac{1}{2}, 0, -\frac{1}{6}, \ldots] \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ \vdots \end{bmatrix}$$

Por sencillez, en este ejemplo hemos utilizado una base que no es ortonormal (enseguida definiremos un producto escalar para espacios de funciones). En general es mejor utilizar bases ortonormales porque simplifican el cálculo de las coordenadas (ver Sec. A.2). Por supuesto, la utilidad de la expansión anterior reside en que si x está cerca de cero se consigue una buena aproximación a la función original con unos pocos términos.

B.2. Señales periódicas

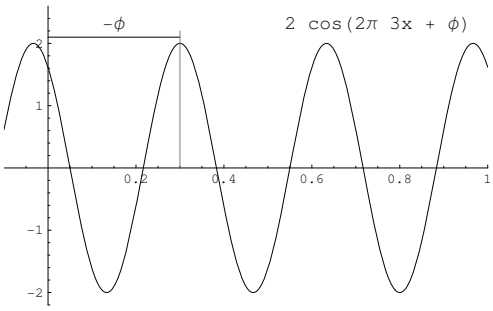
Normalmente deseamos representar las funciones en una base cuyas coordenadas tienen algún significado físico. Las *funciones periódicas* son especialmente ventajosas para analizar fenómenos repetitivos como las vibraciones sonoras o cualquier otra situación en la que los cambios en las señales presentan una cierta estructura no completamente aleatoria. Muchas señales pueden representarse sin apenas pérdida de información como combinación de un número relativamente pequeño de componentes de este tipo:



Las funciones periódicas más utilizadas como base de representación son las de tipo (co)sinusoidal:

$$A\cos(2\pi\omega x + \phi)$$

Se caracterizan por su amplitud A, frecuencia ω y desfase ϕ respecto al origen. En la figura siguiente se muestra un ejemplo con $A = 2$, $\omega = 3$ (hay 3 repeticiones completas en el intervalo $[0, 1]$) y fase $\phi = -0.3/(2\pi\omega)$:



Existen muchas otras funciones periódicas: triangulares, cuadradas, etc. ¿Por qué son tan populares las funciones trigonométricas? Una razón importante es que la superposición (suma) de varios senos (o cosenos, que son equivalentes salvo desfase) de diferentes amplitudes y desfases pero *de la misma frecuencia*, es también una función seno de esa misma frecuencia, con una cierta amplitud y desfase que depende de los ingredientes que hemos sumado. Esta curiosa propiedad es ideal para constituir una base de representación donde se haga explícita la importancia de cada frecuencia en una señal. Como veremos más adelante, las funciones armónicas son las funciones propias (*eigenfunctions*) de los sistemas lineales invariantes a desplazamientos.

B.3. Manejo de la fase

El hecho de que una función sinusoidal requiera para su especificación no sólo la amplitud –un coeficiente que juega el papel de una coordenada ordinaria– sino también la fase ϕ, que actúa *no linealmente* en la función, hace necesario un tratamiento matemático especial para conseguir una expresión elegante del tipo $f(x) = \boldsymbol{F}^{\mathsf{T}}\mathtt{H}(x)$.

Una opción es utilizar una base con dos familias de funciones (senos y cosenos):

$$A\cos(2\pi\omega x + \phi) = \underbrace{(A\cos\phi)}_{F^c_\omega}\cos(2\pi\omega x) + \underbrace{(-A\sin\phi)}_{F^s_\omega}\sin(2\pi\omega x)$$

los coeficientes F^c_ω y F^s_ω codifican simultaneamente la amplitud y la fase de la onda.

Una alternativa matemáticamente más elegante consiste en utilizar la forma compleja de las funciones trigonométricas: $\cos x = \frac{1}{2}(e^{ix} + e^{-ix})$ y $\sin x = \frac{1}{2i}(e^{ix} - e^{-ix})$ (recuerda la representación polar $e^{ia} = \cos a + i\sin a$). La onda anterior puede expresarse entonces como:

$$A\cos(2\pi\omega x + \phi) = \underbrace{\frac{A}{2}e^{i\phi}}_{F_\omega}\, e^{i2\pi\omega x} + \underbrace{\frac{A}{2}e^{-i\phi}}_{F_{-\omega}}\, e^{-i2\pi\omega x}$$

con lo que cambiamos la base de representación de senos y cosenos por la de exponenciales complejas:

$$H_\omega(x) = e^{i2\pi\omega x}$$

Ahora los dos parámetros necesarios para especificar la función cosenoidal, amplitud y fase, también aparecen como una coordenada ordinaria (un coeficiente que multiplica a cada elemento de la base $H_\omega(x)$). En este caso cada coordenada frecuencial es un número complejo $F_\omega = A_\omega e^{i\phi_\omega}$ que tiene la curiosa propiedad de que codifica en su módulo la amplitud de la onda ($|F_\omega| = A_\omega$) y en su argumento (el ángulo de la representación polar) el desfase ($\arg F_\omega = \phi_\omega$). Usamos el subíndice ω para hacer explícito que se trata de la amplitud y la fase correspondiente a esa frecuencia.

Debe quedar bien entendido que para reconstruir cada componente cosenoidal real de frecuencia ω necesitamos las componentes de frecuencia positiva y 'negativa' $H_{+\omega}$ y $H_{-\omega}$ (podemos pensar que el cambio de signo de la componente imaginaria i se lo 'transferimos' a ω). El precio que pagamos para que la fase actúe como un coeficiente multiplicativo es la necesidad de usar frecuencias negativas para extraer automáticamente la parte real. Sin embargo, está claro que sólo necesitamos calcular la coordenada de una cualquiera de las dos, obteniéndose la otra por conjugación: $F_{-\omega} = \overline{F_\omega}$.

En definitiva, cualquier función se puede representar como la superposición de un número, en principio infinito, de armónicos complejos con frecuencias positivas y negativas:

$$f(x) = \frac{1}{2}\sum_{\omega=-\infty}^{\infty} F_\omega H_\omega(x)$$

En forma matricial podemos escribir:

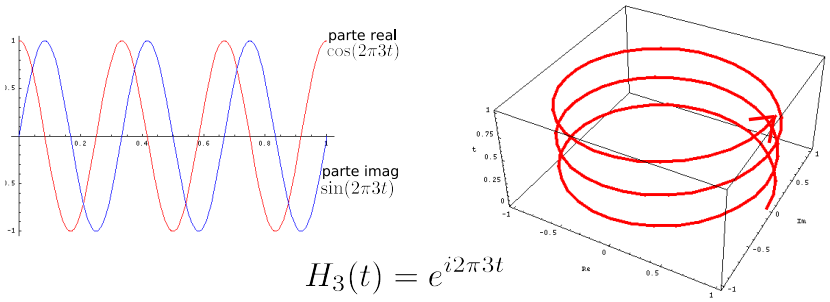
$$f(x) = \boldsymbol{F}^\mathsf{T}\mathtt{H}(x)$$

donde las coordenadas de $f(x)$ son un vector $\boldsymbol{F}^\mathsf{T} = \frac{1}{2}[\ldots, F_{-2}, F_{-1}, F_0, F_1, F_2, \ldots]$ de infinitas componentes complejas.

La base $\mathtt{H}(x)$ tiene la doble ventaja de que permite una representación elegante en términos de frecuencias puras, cuyas coordenadas seleccionan la amplitud y la fase, y además abre la posibilidad de manejar señales complejas $f(x) = g(x)+ih(x)$. En este caso las coordenadas de frecuencias positivas y negativas ya no son (conjugadamente) equivalentes, sino que entre las partes reales e imaginarias de F_ω y $F_{-\omega}$ se codifica, de forma entremezclada, la amplitud y la fase de cada componente cosenoidal de las partes real $g(x)$ e imaginaria $h(x)$ de la señal.

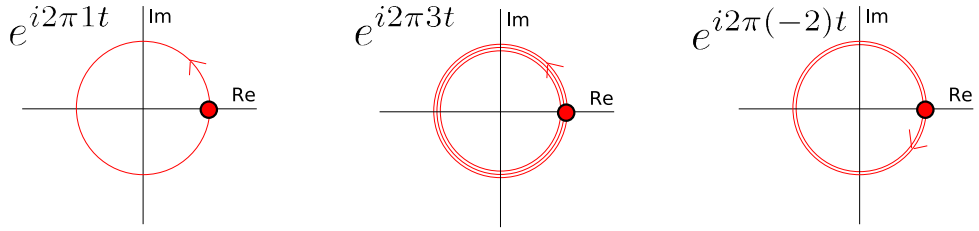
B.4. Interpretación geométrica

Las funciones armónicas $H_\omega(t)$ se pueden interpretar geométricamente como un trayectoria circular, o mejor, en forma de 'muelle'. Al plano complejo le añadimos una tercera dimensión para la variable independiente t (que podemos imaginar como un tiempo o un distancia recorrida) y entonces $H_\omega(t)$ da ω vueltas cuando t recorre $[0, 1]$. Si $\omega > 0$ el giro es en sentido positivo (contrario al de las agujas del reloj) y si $\omega < 0$ el giro es en sentido contrario. P.ej., el tercer armónico positivo se puede representar así:

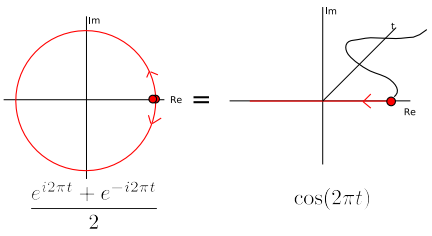


$$H_3(t) = e^{i2\pi 3t}$$

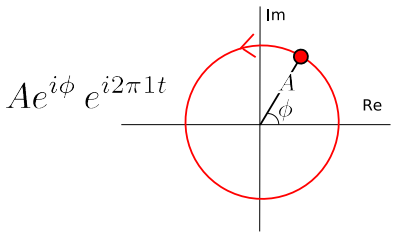
Una representación todavía mejor se consigue aplastando el muelle pero indicando en el dibujo de alguna manera su frecuencia (ω), e incluyendo el punto de partida y el sentido de giro. P.ej.:



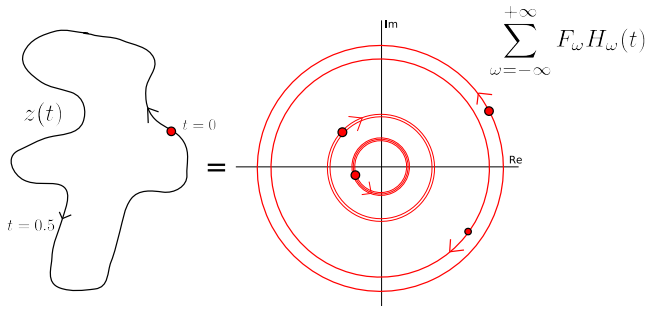
Así es evidente la relación entre la exponencial compleja y las funciones trigonométricas:



La multiplicación de un armónico por un número complejo $z = Ae^{i\phi}$ cambia el radio del muelle en un factor A y lo gira rígidamente un ángulo ϕ:

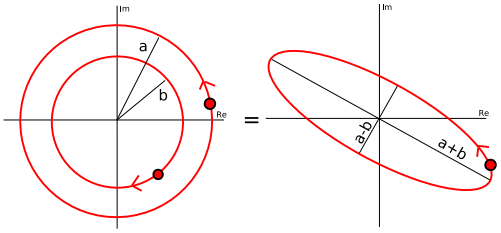


Cualquier función periódica $z : [0, 1] \to \mathbb{C}$ se puede representar como una curva cerrada en el plano complejo, donde la variable independiente (parámetro) queda 'aplastada' y no se observa en el dibujo. (Podemos marcar la trayectoria de alguna manera para indicar la 'velocidad de trazado'.) Su descomposición en frecuencias puras se puede interpretar como la suma de un conjunto de trayectorias circulares, de diferentes tamaños, sentidos, puntos de partida y velocidades de giro:

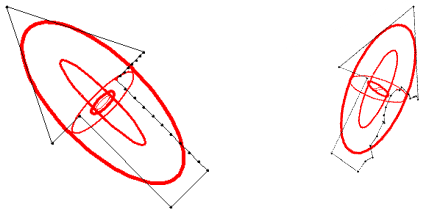


Es como si la figura de la izquierda se dibujara con el extremo de una cadena de lápices, cada uno dando vueltas cada vez más rápidamente desde la punta del anterior.

La interpretación de cada componente frecuencial puede mejorarse todavía más si consideramos conjuntamente las frecuencias opuestas $F_\omega H_\omega(t) + F_{-\omega} H_{-\omega}(t)$. Las trayectorias circulares que giran a la misma velocidad en sentidos opuestos se combinan en una *elipse* que da el mismo número de vueltas ω para $t \in [0, 1]$. El tamaño de los ejes, el sentido de giro y el punto de partida dependen de los coeficientes F_ω y $F_{-\omega}$ de los armónicos constituyentes:



El análisis frecuencial del contorno de una figura es el fundamento del método de reconocimiento de formas estudiado en la sección 2.3. Cada silueta se caracteriza por su conjunto de elipses constituyentes, que están rígidamente unidas a la figura cuando ésta sufre desplazamientos, rotaciones y escalados uniformes. Además, las componentes de baja frecuencia son muy robustas frente al ruido:



Es muy simple obtener invariantes frente a esos cambios basados en propiedades de esas elipses.

B.5. Serie de Fourier

¿Cómo calculamos las coordenadas frecuenciales $\boldsymbol{F}$ de una función cualquiera $f(x)$? Si estuviéramos en un espacio vectorial ordinario las coordenadas de un vector en una base ortonormal se calcularían simplemente como el producto escalar del vector por cada uno de los elementos de la base (ver Sección A.2, pág. 180). En un espacio de funciones el producto escalar se define mediante una integral (las funciones son una especie de vectores con un continuo de coordenadas). Por ejemplo, en $\mathcal{L}^2(a,b)$ (las funciones de cuadrado sumable en el intervalo (a,b)) un producto escalar puede ser:

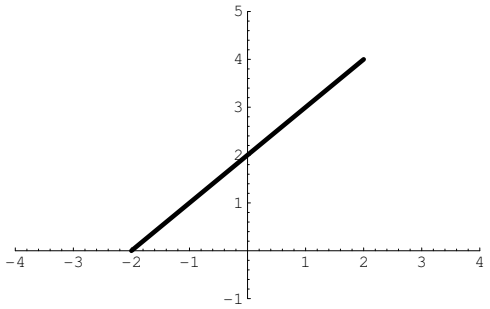
$$f \cdot g = \int_a^b f(x)\overline{g(x)}dx$$

Con este producto escalar la base de exponenciales complejas $H_\omega(x) = \frac{1}{\sqrt{b-a}} e^{i2\pi \frac{\omega}{b-a} x}$ con $\omega = -\infty \ldots \infty$ es ortonormal en $\mathcal{L}^2(a,b)$ (se trata de la familia anterior pero ajustada al intervalo (a,b) y normalizada). Las coordenadas de cualquier función f en esta base se calculan de forma inmediata como $F_\omega = H_\omega \cdot f$, o más explícitamente:

$$F_\omega = \frac{1}{\sqrt{b-a}} \int_a^b f(x) e^{-i2\pi \frac{\omega}{b-a} x} dx$$

que son los coeficientes de la *serie de Fourier* de $f(x)$ en esta base.

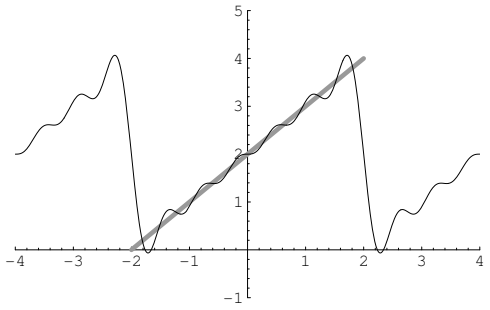
Veamos un ejemplo. La siguiente función es una rampa definida en el intervalo (-2,2):



Sus coordenadas en la base de exponenciales complejas son $\boldsymbol{F} = [\ldots, \frac{\frac{4\,i}{3}}{\pi}, \frac{-2\,i}{\pi}, \frac{4\,i}{\pi}, 4, \frac{-4\,i}{\pi}, \frac{2i}{\pi}, \frac{\frac{-4\,i}{3}}{\pi}, \ldots]$. Como la función de interés es sencilla podemos conseguir una expresión general para las coordenadas:

$$F_0 = 4 \qquad F_\omega = \frac{4i(-1)^\omega}{\pi\omega}$$

Para comprobar que son correctas podemos construir una aproximación a la función utilizando unos cuantos términos, p. ej., desde $\omega = -6$ hasta $\omega = +6$:



Fuera del intervalo (a, b) la expansión reconstruye periódicamente la función original. Como todos los elementos de la base son periódicos y continuos, la aproximación con pocas componentes produce unos artefactos (oscilaciones de Gibbs) cerca de las discontinuidades, donde la serie completa converge al valor medio de los dos extremos. (Más adelante estudiaremos una base de representación ligeramente distinta capaz de reducir significativamente este defecto.)

Observa que cada pareja de términos correspondientes a $-\omega$ y $+\omega$ se combina para dar lugar a un trozo de señal real:

$$f(x) = \sum_{\omega=-\infty}^{+\infty} F_\omega H_\omega(x) = 2 + \underbrace{\frac{2ie^{i\pi\frac{-1}{2}x}}{\pi} + \frac{-2ie^{i\pi\frac{1}{2}x}}{\pi}}_{\frac{4}{\pi}\sin\frac{\pi x}{2}} + \underbrace{\frac{-ie^{i\pi(-1)x}}{\pi} + \frac{ie^{i\pi x}}{\pi}}_{\frac{-2}{\pi}\sin\pi x} + \ldots$$

Se obtiene el mismo resultado si utilizamos la base $\{\frac{1}{\sqrt{b-a}}\cos(2\pi\frac{\omega}{b-a}x), \frac{1}{\sqrt{b-a}}\sin(2\pi\frac{\omega}{b-a}x)\}$, ortonormal en $\mathcal{L}^2(a,b)$ con $\omega = 0\ldots\infty$, que da lugar a la serie de Fourier trigonométrica:

$$f(x) = \sum_{\omega=0}^{\infty}\left(\alpha_\omega \cos\frac{\pi\omega x}{l} + \beta_\omega \sin\frac{\pi\omega x}{l}\right)$$

con coordenadas reales α_ω y β_ω:

$$\alpha_\omega = \frac{1}{l}\int_{-l}^{+l} f(x)\cos\frac{\pi\omega x}{l}dx \qquad \beta_\omega = \frac{1}{l}\int_{-l}^{+l} f(x)\sin\frac{\pi\omega x}{l}dx$$

Por casualidad, en este ejemplo todos los coeficientes α_ω (salvo la constante) son cero (la función es esencialmente *impar*).

B.6. Integral de Fourier

Desde un punto de vista teórico, podríamos estar interesados en trabajar con funciones no periódicas, definidas sobre toda la recta real. En este caso no podemos construir una base numerable de funciones trigonométricas o exponenciales complejas (ni tampoco de polinomios, aunque sí de otras funciones que decrecen suficientemente rápido en el infinito, como las funciones de Hermite o las de Laguerre). No obstante, sí es posible conseguir una representación integral en términos de un conjunto continuo ($\omega \in \mathbb{R}$) de armónicos:

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(w)e^{i\omega x}d\omega$$

No sólo aparecen las frecuencias enteras $\omega = \ldots, -2, -1, 0, 1, 2, \ldots$ sino que también hay componentes de todas las frecuencias intermedias.

Informalmente, esta expresión se consigue llevando al límite $l \to \infty$ el tamaño del intervalo de la serie de Fourier anterior. Observa también que el coeficiente 2π, que se utilizaba para ajustar el período de la función al intervalo deseado, ya no es necesario.

Las (incontables) coordenadas $F(w)$ se calculan de nuevo como el producto escalar de la función y cada elemento de la 'base':

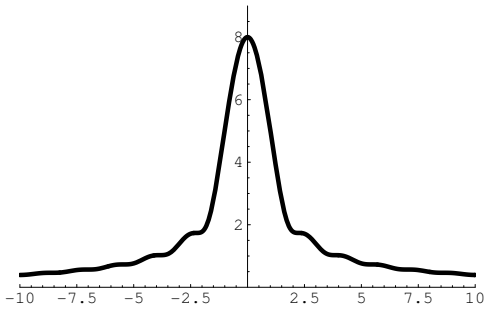
$$F(\omega) = \int_{-\infty}^{+\infty} f(x)e^{-i\omega x}dx$$

La función $F(\omega)$ es la *Transformada de Fourier* de $f(x)$.

La Transformada de Fourier de la rampa anterior es:

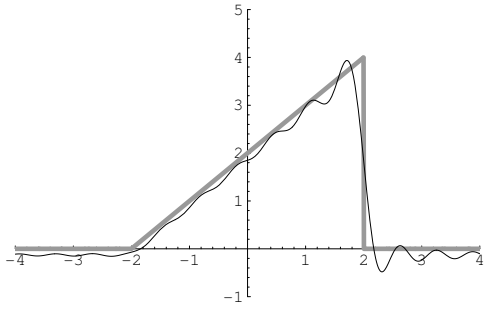
$$F(\omega) = \frac{1}{\omega^2}\left(e^{-2i\omega} - e^{2i\omega} + 4i\omega e^{-2i\omega}\right)$$

Como es una función compleja podemos representar el módulo $|F(\omega)|$ para hacernos una idea de la distribución de frecuencias:



De hecho, la función $|F(\omega)|^2$ es el *espectro de potencia* o *densidad espectral* de la función f, que nos indica la amplitud A^2 de cada armónico (relacionada con la energía de la señal).

La mejor manera de comprobar que la descomposición es correcta es intentar reconstruir la señal original. Mediante una aproximación numérica a la integral anterior (en la que hemos incluido sólo las frecuencias entre $\omega = -10$ y $\omega = +10$ a saltos $\Delta\omega = 0.1$) comprobamos que mediante la integral de Fourier conseguimos representar la señal no periódica en todo el dominio $\mathbb{R}$:



Si $f(x)$ es periódica, entonces su transformada integral $F(\omega)$ es cero en todas las frecuencias intermedias, no enteras (respecto al intervalo considerado). Sólo intervienen los mismos armónicos que aparecerían en la serie de Fourier [1].

