# Nonlinear Kernel-Based Statistical Pattern Analysis

Alberto Ruiz Pedro E. López-de-Teruel
*Department of Computer Science, University of Murcia, Spain*
aruiz@dif.um.es, pedroe@ditec.um.es

**Abstract.** The eigenstructure of the second order statistics of a multivariate random population can be inferred from the matrix of pairwise combinations of inner products of the samples. Therefore, it can be also efficiently obtained in the implicit, high dimensional feature spaces defined by kernel functions. We elaborate on this property to obtain general expressions for immediate derivation of nonlinear counterparts of a number of standard pattern analysis algorithms, including Principal Component Analysis, data compression and denoising, and Fisher's discriminant. The connection between kernel methods and nonparametric density estimation is also illustrated. Using these results we introduce the kernel version of Mahalanobis distance, which originates nonparametric models with unexpected and interesting properties, and also propose a kernel version of the Minimum Squared Error linear discriminant function. This learning machine is particularly simple and includes a number of generalized linear models such as the Potential Functions method or the Radial Basis Functions Network. Our results shed some light on the relative merit of feature spaces and inductive bias in the remarkable generalization properties of the Support Vector Machine. Although in most situations the Support Vector Machine obtains the lowest error rates, exhaustive experiments with synthetic and natural data show that simple kernel machines based on pseudoinversion are competitive in problems with appreciable class overlapping.

**Keywords:** Support Vector Machine, Kernel Expansion, Nonlinear Feature Extraction, Mahalanobis Distance, Fisher's Discriminant Analysis, MSE Estimation, Pseudoinverse, Nonparametric Statistics.

## 1 Introduction

Nonlinear information processing algorithms can be designed by means of linear techniques in implicit feature spaces induced by *kernel* functions. This idea can be traced back to the *Potential Functions* method [3] [1] (see also [24] [55]) and it has been successfully applied to the Support Vector Machine (SVM), a learning method with controllable capacity which obtains outstanding generalization in high (even infinite) dimensional feature spaces [6] [49] [51] [52] [10].

The kernel method can be used if the interactions between elements of the domain occur only through inner products. This suggests the possibility of building nonlinear, kernel-based counterparts of standard pattern analysis algorithms. Recently, a nonlinear feature extraction method has been presented [39] based on a kernel version of Principal Component Analysis (PCA) and [27] proposed a nonlinear kernel version of Fisher Discriminant Analysis. Kernel-based versions of other pattern analysis algorithms have been also proposed in [13] [21] [17] among others; the field of the so-called *Kernel Machines* is now extremely active [38] [43].

The objective of this paper is twofold. First we present a concise expression for the structure of the second order statistics of a multivariate random sample, useful for translation into kernel versions of classical pattern analysis algorithms initially based on linear dependencies or gaussian models. The method is illustrated with immediate derivations of the above mentioned kernel versions of Fisher's discriminant, Principal Component Analysis and associated procedures for nonlinear data compression and denoising. First order methods (e.g. kernel versions of template matching) are also considered and shown to be closely related to standard nonparametric statistical techniques (Parzen's method).

Using this technique we introduce the kernel version of Mahalanobis distance, not previously considered in the literature, which originates nonparametric models with unexpected and interesting properties, and also propose a kernel version of the Minimum Squared Error (MSE) linear machine. This learning algorithm is particularly simple and includes a number of generalized linear models such as the Potential Functions method or the Radial Basis Functions Network. In both cases learning is based on matrix pseudoinversion by diagonalization (complexity O($n^3$)), so the run time essentially depends on the size of the training data, and not on problem complexity (e.g., noise or class overlapping).

The second goal of this paper is to elucidate the relative responsibility of feature spaces and inductive bias in the remarkable properties of Support Vector Machines (SVMs). The SVM combines two powerful ideas: maximum margin classifiers with low capacity and therefore good generalization (based on a quadratic optimization procedure giving rise to sparse solutions), and implicit feature spaces defined by kernel functions. Our experiments show that simple learning algorithms can successfully take advantage of kernel expansions to obtain very satisfactory solutions, although in most cases the highest generalization can only be achieved by the maximum margin strategy of SVMs. However, in situations characterized by large class overlapping, SVM learning is significantly slower and not more accurate than the proposed kernel machines based global statistical properties of the sample.

The paper is organized as follows. Section 2 describes the relationship between the eigenstructures of the sample autocorrelation (or covariance) matrix and the matrix of inner products of the sample. Section 3 extends the results to kernel feature spaces. In sections 4 to 7 we use this result to derive kernel versions of standard pattern analysis algorithms. For clarity, exposition follows logical connections instead of chronology: Section 4 explores the link between first order kernel methods and nonparametric statistics; Section 5 introduces the kernel version of Mahalanobis distance and describes some of its properties; Section 6 derives kernel PCA and related algorithms; and Section 7 considers kernel versions of linear machines, focussing on the MSE criterion. In Section 8 the proposed methods are experimentally contrasted in synthetic and natural problems with state-of-the-art SVM implementations. The last section summarizes the contributions of this paper.