[1] Como veremos después la transformada de una exponencial compleja es un impulso (δ de Dirac).

Las únicas componentes sinusoidales periódicas en un intervalo, p. ej., $[0, 1)$, tienen que ser del tipo $A\cos(2\pi\omega x + \phi)$ con ω entero. En caso contrario, por ejemplo con $\omega = 2.5$, o cualquier otro valor fraccionario, la onda no realizará un número completo de repeticiones. Su serie de Fourier extenderá la reconstrucción periódicamente, repartiendo la carga en frecuencias cercanas, y se producirá el efecto de Gibbs.

B.7. Muestreo y reconstrucción

En la mayoría de las aplicaciones prácticas no tenemos funciones continuas como tales, descritas mediante expresiones matemáticas, sino que obtenemos un conjunto finito de muestras o mediciones de alguna magnitud física de interés. En concreto, supongamos que tomamos n muestras equiespaciadas $\boldsymbol{f} = \{f_0, f_1, \ldots, f_{n-1}\}$ de una funcion $f(x)$ en el intervalo $[a, b)$, donde $f_k = f(a + k\Delta x)$ y $\Delta x = \frac{b-a}{n}$. (Muchas veces conviene normalizar el dominio de la función haciendo $a = 0$ y $b = 1$ y entonces $f_k = f(\frac{k}{n})$.)

La pregunta crucial es: ¿podemos reconstruir completamente la función original, en cualquier punto intermedio, a partir únicamente de sus muestras?

Evidentemente, la respuesta es negativa, a menos que dispongamos de algún tipo de conocimiento externo sobre la clase de funciones que pueden entrar en juego. P. ej., si la función objetivo es algún polinomio de grado no superior a g –y las muestras no están contaminadas con ruido– podemos calcular los coeficientes del polinomio a partir de $n \leq g + 1$ muestras. Por supuesto, en muchas aplicaciones prácticas no disponemos de información tan detallada sobre nuestros datos que, además, pueden estar contaminados con ruido.

En cierto modo, este problema es el mismo que tratan de resolver, desde otro punto de vista, las máquinas de aprendizaje inductivo estudiadas en la Sección 4.

Sería muy útil caracterizar de forma poco restrictiva la clase de funciones que podemos reconstruir con precisión. En el contexto del análisis armónico estamos interesados en algo como lo siguiente: ¿podemos calcular todas las coordenadas frecuenciales de $f(x)$ a partir sólo de la señal muestreada $\boldsymbol{f}$? Con ellas podríamos reconstruir la función en cualquier punto y, por supuesto, analizar sus propiedades frecuenciales o de otro tipo.

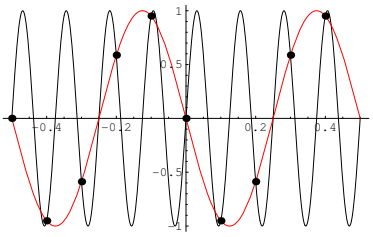
Lógicamente, al disponer solo de un número finito de muestras solo podemos aspirar a reconstruir la función en el intervalo observado (aunque la reconstrucción en una base armónica extenderá la función periódicamente fuera de él). Por tanto, no tiene sentido intentar capturar las componentes frecuenciales no enteras ya que son idénticamente nulas en una función periódica. Sólo necesitamos las frecuencias discretas de la serie de Fourier.

La respuesta a esta pregunta es afirmativa, si se cumplen las condiciones de uno de los resultados más importantes de la teoría de la información y la comunicación: el *Teorema del Muestreo* (Shannon).

La función original debe tener *ancho de banda* limitado, lo que significa que no tiene armónicos superiores a una cierta frecuencia ω_m, esto es, $F(\omega) = 0$ para $|\omega| > \omega_m$. En otras palabras, la función no cambia infinitamente rápido. En estas condiciones, si el intervalo de muestreo es lo suficientemente pequeño como para detectar la frecuencia más alta contenida en la señal, conseguiremos capturar la función sin pérdida de información y reconstruirla exactamente. En concreto, dada una señal definida en el intervalo $x \in [0, 1)$, si la componente armónica de frecuencia más alta es $A\cos(2\pi\omega_m x + \phi)$ necesitamos $n > 2\omega_m$ muestras (la separación entre ellas debe ser $\Delta x < \frac{1}{2\omega_m}$).

> Para demostrar este resultado solo hay que tener en cuenta que muestrear una señal (multiplicarla por un tren de impulsos) equivale a la convolución (ver Sec. B.13) de su espectro con un tren de impulsos tanto más separados en el dominio frecuencial cuanto más juntos estén en el dominio espacial. Esta convolución crea réplicas del espectro que no deben solaparse para poder recuperarlo.

Este tipo de condición impuesta a la frecuencia máxima de la señal, relacionada con la máxima pendiente que puede aparecer, es razonable y poco restrictiva. La detección de cambios rápidos en la señal requiere necesariamente un muestreo muy denso del dominio. Si no se realiza aparece el fenómeno de *aliasing*. P. ej., con 10 muestras la frecuencia más alta que podemos capturar es 4. Una señal de frecuencia 8 se confunde una de frecuencia -2:



En cierto sentido, la interpolación de las muestras se realizará de la manera menos oscilatoria posible compatible con los datos.

B.8. Transformada de Fourier Discreta (DFT)

Supongamos entonces que hemos muestreado suficientemente rápido la función de interés. ¿Cómo calculamos sus coordenadas frecuenciales a partir únicamente de las n muestras $\boldsymbol{f} = \{f_i\}_{i=0}^{n-1}$?

Consideremos las funciones armónicas utilizadas hasta el momento como base del espacio de funciones continuas:

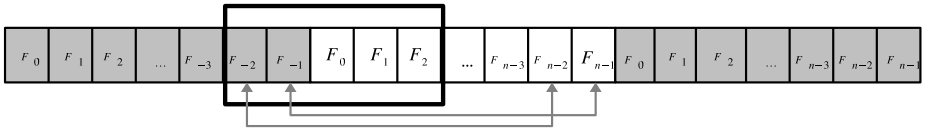
$$H_\omega(x) = e^{i2\pi\omega x}$$

Un resultado fundamental es que las versiones discretizadas con cualquier número n de muestras, que denotaremos como $\boldsymbol{h}_\omega$, son también vectores ortonormales de $\mathbb{R}^n$. Estos vectores se obtienen muestreando cada función en el intervalo $[0, 1)$ donde es periódica y normalizando:

$$\boldsymbol{h}_\omega = \left\{ \frac{1}{\sqrt{n}}\, e^{i2\pi\omega\frac{k}{n}} \right\}_{k=0}^{n-1}$$

Como antes, necesitamos frecuencias positivas y negativas para manejar correctamente la fase. Una base de $\mathbb{R}^n$ contiene n vectores. Y de acuerdo con el teorema del muestreo, la frecuencia más alta que podemos capturar en el intervalo de trabajo es (menos de) la mitad de las muestras disponibles. Por tanto, la matriz $\mathtt{H} = \{\boldsymbol{h}_\omega^\mathsf{T}\}$ con $\omega = -\lfloor\frac{n-1}{2}\rfloor, \ldots, -2, -1, 0, 1, 2, \ldots, \lfloor\frac{n-1}{2}\rfloor$ es una base ortonormal de $\mathbb{R}^n$ [2]. Esta base tiene la interesante propiedad de que proporciona las coordenadas frecuenciales de la señal continua original a partir de la señal muestreada. Cada una de ellas se obtiene como $F_\omega = \boldsymbol{h}_\omega^\mathsf{T}\boldsymbol{f}$ y el conjunto de todas ellas se puede expresar sencillamente como $\boldsymbol{F} = \mathtt{H}\boldsymbol{f}$.

> Es importante tener en cuenta que en lugar de organizar las frecuencias en orden creciente como:
>
> $$\boldsymbol{F} = \{F_{-\frac{n-1}{2}}, F_{-1}, F_0, F_1, \ldots, F_{\frac{n-1}{2}}\}$$
>
> los algoritmos estándar que calculan la DFT nos suministran las frecuencias en el siguiente orden:
>
> $$\boldsymbol{F} = \{F_0, F_1, \ldots, F_{n-1}\}$$
>
> aprovechando el hecho de que transformada discreta es periódica[3] y que, por tanto, las frecuencias negativas aparecen en la parte superior de la transformada:



[2] Esto encaja perfectamente para n impar. Cuando n es par, para completar la base de $\mathbb{R}^n$ se incluye la frecuencia $\frac{n}{2}$, (da igual positiva o negativa) multiplicada por dos. En realidad, de la componente de frecuencia $\pm\frac{n}{2}$ sólo se puede capturar su 'parte coseno'. La 'parte seno' se confunde con la señal idénticamente nula. Con n par capturamos sólo 'media' frecuencia $\frac{n}{2}$.

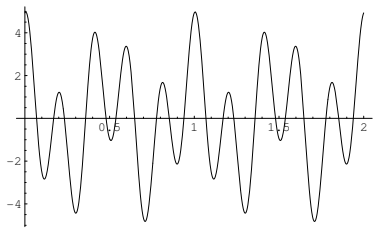
[3] Las frecuencias superiores a $\frac{n}{2}$, que por supuesto no podemos capturar, en su versión discreta coinciden (por el fenómeno de *aliasing*) con las frecuencias inferiores negativas que necesitamos.

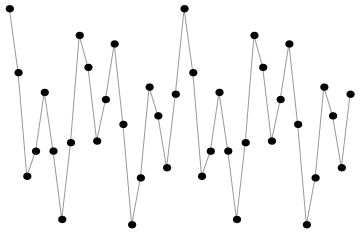
En concreto, $F_{-\omega} = F_{n-\omega}$, o también, si $\boldsymbol{F}$ es un array con primer índice cero, $F_\omega = \boldsymbol{F}[\omega \mod n]$:

$$\boldsymbol{F} = \{F_0, F_1, F_2, \ldots, \underbrace{F_{n-3}}_{F_{-3}}, \underbrace{F_{n-2}}_{F_{-2}}, \underbrace{F_{n-1}}_{F_{-1}}\}$$

Observa que, en principio, la complejidad del cálculo de la DFT es $O(n^2)$. Afortunadamente, es posible organizar los cálculos para que cueste sólo $O(n \log n)$, lo que se conoce como la *Transformada de Fourier Rápida* (FFT). Todos los entornos de cálculo científico suministran algoritmos muy eficientes para calcular la FFT. Pueden diferir en las constantes de normalización utilizadas.

Veamos un ejemplo. Consideremos la función $f(x) = 2\cos(2\pi 2x + 0.1) + 3\cos(2\pi 5x - 0.2)$ y tomemos 40 muestras en $[0, 2)$ ($\Delta x = 0.05$), $\boldsymbol{f} = \{4.93, 2.09, -2.52, \ldots, -2.14, 1.13\}$:





La FFT de la señal (redondeando un poco los valores) es[4]:

$$\boldsymbol{F} \simeq \{0, 0, 0, 0, \underbrace{0.9 - 0.1i}_{F_4}, 0, 0, 0, 0, 0, \underbrace{1.4 + 0.3i}_{F_{10}}, \overbrace{0, \ldots 0}^{19\text{ ceros}}, \underbrace{1.4 - 0.3i}_{F_{30}=F_{-10}}, 0, 0, 0, 0, 0, \underbrace{0.9 + 0.1i}_{F_{36}=F_{-4}}, 0, 0, 0\}$$

El módulo y el argumento de $\boldsymbol{F}$ codifican las amplitudes y fase de las componentes cosinusoidales de $f(x)$:

$$|\boldsymbol{F}| = \{0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1.5, \overbrace{0, \ldots 0}^{19\text{ ceros}}, 1.5, 0, 0, 0, 0, 0, 1, 0, 0, 0\}$$

La amplitud ($2 = F_4 + F_{-4}$) de la componente $2\cos(2\pi 2\omega x + 0.1)$ aparece en la posición ± 4 porque el intervalo de trabajo $[0, 2)$ contiene 4 períodos de la frecuencia $\omega = 2$, definida entre 0 y 1. Análogamente, la amplitud de la otra componente ($3 = F_{10} + F_{-10}$), de frecuencia $\omega = 5$ aparece en la posición ± 10. En general, las posiciones de la FFT indican el número de repeticiones de

[4] Para ajustar la escala, en Matlab/Octave (Sección F) debemos dividir el resultado de `fft` por n y tomar cambiar el signo de la componente imaginaria. En Mathematica debemos dividir el resultado de `Fourier` por $\sqrt{n}$.

la componente en el intervalo muestreado. Por supuesto, si hubiéramos muestreado la señal en $[0, 1)$ las posiciones de la FFT corresponderían directamente a las frecuencias expresadas como $2\pi\omega x$ en la señal[5].

El argumento de la FFT nos indica la fase de cada componente (para las componentes con $|F_\omega| = 0$ el argumento no está definido):

$$|\arg F| = \{\cdot, \cdot, \cdot, \cdot, -0.1, \cdot, \cdot, \cdot, \cdot, \cdot, 0.2, \overbrace{\cdot, \ldots, \cdot}^{19}, -0.2, \cdot, \cdot, \cdot, \cdot, \cdot, 0.1, \cdot, \cdot, \cdot\}$$

B.9. Onda 2D

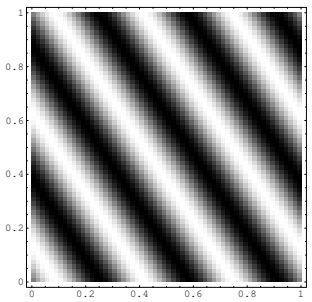
Una función de varias variables $f(\boldsymbol{x})$, donde $\boldsymbol{x} = (x_1, x_2, x_3, \ldots)$, también se puede expresar en una base de componentes periódicos de tipo armónico:

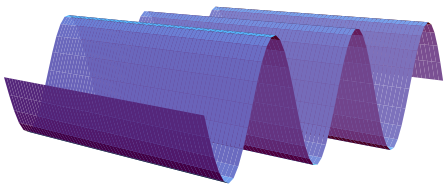
$$H(\boldsymbol{\omega}) = e^{i2\pi\boldsymbol{\omega}\cdot\boldsymbol{x}}$$

caracterizados por una'frecuencia vectorial' $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3, \ldots)$, cuyas componentes son las frecuencias de la onda en cada dirección. En el caso 2D tenemos $\boldsymbol{x} = (x, y)$, $\boldsymbol{\omega} = (u, v)$ y cada componente frecuencial es una onda plana:

$$H(u, v) = e^{i2\pi(ux+vy)}$$

En la figura siguiente se muestra como ejemplo una onda de frecuencia $\boldsymbol{\omega} = (3, 2)$ (observa que hay 3 repeticiones horizontales y 2 verticales):





Todos los conceptos anteriores se mantienen igual en el caso multivariable, con la ventaja de que por la propiedad de *separabilidad* las transformadas n-D se pueden computar mediante múltiples transformadas 1D. Sólo debemos tener en cuenta que

[5]Cuando aparecen frecuencias intermedias se distribuyen en varias posiciones cercanas. Pero ten en cuenta que si en un intervalo de muestreo aparecen frecuencias intermedias, no enteras, entonces la extensión periódica de la señal será discontinua, dando lugar a distorsiones como las que aparecían en la función rampa.

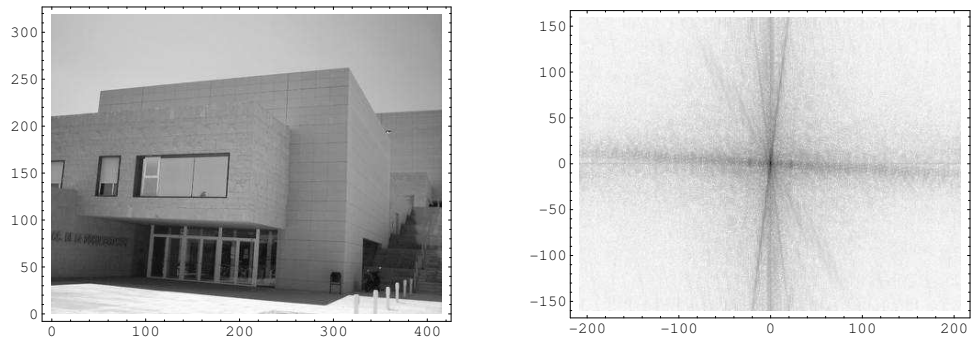
las funciones base son superficies sobre el plano, especificadas por los dos índices u y v y, que la reconstrucción requiere por tanto una suma doble.

P. ej., en el caso discreto, si tenemos una función muestreada en una malla de tamaño $n \times m$:

$$f_{i,j} = \sum_{u=0}^{n-1} \sum_{v=0}^{m-1} F_{u,v}\, e^{i2\pi\left(u\frac{x}{n}+v\frac{y}{m}\right)}$$

(de nuevo, por la periodicidad de la transformada los límites de los sumatorios son equivalentes a utilizar frecuencias desde $u = 0$ hasta $\pm n/2$ y desde $v = 0$ hasta $\pm m/2$). Cada frecuencia (u, v) se asocia con la opuesta $(-u, -v)$ para reconstruir una onda plana real.

Las coordenadas frecuenciales de una imagen (una función de dos variables que devuelve el color o el nivel de gris de cada pixel) se pueden representar como otra imagen:



(Hemos rotado filas y columnas de la transformada para que las bajas frecuencias queden en el centro de la imagen y además hemos representado $\log(1 + |F_{u,v}|)$ para ampliar el rango de valores que podemos apreciar en la imagen.) Se observan ciertas frecuencias predominantes, en las direcciones perpendiculares a los bordes más destacados de la imagen original.

B.10. Propiedades de la Transformada de Fourier

Antes de continuar repasaremos rápidamente algunas propiedades de la Transformada de Fourier. (Mientras no indiquemos lo contrario, estas propiedades son válidas para la serie, la integral y la DFT). Usaremos el siguiente convenio: $F(\omega) = \mathcal{F} f(x)$.

- *Linealidad.* La transformada de la suma de varias señales ponderadas se obtiene directamente a partir de las transformadas de cada componente:

$$\mathcal{F}\left[\alpha f(x) + \beta g(x)\right] = \alpha F(\omega) + \beta G(\omega)$$

En el caso discreto la transformación puede expresarse simplemente como un producto matriz vector (aunque no es computacionalmente tan eficiente como la FFT).

- *Separabilidad dimensional.* La transformada de una señal multidimensional se puede calcular mediante sucesivas transformaciones 1D. P. ej., la transformada de una imagen se puede hacer calculando la transformada de todas las filas y luego la de todas las columnas resultantes del paso anterior.
- *Traslación.* Al desplazar la señal en el espacio (lo que en el caso de una función muestreada, por la suposición de periodicidad es equivalente a 'rotar' la lista de muestras) la transformada sólo cambia su fase, pero no el módulo. Es lógico porque la fase de la onda es relativa a la situación, más o menos arbitraria del origen de coordenadas. La energía de las diferentes componentes frecuenciales, indicada por el módulo de las coordenadas frecuenciales no cambia:

$$\mathcal{F}f(x-a) = e^{i\,a\,\omega}F(\omega) \qquad \mathcal{F}^{-1}F(\omega-\phi) = e^{-i\,\phi\,x}f(x)$$

- *Escalado.* Al estirar la función en el eje x su transformada no cambia más que en un 'reajuste' del tamaño de las frecuencias que tenía anteriormente:

$$\mathcal{F}f(ax) = \frac{1}{|a|}F\left(\frac{\omega}{a}\right)$$

- *Rotación (en 2D).* Debido a que cada componente $F(u,v)$ representa la contribución de una onda plana en dirección (u,v), si rotamos una 'imagen' la transformada rota exactamente igual:
- *Valor medio.* La frecuencia cero (componente constante) nos da la media de la señal:

$$F(0) = \mathrm{E}\{f(x)\}$$

- *Dualidad.* Las propiedades del operador $\mathcal{F}$ y de su inverso son esencialmente iguales. El dominio espacial y frecuencial son intercambiables:

$$f(x) \to^{\mathcal{F}} F(\omega) \to^{\mathcal{F}} f(-t)$$

- *Parseval.* La transformación es ortonormal y por tanto preserva la norma:

$$\int_{\mathbb{R}} |f(x)|^2 dx = \int_{\mathbb{R}} |F(\omega)|^2 d\omega$$

- *Simetría conjugada.* Como vimos en el manejo de la fase, si la señal es real cada pareja de frecuencias $\pm\omega$ contribuye a una componente tipo coseno.

$$F(-\omega) = \overline{F(\omega)}$$

- *Periodicidad DFT.* Por el fenómeno de *aliasing* la DFT es periódica, y las frecuencias negativas se confunden con las positivas:

$$F_{-\omega} = F_{n-\omega}$$

- *Convolución.* La convolución en el dominio espacial se reduce a una multiplicación elemento a elemento en el dominio frecuencial:

$$\mathcal{F}(f * g)(x) = F(\omega)G(\omega)$$

  (algo similar ocurre con la operación de 'correlación'). Cuando hay que aplicar a una señal (p. ej., una imagen) una máscara de convolución de tamaño muy grande, puede ser más rápido hacer una multiplicación ordinaria en el dominio frecuencial.

- *Derivación.* La siguiente propiedad es muy útil para resolver ecuaciones diferenciales (aunque en ciertos casos es mejor usar la transformada de Laplace):

$$\mathcal{F} f'(x) = i\omega F(\omega)$$

  La transformada 2D del laplaciano de una función es por tanto:

$$\mathcal{F}\,\nabla^2 f(x, y) = -(2\pi)^2(u^2 + v^2)F(u, v)$$

- *Transformada de una gaussiana.* Es otra gaussiana de anchura recíproca:

$$\mathcal{F}\mathcal{N}_\sigma(x) = \mathcal{N}_{\frac{1}{\sigma}}(\omega)$$

  Un suavizado de kernel gaussiano deja pasar menos frecuencias cuanto mayor sea el radio de promediado.

- *Transformada de una onda pura.* Es un impulso:

$$\mathcal{F}\cos(2\pi a x) = \frac{1}{2}\left[\delta(\omega + a) + \delta(\omega - a)\right]$$

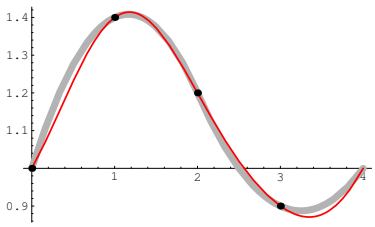
  Por eso las funciones periódicas se pueden representar como una suma de frecuencias discretas.

B.11. Transformada Discreta del Coseno (DCT)

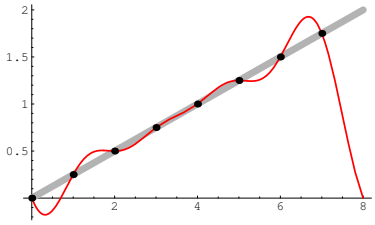
¿Realmente podemos reconstruir la función en puntos no observados? Si cumplimos el criterio de Nyquist muestreando a una frecuencia doble que la de la mayor componente de la señal, trivialmente podremos reconstruir la función, ya que su serie de Fourier no tendrá componentes más altas que las que se pueden capturar con n elementos de la base del espacio discreto $\mathbb{R}^n$ (se necesita una pareja, positiva y negativa, para cada frecuencia). La cuestión, en realidad, es si es fácil cumplir esta condición.

Si la señal es periódica y 'suave', sí es cierto que las frecuencias altas serán despreciables. P. ej., la siguiente señal está formada por tramos de parábola ajustados para

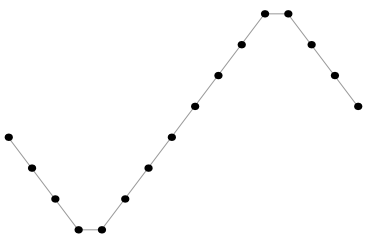
que el recorrido completo sea suave. Las funciones cuadráticas no tienen mucho que ver con la base de senos y cosenos pero, aún así, la reconstrucción con sólo 4 muestras es excelente:



Si la señal es suave pero no es realmente periódica (como en el caso de la rampa anterior), o si tiene otro tipo de discontinuidades (la propia función o su pendiente), entonces aparecerán componentes distintas de cero en todas las frecuencias. Por supuesto, siempre se podrá conseguir una aproximación aceptable usando un número suficientemente grande de componentes. Pero la condición de periodicidad es bastante restrictiva, ya que casi nunca tendremos la suerte de que la señal observada acabe a la misma altura, y con la misma pendiente, que donde empezó. La no periodicidad crea artefactos que pueden ser inaceptables:



Una variante de la Transformada Discreta de Fourier que resuelve bastante bien este inconveniente es la Transformada Discreta del Coseno (DCT). Está basada en una idea muy ingeniosa que resuelve dos problemas de una sola vez. Dado que la muestra de interés no tiene por qué ser periódica, podemos hacerla periódica nosotros concatenando una imagen especular de la función original. Esto garantiza que no hay discontinuidades en la version periódica de la función (aunque sí puede haber discontinuidad en la pendiente). Por ejemplo, en el caso de la rampa, la señal modificada que podríamos representar sería la siguiente (con media imagen especular por cada extremo):



Las coordenadas frecuenciales de esta señal desdoblada tienen muchas menos componentes de alta frecuencia ya que no es necesario ajustar bien ninguna discontinuidad. Pero, además, la señal desdoblada es par (simétrica respecto al eje vertical) y, por tanto, sólo aparecen las componentes de tipo coseno; en definitiva, todas las coordenadas frecuenciales son reales[6]. Por ejemplo, el módulo de la DFT de la rampa anterior discretizada en 8 muestras es:

$$\{0.88, 0.33, 0.18, 0.14, 0.13\}$$

(omitimos las 3 últimas ya que por simetría conjugada se deducen de $F_1 \ldots F_3$). Todas las componentes frecuenciales son necesarias para conseguir una aproximación que no es excesivamente buena. La función de reconstrucción continua es:

$$0.88 - 0.25\cos(\frac{\pi x}{4}) - 0.25\cos(\frac{\pi x}{2}) - 0.25\cos(\frac{3\pi x}{4}) - 0.13\cos(\pi x) - 0.60\sin(\frac{\pi x}{4}) - 0.25\sin(\frac{\pi x}{2}) - 0.10\sin(\frac{3\pi x}{4})$$

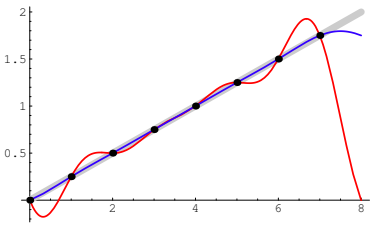
La DCT de la misma función es:

$$\{2.47, -1.61, 0, -0.17, 0, -0.05, 0, -0.01\}$$

A diferencia con la DFT, tiene varias componentes idénticamente nulas y poco peso en las de alta frecuencia. La función de reconstrucción correspondiente es:

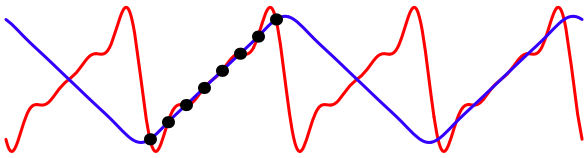
$$0.88 - 0.81\cos(\frac{\pi\ (1+2x)}{16}) - 0.08\cos(\frac{3\pi\ (1+2x)}{16}) - 0.03\cos(\frac{5\pi\ (1+2x)}{16}) - 0.006\cos(\frac{7\pi\ (1+2x)}{16})$$

La reconstrucción producida por la DCT (en azul) es mejor que la de la DFT (en rojo):



La extensión periódica producida por las dos funciones es diferente, siendo mucho más favorable la utilizada por la DCT:

[6] En el caso discreto para que realmente la función desdoblada sea par hay que desplazar la señal *media* muestra hacia la derecha, dejando el eje vertical entre las dos copias de la primera muestra. Ese desplazamiento se simula multiplicando cada componente F_ω de la DFT por $e^{i2\pi\omega(\frac{1}{2n}/2)}$



Los coeficientes de la DCT pueden obtenerse de manera sencilla a partir de la FFT de la señal desdoblada. El coste de la DCT es por tanto algo superior al de la FFT ordinaria pero está justificado por la calidad de la representación obtenida.

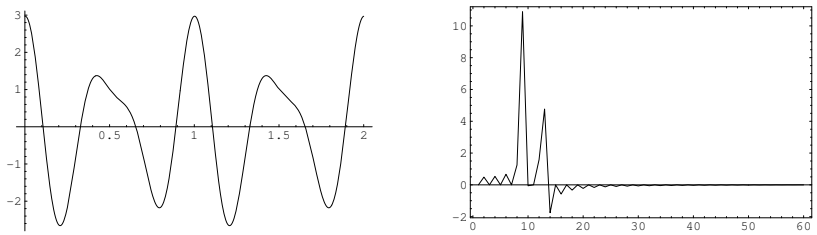
> En múltiples aplicaciones los elementos de la base están precalculados, por lo que no existe ninguna penalización. En el formato jpeg la imagen se particiona en bloques 8×8, cuya base de representación en la DCT 2D están precalculados.

Todas estas razones justifican que la DCT es el estándar de compresión de imágenes utilizado en los formatos jpeg, video digital, etc. En la Sección 6 se muestra el aspecto de la DCT 2D de varias imágenes.

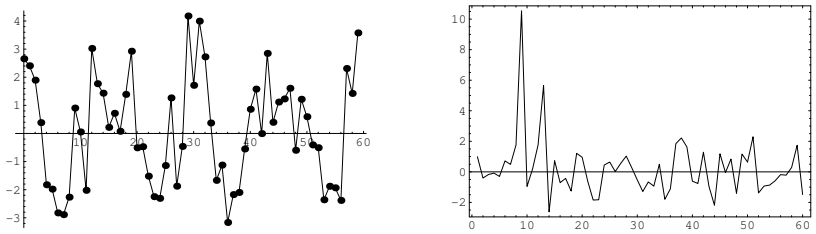
B.12. Filtrado

Una vez descompuesta una señal en sus componentes frecuenciales tenemos la posibilidad de modificar esta representación para conseguir los efectos deseados. Esta operación se denomina *filtrado*. Es exactamente lo mismo que hace el ecualizador de un equipo musical. Podemos atenuar o amplificar bandas de frecuencia según nuestros intereses.

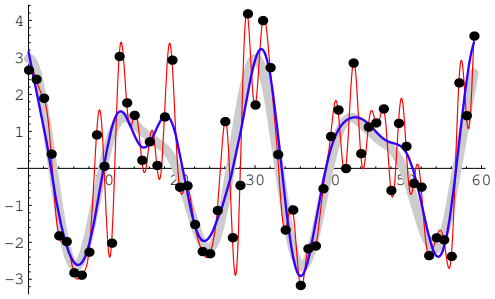
Una aplicación típica es la eliminación de ruido. Supongamos que estamos interesados en una señal suave como la siguiente, cuya DCT muestra simplemente dos picos claros alrededor de las frecuencias 8/2 y 12/2:



Desgraciadamente, en el proceso de muestreo se añade una componente de ruido aleatorio. La señal original queda completamente distorsionada y en la DCT observada aparecen componentes en todas las frecuencias (ruido blanco: mezcla de todos los colores/frecuencias):



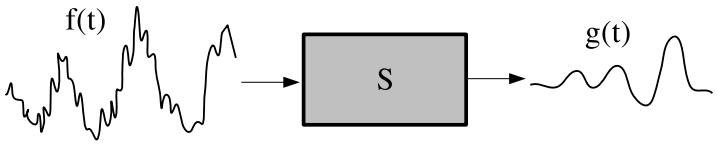
Por supuesto, es imposible eliminar perfectamente el ruido: por una parte la señal original siempre tiene alguna componente de alta frecuencia; y el ruido blanco tiene también componentes de baja frecuencia como los de la señal original. A pesar de todo, la eliminación o atenuación de los compontentes de alta frecuencia (digamos, a partir de 20) permite reconstruir una versión bastante aceptable de la señal original:



Este proceso se puede aplicar también a señales multidimensionales. En la sección Sección 6 se muestran algunos ejemplos de filtrado frecuencial de imágenes.

B.13. Convolución

Considera un sistema 'dinámico' S, cuya entrada es una función $f(t)$ (supongamos por ahora que depende del tiempo) y su respuesta es otra función $S[f(t)] = g(t)$:



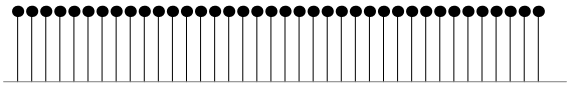
La clase más simple de sistemas dinámicos son los llamados 'lineales e invariantes a desplazamiento' (SLI). Ante una superposición de funciones responden superponiendo las respuestas individuales:

$$S[af_1(t) + bf_2(t)] = aS[f_1(t)] + bS[f_2(t)]$$

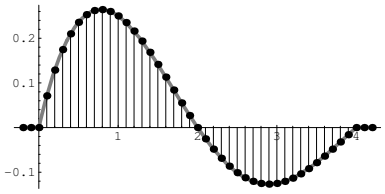
$$\text{Si } S[f(t)] = g(t) \text{ entonces } S[f(t-a)] = g(t-a)$$

Estos sistemas tan simples son una idealización que no cumple ningún sistema real, pero que es bastante aceptable en ciertos rangos de funcionamiento. Matemáticamente son muy cómodos de analizar: al ser lineales, la respuesta queda determinada por la respuesta que tienen frente a los elementos de una base (cualquiera) del espacio de funciones.

Una especie de base elemental es el 'tren de impulsos' (infinitamente pegados):

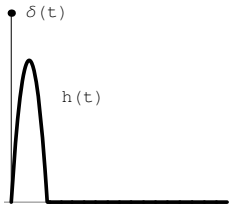


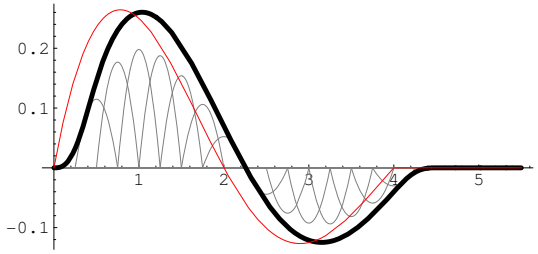
Cualquier señal se puede interpretar como un tren de impulsos que van ponderados por los valores que toma la señal en cada posición:



$$f(t) = \int_{-\infty}^{+\infty} f(a)\delta(t-a)da$$

Por tanto, por linealidad e invarianza, si frente a un único impulso $\delta(t)$ el sistema reacciona con una respuesta $h(t) = S[\delta(t)]$ entonces frente a todo el tren ponderado con los valores de la función la respuesta del sistema será simplemente la superposición de la respuesta a los diferentes impulsos que van llegando, cada uno con su ponderación y su retraso en el tiempo:





Por esta razón, un SLI se caracteriza completamente por su respuesta al impulso $h(t)$, y la operación anterior que nos da la salida del sistema para una entrada cualquiera $f(t)$ se denomina *convolución* de $f(t)$ con $h(t)$:

$$g(t) = f(t) * h(t) = \int_{-\infty}^{+\infty} f(a)h(t-a)da$$

En el caso de señales muestreadas la integral se sustituye por un sumatorio (convolución discreta).

Teorema de la convolución. Otra base interesante es la de componentes frecuenciales $H_\omega(x)$, cuyas coordenadas se obtienen mediante la transformada de Fourier, que da lugar a un resultado fundamental de la teoría de sistemas: un SLI queda completamente determinado por su 'función de transferencia', que indica cómo se modifica, individualmente, cada componente frecuencial de la señal original. Si $\mathcal{F}f(t) = F(\omega)$ y $\mathcal{F}h(t) = H(\omega)$ entonces:

$$\mathcal{F}(f * h)(t) = F(\omega)H(\omega)$$

Este resultado se deduce fácilmente de las propiedades de la exponencial:

$$(f * h)(t) = \int_a f(a)h(t-a)da =$$

$$= \int_a f(a)\left[\int_\omega H(\omega)\exp(i\omega(t-a))d\omega\right]da = \int_a f(a)\left[\int_\omega H(\omega)\exp(i\omega t)\exp(-i\omega a)d\omega\right]da =$$

$$= \int_\omega\int_a f(a)H(\omega)\exp(i\omega t)\exp(-i\omega a)da\,d\omega = \int_\omega H(\omega)\exp(i\omega t)\underbrace{\int_a f(a)\exp(-i\omega a)da}_{F(\omega)}d\omega =$$

$$= \int_\omega F(\omega)H(\omega)\exp(i\omega t)d\omega$$

La respuesta de un SLI a una función armónica es una función armónica de la misma frecuencia, aunque con una amplitud y fase diferente. (Un SLI no puede cambiar la frecuencia de una señal.) En otras palabras, las exponenciales complejas son las *funciones propias* de los SLI; constituyen una base en la que el SLI tiene forma diagonal. Son el equivalente a los autovectores (Sección A.3) en un espacio de dimensión infinita.

Si tenemos que implementar un SLI podemos hacerlo mediante una convolución en el dominio espacial/temporal o mediante un producto ordinario en el dominio frecuencial, lo que resulte computacionalmente más económico. (Existe un resultado equivalente relacionado con la *correlación* entre dos señales.)

En la práctica suelen implementarse filtros paso bajo, paso alto o paso banda mediante máscaras de convolución. También se aplican filtros no lineales, que modifican cada muestra mediante algoritmos más generales que una simple suma ponderada de las muestras vecinas. En el apartado 6 se muestran algunos ejemplos de procesamiento digital de imágenes en el dominio espacial y frecuencial.

Filtrado inverso. Supongamos que nuestra señal ha sufrido un cierto proceso que se puede modelar bien mediante un SLI. P. ej., un emborronado o desenfoque, una foto movida, etc. En principio, si conocemos con precisión el efecto sufrido sobre un único pixel (la respuesta al impulso o *point-spread function*), podemos deshacerlo en el dominio de la frecuencia. Si $g(t) = (f * h)(t)$, $\mathcal{F}h(t) = H(\omega)$ y $\mathcal{F}g(t) = G(\omega)$ entonces:

$$f(t) = \mathcal{F}^{-1}\left[\frac{G(\omega)}{H(\omega)}\right]$$

Este proceso de 'desconvolución' trata de reconstruir las modificaciones sufridas por el espectro de frecuencias de la señal. El problema surge cuando la función de transferencia $H(\omega)$ se hace cero o muy pequeña. Las frecuencias que no deja pasar no pueden reconstruirse. Se han propuesto algunos métodos de restauración de imágenes basados en la optimización de ciertos criterios [ref]. En general consiguen resultados aceptables solo cuando la degradación es pequeña y conocida. (En muchas situaciones reales la degradación ni siquiera es lineal o invariante a desplazamiento.)

B.14. Wavelets

Pendiente.

B.15. Alineamiento de contornos

En el apartado 2.3 se propuso un sencillo vector de propiedades de formas planas basado en la transformada de Fourier del contorno, que es invariante a posición, rotación, escala, punto de partida en la polilínea y pequeñas deformaciones. Sólo requiere que los nodos estén aproximadamente equiespaciados. En realidad, a partir de las componentes frecuenciales es posible definir una función de distancia entre dos siluetas que tiene un significado geométrico muy razonable, y que puede adaptarse fácilmente para trabajar con polilíneas irregularmente espaciadas. Mediante una normalización previa puede hacerse invariante a transformaciones afines (escalados diferentes en dos direcciones), lo que permite modelar deformaciones de perspectiva débil.

B.15.1. Distancia entre contornos

- distancia ideal
- mse (Parseval)
- se mantiene en rotación, podemos usar coordenadas frecuenciales
- interesa normalizar (quizá escala) rotación y sobre todo punto de partida

- el valor absoluto resuelve, pero pierde mucha info (se queda solo con los tamaños de las elipses)

B.15.2. Invarianza afín

Para conseguir un representante afín de un contorno podemos aplicar una transformación de *whitening* (C.2). Para ello necesitamos la media y matriz de covarianza

$$\boldsymbol{\mu} = (\mu_x, \mu_y) \qquad \Sigma = \begin{bmatrix} c_{xx} & c_{xy} \\ c_{xy} & c_{yy} \end{bmatrix}$$

de los infinitos puntos $\boldsymbol{z} = (x, y)$ que constituyen el interior de la figura. (Dentro de un momento veremos cómo se puede calcular $\boldsymbol{\mu}$ y Σ de forma sencilla.) La transformación de normalización afín de la figura se obtiene a partir de los valores y vectores propios de $\Sigma = \mathtt{V}^\mathsf{T}\Lambda\mathtt{V}$:

$$\boldsymbol{z}' = \Lambda^{-\frac{1}{2}}\mathtt{V}^\mathsf{T}(\boldsymbol{z} - \boldsymbol{\mu})$$

Aunque Λ y $\mathtt{V}$ se pueden calcular mediante las herramientas de álgebra lineal usuales, al tratar con objetos de 2 dimensiones podemos obtener la descomposición de forma cerrada. En realidad sólo hay 3 parámetros esenciales, los dos valores propios $\lambda_1 \geq \lambda_2$ y el ángulo θ que forma la dirección principal:

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \qquad \mathtt{V} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

que se pueden obtener directamente de los elementos de Σ:

$$\lambda_1 = \frac{1}{2}(c_{xx} + c_{yy} + r) \qquad \lambda_2 = \frac{1}{2}(c_{xx} + c_{yy} - r) \qquad \theta = \arctan\frac{c_{xx} - c_{yy} + r}{2c_{xy}}$$

donde

$$r^2 = c_{xx}^2 + + c_{yy}^2 + 4c_{xy}^2 - 2c_{xx}c_{yy}$$

B.15.3. Cálculo de μ y Σ de una región a partir de su contorno.

Consideremos una figura plana $\mathcal{S}$. Los parámetros que nos interesan pueden definirse a partir de los momentos estadísticos:

$$m_{pq} = \iint_{\mathcal{S}} x^p y^q dx dy$$

A partir de ellos podemos obtener los momentos centrados (m_{00} es el área de la figura):

$$m'_{pq} = \frac{m_{pq}}{m_{00}}$$

Y entonces la media (centro de masas) es

$$\mu = (\mu_x, \mu_y) = (m'_{10}, m'_{01})$$

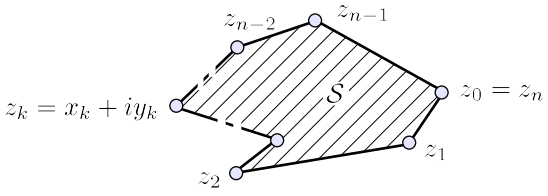
y los elementos de la matriz de covarianza (relacionada con los momentos de inercia de la figura) son:

$$c_{xx} = m'_{20} - (m'_{10})^2 \qquad c_{yy} = m'_{02} - (m'_{01})^2 \qquad c_{xy} = m'_{11} - m'_{10}m'_{01}$$

En principio puede parecer necesario calcular explícitamente las integrales dobles que definen los momentos estadísticos m_{pq} pero, afortunadamente, usando el Teorema de Green[7] pueden sustituirse por integrales simples sobre el contorno[8]:

$$m_{pq} = \iint_{\mathcal{S}} x^p y^q dxdy = \frac{1}{2} \oint_Z \frac{x^p y^{q+1}}{q+1} dx - \frac{x^{p+1} y^q}{p+1} dy$$

Representaremos el contorno de la región $\mathcal{S}$ mediante una polilínea con n nodos $Z = \{z_0, z_1, \dots z_{n-1}\}$, donde $z_k = (x_k, y_k)$. Para facilitar la notación cerraremos el contorno con un nodo adicional $z_n = z_0$.



La integral de contorno puede descomponerse en las contribuciones de cada tramo recto:

$$m_{pq} = \frac{1}{2} \sum_{k=0}^{n-1} \int_{z_k}^{z_{k+1}} \frac{x(t)^p y(t)^{q+1}}{q+1} \frac{dx}{dt} dt - \frac{x(t)^{p+1} y(t)^q}{p+1} \frac{dy}{dt} dt = \frac{1}{2} \sum_{k=0}^{n-1} A_k - B_k$$

En cada tramo desde z_k hasta z_{k+1} el contorno se puede parametrizar como $z(t) = z_k + (z_{k+1} - z_k)t$, donde $t \in (0, 1)$, por lo que podemos escribir:

[7] Es la particularización al plano del Teorema de Stokes, la extensión multidimensional del Teorema Fundamental del Cálculo.