## 2 The Essential Relation between $X^T X$ and $X X^T$

Objects in a certain domain are usually described by $l$-dimensional attribute vectors $x \in \mathbb{R}^l$, which can be considered as realizations of a multivariate random variable. When the probability density function $p(x)$ is unknown or its shape is assumed to be simple, the population is typically characterized in a first approximation by its first and second order moments: the mean vector $\eta = \mathrm{E}\{x\}$ and the covariance matrix $\Sigma = \mathrm{E}\{(x-\eta)(x-\eta)^T\}$. They describe the location and dispersion of the population and are the sufficient statistics for many pattern recognition and data analysis algorithms.

In practice, we estimate these parameters from a finite sample. Let $X$ be a $m \times l$ sample matrix containing as rows $m$ random observations $x_i$, $i$=1..$m$, drawn from the underlying law $p(x)$. Consider the following statistics, concisely expressed in terms of the sample matrix:

$$\mu = X^T \vec{1} \qquad \text{(sample mean)} \quad (1)$$

$$S = \tfrac{1}{m} X^T X \qquad \text{(sample autocorrelation matrix)} \quad (2)$$

$$C = \tfrac{1}{m} X^T X - X^T \vec{1}\, \vec{1}^T X \qquad \text{(sample covariance matrix)} \quad (3)$$

where $\vec{1}$ denotes a $m$-dimensional vector with all the components equal to $m^{-1}$.

Any real symmetric and positive semidefinite matrix $M$ of rank $d$ can be diagonalized as

$$M_{m\times m} = P^T{}_{m\times d}\, G_{d\times d}\;\; P_{d\times m}$$

where $G$ is a (nonsingular) diagonal matrix containing only the positive eigenvalues of $M$, and $P$ is a row orthonormal system ($P\,P^T = I_d$) containing the associated eigenvectors, which span the subspace not degenerate of $M$. $I_d$ denotes the identity matrix of dimension $d$. We will denote this decomposition by the terms "eigenstructure" or "spectral decomposition". The eigenstructure of the covariance matrix discloses some intrinsic geometric properties of the random population, independent from the reference system (e.g., principal directions and associated variances).

It turns out that the spectral decompositions of $S$ and $C$ can be obtained respectively from the $m \times m$ symmetric matrix $K$ of pairwise combinations of dot products of the samples:

$$K = \{x_i \cdot x_j\}_{i,j} = X\,X^T \quad (4)$$

and from the centered matrix $K_c$ defined by

$$K_c = K - \mathbf{1}K - K\mathbf{1} + \mathbf{1}K\mathbf{1} \quad (5)$$

where $\mathbf{1}$ denotes a $m \times m$ matrix with all entries equal to $1/m$. The following theorem and corollaries are proved in the Appendix:

**Theorem**. *Let the eigenstructures of S, C, K and $K_c$ be denoted by:*

$$S = U^T D\, U \tag{6}$$

$$C = V^T E\, V \tag{7}$$

$$K = \alpha^T \Lambda\, \alpha \tag{8}$$

$$K_c = \beta^T \Omega\, \beta \tag{9}$$

*then the following relations are valid*:

$$D = \frac{1}{m}\Lambda \tag{10}$$

$$U = \Lambda^{-1/2}\ \alpha\, X \tag{11}$$

$$E = \frac{1}{m}\Omega \tag{12}$$

$$V = \Omega^{-1/2}\ \beta\, X \tag{13}$$

This fundamental relation between $X^T X$ and $X\, X^T$ is typically applied to efficient computation of eigenstructures in the small sample size case [14](p.39). It is particularly useful for standard PCA in image representation [48](p.268).

The pseudoinverse $M^+$ of a symmetric square matrix $M$ with eigenstructure $M{=}P^T G\, P$ is computed by inversion of the nonzero eigenvalues: $M^+{=}P^T G^{-1}\, P$. It gives the minimum squared error approximation $I_M = M^+M = P^TP$ to the identity and acts as the true identity for both $M$ and $M^+$ (e.g. $I_M\, M = M$, $M^+I_M = M^+$, an so on). The pseudoinverse $R^+$ of a rectangular matrix $R$ is computed in terms of the pseudoinverse of the symmetric $R^TR$ as $R^+{=}(R^TR)^+R^T$, with the property that $R^+R$ acts on $R$ as a right identity: $RR^+R = R$. In general, the pseudoinverse of any matrix can be computed by inversion of its nonzero singular values. It is typically used to obtain minimum squared error solutions of overconstrained systems of linear equations. Loosely speaking, pseudoinversion restricts inversion to the range of the operator, i.e., the subspace where it is not degenerate. This is unavoidable in high dimensional feature spaces. We will show that pseudoinversion provides reasonable approximations in the algorithms of interest.