[8] La derivación presentada aquí debería cambiarse para usar el sentido positivo (antihorario) de recorrido del contorno.

$$A_k = \int_0^1 \frac{(x_k + (x_{k+1} - x_k)t)^p (y_k + (y_{k+1} - y_k)t)^{q+1}}{q+1} (x_{k+1} - x_k) dt$$

$$B_k = \int_0^1 \frac{(x_k + (x_{k+1} - x_k)t)^{p+1} (y_k + (y_{k+1} - y_k)t)^q}{p+1} (y_{k+1} - y_k) dt$$

Estas integrales son muy fáciles de calcular para valores pequeños de p y q. En concreto, los términos $(A_k - B_k)/2$ para los valores de p y q que nos interesan son:

$$\begin{aligned}
m_{00} &\to (x_1 y_2 - x_2 y_1)/2 \\
m_{10} &\to (2x_1 x_2 (y_2 - y_1) - x_2^2 (2y_1 + y_2) + x_1^2 (2y_2 + y_1))/12 \\
m_{01} &\to (-2y_1 y_2 (x_2 - x_1) + y_2^2 (2x_1 + x_2) - y_1^2 (2x_2 + x_1))/12 \\
m_{20} &\to ((x_1^2 x_2 + x_1 x_2^2)(y_2 - y_1) + (x_1^3 - x_2^3)(y_1 + y_2))/12 \\
m_{02} &\to (-(y_1^2 y_2 + y_1 y_2^2)(x_2 - x_1) - (y_1^3 - y_2^3)(x_1 + x_2))/12 \\
m_{11} &\to ((x_1 y_2 - x_2 y_1)(x_1 (2y_1 + y_2) + x_2 (y_1 + 2y_2)))/24)
\end{aligned}$$

donde el subíndice 1 representa la posición k y el 2 la posición $k+1$ de cada coordenada en el tramo que empieza en z_k. Los momentos estadísticos de la región $\mathcal{S}$ se obtienen acumulando estas expresiones a lo largo de todo su contorno.

Una alternativa es calcular la media y covarianza sólo del contorno de la figura (considerada como una especie de alambre), sin tener en cuenta los puntos del interior. Las expresiones necesarias son un poco más sencillas, pero no dan lugar a un invariante afín exacto.

→ interpretación geométrica alternativa como suma y resta de áreas.

B.15.4. Serie de Fourier exacta de una función lineal a trozos

El procedimiento habitual para obtener invariantes espectrales es aproximar los coeficientes de la serie de Fourier (para funciones continuas) mediante la transformada discreta. Esto es aceptable cuando el número de puntos es suficientemente grande y están uniformemente espaciados. Pero podría objetarse que no tiene sentido almacenar todos los puntos de un tramo rectilíneo de contorno, deberían ser suficientes los extremos. Además, si construimos una versión invariante afín, se producirá un estiramiento anisotrópico en los nodos, invalidando los resultados que se obtendrían usando una FFT ordinaria. Sería conveniente obtener las coordenadas frecuenciales a partir de una polilínea arbitraria, cuyos nodos se pueden encontrar a diferentes distancias. Hay que optar entre un muestreo uniforme (lo que implicar un posible remuestreo de la versión afín invariante) que permita usar la FFT (complejidad $O(n \log n)$), o el uso de polilíneas reducidas con espaciado irregular, mediante un procedimiento más

laborioso (complejidad $O(Mn')$) para obtener el número deseado M (normalmente pequeño, del orden de 10) de coordenadas, donde n' es el número de puntos de la polilínea reducida. La solución más eficiente dependerá de los requisitos de cada aplicación, pero en cualquier caso es interesante tener la posibilidad de manejar directamente polilíneas reducidas. A continuación se muestra un algoritmo para calcular la serie de Fourier exacta de una función periódica compleja lineal a trozos, definida por el conjunto de nodos $Z = \{z_0, z_1, \ldots z_{n-1}\}$, añadiendo igual que antes un nodo $z_n = z_0$ para cerrar la figura.

Cada componente frecuencial es una suma de tramos rectos desde (t_k, x_k) hasta (t_{k+1}, z_{k+1}):

$$F_\omega = \int_0^1 e^{-2\pi i\omega t} z(t)dt = \sum_{k=0}^{n-1} \int_{t_k}^{t_{k+1}} e^{-2\pi i\omega t}(z_k + \frac{z_{k+1} - z_k}{t_{k+1} - t_k}(t - t_k))dt$$

Los valores del parámetro de integración t_k en cada nodo son la fracción de longitud recorrida desde el punto inicial z_0. Por tanto $t_0 = 0$, $t_n = 1$ y $t_k = t_{k-1} + |z_k - z_{k-1}|/l$, donde l es la longitud de la curva. (Es preciso recorrer el contorno una primera vez para calcular todos los t_k, la contribución de cada tramo depende de su posición *absoluta* en el recorrido.)

En principio es posible obtener los F_ω deseados resolviendo directamente la integral anterior. Si se desea un algoritmo eficiente es conveniente reorganizar los sumandos para conseguir que tomen una forma más sencilla en función de unos valores que se pueden precalcular y son válidos para todos los ω.

Veamos cómo puede hacerse. En primer lugar escribimos

$$F_\omega = \sum_{k=0}^{n-1} \int_{t_k}^{t_{k+1}} e^{-2\pi i\omega t}(\beta_k + \alpha_k t)dt$$

en función de unos términos α_k y β_k que no dependen de ω y pueden precalcularse una sola vez a partir de los z_k originales:

$$\alpha_k \equiv \frac{z_{k+1} - z_k}{t_{k+1} - t_k}, \qquad \beta_k \equiv z_k - \alpha_k t_k$$

La integral de cada tramo es:

$$F_\omega = \sum_{k=0}^{n-1} \frac{\beta_k}{-2\pi i\omega}(e^{-2\pi i\omega t_{k+1}} - e^{-2\pi i\omega t_k}) +$$

$$+ \sum_{k=0}^{n-1} \frac{\alpha_k}{-2\pi i\omega}(e^{-2\pi i\omega t_{k+1}}(t_{k+1} - \frac{1}{-2\pi i\omega}) - e^{-2\pi i\omega t_k}(t_k - \frac{1}{-2\pi i\omega}))$$

Si definimos $H_k \equiv e^{-2\pi i t_k}$, que también se puede precalcular, y $\tau_k(\omega) \equiv t_k + 1/(2\pi i\omega)$, podemos simplificar un poco la expresión anterior:

$$F_\omega = \frac{1}{-2\pi i\omega}\sum_{k=0}^{n-1}[\beta_k(H_{k+1}^\omega - H_k^\omega) + \alpha_k(H_{k+1}^\omega\tau_{k+1}(\omega) - H_k^\omega\tau_k(\omega))]$$

Sin embargo, no tiene mucho sentido operar 4 veces en cada tramo con los H_k^ω. Teniendo en cuenta factores comunes en términos consecutivos del sumatorio y aprovechando que $\alpha_0 = \alpha_n$, $H_0^\omega = H_n^\omega = 1$, y que $-\beta_0 - \tau_0(\omega)\alpha_0 = -\beta_n - \tau_n(\omega)\alpha_n$ (lo que nos permite tratar los extremos sueltos como un término n normal), podemos reagrupar los términos así:

$$F_w = \frac{1}{-2\pi i\omega}\sum_{k=1}^{n} H_k^\omega[(\beta_{k-1} - \beta_k) + \tau_k(\omega)(\alpha_{k-1} - \alpha_k)]$$

de modo que podemos precalcular algo más: $A_k \equiv \alpha_{k-1} - \alpha_k$, $B_k \equiv \beta_{k-1} - \beta_k$ y $C_k \equiv B_k + t_k A_k$.

$$F_w = \frac{1}{-2\pi i\omega^2}\sum_{k=1}^{n} H_k^\omega(\omega C_k + \frac{A_k}{2\pi i})$$

Es faćil comprobar que los coeficientes C_k son idénticamente nulos, por lo que finalmente conseguimos una expresión bastante compacta[9]:

$$F_w = \frac{1}{(2\pi\omega)^2}\sum_{k=1}^{n} H_k^\omega A_k$$

La componente cero requiere un tratamiento especial:

$$F_0 = \frac{1}{2}\sum_{k=0}^{n-1}(z_{k+1} + z_k)(t_{k+1} - t_k)$$

A partir del contorno original z_k se precalculan las secuencias t_k y α_k, y con ellas H_k y A_k, lo que permite obtener después cada componente frecuencial mediante solo 3 operaciones aritméticas (complejas) por cada nodo.

[9] Agradezco a Miguel Ángel Davó la notificación de que los términos C_k se cancelan, y a Adrián Amor la observacion sobre la agrupación de términos consecutivos del sumatorio y la detección de varias erratas.

B.15.5. Ejemplo

La imagen siguiente muestra unas cuantas formas planas y sobreimpresas sus versiones invariantes afines (aunque mantenemos la posición, tamaño y orientación para poder comparar las figuras):

[ej]

La imagen siguiente muestra el alineamiento afín de una forma considerada como prototipo sobre diferentes vistas con una inclinación muy marcada, lo que sugiere que un modelo afín puede ser aceptable incluso para deformaciones proyectivas moderadas.

[ej]

Apéndice C

Probabilidad

The true logic of this world is in
the calculus of probabilities.

–J. C. Maxwell

Existen experimentos cuya repetición en condiciones esencialmente iguales produce resultados diferentes; decimos que son *aleatorios*. Muchos de ellos cumplen bastante bien un modelo *estocástico*: aunque cada resultado concreto es impredecible, su frecuencia relativa de aparición en una secuencia infinita de repeticiones tiende a un límite que, curiosamente, es el mismo para cualquier subsecuencia previamente especificada. Esos límites se llaman *probabilidades*.

La Teoría de la Probabilidad [13, 19] nos ayuda a construir un modelo abstracto, aproximado, de un fenómeno físico. Es útil en el 'mundo real' porque es razonable admitir que los sucesos con una probabilidad 'muy pequeña' no ocurren en un único intento. Si para explicar un fenómeno proponemos una hipótesis de la que se deducen probabilidades muy bajas para un cierto suceso, y ese suceso ocurre, debemos rechazar la hipótesis (este es uno de los fundamentos del Método Científico).

Hay otras aproximaciones muy interesantes a la aleatoriedad, como p. ej., la *aleatoriedad algorítmica* [15], de acuerdo con la cual una secuencia finita se considera aleatoria si no puede comprimirse.

C.1. Cálculo de Probabilidades

Los posibles resultados concretos de un experimento aleatorio se llaman *resultados elementales*. P. ej., al lanzar una moneda los resultados elementales son {'cara', 'cruz'}; al lanzar un dado, los resultados elementales son {'1','2'... '6'}. Un *suceso* es un conjunto de resultados elementales. P. ej., *par* = {'2','4','6'}. De acuerdo con las condiciones del problema se asigna a cada resultado elemental un número positivo, su

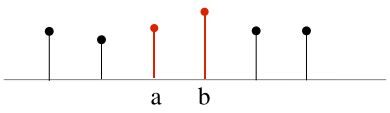
probabilidad, de manera que entre todos sumen uno. P. ej., si el dado está equilibrado todas las caras tienen probabilidad $1/6$; una moneda cargada podría tener probabilidades $\mathrm{P}\{\text{'cara'}\} = 0.6$ y $\mathrm{P}\{\text{'cruz'}\} = 0.4$. La probabilidad de un suceso es la suma de las probabilidades de los resultados elementales que contiene. P. ej., en un dado equilibrado $\mathrm{P}\{\textit{par}\} = 1/2$.

Cuando el conjunto de resultados elementales de un experimento es continuo (p. ej., la posición de un dardo en una diana) las probabilidades no se asignan a posiciones concretas sino a intervalos o regiones.

Como su propio nombre indica, el Cálculo de Probabilidades tiene como objetivo calcular las probabilidades de unos sucesos de interés a partir otras probabilidades conocidas.

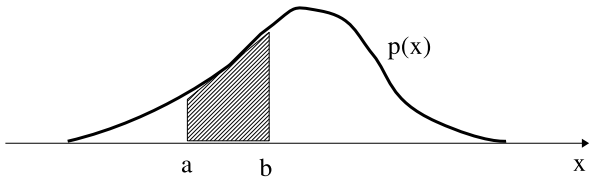
Variable aleatoria. Es conveniente etiquetar cada resultado de un experimento aleatorio con un código númerico. Cualquier variable aleatoria x queda perfectamente caracterizada por su función de distribución de probabilidad acumulada $F(z) = \mathrm{P}\{x \leq z\}$, que indica la probabilidad de que una realización del experimento produzca un resultado menor o igual que cualquier valor z. Es una forma cómoda de resumir las probabilidades asignadas a todos los sucesos elementales del experimento.

Densidad de Probabilidad. En el caso discreto (p. ej., el lanzamiento de un dado), la probabilidad de un suceso es la suma de las probabilidades que lo componen:



$$\mathrm{P}\{a \cup b\} = \mathrm{P}\{a\} + \mathrm{P}\{b\}$$

Las variables aleatorias continuas se pueden caracterizar también por su función de densidad de probabilidad $p(x)$. La probabilidad de que la variable aparezca en una región del espacio es la integral en esa región de la función de densidad:



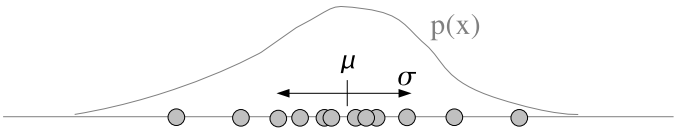
$$\mathrm{P}\{x \in (a,b)\} = \int_a^b p(x)dx$$

Valor esperado. Cada resultado individual de una variable aleatoria es impredecible, pero el valor esperado (el promedio sobre todos los posibles resultados) de funciones de la variable es una magnitud concreta, bien definida, que *puede* conocerse:

$$\mathrm{E}_{p(x)}\{f(x)\} = \int_{-\infty}^{+\infty} f(x)p(x)dx$$

(cuando la densidad involucrada $p(x)$ se sobreentiende no aparece explícitamente en la notación.)

Media y varianza. Aunque toda la información sobre una variable aleatoria continua x está contenida implícitamente en su función de densidad $p(x)$, en muchos casos es suficiente con tener una idea de su localización y dispersión, lo que puede expresarse con la media $\mu = \mathrm{E}\{x\}$ y la desviación media σ o la varianza $\sigma^2 = \mathrm{E}\{(x-\mu)^2\}$.



Estos parámetros pueden estimarse a partir de múltiples observaciones de la variable aleatoria.

Promedios. El valor esperado del promedio de n observaciones de una variable aleatoria no modifica la media pero divide su desviación por $\sqrt{n}$, haciendo 'menos aleatorio' el resultado. Esto permite estimar parámetros de una variable aleatoria con la precisión deseada (si tenemos muestras suficientes).

Desigualdad de Markov. Cuando una variable aleatoria es positiva ($x > 0$):

$$\mathrm{P}\{x > a\} < \frac{\mathrm{E}\{x\}}{a}$$

Desigualdad de Tchevichev. Permite acotar la probabilidad de que una realización de una variable aleatoria se aleje r desviaciones de la media:

$$\mathrm{P}\{|x-\mu| > r\sigma\} < \frac{1}{r^2}$$

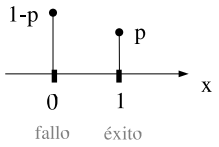
Por ejemplo, la probabilidad de que *cualquier* variable aleatoria (con varianza finita) se aleje más de dos desviaciones de la media es menor del 25 %. Si la variable es *normal* (véase la Sección C.3, pág. 228) la probabilidad es concretamente del 4.6 %. Y si la variables es uniforme, la probabilidad es cero.

Ley de los grandes números. Las frecuencias tienden (¡en un sentido probabilístico!) a las probabilidades. La desigualdad de Tchevichev nos ofrece una estimación cuantitativa de esta convergencia:

$$\mathrm{P}\{|\hat{p}_n - p| > \epsilon\} < \frac{1}{4n\epsilon^2}$$

En esta expresión p es la probabilidad real de un suceso y $\hat{p}_n$ es la frecuencia observada en n realizaciones. P. ej., con un 99 % de confianza $|p - \hat{p}_n| < \sqrt{20/n}$.

El resultado anterior se consigue utilizando la desigualdad de Tchevichev sobre una variable aleatoria de tipo Bernoulli, cuyos resultados son éxito ($x = 1$), con probabilidad p, o fracaso ($x = 0$), con probabilidad $1 - p$.



Esta variable aleatoria tiene media p y varianza $p(1-p) < 1/4$. Por tanto, $\hat{p}_n$ también tiene media p y la varianza se reduce a $p(1-p)/n < 1/4n$.

Momentos estadísticos. Son los valores esperados de las potencias de la variable aleatoria: $m_n = \mathrm{E}\{x^n\}$. Permiten calcular el valor esperado de cualquier función (expresada como serie de potencias). Contienen más o menos la misma información que la densidad de probabilidad, pero en una forma que podría estimarse mediante promedios.

El momento de orden 1 es la media ($\mu = m_1$), el de orden 2 está relacionado con la con la varianza ($\sigma^2 = m_2 - m_1^2$), el de orden 3 con la asimetría de la densidad (*skewness*), el de orden 4 con el 'afilamiento' (*kurtosis*), etc.

Estadística. Sirve para hacer inferencias sobre las densidades de probabilidad de una variable aleatoria, o sobre parámetros importantes de ella, a partir de experimentos aleatorios.

Muestra aleatoria. La estadística utiliza normalmente muestras aleatorias *i.i.d.*, cuyas observaciones son *independientes* e *igualmente distribuidas* ('intercambiables').

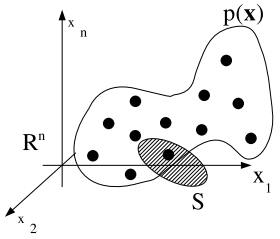
$$p(x_1, x_2, \ldots x_n) = \prod_i p(x_i)$$

Temas relacionados

- Calcula la distancia media entre observaciones de una población y la desviación de esa distancia media (calcular la dispersión pero sin referirla a la media).
- ¿Cuál es la media de la suma de dos variables aleatorias? ¿Y la varianza?
- ¿Cómo generaramos números (pseudo)aleatorios con cualquier distribución de probabilidad?

C.2. Descripción de poblaciones aleatorias multidimensionales

Densidad conjunta. Toda la información sobre una variable aleatoria n-dimensional $\boldsymbol{x}$ está incluida en su función de densidad de probabilidad conjunta $p(\boldsymbol{x}) = p(x_1, x_2, \ldots, x_n)$. Nos permite calcular la probabilidad de que la variable aparezca en cualquier región:

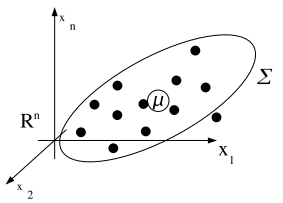


$$\mathrm{P}\{\boldsymbol{x} \in S\} = \int_S p(\boldsymbol{x}) d\boldsymbol{x}$$

Media y matriz de covarianza. En muchas aplicaciones prácticas la función de densidad es desconocida. Aunque existen métodos de estimación de densidades de probabilidad a partir de muestras aleatorias, la estimación de densidades multidimensionales es un problema muy complejo.

A veces es suficiente disponer de una información básica sobre la variable: su localización y dispersión. La localización puede describirse simplemente mediante el vector de medias de cada componente: $\boldsymbol{\mu} = \mathrm{E}\{\boldsymbol{x}\}$.

Para medir la 'dispersión multidimensional' no sólo necesitamos la varianza de cada componente por separado, sino también los coeficientes de covarianza de cada pareja de ellos. Esto es así para poder determinar la 'anchura' de la 'nube' de puntos en cualquier dirección del espacio. El conjunto de varianzas y convarianzas se representa en la matriz de covarianza Σ:



$$\Sigma = \mathrm{E}\{(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^{\mathsf{T}}\}$$

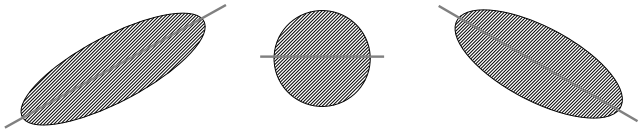
Cada elemento de la matriz Σ se define como $\sigma_{i,j} = \mathrm{E}\{(x_i - \mu_i)(x_j - \mu_j)\}$. En la diagonal están las varianzas de cada variable $\sigma_{i,i} = \sigma_i^2$, y fuera de la diagonal los coeficientes de covarianza σ_{ij} de cada pareja de variables (la matriz es simétrica):

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1i} & \dots & \sigma_{1j} & \dots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \dots & \sigma_{2i} & \dots & \sigma_{2j} & \dots & \sigma_{2n} \\ \vdots & \vdots & \ddots & & & & & \\ \sigma_{i1} & \sigma_{i2} & \dots & \sigma_i^2 & \dots & \sigma_{ij} & \dots & \sigma_{in} \\ \vdots & \vdots & & & \ddots & & & \\ \sigma_{j1} & \sigma_{j2} & \dots & \sigma_{ji} & \dots & \sigma_j^2 & \dots & \sigma_{jn} \\ \vdots & \vdots & & & & & \ddots & \\ \sigma_{n1} & \sigma_{n2} & \dots & \sigma_{ni} & \dots & \sigma_{nj} & \dots & \sigma_{nn} \end{bmatrix}$$

La matriz de covarianza también se puede escribir como $\Sigma = \mathtt{S} - \boldsymbol{\mu}\boldsymbol{\mu}^\mathsf{T}$ donde $\mathtt{S} = \mathrm{E}\{\boldsymbol{x}\boldsymbol{x}^\mathsf{T}\}$ es la matriz de *autocorrelación*. Es la versión multivariable de la típica expresión $\sigma^2 = \mathrm{E}\{x^2\} - \mathrm{E}\{x\}^2$.

Coeficiente de correlación lineal. La dependencia *lineal* entre cada pareja de variables x_i y x_j se puede medir mediante el coeficiente de correlación $\rho_{i,j} = \frac{\sigma_{i,j}}{\sigma_i \sigma_j} \in [0, 1]$.