**Corollary 1**. *The following equalities are valid*:

$$S^+ = m\, X^T\ \alpha^T\ \Lambda^{-2}\ \alpha\, X \tag{14}$$

$$C^+ = m\, X^T\ \beta^T\ \Omega^{-2}\ \beta\, X \tag{15}$$

Therefore the action of $S^+$ or $C^+$ over a target vector $t$ can be obtained from certain associated operators working on the linear span of the whole sample. Let $t_X$ denote the expansion of $t$ in terms of the inner products with the elements in $X$:

$$t_X = X\, t = (x_1{\cdot}t,\, x_2{\cdot}t,\, \ldots,\, x_m{\cdot}t)^T \tag{16}$$

For example, the counterpart of $C^+$ in the space of inner products is $m\beta^T\Omega^{-2}\beta$, since we can write:

$$t^T C^+ t = m\, t_X^T\ \beta^T\ \Omega^{-2}\ \beta\, t_X$$

This property is crucial for working in implicit high dimensional feature spaces defined by kernel functions, as proposed in [39] for nonlinear PCA.

The above relations are particular cases of

**Corollary 2**. *Let A be any symmetric matrix. Then the following equality is valid for any vectors t and u and any integer power n:*

$$t^T\ (X^T AX)^n\ u \quad = \quad t_X{}^T\ \left(A^{1/2}\left(A^{1/2}\ K\ A^{1/2}\right)^{n-1} A^{1/2}\right) u_X \tag{17}$$

The equality is trivial for positive powers, but it also holds for negative powers in the pseudoinverse sense[1]. For appropriate $A$ and $n$, expression (17) provides a conceptually elegant tool for building nonlinear versions of standard algorithms. Note that the left side contains matrix-vector products of dimension $l$, very large or even infinite when working in feature spaces, while the dimension involved in the right side equals the sample size $m$. In practical situations we will find sometimes preferable to work with the explicit decomposition (15), since $A^{1/2}$ requires additional computing effort.

# 3 Extension to Kernel-based Feature Spaces

Let the kernel function $k$ define the inner product in some implicit feature space with associated transformation $\phi$, which need not be computed explicitly:

$$k(x, y) = \phi(x) \cdot \phi(y)$$

For instance, the polynomial kernel [51], denoted by $\mathcal{P}_d$, induces features which include products of up to $d$ attributes:

$$k(x,y) = \mathcal{P}_d(x,y) \equiv (1 + x \cdot y)^d \qquad (18)$$

The gaussian or RBF (Radial Basis Function) kernel [51], denoted by $\mathcal{G}_\gamma$, induces an infinite dimensional feature space in which all image vectors have the same norm and become orthogonal when they are distant in the input space with respect to the scale or regularization parameter $\gamma$:

$$k(x,y) = \mathcal{G}_\gamma(x,y) \equiv e^{-\frac{1}{2}\left(\frac{\|x-y\|}{\gamma}\right)^2} \qquad (19)$$

Other kernel generating functions are also used, giving rise to neural network structures, splines or Fourier expansions. Any function verifying Mercer's condition [52] [9] can be used as a kernel.

Since the interaction between target vectors occurs only in the form of inner products, the results in section 2 are also valid in the feature space defined by any desired kernel function $k$. To do this, the matrix $K$ must be replaced by

$$K = \{k(x_i, x_j)\}_{i,j} \qquad (20)$$

and the expansion $t_X$ must be replaced by

$$t_X = (k(x_1, t), k(x_2, t), \ldots, k(x_m, t))^T \qquad (21)$$

The above transformation is sometimes referred to as the *empirical kernel map* [41]. In the following we will take advantage of the results of the previous section to write kernel versions of standard pattern analysis algorithms.

# 4 First Order Kernel Methods and Nonparametric Statistics

One of the simplest pattern classification situations occurs when the attribute vector is normally distributed in each class and the attributes are conditionally independent with the same variance (i.e., spherical classes of the same size). In this case the optimum classification rule depends essentially on the distances $\|t - \mu_i\|$ from the input $t$ to the mean value $\mu_i$ of each class, which can be reduced to comparison, with an adequate threshold, of the "correlations" $t^T\mu_i$ [12] [14]. In many realistic situations this simplifying assumption is too inaccurate and more expressive models are required. However, it is interesting to realize that some of these more powerful methods are closely related to minimum distance classifiers in certain feature spaces.

[1] Depending on the properties of $A$ and X and on the dimensions $m$ and $l$, any of the matrices raised to $n$ and $n$–1 respectively in the left and right sides of the above expression may be singular; $A^{1/2}$ can be directly computed from the spectral decomposition of $A$, and it will be also symmetric.

Consider Parzen's method [32] for nonparametric estimation of the probability density $p(\cdot)$ of a random sample $X=\{x_i\}$, $i=1..m$. Let $b_h(\cdot)$ be a simple (e.g., normal) density with zero mean and standard deviation (width) $h$. Then the unknown density $p(\cdot)$ can be estimated as the average value of copies of $b_h(\cdot)$ centered on the available samples:

$$p(t) \cong \hat{p}_h(t) = \frac{1}{m}\sum_{i=1}^{m} b_h\left(\|t-x_i\|\right)$$

The smoothing parameter $h$ controls the amount of regularization, and in practice (finite samples) it can be adjusted by cross-validation [26]. Note that the Parzen estimator can be written as the average value of the empirical map (21) with the smoothing density acting as a kernel ($k(x,y) = b_h(\|x-y\|)$):

$$\hat{p}_h(t) = t_X \cdot \vec{1} \qquad (22)$$

This suggests a fundamental relationship between kernel methods and nonparametric statistics. Using (1), $t_X \cdot \vec{1}$ can be interpreted in feature space as $\phi(t)^T\mu_\phi$, the key term for minimum distance classification ($\mu_\phi$ denotes the mean value of the samples in feature space). This result follows from the fact that Parzen estimations are just low-pass filtered versions of the original density, computed by Monte Carlo integration:

$$\begin{aligned} p(t) &\cong p(t) * b_h(t) = \int_X p(x) b_h(t-x)dx = E_{p(x)}\{b_h(t-x)\} = E_{p(x)}\{k(t,x)\} = \\ &= E_{p(x)}\{\phi(t)\phi(x)\} = \phi(t)E_{p(x)}\{\phi(x)\} = \phi(t)\cdot\mu_\phi \end{aligned} \qquad (23)$$

In general, expanding $d(t) = \|t-\mu\|$ in terms of inner products and using (1), (4), (16) and redefinitions (20) and (21), the distance to the mean in feature space (denoted by KD) can be written as:

$$\begin{aligned} d^2(t) &= t^T t + \mu^T\mu - 2t^T\mu = t^T t + \vec{1}XX^T\vec{1} - 2\,t^T X^T\vec{1} = \\ &= k(t,t) - \vec{1}K\vec{1} - 2t_X\vec{1} \end{aligned} \qquad (24)$$

For the above case of a kernel derived from a density, we have:

$$d^2(t) = c - 2\hat{p}_h(t)$$

where constant $c = b_h(0) - \vec{1}K\vec{1}$ (independent from $t$) can be precomputed. Therefore Parzen density estimation (a very flexible kind of model) is equivalent to simple distance to the mean in the implicit feature space induced by the smoothing density $b_h(\cdot)$.

Different kind of interpolating functions (e.g. polynomial) in Parzen estimations need not produce legitimate densities nor reasonable data models. A natural question arises on the modeling quality of KD, the full-fledged distance to the mean in the feature space (24), induced by arbitrary but otherwise perfectly admissible kernels, where the Parzen-like term is adjusted by $k(t,t)$ and the average of the $K$ entries.

In general, this modification –suggested by the theory of kernel expansions, without immediate interpretation in the context of nonparametric statistics– actually creates somewhat crude and inaccurate data models which can be nevertheless useful in particular circumstances (e.g. pattern classification with boundaries of low complexity), although local closeness to data is in general not correctly measured. First-order kernel models are not flexible enough to capture arbitrary data distributions, unless specific kernels are used (e.g., derived from densities, hence producing ordinary Parzen estimations). However, in the following sections we will show that second order kernel methods, which take into account data dispersion in feature space, originate remarkably satisfactory models with novel and unexpected properties not found in standard nonparametric methods.

We conclude this section with a brief commentary on the possibility of constructing kernel versions of nonparametric methods themselves. Since they are in some sense equivalent to kernel methods (e.g., $\hat{p}_h(t) = \phi(t)\cdot\mu_\phi$), they actually take advantage of the infinite set of features implicitly induced by the smoothing function. Therefore, an additional "kernelization" will have probably no substantial effects[2]. In kernel feature

[2] For instance, Parzen estimation in feature space leads to the concept of *iterated kernel*, where the distance required by the argument of the interpolating density is derived from a generalized inner product. However, neither the iterated kernel nor the corresponding estimate

spaces, simple (first or second order, e.g., gaussian) models and complex (higher order, semiparametric or nonparametric) ones are essentially equivalent.