Si $\rho_{ij} \simeq 1$, cuando x_i crece también tiende a hacerlo x_j. Si $\rho_{ij} \simeq -1$, cuando una variable crece la otra decrece. Finalmente, cuando $\rho_{ij} \simeq 0$ no hay dependencia *lineal* entre las variables:



$\rho \simeq +1$ $\rho \simeq 0$ $\rho \simeq -1$

El hecho de que $\rho = 0$ no significa que las variables sean estadísticamente independientes. Pueden tener una dependencia *no lineal* indetectable por el coeficiente de correlación.

Matriz de covarianza de una transformación lineal. Si una población $\boldsymbol{x}$ tiene matriz de covarianza $\Sigma_{\boldsymbol{x}}$, la matriz de covarianza $\Sigma_{\boldsymbol{y}}$ de una transformación lineal $\boldsymbol{y} = \mathtt{A}\boldsymbol{x}$ (ver Sec. A.1) es simplemente $\Sigma_{\boldsymbol{y}} = \mathtt{A}\,\Sigma_{\boldsymbol{x}}\,\mathtt{A}^\mathsf{T}$. Además, $\boldsymbol{\mu}_{\boldsymbol{y}} = \mathtt{A}\boldsymbol{\mu}_{\boldsymbol{x}}$.

Esto se comprueba fácilmente teniendo en cuenta que el valor esperado es un operador lineal:

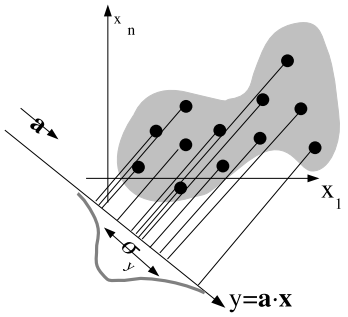
$$\mu_y = \mathrm{E}\{\boldsymbol{y}\} = \mathrm{E}\{\mathtt{A}\boldsymbol{x}\} = \mathtt{A}\mathrm{E}\{\boldsymbol{x}\} = \mathtt{A}\boldsymbol{\mu}_x$$

$$\Sigma_y = \mathrm{E}\{(\boldsymbol{y}-\boldsymbol{\mu}_y)(\boldsymbol{y}-\boldsymbol{\mu}_y)^\mathsf{T}\} = \mathrm{E}\{(\mathtt{A}\boldsymbol{x}-\mathtt{A}\boldsymbol{\mu}_x)(\mathtt{A}\boldsymbol{x}-\mathtt{A}\boldsymbol{\mu}_x)^\mathsf{T}\} =$$
$$= \mathrm{E}\{\mathtt{A}(\boldsymbol{x}-\boldsymbol{\mu}_x)(\boldsymbol{x}-\boldsymbol{\mu}_x)^\mathsf{T}\mathtt{A}^\mathsf{T}\} = \mathtt{A}\mathrm{E}\{(\boldsymbol{x}-\boldsymbol{\mu}_x)(\boldsymbol{x}-\boldsymbol{\mu}_x)^\mathsf{T}\}\mathtt{A}^\mathsf{T} = \mathtt{A}\Sigma_x\mathtt{A}^\mathsf{T}$$

Ejercicios:

- La expresión anterior nos permite calcular la distancia media a puntos cualesquiera. Por ejemplo, calcula $E\{||\boldsymbol{x}||^2\}$, $E\{d^2(\boldsymbol{x},\boldsymbol{p})\}$, $E\{d^2(\boldsymbol{x},\boldsymbol{\mu})\}$, $E\{d^2(\boldsymbol{x}_1,\boldsymbol{x}_2)\}$, $E\{(\boldsymbol{x}-\boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\}$, etc., usando las propiedades siguientes: $\mathrm{trz}(\boldsymbol{x}\cdot\boldsymbol{y}^\mathsf{T}) = \boldsymbol{x}^\mathsf{T}\boldsymbol{y}$, $\mathrm{trz}(\mathtt{AB}) = \mathrm{trz}(\mathtt{BA})$, $||\boldsymbol{x}||^2 = \boldsymbol{x}^\mathsf{T}\cdot\boldsymbol{x}$, $d(\boldsymbol{x},\boldsymbol{p}) = ||\boldsymbol{x}-\boldsymbol{p}||$. La traza de una matriz $\mathtt{A}$, $\mathrm{trz}\,\mathtt{A}$, es la suma de los elementos de la diagonal.

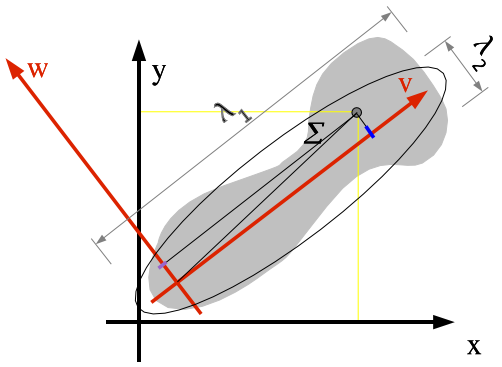
Direcciones principales. La media de las proyecciones $y = \boldsymbol{a}^\mathsf{T}\boldsymbol{x}$ de una población de vectores $\boldsymbol{x}$ sobre una dirección $\boldsymbol{a}$ es $\mu_y = \boldsymbol{a}^\mathsf{T}\boldsymbol{\mu}_x$ y la varianza de las proyecciones es $\sigma_y^2 = \boldsymbol{a}^\mathsf{T}\Sigma_x\boldsymbol{a}$:



Para cualquier dirección, definida por un vector unitario $\boldsymbol{a}$, tenemos una proyección de los elementos $\boldsymbol{x}$ y una varianza asociada a esa proyección. Es importante saber en qué direcciones una población tiene mayor (y menor) dispersión. Es fácil demostrar que los vectores propios de la matriz de covarianza Σ son las direcciones principales de la población y los valores propios correspondientes son las varianzas de la población en esas direcciones.

> Hemos visto que cuando las observaciones sufren una transformación $\boldsymbol{y} = \mathtt{A}\boldsymbol{x}$ la matriz de covarianza cambia a $\Sigma_y = \mathtt{A}\Sigma_x\mathtt{A}^\mathsf{T}$. Un tipo especial de transformación lineal es la rotación, que gira los objetos alrededor del origen. Es equivalente a cambiar a otros ejes de coordenadas. Una matriz de rotación $\mathtt{R}$ se caracteriza por tener $\det\mathtt{R} = 1$. Está compuesta por un conjunto de filas (o de columnas) ortonormales ($||\boldsymbol{r}_i|| = 1$ y $\boldsymbol{r}_i^\mathsf{T}\boldsymbol{r}_j = 0$). Por otra parte, la matriz simétrica, real y semidefinida positiva Σ_x puede expresarse como $\Sigma_x = \mathtt{V}^\mathsf{T}\Lambda\mathtt{V}$ donde Λ es una matriz diagonal que contiene los valores propios de Σ y $\mathtt{V}$ es la correspondiente

matriz ortonormal de vectores propios (Sección A.3). Esto significa que la transformación lineal $\boldsymbol{y} = \mathtt{V}\boldsymbol{x}$ hace que la matriz de covarianza resultante se haga diagonal: $\Sigma_y = \Lambda$. Aplicando esa transformación la nube de puntos se expresa de la forma más sencilla posible: todos los coeficientes de covarianza se hacen cero; las nuevas variables quedan 'descorreladas'. Se ha hecho un cambio de base y los nuevos ejes son las *direcciones principales* de la población:



En la base original $\{\boldsymbol{x}, \boldsymbol{y}\}$ las observaciones tienen varianza similar pero son bastante redundantes. En la base de componentes principales $\mathtt{V} = \{\boldsymbol{v}, \boldsymbol{w}\}$ hay una componente muy informativa ($\boldsymbol{v}$), con gran varianza λ_1. Es la componente principal de la población. Observa que si las componentes secundarias tienen muy poca varianza podrían despreciarse sin gran deterioro de las muestras originales. Esta idea es la base del método de reducción de dimensión explicado en la Sección 2.2.1.

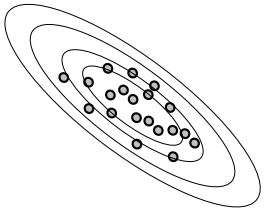
Como el determinante de un operador es invariante a transformaciones ortonormales, $\sqrt{\det \Sigma}$ es proporcional al 'volumen' ocupado por la nube (es el producto de los tamaños de los ejes del hiperelipsoide que engloba a la nube). La traza también es invariante a rotaciones, por lo que la $\mathrm{trz}\,\Sigma$ nos da la suma de varianzas en los ejes principales.

Normalización. El tamaño de las coordenadas de una cierta observación depende de la escala y el origen de coordenadas elegido. (P. ej., dependiendo de si utilizamos kilómetros o milímetros, el número que describe una longitud será grande o pequeño.) Por tanto es conveniente utilizar medidas normalizadas, que tienen media cero y varianza (y desviación) unidad. En el caso multidimensional la normalización produce variables aleatorias con media cero y matriz de covarianza identidad.

Sea $\Sigma = \mathtt{V}^\mathsf{T}\Lambda\mathtt{V}$ la descomposición en valores y vectores propios de la población $\boldsymbol{x}$ y sea $\boldsymbol{\mu}$ su valor medio. La transformación $\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{\mu}$ produce observaciones centradas en el origen. La transformación $\boldsymbol{y} = \mathtt{V}(\boldsymbol{x} - \boldsymbol{\mu})$ produce variables centradas y descorreladas (con matriz de covarianza diagonal Λ). Finalmente, la transformación $\boldsymbol{y} = \Lambda^{-\frac{1}{2}}\,\mathtt{V}(\boldsymbol{x} - \boldsymbol{\mu})$ produce variables normalizadas con media cero y matriz de

covarianza indentidad (*whitening*): las variables están descorreladas, y tienen todas desviación típica unidad. La población queda de forma esférica.

La **distancia de Mahalanobis** de un punto $\boldsymbol{x}$ a una distribución con media $\boldsymbol{\mu}$ y matriz de covarianza Σ se define como:



$$d^2(\boldsymbol{x}) = (\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$$

Es la distancia a la media de la población, expresada en unidades de desviación típica en la dirección de la muestra. Si Σ es la matriz identidad la distancia de Mahalanobis se reduce a la distancia euclídea a la media.

Es fácil comprobar que $\mathrm{E}\{d^2(\boldsymbol{x})\} = n$, donde n es la dimensión del espacio. La desigualdad de Markov garantiza que $\mathrm{P}\{d(\boldsymbol{x}) > k\} < n/k^2$.

> En concreto, si la población es gaussiana (Sección C.3) la distribución de la distancia de Mahalanobis (al cuadrado) tiene media n y desviación $\sqrt{2n}$. Curiosamente, en dimensión elevada las muestras se distribuyen en una especie de 'hiperdonut', dejando un hueco en el origen. Es muy improbable que todas las coordenadas tengan valores pequeños.

Poblaciones degeneradas. Decimos que un conjunto de muestras está 'degenerado' cuando todas ellas se encuentran en una región del espacio que no ocupa volumen; cuando su 'envolvente' está completamente aplastada (p. ej., tres puntos en el espacio siempre están en un plano). En este caso una o más direcciones principales tienen varianza nula y el determinante de la matriz de covarianza es cero. Esto ocurre seguro si el número de observaciones es menor que la dimension el espacio.

Cuando hay un número suficiente de muestras (mayor que la dimensión del espacio) y aún así están degeneradas debemos sospechar que existe algún problema: si la densidad de probabilidad que las ha generado es continua eso no podría ocurrir.

Cuando la población está degenerada no podemos calcular directamente sus direcciones más discriminantes (Sección 2.2.2) o la distancia de Mahalanobis (Sección 3.1). Una alternativa es calcular esas magnitudes en el subespacio de las componentes principales.

Distancia entre dos poblaciones. A veces es necesario determinar el grado de similitud entre dos poblaciones aleatorias. Es posible definir funciones de distancia entre nubes de puntos, caracterizadas por sus valores medios y sus matrices de covarianza.

Por ejemplo, una medida de distancia entre dos poblaciones podría ser la distancia media entre muestras de una y otra población:

$$\mathrm{E}\{||\boldsymbol{x}_1 - \boldsymbol{x}_2||^2\} = \mathrm{trz}\left(\Sigma_1 + \Sigma_2 + \boldsymbol{\mu}_1\boldsymbol{\mu}_1^\mathsf{T} + \boldsymbol{\mu}_2\boldsymbol{\mu}_2^\mathsf{T} - 2\boldsymbol{\mu}_1\boldsymbol{\mu}_2^\mathsf{T}\right)$$

Pero tiene la desventaja de que no tiene en cuenta en ningún momento la correlación entre variables (!).

Otra posibilidad es la la distancia de Bhattachariya:

$$\mathcal{B} = \frac{1}{8}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\left[\frac{\Sigma_1 + \Sigma_2}{2}\right]^{-1} + \frac{1}{2}\ln\frac{\left|\frac{\Sigma_1+\Sigma_2}{2}\right|}{\sqrt{|\Sigma_1||\Sigma_2|}}$$

Está relacionada con la probabilidad de error óptima (Error de Bayes, Sección 3.3.12) en el problema de clasificación de muestras de esas poblaciones. En concreto, EB $< \sqrt{P_1P_2}\exp(-\mathcal{B})$, donde P_1 y P_2 son las probabilidad *a priori* de las clases.

También podríamos calcular el valor esperado de la distancia de Mahalanobis de una población a la otra:

$$\mathrm{E}\{d_1^2(\boldsymbol{x}_2)\} = d_1^2(\mu_2) + \Sigma_1^{-1} \otimes \Sigma_2$$

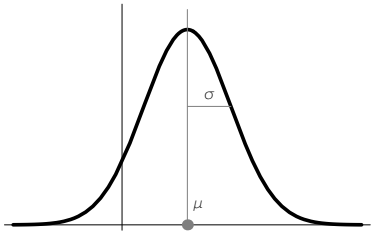
donde el operador $\otimes$ indica la multiplicación elemento a elemento de dos matrices y la suma de todos los elementos del resultado.

En principio, un clasificador por mínima distancia, en este caso normalizada, funciona bien cuando la distancia de una muestra a su clase casi siempre es menor que a la clase contraria. Por tanto una medida de distancia entre dos poblaciones puede ser:

$$\mathcal{D} = \mathrm{mín}(\mathrm{E}\{d_1^2(\boldsymbol{x}_2) - n\}, \mathrm{E}\{d_2^2(\boldsymbol{x}_1) - n\})$$

C.3. Modelo Gaussiano

La densidad de probabilidad más importante es la *normal* o gaussiana:

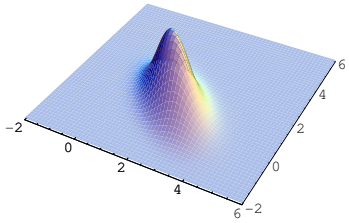


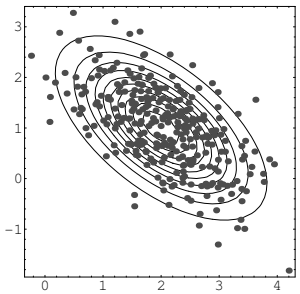
$$\mathcal{N}(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\,\sigma}\, exp - \frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2$$

En el caso multidimensional la expresamos como:

$$\mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{|\Sigma|}}\, exp -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$$

Los puntos de igual densidad de probabilidad son elipses centradas en la media $\boldsymbol{\mu}$ cuyos ejes principales son los vectores propios de la matriz de covarianza Σ. A continuación se muestra un ejemplo de campana de Gauss bidimensional y sus parámetros:



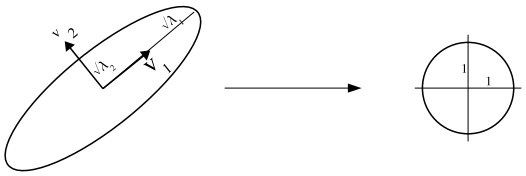


$$\boldsymbol{\mu} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

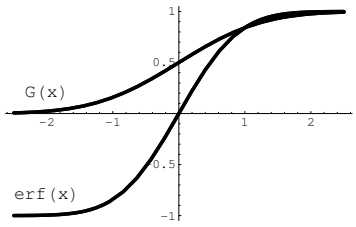
$$\Sigma = \begin{bmatrix} 0.625 & -0.375 \\ -0.375 & 0.625 \end{bmatrix}$$

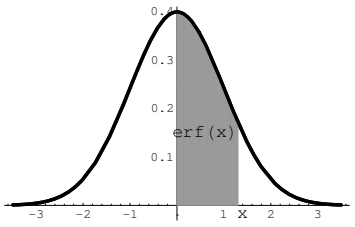
Algunas propiedades interesantes de la densidad normal son las siguientes:

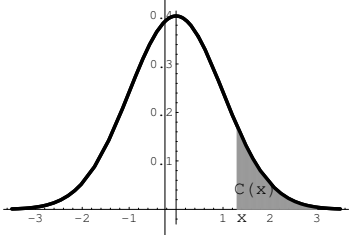
- Queda completamente determinada por los momentos estadísticos de primer y segundo orden ($\boldsymbol{\mu}$ y Σ).
- Es un buen modelo de medidas de una magnitud real, bien definida, contaminada con ruido.
- Es el modelo del resultado de muchos efectos (teorema del límite central: la suma de muchas variables aleatorias independientes de casi cualquier distribución produce una distribución normal).
- Es la densidad de mayor *entropía* informativa $H = -\int p(x)\log p(x)dx$ de todas las que tiene igual $\boldsymbol{\mu}$ y Σ. En general $H = 1/2(\log|\Sigma| + n\log 2\pi + 1)$ y en el caso unidimensional $H = \log\sigma\sqrt{2\pi e}$). Es el modelo más sencillo, no añade información adicional a $\boldsymbol{\mu}$ y Σ.
- Podemos escribir $\mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}, \Sigma) = K\exp -\frac{1}{2}d^2(\boldsymbol{x})$ donde $d^2(\boldsymbol{x})$ es la distancia de Mahalanobis y K es una constante de normalización.
- Una gaussiana multidimensional es el producto de varias gaussianas unidimensionales a lo largo de los ejes principales.
- Se pueden normalizar y 'descorrelar' las componentes de $\boldsymbol{x}$ con la transformación $\boldsymbol{y} = \Lambda^{-\frac{1}{2}}\mathtt{V}^\mathsf{T}(\boldsymbol{x}-\boldsymbol{\mu})$, donde $\Sigma = \mathtt{V}^\mathsf{T}\Lambda\mathtt{V}$ es la descomposición de valores y vectores propios de Σ. Las nuevas variables $\boldsymbol{y}$ tendrán media cero y matriz de covarianza identidad (forma esférica de desviación unidad):



- La magnitud $\delta = -\ln \mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}, \Sigma) = \frac{n}{2}\ln 2\pi + \frac{1}{2}\ln|\Sigma| + \frac{1}{2}d^2(\boldsymbol{x})$ es una especie de distancia probabilística de $\boldsymbol{x}$ a la población (útil para construir clasificadores gaussianos multiclase (Sección 3.4.1)).
- Si $\boldsymbol{x}$ tiene una distribución normal $\{\boldsymbol{\mu}$,$\Sigma\}$ cualquier transformación lineal $\boldsymbol{y} = \mathtt{A}\boldsymbol{x}$ hace que las nuevas variables $\boldsymbol{y}$ sigan siendo normales con media y matriz de covarianza $\{\mathtt{A}\boldsymbol{\mu}, \mathtt{A}\Sigma\mathtt{A}^\mathsf{T}\}$.
- La densidad normal unidimensional estándar es $\mathcal{N}(x) = \mathcal{N}(x, 0, 1)$. En lo que sigue, donde dice x nos referimos a la variable normalizada $\frac{x-\mu}{\sigma}$.
- La distribución acumulada es $G(x) \equiv \int_{-\infty}^{x} \mathcal{N}(x)dx$. $G(\infty) = 1, G(-x) = 1 - G(x)$. Se suele expresar como $G(x) = \frac{1}{2} + \frac{1}{2}erf(\frac{x}{\sqrt{2}})$ en términos de la 'función de error' $erf(x) = \frac{2}{\sqrt{\pi}}\int_0^y e^{-y^2}dy$, que es una función especial cuyos valores están tabulados o disponibles en los entornos de cálculo científico. Una buena aproximación para valores extremos es $G(x) \xrightarrow[x\to\infty]{} 1 - \frac{1}{x}\mathcal{N}(x)$.
- La cola de la distribución acumulada es $C(x) = 1 - G(x)$, de modo que $\mathrm{P}\{x > a\} = C(a)$, y $\mathrm{P}\{|x| > a\} = 2C(a)$ (dos colas).







- El *percentil* x_u de la distribución normal es el x tal que $\mathrm{P}\{x < x_u\} = u$, o, lo que es lo mismo, la inversa de la distribución acumulada: $x_u = G^{-1}(u)$. Por tanto, $\mathrm{P}\{x < x_u\} = G(x_u) = u$.
- Los momentos impares son cero, y los momentos pares son $\mathrm{E}\{x^n\} = 1 \cdot 3 \cdots (n-1) \cdot \sigma^n$.
- Aquí tenemos algunos percentiles de la normal y el valor de las colas, útiles para fabricar intervalos de confianza (todas las columnas excepto la primera se dan en %):

α	$C(\alpha)$	$2C(\alpha)$	$1-C(\alpha)$	$1-2C(\alpha)$
1.00	16	32	84	68
1.29	10	20	90	80
1.65	5	10	95	90
2.00	2.3	4.6	97.7	95.4
2.33	1	2	99	98
2.58	0.5	1	99.5	99
3.08	0.1	0.2	99.9	99.8
x_u			u	
			$\mathrm{P}\{x<\alpha\}$	$\mathrm{P}\{\lvert x\rvert<\alpha\}$

Por ejemplo, con 95.4 % de confianza $x \simeq \mu \pm 2\sigma$.