## 5 Kernel Mahalanobis Distance (KM)

The Mahalanobis distance $\delta$ from a vector $t$ to a random population $X$ with sample mean $\mu$ and sample covariance matrix $C$ is given by the quadratic form

$$\delta^2(t) = (t-\mu)^T C^{-1}(t-\mu) \tag{25}$$

which in terms of the sample matrix $X$ can be expressed as

$$\delta^2(t) = (t - X^T \vec{1})^T (X^T Z^2\ X)^{-1}(t - X^T \vec{1}) \tag{26}$$

where $Z$ is the square root of the centering matrix required to write $C$ with structure $X^TZX$:

$$Z = \left(\frac{1}{m}(I_m - \mathbf{1})\right)^{1/2} \tag{27}$$

Using the main relation (17) we can write:

$$\delta^2(t) = (t - X^T \vec{1})^T X^T Z\ (Z\ K\ Z)^{-2} Z\ X(t - X^T \vec{1}) \tag{28}$$

Let us introduce the following notation:

$$\mu_X = X\mu = XX^T \vec{1} = K\vec{1} \tag{29}$$

$$M = Z\ (Z\ K\ Z)^{-2} Z \tag{30}$$

where exponent –2 actually denotes the squared pseudoinverse. Clearly, $\mu_X$ and $M$ , which are directly obtained from the matrix of inner products $K$, play respectively the role of the mean vector and the inverse of the covariance matrix in the space of projections over $X$. Using the above notation the Mahalanobis distance can be concisely written as:

$$\delta^2(t) = (t_X - \mu_X)^T M\ (t_X - \mu_X) \tag{31}$$

In practice it is more efficient to write the Mahalanobis distance as the norm of a transformed vector, by using the spectral decomposition for the covariance matrix $C$ in terms of $K_c$ (eqs. (7) (12) and (13)). Then we have

$$\delta(t) = \|\ \Theta\ (t_X - \mu_X)\ \| \tag{32}$$

where

$$\Theta = \sqrt{m}\ \Omega^{-1}\ \beta \tag{33}$$

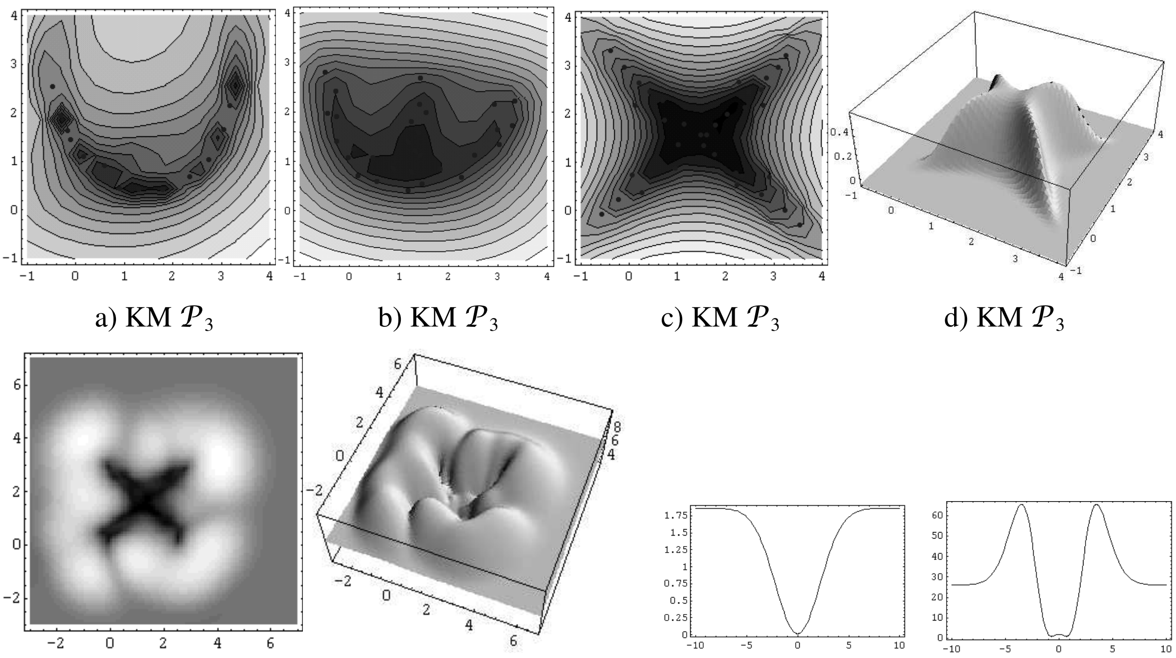
plays the role of the whitening transformation in the projection space.

Through appropriate redefinition of $K$ and $t_X$, expressions (31) or (32) allow immediate computation of Mahalanobis distance in implicit feature spaces defined by kernel functions. Fig. 1 shows illustrative 2D data sets and the level curves and surface plots of the corresponding Mahalanobis distance in different feature spaces[3]. Simple (essentially gaussian) models in the feature space project back into powerful and expressive models of complex distributions in the input space.

---

are legitimate densities in input space (in the feature space induced by $\mathcal{G}_\gamma$ no transformed vector lies at an infinite distance of any other transformed vector, so the iterated kernel never vanishes). In any case, the regularization effect of the iterated kernel can be easily accomplished by an ordinary smoothing density with equivalent bandwidth. Other nonparametric methods such as the *k-n* nearest neighbor classification rule also remain essentially unchanged in the feature space induced by the usual kernel functions.

[3] In figures we often use logarithmic or negative exponential scaling functions to emphasize model structure.



a) KM $\mathcal{P}_3$ b) KM $\mathcal{P}_3$ c) KM $\mathcal{P}_3$ d) KM $\mathcal{P}_3$

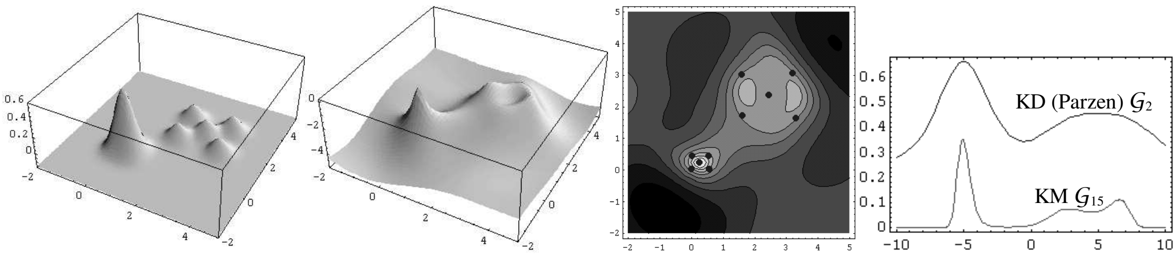
e) KM $\mathcal{G}_2$ f) KM $\mathcal{G}_2$ g) KD (Parzen) $\mathcal{G}_2$ h) KM $\mathcal{G}_2$

**Figure 1**. Contour lines and surface plots of nonlinear Mahalanobis distance for illustrative distributions: a) *U* shape. b) *fork* shape. c-d) *X* shape. e-f) Overshooting effect of RBF kernel (see text) for the *X* shape. g-h) Overshooting effect of KM compared with standard Parzen estimation in the illustrative 1D dataset $X=\{-1, 0, 1\}$.

Through variance equalization in feature space, Kernel Mahalanobis distance (in the following denoted by KM), produces a new kind of nonlinear model with some unexpected properties not found in standard nonparametric methods. Note first that highly satisfactory models can be obtained even with polynomial kernels, which in general are not appropriate for models based only on first order statistics and are completely useless for Parzen estimation.

Furthermore, KM presents interesting properties in the feature space induced by gaussian kernels. With adequate smoothing parameters there is an "overshooting" or "Mexican hat" effect (see Fig. 1.e-h), which in some sense detects the boundaries of the distribution. Distance to the population is higher in the neighborhood of the data than in farther locations. This behavior may be useful in certain applications (see Section 8).

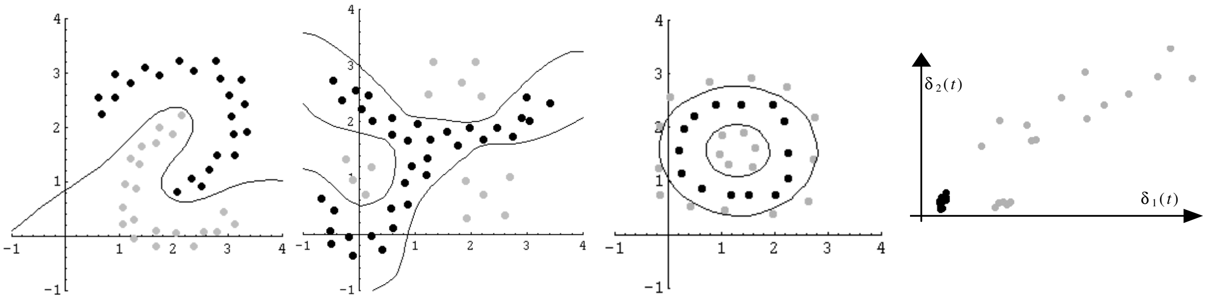
A well-known problem of certain nonparametric methods is the selection of the smoothing parameter for data distributions containing clusters of different dispersion. Large values required by the more disperse clusters destroy density details in the more concentrated clusters, and conversely, small values create artifacts in low dispersion zones. Such mixture distributions require variable parameter techniques [22]. Gaussian KM presents a remarkable ability to perform automatic, data dependent smoothing. For specific values of the kernel parameter $\gamma$, variance normalization in feature space automatically adapts the effective local bandwidth to reasonable values in different locations of the input space. Structure can be detected and modeled at a much smaller scale than the one attributed to the kernel width in ordinary Parzen estimates, as shown in Figure 2.