- El número a de éxitos en n intentos de un suceso con probabilidad p tiene una distribución binomial:

$$\mathcal{B}(a,p,n) = \binom{n}{a} p^a (1-p)^{n-a}$$

Cuando p no es muy extrema $\mathcal{B}(a,p,n) \simeq \mathcal{N}(a, np, \sqrt{np(1-p)})$. Por tanto la frecuencia $\hat{p}_n \equiv \frac{a}{n}$ se distribuye aproximadamente como $\mathcal{N}(\hat{p}_n, p, \sqrt{p(1-p)/n})$. Si n no es muy pequeño podemos estimar la desviación como $\sqrt{\hat{p}_n(1-\hat{p}_n)/n}$ y usar el percentil deseado de la densidad normal para establecer un intervalo de confianza para p a partir de $\hat{p}_n$ y n. Con confianza 99 %:

$$p \simeq \hat{p}_n \pm 2.58\sqrt{\hat{p}_n(1-\hat{p}_n)/n}$$

Por ejemplo, si observamos un suceso 10 veces en 100 intentos $p \simeq (10 \pm 8)\,\%$ (entre un 2 % y un 18 %(!)). Si lo observamos 1000 veces en 10000 intentos $p \simeq (10 \pm 0.8)\,\%$ (entre un 9 % y un 11 %).
Estos intervalos son más pequeños que los conseguidos con la desigualdad de Tchevichev, que es válida para cualquier p y n y por tanto más conservadora. Con $n = 100$ obtenemos $p \simeq (10 \pm 50)\,\%$ (?) y con $n = 10000$ obtenemos $p \simeq (10 \pm 5)\,\%$.

- El producto de dos gaussianas es otra gaussiana:

$$\mathcal{N}(x,\mu_1,\sigma_1)\,\mathcal{N}(x,\mu_2,\sigma_2) = \mathcal{N}\left(x, \frac{\sigma_1^2\mu_2 + \sigma_2^2\mu_1}{\sigma_1^2+\sigma_2^2}, \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}\right)\mathcal{N}\left(\mu_1,\mu_2,\sqrt{\sigma_1^2+\sigma_2^2}\right)$$

Esta propiedad (conjugación) es muy importante en aplicaciones de inferencia probabilística.

- La transformada de Fourier (Sección B) de una gaussiana es otra gaussiana con desviación inversa:

$$\mathcal{F}\mathcal{N}(x,0,\sigma) = \mathcal{N}(\omega,0,\sigma^{-1})$$

Cuanto mayor es el radio σ de un suavizado gaussiano (Sección 6) más estrecha es la banda de frecuencias que se preservan en la señal.

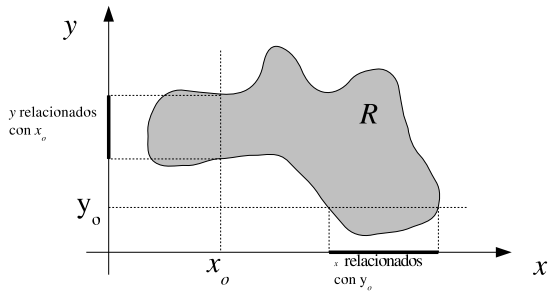
- Distribución χ^2.

C.4. Inferencia Probabilística

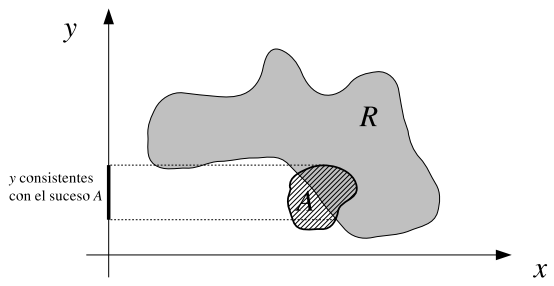
Simplificando al máximo, muchos problemas de la ciencia y la ingeniería podrían formularse como el estudio de la relación R entre dos variables x e y, con el objetivo de predecir lo mejor posible una de ellas a partir de la otra. La forma que posee la relación entre ellas indica el grado de dependencia. Dada una observación x_o, nuestra predicción será el conjunto de valores de y relacionados con x_o.

Es conveniente recordar conceptos de teoría de conjuntos como pertenencia, inclusión, unión, intersección, complemento, producto cartesiano, cardinalidad, ordenación, conjunto potencia, etc. Un subconjunto se asocia con un predicado o concepto. Una relación es un subconjunto del producto cartesiano de los conjuntos involucrados: $R \subset X \times Y$. Una función es una relación en la que el primer elemento de cada par aparece una sola vez. La función es total si todos los primeros elementos aparecen. Si es 'uno a muchos' podríamos decir que es 'no determinista'. Si es 'muchos a uno' no es invertible.

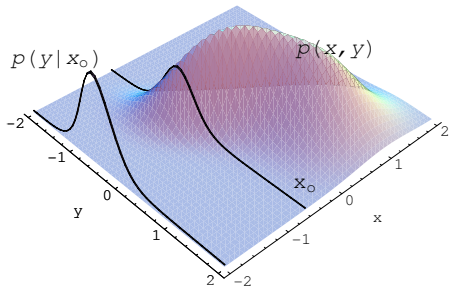
La 'inferencia' consiste en averiguar lo desconocido (y) a partir de lo conocido (x), dada una cierta relación R entre ellos. Normalmente deseamos obtener el subconjunto de los y relacionados con un x_o observado:



También puede plantearse de manera más general como la obtención de los y consistentes con un suceso A (otra relación entre x e y):



Si a la relación R le añadimos una función $p(x, y)$ que mide la frecuencia con que aparece cada par (x, y), la predicción de los valores y relacionados con una observación x_o puede mejorarse, pasando de un simple conjunto de posibilidades a una función (densidad de probabilidad condicionada) que indica la frecuencia relativa con que aparecerá cada uno de los posibles valores y relacionados con x_o:

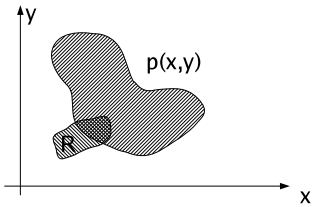


$$p(y|x_o) = \frac{p(x_o, y)}{p(x_o)} = \frac{p(x_o, y)}{\int p(x_o, y)dy}$$

El resultado de la inferencia es la densidad condicionada $p(y|x_o)$. Se calcula sencillamente como el corte (normalizado) de la densidad conjunta a través del valor observado x_o.

En principio la inferencia es 'reversible': los papeles de x e y son intercambiables; de acuerdo con nuestros intereses podemos calcular $p(x|y)$ o $p(y|x)$. Si la variable deseada es continua hablamos de *regresión* y si es discreta de *clasificación*.

Por supuesto, es posible obtener densidades condicionadas a sucesos más generales que la simple observación de una variable. P. ej., podemos restringir ambas variables a un cierto suceso (región) R:

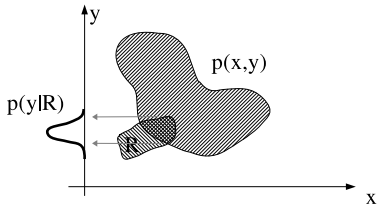


$$p(x, y|R) = \frac{p(x, y)I_R}{\mathrm{P}\{R\}}$$

$$\mathrm{P}\{R\} = \int \int_R p(x, y)dxdy$$

I_R denota la función indicadora del conjunto R: $I_R(x, y) = 1$ si $(x, y) \in R$ y $I_R(x, y) = 0$ si $(x, y) \notin R$.

Si sólo nos interesa una de las variables marginalizamos la otra:



$$p(y|R) = \int p(x, y|R) = \frac{\int_R p(x, y)dx}{\mathrm{P}\{R\}}$$

Reglas probabilísticas. Dada una expresión del tipo $p(x, y|z, v)$ las reglas para introducir o eliminar variables en cualquier lado del condicionamiento son las siguientes:

- *Conjunción* (añadir a la izquierda):

$$p(x|z) \rightarrow p(x, y|z) = p(x|y, z)p(y, z)$$

  en particular, $p(x, y) = p(x|y)p(y)$.
- *Condicionamiento* (añadir a la derecha):

$$p(x|v) \rightarrow p(x|z, v) = \frac{p(x, z|v)}{p(z|v)}$$

  en particular, $p(x|z) = p(x, z)/p(z)$.
- *Marginalización* (eliminar a la izquierda):

$$p(x, y|z) \rightarrow p(x|z) = \int p(x, y|z)dy$$

  en particular $p(x) = \int p(x, y)dy$.
- *Desarrollo* (eliminar a la derecha):

$$p(x|z, v) \rightarrow p(x|v) = \int p(x|z, v)p(z|v)dz$$

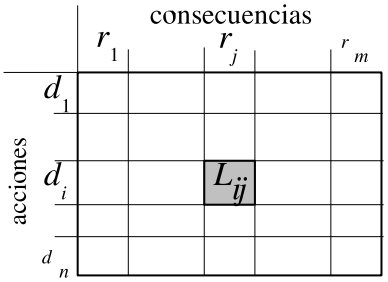
  en particular, $p(x) = \int p(x|z)p(z)dz$.

Estimadores. Además de la densidad condicionada resultante de la inferencia, en muchos casos nos gustaría dar una predicción concreta, calcular un estimador $\hat{y}$. Éste dependerá del 'coste' que deseamos minimizar. P. ej., si la variable y que deseamos predecir es continua y lo que deseamos es acercarnos lo más posible a valor real en el sentido del error cuadrático medio $MSE = E\{(\hat{y} - y)^2\}$, el estimador óptimo es la *media* de la densidad condicionada: $\hat{y}(x_o) = E\{y|x_o\} = \int p(y|x_o)dy$. Si preferimos minimizar las desviaciones absolutas $MAE = E\{|\hat{y} - y|\}$ (en cierto sentido

un criterio más robusto), entonces el estimador óptimo es la *mediana* de la densidad condicionada. Cuando deseamos predecir variables nominales (en problemas de clasificación) no tiene mucho sentido 'acercarnos' más o menos al valor real, sino acertar ese valor. En este caso se debe elegir como estimador la *moda* de la densidad condicionada, que minimiza la probabilidad de error (función de coste 0-1, ambos errores tienen el mismo coste).

En resumen, toda la información sobre y dado x_o está contenida en la función $p(y|x_o)$, que es la que habría que suministrar como resultado; a pesar de todo, esa función se suele resumir mediante ciertas propiedades como la media, la mediana, la moda, intervalos de confianza, etc. Pero estas formas resumidas de la densidad condicionada no tienen mucho sentido, p. ej., si las densidades son 'multimodales'.

Teoría de la Decisión. Supongamos que debemos tomar una decisión o predicción d_i a elegir entre un cierto conjunto de posibilidades, y que el mundo externo puede 'contestar' con un estado real o consecuencia r_j. Construyamos una tabla $L_{i,j}$ con el coste (*loss*) que supone predecir d_i cuando la realidad es r_j (en el caso continuo tendríamos una función $L(d, r)$):

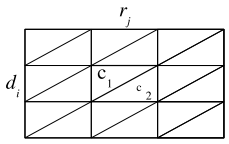


Obviamente, si conocemos r_j debemos decidir la d_i de mínimo coste. Si no se conoce –y el mundo externo no es un adversario inteligente con sus propios intereses– podemos adoptar p. ej., una estrategia *minimax*.

Cuando tenemos información probabilística $p(r)$ sobre el estado real (típicamente $p(r|x_o)$, donde x_o es el resultado de alguna medida relacionada con el problema) podemos evaluar el *riesgo* $R(d)$ de cada alternativa, que es el valor esperado del coste al tomar esa decisión. En el caso discreto tenemos $R(d_i) = \sum_j L(d_i, r_j)p(r_j)$ (la suma de cada fila de L ponderada con las probabilidades de r). En el caso continuo $R(d) = \int_r L(d, r)p(r)dr$.

La decisión óptima en vista de la información disponible $p(r)$ y nuestra función de coste L es la que minimiza el riesgo $R(d)$.

> Ten en cuenta que si nos enfrentamos a un adversario inteligente, éste tendrá su propia tabla de costes: entramos en el dominio de la Teoría de Juegos:



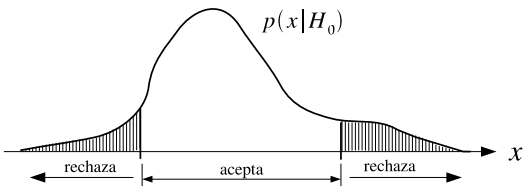
En este caso podría ser necesario 'randomizar' nuestra estrategia de decisión para resultar imprevisibles.

Inferencia Bayesiana. Dada una nueva observación x, calculamos la distribución *a posteriori* $p(r|x)$ de la variable de interés r dada la observación x y elegimos la decisión que minimiza el riesgo *a posteriori* : $R'(d) = E_{p(r|x)}\{L(d,r)\}$. Nuestro modelo de la incertidumbre sobre r se actualiza con x. La distribución *a priori* $p(r)$ se convierte en la distribución *a posteriori* $p(r|x)$.

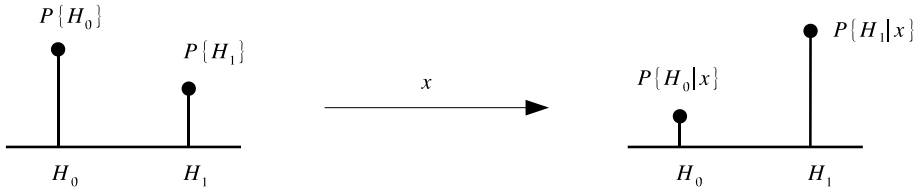
En general, la Teoría Estadística de la Decisión utiliza información empírica para construir modelos probabilísticos de la incertidumbre. El enfoque *bayesiano* tiene la ventaja de que hace explícito:

- La información *a priori* .
- El coste a minimizar en la decisión.

Otros enfoques tienen esos ingredientes implícitos. P. ej., un test de hipótesis clásico establece una hipótesis 'nula' H_0 (no ocurre nada especial) y construye la distribución del observable bajo esa hipótesis: $p(x|H_0)$. Si la observación no es muy 'extrema' mantenemos la hipótesis nula: nuestra observación es compatible con una fluctuación aleatoria típica. En caso contrario, si la observación es significativamente extrema, rechazamos la hipótesis nula y adoptamos la alternativa: sí ocurre algo especial:



Sin embargo, el procedimiento anterior no nos dice nada sobre la probabilidad que tienen las hipótesis nula o la alternativa de ser ciertas. En contraste, en el enfoque bayesiano se especifican unas probabilidades *a priori* para H_0 y H_1 (podemos elegir distribuciones *a priori no informativas*) y se actualizan con la observación x para obtener $\mathrm{P}\{H_0|x\}$ y $\mathrm{P}\{H_1|x\}$:



C.5. Inferencia Probabilística en el caso Gaussiano

Las reglas de condicionamiento y marginalización que aparecen en la regla de Bayes se puedean aplicar de forma exacta y eficiente (en forma cerrada) cuando las variables de interés siguen una ley de probabilidad conjunta Gaussiana. Esto da lugar a muchas aplicaciones prácticas: Filtro de Kalman (C.6), Procesos Gaussianos (4.8), etc.

Si reordenamos las variables de forma que haya dos grupos $(\boldsymbol{x}, \boldsymbol{y})$ la densidad conjunta se puede expresar en función de las medias y matrices de covarianza de cada grupo y la covarianza cruzada entre los dos grupos de variables.

$$p(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}\Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{a} \\ \boldsymbol{b}\end{bmatrix}, \begin{bmatrix}\mathtt{A} & \mathtt{C} \\ \mathtt{C}^\mathsf{T} & \mathtt{B}\end{bmatrix}\right)$$

C.5.1. Marginalización

La densidad marginal de cualquier grupo se obtiene simplemente seleccionando las variables deseadas tanto en la media como en la matriz de covarianza. Por ejemplo, para $\boldsymbol{y}$:

$$p(\boldsymbol{y}) \sim \mathcal{N}\left(\boldsymbol{\mu}_y \, \Sigma_{yy}\right) = \mathcal{N}\left(\boldsymbol{b}, \mathtt{B}\right)$$

C.5.2. Condicionamiento

La densidad de un grupo de variables condicionada a la observación de otro grupo de variables es también gaussiana y se puede expresar de la siguiente forma:

$$p(\boldsymbol{y}|\boldsymbol{x}) \sim \mathcal{N}\left(\boldsymbol{b} + \mathtt{C}^\mathsf{T}\mathtt{A}^{-1}(\boldsymbol{x} - \boldsymbol{a}), \mathtt{B} - \mathtt{C}^\mathsf{T}\mathtt{A}^{-1}\mathtt{C}\right) \tag{C.1}$$

Es interesante observar que la matriz de covarianza de la densidad condicionada es el "complemento de Schur[1]" del bloque que condiciona (seleccionamos las filas y columnas de las variables de interés de la matriz de covarianza inversa), y no depende de la observación concreta $\boldsymbol{x}$. Por supuesto, la media condicionada sí depende: su valor es la "hipersuperficie" de regresión, donde juega un papel esencial al covarianza cruzada

[1]Es el nombre que recibe el proceso de eliminación gaussiana por bloques al resolver sistemas de ecuaciones.

entre los dos grupos de variables. De hecho, la media condicionada se puede reinterpretar fácilmente en términos de la pseudoinversa que aparece en la regresión lineal por mínimo error cuadrático. Si deseamos encontrar una transformacion $\mathtt{W}$ tal que $\boldsymbol{y} \simeq \mathtt{W}\boldsymbol{x}$ y tenemos un conjunto de pares $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ organizados por filas en las matrices $\mathtt{X}$ y $\mathtt{Y}$, la transformación que minimiza $||\mathtt{X}\mathtt{W}^\mathsf{T} - \mathtt{Y}||^2$ es $\mathtt{W}^\mathsf{T} = \mathtt{X}^+\mathtt{Y} = (\mathtt{X}^\mathsf{T}\mathtt{X})^{-1}\mathtt{X}^\mathsf{T}\mathtt{Y}$, donde encontramos las estimaciones a partir de muestras de $\Sigma_{xx} = \mathtt{A}$ y $\Sigma_{yx} = \mathtt{C}^\mathsf{T}$ en la expresión anterior.

$\rightarrow$ demostración

$\rightarrow$ ejemplo

En ocasiones estamos interesados realizar inferencia sobre unas variables $\boldsymbol{x}$ a partir de la observación de una cierta función de ellas: $\boldsymbol{y} = f(\boldsymbol{x})$. Si $\boldsymbol{x}$ es normal y la función f es lineal podemos obtener fácilmente la densidad conjunta $p(\boldsymbol{x}, \boldsymbol{y})$ y realizar el condicionamiento como se acaba de explicar.

Concretamente, sea $p(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \mathtt{P})$ y $f(\boldsymbol{x}) = \mathtt{H}\boldsymbol{x}$ con ruido gaussiano aditivo de media $\boldsymbol{o}$ y covarianza $\mathtt{R}$. Esto significa que $p(\boldsymbol{y}|\boldsymbol{x}) \sim \mathcal{N}(\mathtt{H}\boldsymbol{x} + \boldsymbol{o}, \mathtt{R})$. Entonces la densidad conjunta $p(\boldsymbol{x}, \boldsymbol{y})$ es normal y tiene la siguiente media y matriz de covarianza[2]:

$$p(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \mathtt{H}\boldsymbol{\mu} + \boldsymbol{o} \end{bmatrix}, \begin{bmatrix} \mathtt{P} & \mathtt{P}\mathtt{H}^\mathsf{T} \\ \mathtt{H}\mathtt{P} & \mathtt{H}\mathtt{P}\mathtt{H}^\mathsf{T} + \mathtt{R} \end{bmatrix}\right) \tag{C.2}$$

La densidad marginal $p(\boldsymbol{y})$ es:

$$p(\boldsymbol{y}) \sim \mathcal{N}\left(\mathtt{H}\boldsymbol{\mu} + \boldsymbol{o}, \mathtt{H}\mathtt{P}\mathtt{H}^\mathsf{T} + \mathtt{R}\right)$$

Y la densidad condicionada contraria $p(\boldsymbol{x}|\boldsymbol{y})$ es:

$$p(\boldsymbol{x}|\boldsymbol{y}) \sim \mathcal{N}\left(\boldsymbol{\mu} + \mathtt{K}(\boldsymbol{y} - \mathtt{H}\boldsymbol{\mu} - \boldsymbol{o}), (\mathtt{I} - \mathtt{K}\mathtt{H})\mathtt{P}\right)$$

donde

$$\mathtt{K} = \mathtt{P}\mathtt{H}^\mathsf{T}(\mathtt{H}\mathtt{P}\mathtt{H}^\mathsf{T} + \mathtt{R})^{-1}$$

Esta expresión está construida de manera que a partir de la observación $\boldsymbol{y}$ corregimos la información sobre $\boldsymbol{x}$ con una "ganancia" $\mathtt{K}$ que depende del balance entre la incertidumbre a priori $\mathtt{P}$, el ruido de la medida $\mathtt{R}$, y el modelo de medida $\mathtt{H}$.

Una expresión alternativa para la inferencia bayesiana con modelos gaussianos es la siguiente:

[2] La media y covarianza de $\boldsymbol{y}$ y la covarianza cruzada entre $\boldsymbol{x}$ y $\boldsymbol{y}$ que aparecen en esta expresión se deducen fácilmente de la linealidad del valor esperado.

modelo de medida	prior	=	posterior	evidencia
$p(\boldsymbol{y}\|\boldsymbol{x})$	$p(\boldsymbol{x})$	=	$p(\boldsymbol{x}\|\boldsymbol{y})$	$p(\boldsymbol{y})$
$\mathcal{N}(\boldsymbol{y}\|\mathtt{H}\boldsymbol{x}+\boldsymbol{o},\mathtt{R})$	$\mathcal{N}(\boldsymbol{x}\|\boldsymbol{\mu},\mathtt{P})$	=	$\mathcal{N}(\boldsymbol{x}\|\boldsymbol{\eta}_{\boldsymbol{y}},\mathtt{Q})$	$\mathcal{N}(\boldsymbol{y}\|\mathtt{H}\boldsymbol{\mu}+\boldsymbol{o},\mathtt{H}\mathtt{P}\mathtt{H}^\mathsf{T}+\mathtt{R})$

La incertidumbre inicial sobre $\boldsymbol{x}$ era $\mathtt{P}$, que se reduce a $\mathtt{Q}$ tras la observación de $\boldsymbol{y}$:

$$\mathtt{Q} = (\mathtt{P}^{-1} + \mathtt{H}^\mathsf{T}\mathtt{R}^{-1}\mathtt{H})^{-1}$$

Y el estimador de $\boldsymbol{x}$ se actualiza de $\boldsymbol{\mu}$ a $\boldsymbol{\eta}_{\boldsymbol{y}}$, que puede expresarse como una combinación ponderada de la observación y la información a priori:

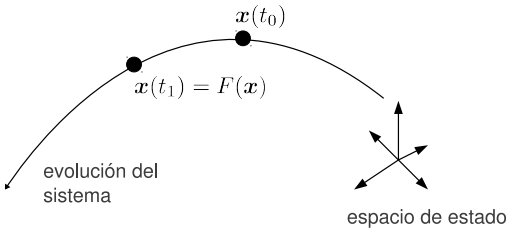
$$\boldsymbol{\eta}_{\boldsymbol{y}} = (\mathtt{Q}\mathtt{H}^\mathsf{T}\mathtt{R}^{-1})(\boldsymbol{y}-\boldsymbol{o}) + (\mathtt{Q}\mathtt{P}^{-1})\boldsymbol{\mu}$$

La "evidencia" $p(\boldsymbol{y})$ es la verosimilitud de la medida $\boldsymbol{y}$ teniendo en cuenta todos los posibles $\boldsymbol{x}$ (convolución de dos gaussianas). Juega un papel esencial en la selección de modelos.