a) KD (Parzen) $\mathcal{G}_{0.3}$ b) KM $\mathcal{G}_{1.5}$ c) KM $\mathcal{G}_{1.5}$ d) KD vs. KM

**Figure 2**. Illustration of variable effective smoothing by Kernel Mahalanobis distance (KM). a-c) Comparison of KD (Parzen) and KM for the 2D distribution shown in c). a) The small $\gamma$ required to obtain a sharp peak is inappropriate for the more disperse cluster. b) KM obtains a more satisfactory model on both regions using a unique regularization parameter. c) KM contour plot and sample locations for (a)-(b). d) Comparison for the 1D dataset $X$={-6,-5,-4,2,5,8}. In this case we adjust Parzen estimation to obtain a good model for the large cluster, at the cost of destroying localization of the small one. Again, KM obtains a satisfactory model for the whole distribution.

Kernel Mahalanobis distance can be used to design very satisfactory pattern classifiers, as illustrated in Fig. 3. The classification boundaries are polynomial hypersurfaces of degree 2. So in the input space the boundaries have double degree that the expansion induced by the kernel. Nevertheless, as demonstrated in the experimental section, they don't seem to produce extreme overfitting when compared with kernel linear machines (see the next section) or with Parzen (first order) models.




a) KM $\mathcal{P}_2$ b) KM $\mathcal{G}_5$ c) KM $\mathcal{P}_2$ d) Distance scatterplot of c)

**Figure 3**: a-c) Classification boundaries based on kernel Mahalanobis distance in illustrative toy problems. d) scatter plot of dataset c) in terms of KM distances to both classes, showing that classification based just on minimum KM distance will not work, requiring a more elaborated, though still simple (e.g. linear) decision rule in this space (see text).

Parzen estimates correspond to spherical gaussian densities in feature space and, by construction, they are legitimate densities in input space. KM distance is based essentially on full covariance gaussian models in feature space, but unfortunately they do not induce normalized functions in input space. In probabilistic classification, normalization constants affect the decision threshold of the likelihood ratio, which also depends on the *a priori* probability of the classes, so classifiers directly based on minimum Mahalanobis distance may sometimes show poor performance. Although some normalization strategies could be conceived, in practice KM should be considered just as some kind of relative likelihood function, and classification design must be based on the well-known process of quadratic classifier design explained in [14](p.154). Distances from the input vector to the classes of interest are used as new attributes of the input vectors, and it is in this highly discriminant and low-dimensional feature space where a simple decision function (i.e. linear) can be built with negligible computing effort. In binary classification we can even display 2D scatter plots (see Fig 3-d), allowing some kind of human supervision. In this paper most KM classifiers will be automatically designed using linear discriminant functions with minimum squared error (MSE) to the desired output on the space of KM distances.

The interpretation and effects of pseudoinversion are fundamental issues in our approach. In the high dimensional feature spaces considered here, the sample covariance matrices are singular, so the ordinary Mahalanobis distance is infinite almost everywhere. Pseudoinversion computes Mahalanobis distance in the subspace spanned by the samples, extrapolating to the rest of the space by orthogonal projection. This behavior is reasonable in our context, since distances, as a function of the original variables, should not change when the samples are embedded in a space with additional irrelevant attributes. However, pseudoinversion is based on inversion of the nonzero eigenvalues, which raises a numerical stability problem. Due to round-off errors it is not easy to identify the true null eigenvalues if they exist, and what is a more serious problem, certain kernels (e.g., $\mathcal{G}_\gamma$ with large $\gamma$) often produce an extremely ill-conditioned matrix $K$. Model quality and classifier accuracy depend on the amount of spectral mass assumed to be zero. Except when explicitly mentioned, in the experimental section we will always perform pseudoinversion discarding the proportion of the spectral mass $P = 10^{-5}$, corresponding to the lowest eigenvalues, although fine tuning of this proportion may improve the results in some cases.