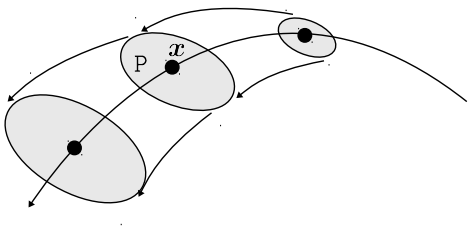
C.6. Filtro de Kalman

C.6.1. Motivación

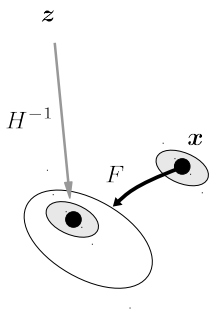
Consideremos un sistema que evoluciona de acuerdo con una ley dinámica $\boldsymbol{x}_{t+1} = F(\boldsymbol{x}_t)$.



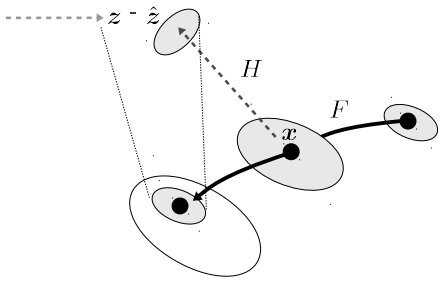
Nuestra información acerca del estado tiene incertidumbre, que va aumentando a medida que pasa el tiempo:



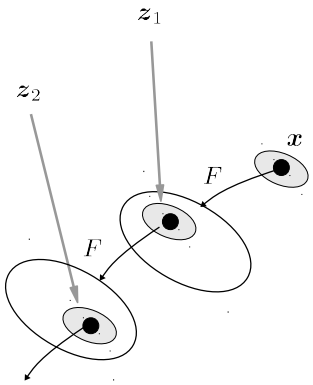
Supongamos que podemos observar alguna propiedad del estado $\boldsymbol{z} = H(\boldsymbol{x})$: Lógicamente, lo ideal sería poder medir "directamente" el estado (en cuyo caso $\boldsymbol{z} = \boldsymbol{x}$) y el problema sería trivial. Pero lo normal será que tengamos información *incompleta* y ruidosa: $\boldsymbol{z}$ será alguna propiedad que depende de unas pocas componentes de $\boldsymbol{x}$. Por ejemplo, si deseamos estimar la trayectoria de un objeto móvil usando una cámara, el estado incluye coordenadas de posición y velocidad, pero la observación sólo registra información de posición. A primera vista, el problema parece imposible de resolver, ya que estamos intentando obtener una especie de "modelo inverso" H^{-1} del proceso de medida, capaz de reconstruir el estado completo a partir de información parcial:



Aunque ese modelo inverso no puede existir, la información proporcionada por la observación reduce la incertidumbre sobre el estado completo. La figura siguiente muestra intuitivamente que la discrepancia entre la medida prevista $\hat{\boldsymbol{z}}$ y la realmente observada $\boldsymbol{z}$ actualiza la distribución de probabilidad del estado:



En la práctica tenemos una serie de observaciones $\boldsymbol{z}_t$ tomadas a lo largo de la evolución del sistema, con las que podemos aplicar recursivamente el proceso de inferencia en una secuencia de pasos de predicción-corrección del estado $\boldsymbol{x}_t$:



El resultado efectivo es que conseguimos "invertir" el modelo del proceso de observación, aunque H^{-1} como tal no exista. La reducción global de incertidumbre sobre el estado a partir de información incompleta se consigue porque la ley de evolución dinámica establece una dependencia entre las variables de estado que tienen reflejo en la observación y las de tipo "latente".

> En el ejemplo del objeto móvil, cuyas variables de estado son posición y velocidad, y la observación depende sólo de la posición, la velocidad puede estimarse a través de la dependencia de la posicion siguiente respecto a la posición y velocidades anteriores. Lo interesante es que una estimación de esa velocidad se conseguirá de forma automática, sin necesidad de construirla explícitamente a partir de las observaciones o las estimaciones de posición.
>
> Debido a esto, no tiene mucho sentido procesar las magnitudes observables "crudas" tratando de obtener propiedades más o menos interesantes que faciliten el proceso de inferencia, sino que deberían utilizarse directamente en $\boldsymbol{z}$, dejando que las leyes de la probabilidad hagan su trabajo.

Vamos a modelar matemáticamente el problema. En el instante t la información que tenemos sobre el estado se describe mediante una densidad de probabilidad $p(\boldsymbol{x}_t)$. La ley de evolución F y el modelo de medida H son aproximaciones más o menos razonables a determinandos procesos reales, en donde tenemos que incluir componentes de "ruido" R_F y R_H:

$$\boldsymbol{x}_{t+1} = F(\boldsymbol{x}_t) + R_F$$

$$\boldsymbol{z}_{t+1} = H(\boldsymbol{x}_{t+1}) + R_H$$

Nuestro objetivo es actualizar la información sobre el estado a partir de una nueva observación:

$$p(\boldsymbol{x}_t) \rightarrow p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{t+1}) = \frac{p(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1})}{p(\boldsymbol{z}_{t+1})}$$

En general es muy difícil obtener una expresión compacta para esta densidad condicionada. Solo puede hacerse de forma analítica con familias muy concretas de densidades de probabilidad. Además, hay que calcular cómo evoluciona la incertidumbre acerca del estado a causa de las leyes dinámicas, y también la distribución de probabilidad de la observación. Estos pasos requieren calcular la distribución de una función de una variable aleatoria, que tampoco tiene solución analítica general.

C.6.2. Algoritmo

A pesar de estas dificultades, el problema se puede resolver de forma satisfactoria en muchas situaciones prácticas. En el caso más simple, tanto la ley dinámica F como el modelo de observación H son funciones lineales y la incertidumbre sobre el estado, el ruido del sistema y el de la observación son aditivos y de tipo gaussiano. En este caso, el procedimiento de estimación recursivo del estado a partir de las sucesivas observaciones se conoce como *Filtro de Kalman*, y tiene una expresión algorítmica particularmente elegante.

Partimos de que la incertidumbre sobre el estado es normal: $p(\boldsymbol{x}_t) \sim \mathcal{N}(\boldsymbol{\mu}, \mathtt{P})$. Como la ley dinámica es lineal, $\boldsymbol{x}_{t+1} = \mathtt{F}\boldsymbol{x}_t$, y el ruido del sistema es aditivo, independiente del estado, y gaussiano con media cero y covarianza $\mathtt{S}$, la dinámica se expresa como $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t) \sim \mathcal{N}(\mathtt{F}\boldsymbol{x}_t, \mathtt{S})$. Desarrollando con $p(\boldsymbol{x}_t)$ conseguimos una expresión compacta para la incertidumbre del estado en el instante de tiempo siguiente: $p(\boldsymbol{x}_{t+1}) \sim \mathcal{N}(\mathtt{F}\boldsymbol{\mu}, \mathtt{FPF}^\mathsf{T} + \mathtt{S})$. Por simplicidad, definimos $\boldsymbol{\mu}' = \mathtt{F}\boldsymbol{\mu}$ y $\mathtt{P}' = \mathtt{FPF}^\mathsf{T} + \mathtt{S}$. La función de observacion también es lineal, $\boldsymbol{z}_{t+1} = \mathtt{H}\boldsymbol{x}_{t+1}$, con ruido gaussiano de covarianza $\mathtt{R}$, por lo que el modelo probabilístico de observación es $p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{t+1}) \sim \mathcal{N}(\mathtt{H}\boldsymbol{x}_{t+1}, \mathtt{R})$. Estos dos ingredientes nos permiten construir la densidad conjunta usando la eq. C.2:

$$p(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{t+1})p(\boldsymbol{x}_{t+1}) = p(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1}) \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}' \\ \mathtt{H}\boldsymbol{\mu}'\end{bmatrix}, \begin{bmatrix}\mathtt{P}' & \mathtt{P}'\mathtt{H}^\mathsf{T} \\ \mathtt{HP}' & \mathtt{HP}'\mathtt{H}^\mathsf{T} + \mathtt{R}\end{bmatrix}\right)$$

La única diferencia, aparte del cambio de nombre de variable $\boldsymbol{z}$ por $\boldsymbol{y}$, es que por simplicidad hemos omitido el posible *offset* de F y H, y que es necesario un paso intermedio para obtener la predicción de $\boldsymbol{\mu}'$ y $\mathtt{P}'$ con la ley dinámica. El valor medio y la matriz de covarianza de la estimación de $\boldsymbol{x}_{k+1}$ actualizado al incorporar la observación $\boldsymbol{z}_{t+1}$ puede expresarse con cualquiera de las formas mostradas en la sección anterior, aunque es tradicional usar la ganancia de Kalman $\mathtt{K}$.

El algoritmo quedaría así: dadas las transformaciones $\mathtt{F}$ y $\mathtt{H}$, con ruido modelado mediante matrices de covarianza $\mathtt{S}$ y $\mathtt{R}$ respectivamente, la información sobre $\boldsymbol{x}$, en forma de media $\boldsymbol{\mu}$ y matriz de covarianza $\mathtt{P}$, se actualiza con una nueva observación $\boldsymbol{z}$ de la siguiente forma:

- $\boldsymbol{\mu}' \leftarrow \mathtt{F}\boldsymbol{\mu}$
- $\mathtt{P}' \leftarrow \mathtt{FPF}^\mathsf{T} + \mathtt{S}$ (evolución del estado)
- $\boldsymbol{\epsilon} \leftarrow \boldsymbol{z} - \mathtt{H}\boldsymbol{\mu}'$ (discrepancia entre la observación y su predicción)
- $\mathtt{K} \leftarrow \mathtt{P}'\mathtt{H}^\mathsf{T}(\mathtt{HP}'\mathtt{H}^\mathsf{T} + \mathtt{R})^{-1}$ (ganancia de Kalman)
- $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu}' + \mathtt{K}\boldsymbol{\epsilon}$
- $\mathtt{P} \leftarrow (\mathtt{I} - \mathtt{KH})\mathtt{P}'$ (corrección)

Ejercicios:

- Comprueba el funcionamiento del filtro de Kalman en algún caso muy simple no trivial. Por ejemplo, podemos estudiar un movimiento rectilíneo uniforme con velocidad desconocida, donde observamos la posición con ruido gaussiano σ.

C.7. UKF

→ pendiente

C.8. Filtro de Partículas

→ pendiente

Apéndice D

Optimización

D.1. Operadores diferenciales

La generalización del concepto de derivada para funciones de varias variables se basa en el operador diferencial

$$\nabla \equiv \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \vdots \\ \frac{\partial}{\partial x_n} \end{bmatrix}$$

Si lo aplicamos a una función escalar $f : \mathbb{R}^n \to \mathbb{R}$ produce el *gradiente* de la función: un vector de $\mathbb{R}^n$ que apunta en la dirección de máximo crecimiento. Por ejemplo, el gradiente de $f(x_1, x_2) = x_1 \sin(x_2)$ es el vector $\nabla f = [\sin x_2,\ x_1 \cos x_2]^\mathsf{T}$. En el caso de funciones lineales y cuadráticas podemos escribir expresiones compactas:

$$\nabla \boldsymbol{a} \cdot \boldsymbol{x} = \boldsymbol{a}$$

$$\nabla ||\boldsymbol{x}||^2 = 2\boldsymbol{x}$$

$$\nabla \boldsymbol{x}^\mathsf{T} \mathtt{A} \boldsymbol{x} = 2\mathtt{A}\boldsymbol{x}$$

Esta última expresión es similar a la de la derivada de la función escalar ax^2. [1]

Cuando tenemos una transformación (una función que a partir de un vector produce otro), podemos aplicar el operador ∇ de dos maneras. Como producto escalar ($\nabla^\mathsf{T} \boldsymbol{f}$) obtenemos la *divergencia* del vector (un escalar). Por ejemplo:

[1] En realidad ésto sólo es válido si $\mathtt{A}$ es simétrica. En caso contrario tendríamos $(\mathtt{A} + \mathtt{A}^\mathsf{T})\boldsymbol{x}$, pero cualquier función $\boldsymbol{x}^\mathsf{T}\mathtt{A}\boldsymbol{x}$ se puede escribir como $\boldsymbol{x}^\mathsf{T}\mathtt{A}'\boldsymbol{x}$, en función de la matriz simétrica $\mathtt{A}' = 1/2(\mathtt{A} + \mathtt{A}^\mathsf{T})$.

$$\nabla \cdot \mathtt{A}\boldsymbol{x} = \mathrm{trz}\mathtt{A}$$

Alternativamente, si lo aplicamos a cada componente de la función ($\nabla \boldsymbol{f}^\mathsf{T}$), obtenemos una matriz J cuyos elementos son las derivadas de cada componente respecto a cada variable:

$$J_{i,j} = \frac{\partial f_j}{\partial x_i}$$

Su determinante es el *jacobiano* de la transformación, utilizado p.ej. para hacer cambios de variable en integrales multidimensionales.

> Supongamos que deseamos integrar una función $f(\boldsymbol{x})$. Si cambiamos a un sistema de coordenadas $\boldsymbol{y}(\boldsymbol{x})$ (donde típicamente la integración es más fácil), el 'elemento de volumen' $d\boldsymbol{x}$ cambia de acuerdo con el jacobiano de la transformación:
>
> $$f(\boldsymbol{x})d\boldsymbol{x} = f(\boldsymbol{x}(\boldsymbol{y}))\frac{d\boldsymbol{y}}{|\mathtt{J}(\boldsymbol{y})|}$$

Si lo aplicamos dos veces aparecen segundas derivadas. Pero como es un vector, podemos combinarlo de dos maneras: $\nabla^\mathsf{T}\nabla$ o $\nabla\nabla^\mathsf{T}$. En el primer caso se obtiene un escalar (el *laplaciano*):

$$\nabla \cdot \nabla = \nabla^2 = \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2}$$

y en el segundo una matriz de derivadas segundas (el *hessiano*):

$$\nabla\nabla^\mathsf{T} = \left[\frac{\partial^2}{\partial x_i \partial x_j}\right]_{i=1,j=1}^{n}$$

que es útil por ejemplo para hacer una expansión de segundo orden de una función de varias variables:

$$E(\boldsymbol{x}) \simeq E(\boldsymbol{x}_0) + \boldsymbol{g}^\mathsf{T}(\boldsymbol{x} - \boldsymbol{x}_0) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_0)^\mathsf{T}\mathtt{H}(\boldsymbol{x} - \boldsymbol{x}_0)$$

donde $\boldsymbol{g} = \nabla E(\boldsymbol{x}_0)$ es el gradiente y $\mathtt{H} = \nabla\nabla^\mathsf{T} E(\boldsymbol{x}_0)$ es el hessiano de la función $E(\boldsymbol{x})$ en el punto de interés $\boldsymbol{x}_0$.

D.2. Sistemas de ecuaciones lineales sobredeterminados.

D.2.1. No homogéneos

Consideremos un sistema de ecuaciones lineales determinado o sobredeterminado no homogéneo, con m ecuaciones y n incógnitas, donde $m \geq n$, que expresamos de forma compacta como:

$$\mathtt{A}\boldsymbol{x} = \boldsymbol{b}$$

En general no será posible obtener una solución que resuelva exactamente todas las ecuaciones. Buscaremos una solución aproximada, que minimice alguna función de coste razonable. La más sencilla es el error cuadrático:

$$E_{\mathrm{RMS}}(\boldsymbol{x}) = \frac{1}{2}||\mathtt{A}\boldsymbol{x} - \boldsymbol{b}||^2$$

La solución óptima $\boldsymbol{x}^*$ verifica $\nabla E_{\mathrm{RMS}}(\boldsymbol{x}^*) = 0$. Expandiendo la norma y calculando el gradiente obtenemos la siguiente condición:

$$\nabla E_{\mathrm{RMS}}(\boldsymbol{x}^*) = \mathtt{A}^\mathsf{T}\mathtt{A}\boldsymbol{x}^* - \mathtt{A}^\mathsf{T}\boldsymbol{b} = 0$$

de donde se deduce que

$$\boldsymbol{x}^* = (\mathtt{A}^\mathsf{T}\mathtt{A})^{-1}\mathtt{A}^\mathsf{T}\boldsymbol{b}$$

que puede abreviarse como

$$\boldsymbol{x}^* = \mathtt{A}^+\boldsymbol{b}$$

donde la matriz $\mathtt{A}^+ \equiv (\mathtt{A}^\mathsf{T}\mathtt{A})^{-1}\mathtt{A}^\mathsf{T}$ es la *pseudoinversa* de $\mathtt{A}$.

En un sistema de ecuaciones sobredeterminado la pseudoinversa $\mathtt{A}^+$ juega el mismo papel que el de la inversa ordinaria $\mathtt{A}^{-1}$ en un sistema determinado (p.ej., si $m = n$ la solución óptima se reduce a $\boldsymbol{x}^* = \mathtt{A}^{-1}\boldsymbol{b}$). Hay más información sobre la pseudoinversa en el apéndice A.4.

D.2.2. Homogéneos

Véase A.4.

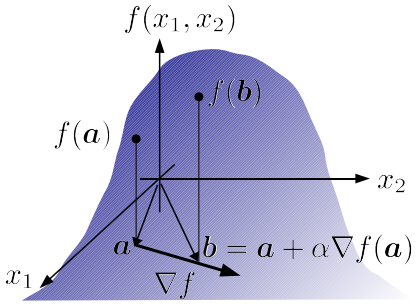
D.3. Descenso de Gradiente

Una forma de buscar un máximo local de una función continua $f : \mathbb{R}^n \to \mathbb{R}$ consiste en explorar el entorno de una solución tentativa y moverse poco a poco hacia posiciones mejores. Evidentemente, este tipo de optimización local de fuerza bruta es inviable en un espacio de dimensión elevada, ya que cada punto tiene un número

enorme de direcciones a su alrededor. Sin embargo, si f es *diferenciable*, a la vez que calculamos el coste de una aproximación a la solución $\boldsymbol{x}_k$, podemos obtener con muy poco esfuerzo adicional la dirección $\nabla f(\boldsymbol{x}_k)$ en la que un movimiento 'pequeño' aumentará el valor de la función. La corrección se expresa típicamente así:

$$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \alpha \nabla E(\boldsymbol{x}_k)$$

y puede representarse gráficamente como:



Los extremos (máximos o mínimos) de la función son *púntos críticos*, que verifican $\nabla f = 0$. Cuando se llega a ellos la corrección se hace cero.

Hay que tener en cuenta que la optimización sólo progresará si el movimiento (controlado por el coeficiente α) es suficientemente pequeño. Es un método local: si la función tiene varios máximos locales este método encontrará alguno que depende del punto de partida. A veces se producen trayectorias en *zig-zag*, debido a que la dirección de máximo crecimiento en un punto no es exactamente la misma que la que nos llevaría en línea recta a la solución. (Para remediar esto existen técnicas más avanzadas como la del *gradiente conjugado*.)

El método anterior tiene en cuenta información de primer orden, que modela f como un hiperplano. Pero evidentemente, si tiene un extremo, cerca de él será parecida a una montaña más o menos suave. El modelo más simple de este tipo es una función cuadrática (paraboloide), que podemos estimar usando segundas derivadas para intentar llegar directamente a la solución en un solo paso. Como la aproximación no será perfecta, en realidad serán necesarios varios pasos de optimización. Pero cada uno de ellos nos acercará mucho más a la solución que las correcciones basadas únicamente en el gradiente. La forma de hacer esto se explica en el apartado siguiente.

D.4. Método de Newton

Supongamos que deseamos resolver el sistema de ecuaciones no lineal $\boldsymbol{f}(\boldsymbol{x}) = 0$, donde $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$. Podemos hacer un desarrollo en serie de primer orden:

$$\boldsymbol{f}(\boldsymbol{x}) \simeq \boldsymbol{f}(\boldsymbol{x}_0) + \mathtt{L}(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0)$$

donde $\mathtt{L}(\boldsymbol{x}) = \boldsymbol{\nabla f}(\boldsymbol{x})$ es una matriz $(m \times n)$ que tiene como filas los gradientes de cada una de las componentes de $\boldsymbol{f}$. Para conseguir $\boldsymbol{f}(\boldsymbol{x}) = 0$ tenemos que resolver un sistema lineal de ecuaciones para la corrección $\Delta\boldsymbol{x} = (\boldsymbol{x} - \boldsymbol{x}_0)$:

$$\mathtt{L}(\boldsymbol{x}_0)\Delta\boldsymbol{x} = -\boldsymbol{f}(\boldsymbol{x}_0)$$

Si $m = n$ es un sistema determinado ordinario. Si hay más ecuaciones que incógnitas ($m > n$) tenemos un sistema sobredeterminado que podemos resolver con la pseudoinversa:

$$\Delta\boldsymbol{x} = -\mathtt{L}(\boldsymbol{x}_0)^+\boldsymbol{f}(\boldsymbol{x}_0)$$

La trayectoria hacia la solución se puede escribir como una secuencia de correcciones a partir de un punto arbitrario $\boldsymbol{x}_0$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \mathtt{L}(\boldsymbol{x}_k)^+\boldsymbol{f}(\boldsymbol{x}_k)$$

Si en lugar de resolver un sistema de ecuaciones $\boldsymbol{f}(\boldsymbol{x}) = 0$ deseamos encontrar el máximo (o el mínimo) de una función escalar $g(\boldsymbol{x})$, podemos utilizar el método anterior con $\boldsymbol{f}(\boldsymbol{x}) = \nabla g(\boldsymbol{x})$ para buscar algún punto crítico de g. Esta aproximación lineal al gradiente equivale a una aproximación cuadrática a la función, y entonces la matriz $\mathtt{L}(\boldsymbol{x}) = \nabla\nabla^\mathsf{T} g$ es el hessiano de g. Cada iteración tiene la forma:

$$(\nabla\nabla^\mathsf{T} g)\Delta\boldsymbol{x} = -\nabla g$$

Si g es realmente cuadrática se llega a la solución en un solo paso.

> Veamos esto en detalle. Sea $g(\boldsymbol{x}) = a + \boldsymbol{b}^\mathsf{T}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^\mathsf{T}\mathtt{C}\boldsymbol{x}$, donde $\mathtt{C}$ es simétrica. Su gradiente es $\nabla g(\boldsymbol{x}) = \boldsymbol{b} + \mathtt{C}\boldsymbol{x}$ y por tanto el extremo es la solución de $\nabla g(\boldsymbol{x}^*) = 0$, que es $\boldsymbol{x}^* = -\mathtt{C}^{-1}\boldsymbol{b}$. En realidad no conocemos $\boldsymbol{b}$ y $\mathtt{C}$, sino que estamos en una posición $\boldsymbol{x}_0$ más o menos alejada de la solución. Pero es inmediato calcularlos 'numéricamente' a partir del gradiente y el hessiano en ese punto. La matrix $\mathtt{C}$ es directamente el hessiano $\nabla\nabla^\mathsf{T} g|_{\boldsymbol{x}_0}$ que no depende de la posición debido a que suponemos que g es sólo cuadrática, no tiene términos de tercer orden. El gradiente sí depende de la posición, pero como conocemos $\mathtt{C}$ es inmedianto deducir $\boldsymbol{b} = \boldsymbol{d} - \mathtt{C}\boldsymbol{x}_0$, donde $\boldsymbol{d} = \nabla g|_{\boldsymbol{x}_0}$. Por tanto el extremo de g se encuentra en:
>
> $$\boldsymbol{x}^* = -\mathtt{C}^{-1}(\boldsymbol{d} - \mathtt{C}\boldsymbol{x}_0) = \boldsymbol{x}_0 - \mathtt{C}^{-1}\boldsymbol{d}$$

Si el modelo cuadrático no es apropiado puede usarse el método de *Levenverg-Marquardt*, que combina inteligentemente el método de Newton y el descenso de gradiente básico. Consiste en multiplicar la diagonal del hessiano $(\nabla\nabla^\mathsf{T} g)$ por un coeficiente $(1 + \lambda)$, donde λ se adapta dinámicamente tomando un valor grande en las zonas de la trayectoria en las que la corrección cuadrática normal no disminuye significativamente el error. En esos casos el hessiano modificado se parece más a

(algo proporcional a) la matriz identidad y la corrección se basa esencialmente en el gradiente.

D.5. Aproximación de Gauss-Newton

De nuevo deseamos resolver un sistema $\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}$ (normalmente sobredeterminado) de m ecuaciones no lineales con n incógnitas. La función de coste global para los residuos matemáticamente más sencilla es el error cuadrático:

$$C(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{f}^\mathsf{T}(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x})$$

Para resolverla mediante el método de Newton necesitamos el gradiente y el Hessiano (por sencillez de notación omitimos la dependencia de $\boldsymbol{x}$ de todos los símbolos):

$$\nabla C = \boldsymbol{f}^\mathsf{T}\mathtt{J}$$

$$\nabla\nabla^\mathsf{T} C = \mathtt{J}^\mathsf{T}\mathtt{J} + \mathtt{K}\boldsymbol{f}$$

donde $\mathtt{K} = \nabla\mathtt{J}^\mathsf{T}$ es un array 3D que contiene las derivadas segundas de cada una de las componentes de $\boldsymbol{f}$.

[esquema de los arrays]

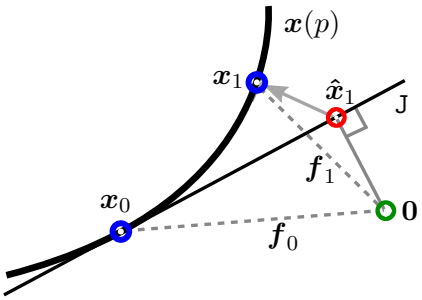
La aproximación de Gauss-Newton consiste en suponer que $\mathtt{K} = 0$, que es equivalente a hacer una aproximación lineal a las ecuaciones $\boldsymbol{f}$, dando lugar a una expresión sencilla para la aproximación al Hessiano. En cada paso se aplica una linealización de las ecuaciones alrededor de la solución actual. La aproximación de Gauss-Newton da lugar a un ajuste linealizado de mínimos cuadrados sobre los residuos basado en la pseudoinversa:

$$\Delta\boldsymbol{x} = -(\nabla\nabla^\mathsf{T} C)^{-1}\nabla C = -(\mathtt{J}^\mathsf{T}\mathtt{J})^{-1}\mathtt{J}^\mathsf{T}\boldsymbol{f}$$

D.5.1. Resumiendo

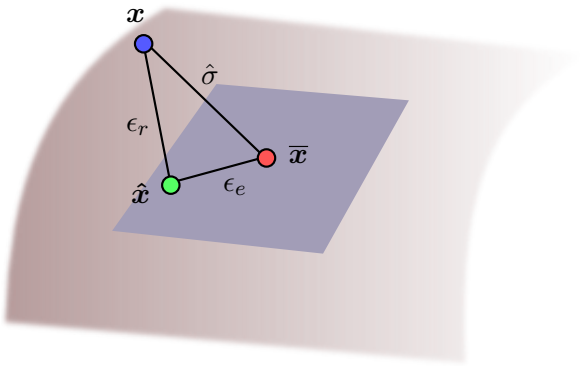
Dado un vector de residuos $\boldsymbol{f}$ con Jacobian $\mathtt{J} = \partial\boldsymbol{f}/\partial\boldsymbol{p}$, el error cuadrático $C = 1/2\boldsymbol{f}^\mathsf{T}\boldsymbol{f}$ puede reducirse progresivamente mediante la regla $\Delta\boldsymbol{p} = -\mathtt{H}^{-1}\nabla C$, donde $\nabla C = \mathtt{J}^\mathsf{T}\boldsymbol{f}$ y el Hessiano se aproxima mediante $\mathtt{H} = \mathtt{J}^\mathsf{T}\mathtt{J}$.

La figura siguiente muestra el caso más sencillo en el que hay un parámetro y dos residuos.



D.5.2. Ruido, error residual y error de estimación

Notación: $\overline{\boldsymbol{x}}$ is el valor real, desconocido; $\boldsymbol{x}$ es la observación ruidosa disponible, y $\hat{\boldsymbol{x}}$ es el valor estimado tras un proceso de optimización, que idealmente proyectará la observación sobre el *manifold* de soluciones admisibles.



El vector de parámetros tiene dimensión d y hay N observaciones. Si hay un ruido σ común en todas, la distribución de $\boldsymbol{x}$ es esférica con covarianza $\sigma\mathtt{I}$ alrededor de $\overline{\boldsymbol{x}}$. El error RMS de la observación con respecto al *ground truth* es $\hat{\sigma} \to \sigma$. El error *residual* RMS de la observación con respecto a la solución obtenida es ϵ_r (el "error de reproyección" que indica lo bien que el modelo se ajusta a las observaciones). El error RMS de *estimación* respecto al *ground truth* es ϵ_e. Si el *manifold* es suave podemos aproximarlo linealmente cerca de $\overline{\boldsymbol{x}}$, por tanto la estimación óptima será la proyección ortogonal a un hiperplano, dando lugar la siguientes relaciones [10]:

$$\hat{\sigma}^2 = \epsilon_r^2 + \epsilon_e^2$$

$$\epsilon_r = \sqrt{||\hat{\boldsymbol{x}} - \boldsymbol{x}||^2/N} \qquad \epsilon_e = \sqrt{||\hat{\boldsymbol{x}} - \overline{\boldsymbol{x}}||^2/N} \qquad \hat{\sigma} = \sqrt{||\boldsymbol{x} - \overline{\boldsymbol{x}}||^2/N}$$

El valor esperado de los errores es:

$$E\{\hat{\sigma}\} = \sigma \quad E\{\epsilon_r\} = \sigma\sqrt{1 - \frac{d}{N}} \quad E\{\epsilon_e\} = \sigma\sqrt{\frac{d}{N}}$$

Esto es más interesante de lo que parece: las estimaciones tendrán, por su propia naturaleza, menos error que la realidad (!). Esto se debe a que en la estimación quitamos 'variables superfluas' y proyectamos a un subsespacio de menos variables (con el mismo ruido). No debe interpretarse como un "sobreaprendizaje", a menos que su valor sea mucho menor que le corresponde en función de d y N. Por otra parte el error de estimación, que es el que realmente nos interesa, se puede hacer tan pequeño como queramos simplemente añadiendo observaciones (N), simpre que los parámetros libres no aumenten (d). Por ejemplo, en un problema de estimación de posición de una cámara a partir de correspondencias mundo-imagen en principio podemos aumentar todo lo que queramos la precisión usando más correspondencias, independientemente del nivel de ruido existente. Pero en un problema de *structure from motion*, donde cada nueva imagen de la secuencia añade parámetros desconocidos (nuevos puntos 3D y posición de cámara), hay un límite en la precisión que se puede conseguir, que tiene que ver con el ratio d/N.

D.5.3. Jacobianos interesantes

Para obtener el gradiente y el Hessiano de la función de coste necesitamos el Jacobiano de los residuos. Para calcular sus elementos podemos usar las reglas elementales de derivación, posiblemente con la ayuda de herramientas de cálculo simbólico, pero muchas veces es conveniente estudiar la estructura de las funciones involucradas para intentar organizar los cálculos de forma eficiente, reaprovechando cálculos de forma inteligente. Dos aplicaciones muy importantes en las que esto ocurre son los siguientes:

Bundle Adjustment: en la reconstrucción 3D a partir de múltiples vistas, cada coordenada de cada punto observado en cada imagen da lugar a un residuo. Las variables son las coordenadas de todos los puntos junto con los parámetros de todas las cámaras, dando lugar a un sistema de altísima dimensión pero muy disperso (*sparse*) que debe resolverse mediante técnicas especiales.

→ parámetros de cámara

Lucas-Kanade: cuando deseamos estimar la transformación que ha sufrido una imagen, cada pixel da lugar a un residuo, ya que su nivel de gris o color debe coincidir con el de la imagen transformada. Estos píxeles se muestrean de forma discreta en la optimización, pero el modelo de transformación supone un dominio continuo que tenemos que tener en cuenta para obtener el Jacobiano. Ahora $\boldsymbol{x}$ es el espacio de imagen, y $\boldsymbol{p}$ las variables de optimización. En cualquier posicion $\boldsymbol{x}$ de la imagen tenemos el residuo

$$f(\boldsymbol{x}, \boldsymbol{p}) = I(\boldsymbol{W}(\boldsymbol{x}, \boldsymbol{p})) - T(\boldsymbol{x})$$

donde I es la imagen observada y T es un *template* o prototipo. Para obtener los residuos transformamos (*warping*) la imagen I para situarla en región $\boldsymbol{x}$ ocupada por el template[2].

[esquema de las transferencia]

Los residuos se forman mediante la composición de las transformaciónes I y W, por lo que el Jacobiano, respecto a las variables de interés $\boldsymbol{p}$, se consigue mediante la aplicación de la regla de la cadena en su versión general multivariable [ref]. Cada elemento del Jacobiano es:

$$\frac{\partial f_k}{\partial p_j} = \frac{\partial I_k}{\partial x}|_{W(\boldsymbol{x},\boldsymbol{p})} \frac{\partial x}{\partial p_j} + \frac{\partial I_k}{\partial y}|_{W(\boldsymbol{x},\boldsymbol{p})} \frac{\partial y}{\partial p_j}$$

Es importante tener en cuenta que el gradiente hay que calcularlo en el dominio de imagen pero transformarlo al dominio del prototipo para que las derivadas se tomen en el lugar correcto. Esto se puede expresar de forma compacta como:

$$\nabla f = \nabla^\mathsf{T} I' \mathrm{J}_W$$

donde $\nabla^\mathsf{T} I'$ representa el gradiente transformado. En cada punto de imagen los elementos del Jacobiano de los residuos surgen de las contracciones de los gradientes de imagen con las dos componentes del Jacobiano de la transformación.

Además del prototipo, la imagen transformada y su gradiente, los residuos y las derivadas parciales se pueden organizar con la misma estrucutura rectangular de la imagen, de modo que el gradiente y hessiano necesario para la optimización de Gauss-Newton se puede expresar con operaciones elementales de proceso de imágenes.

En la práctica es preferible usar la variante "composicional inversa", que permite precalcular el Hessiano y el Jacobiano. El proceso de optimación es muy eficiente ya que solo es necesario transformar la imagen con la solución actual, obtener los residuos, y contraerlos con las imágenes de las derivadas respecto a los parámetros.

D.6. Optimización con restricciones

En ciertos casos deseamos obtener el extremo (máximo o mínimo) de una función $f : \mathbb{R}^n \to \mathbb{R}$ sobre una región restringida del espacio, definida mediante un conjunto de condiciones $\phi_k(\boldsymbol{x}) = 0$, o con desigualdades $\psi_k(\boldsymbol{x}) > 0$.

[2]Esto implica recorrer el dominio de T y obtener los valores en cada localización indicada por W, con una posible interpolación para mejorar la precisión. Las bibliotecas de procesamiento de imagen disponen de funciones para hacer *warping*, con las variantes directa e inversa dependiendo de en qué dominio se especifica el rectángulo de trabajo.

Veamos primero las restricciones de igualdad. La condición que cumple un extremo sin restricciones es $\nabla f(\boldsymbol{x}) = 0$. En principio podríamos introducir las restricciones explícitamente haciendo un cambio de coordenadas. Si hay r restricciones las n variables originales se pueden convertir en $n - r$ variables nuevas, ya sin restricciones. La función objetivo f se expresaría en función de estas nuevas coordenadas, y podríamos encontrar el extremo usando el procedimiento habitual.

> Supongamos que queremos encontrar el mínimo de $f(x, y) = 1/2(x^2 + y^2)$ en la curva $\phi(x, y) = y - \ln x = 0$. Si eliminamos $y = \ln x$ tenemos la función 'restringida' $f_\phi(x) = 1/2(x^2 + 2\ln x)$, cuyo extremo verifica $\nabla f_\phi = 0$, que en este caso se reduce a $x + x^{-1} \ln x = 0$, cuya solución es $x \simeq 0.65$. Y de ahí obtenemos $y \simeq -0.43$.
>
> → cambiar el ejemplo

Sin embargo, el método de los *multiplicadores de Lagrange* es mucho más sencillo y directo. Consiste en definir una nueva función objetivo (llamada *lagrangiano*) que incluye la original y una combinación lineal de las restricciones:

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \lambda_1 \phi_1(\boldsymbol{x}) + \lambda_2 \phi_2(\boldsymbol{x}) + \dots$$

El lagrangiano tiene más variables que la función original, pero, a cambio, sus puntos críticos ($\nabla L(\boldsymbol{x}, \boldsymbol{\lambda}) = 0$), ya sin restricciones, coinciden con los extremos de f sujeta a las restricciones $\boldsymbol{\phi}$. Las variables auxiliares λ (multiplicadores) suelen descartarse, aunque en algunos casos pueden tener una interpretación interesante.

> Por ejemplo, el problema anterior se resuelve creando la función $L(x, y, \lambda) = 1/2(x^2 + y^2) + \lambda(y - \ln x)$. Sus puntos críticos, $\nabla L = 0$, dan lugar a las condiciones:
>
> $$\frac{\partial L}{\partial x} = x - \frac{\lambda}{x} = 0, \qquad \frac{\partial L}{\partial y} = y + \lambda = 0, \qquad \frac{\partial L}{\partial \lambda} = y - \ln x = 0$$
>
> De las dos primeras eliminamos λ y deducimos que $x = -y/x$. Con la tercera obtenemos $x = -x^{-1} \ln x$, exactamente la misma condición que antes.

Las ventajas aparecen realmente cuando hay un número grande de restricciones complicadas, ya que evita reparametrizar la función con algún conjunto adecuado de variables, lo que normalmente es inviable. Pero hay que tener en cuenta que el punto crítico del lagrangiano es un 'punto de silla' (un máximo respecto a unas variables y un mínimo respecto a otras), por lo que su computación numérica no puede hacerse de forma ingenua con una simple minimización.

> El fundamento de este método puede entenderse de forma intuitiva. Las restricciones son hipersuperficies de $\mathbb{R}^n$ donde tiene que estar la solución; imagina una

curva en el plano. La función objetivo puede representarse mediante curvas de nivel. El máximo se conseguirá en algún punto de la curva de restricción que toque con una curva de nivel, pero que *no la cruce* (si la cruza significa que hay posiciones vecinas permitidas con valores mayores y menores de la función). Si la función objetivo y las restricciones son suaves en el punto de contacto tendrán la misma tangente, sus vectores normales serán paralelos. El lagrangiano L es una forma compacta de expresar esta condición en general: las derivadas de L respecto a $\boldsymbol{x}$ imponen la condición de que la normal a las curvas de nivel (∇f) es una combinación lineal de las normales a las restricciones ($\nabla \phi_k$), y las derivadas respecto a $\boldsymbol{\lambda}$ hacen que las restricciones $\phi_k(\boldsymbol{x})$ tomen exactamente el valor 0.

Cuando hay restricciones de desigualdad, del tipo $\psi_k(\boldsymbol{x}) > 0$, se puede utilizar un procedimiento similar al anterior teniendo en cuenta que, en la solución óptima, algunas desigualdades se cumplirán estrictamente (como igualdades) y otras con margen. Éstas últimas se pueden 'ignorar' en el análisis. Para formalizar esta idea se utilizan las condiciones KKT (Karush-Kuhn-Tucker).

→ explicar mejor

D.7. Programación lineal

D.8. Programación cuadrática

D.9. RANSAC

D.10. Minimización L1

Apéndice E

Expresiones útiles

E.1. Woodbury

La inversa de una matriz que sufre una actualización (típicamente de rango pequeño) se puede obtener mediante la siguiente expresión:

$$(\mathtt{C} + \mathtt{AB})^{-1} = \mathtt{C}^{-1} - \mathtt{C}^{-1}\mathtt{A}(\mathtt{I} + \mathtt{BC}^{-1}\mathtt{A})^{-1}\mathtt{BC}^{-1}$$

E.2. Completar el cuadrado

Para eliminar el término lineal en una expresión cuadrática hacemos lo siguiente:

$$ax^2 + bx + c = a(x - h)^2 + k$$

donde

$$h = \frac{-b}{2a}$$

$$k = c - \frac{b^2}{4a}$$

E.3. Producto kronecker y operador "vec"

Útil para extraer las incógnitas de una ecuación matricial:

$$\operatorname{vec}(\mathtt{AXB}) = (\mathtt{B}^{\mathsf{T}} \otimes \mathtt{A}) \operatorname{vec}(\mathtt{X})$$

(El operador vec concatena todas las columnas de una matriz en un vector.)

E.4. Subespacio nulo de la representación "outer"

Apéndice F

Entornos de Cálculo Científico

F.1. Matlab/Octave/R

F.2. Mathematica/Maxima

Bibliografía

[1] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.

[2] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] R.A. Brooks. `http://www.ai.mit.edu/people/brooks/index.shtml`.

[4] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2(2), (`www.kernel-machines.org`), 1998.

[5] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons (hay una nueva edición del 2000), 1973.

[6] M. Ebden. Gaussian processes for regression: A quick introduction.

[7] Página web de *GeoBot*. `http://dis.um.es/~alberto/geobot/geobot.html`.

[8] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Univ Pr; 3rd edition, 1996.

[9] R.C. Gonzalez and R.E. Woods. *Digital Image Processing (2nd)*. Prentice Hall, 2002.

[10] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2003.

[11] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.

[12] D.R. Hofstadter and FARG. *Fluid Concepts and Creative Analogies*. Basic Books, 1995.

[13] E.T. Jaynes. *Probability Theory. The Logic of Science*. Cambrigde University Press, 2003.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[15] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications (2 ed.).* Springer Verlag, 1997.

[16] P.E. López-de-Teruel. *Una Arquitectura Eficiente de Percepció de Alto Nivel: Navegació Visual para Robots Autónomos en Entornos Estructurados.* Tesis Doctoral, Universidad de Murcia, `http://ditec.um.es/~pedroe/documentos/Tesis.pdf`, 2003.

[17] G.J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions.* John Wiley and Sons, 1997.

[18] K.-R. Müler, S. Mika, G. Rätsh, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Trans. on Neural Networks*, 12(2):181–202, 2001.

[19] A. Papoulis. *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill; 3rd edition (hay una 4 edición del 2001), 1991.

[20] R. Penrose. *Lo grande, lo pequeño y la mente humana.* Cambrigde University Press, 1997.

[21] M. Pollefeys. *Tutorial on 3D Modeling from Images.* `http://www.esat.kuleuven.ac.be/~pollefey`, 2000.

[22] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[23] Carl Edward Rasmussen and Chris Williams. *Gaussian Processes for Machine Learning.* 2006.

[24] J.R. Searle. *El misterio de la conciencia.* Paidós, 1997.

[25] Richard Szeliski. *Computer Vision: Algorithms and Applications.* Springer, 2010.

[26] E. Trucco and A. Verri. *Introductory Techniques for 3D Computer Vision.* Prentice Hall, 1998.

[27] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, 1995.

[28] V. Vapnik. *Statistical Learning Theory.* Wiley, 1998.