# 6 Kernel Principal Component Analysis and Nonlinear Denoising

The Kernel PCA algorithm introduced in [39] is a very fruitful contribution to nonlinear pattern processing via implicit feature expansions. Using (15), it can be concisely expressed as:

$$t_{PCA} = \Omega^{-1/2}\beta\,(t_X - \mu_X) \tag{34}$$

where the components of the transformed vector are usually limited to those corresponding to the dominant eigenvalues in $\Omega$. We are inspired in that work, since the principal components are related to the transformations required to compute the Mahalanobis distance and the rest of learning machines proposed here. They show the level curves of some nonlinear principal features in 2D toy distributions, which tend to correctly separate different subgroups capturing the underlying geometric structure. They also use the transformed vectors as improved inputs for support vector machines in classification tasks. Our results continue this research line in several directions:

First, our experiments suggest that the variance captured by the different eigenvalues may sometimes be a misleading cue to the importance of the associated eigenvector, since some forms of nonlinearity (e.g. powers) may increase dispersion just because of the scale of measurements. In our approach all the information in the second order statistics is used, i.e., all the nonnegligible eigenvectors are taken into account with suitable weights.

On the other hand, the two-stage approach to classification mentioned above may not improve significantly the error rates with respect to a SVM with the same kernel working on the original data. In the implicit feature space the SVM must find a separating hyperplane with maximal margin (traded-off with error cost), and this cannot be facilitated by any linear transformation of the data. And the situation becomes worse if dimensionality is reduced. Note also that feature extraction for classification problems should be based on statistical separability criteria (e.g. Fisher's) instead of PCA, oriented to optimal data reconstruction [14].

The efficiency in the execution stage of SVMs is also lost due to the nonsparsity of the feature extraction stage. A more reasonable approach with nearly the same computing effort is building directly a classifier such as some variety of Kernel MSE (proposed in Section 7) or minimum Kernel Mahalanobis distance. As Kernel PCA, these methods do not obtain sparse solutions[4], but in many cases the classification problem will be essentially solved.

We will study now the problem of approximate reconstruction of a vector from its principal components in nonlinear feature spaces. The main problem is that not every vector in a high dimensional feature space has a preimage. We would like to find the vector $z$ in the input space whose image $\phi(z)$ is closest to a desired vector $\psi$ in the feature space. The kernel technique can be used when $\psi$ is expressed as a linear combination of images of input vectors. Taking advantage of the results in Section 2, we can easily derive a simple procedure for approximate reconstruction of a vector using nonlinear PCA. A more complete treatment can be found in [41], which shows attractive results in digit denoising.

Let $\psi$ be the reconstruction of a vector $t$ from its principal components:

[4] In fact, in classification problems with large intrinsic error, requiring high degrees of regularization, sparsity of SVM solutions also deteriorates considerably (see the experiments in Section 8).

$$\psi = V^T t_{PCA} + \mu \tag{35}$$

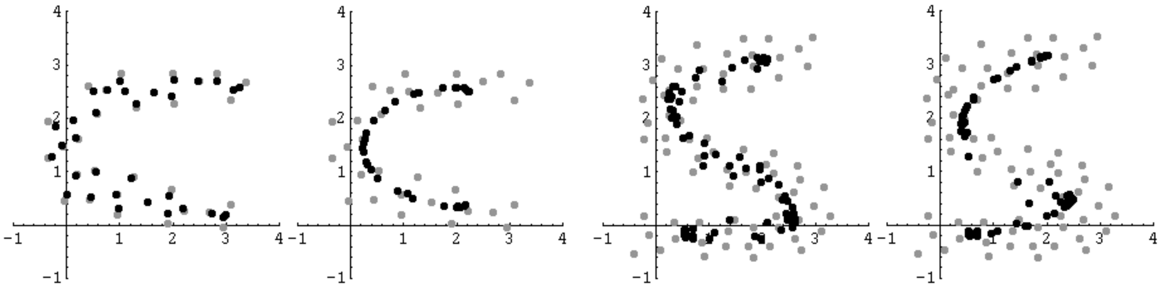
where $t_{PCA}$ are coordinates of $\phi(t)$ in the base of eigenvectors of the sample covariance matrix, given by (34). Inserting (34), (13) and (1) into the above expression we have:

$$\psi = \left(\Omega^{-1/2}\beta\, X\right)^T \left[\Omega^{-1/2}\beta(t_X - \mu_X)\right] + X^T \vec{1} \tag{36}$$

Minimization of $\|\psi - \phi(z)\|^2$ reduces to minimization of $J(z) = k(z, z) - 2\ \psi^T\phi(z)$, which after some manipulation can be concisely written as:

$$J(z) = k(z,z) - 2\ z_X^T \left[\beta^T \Omega^{-1} \beta (t_X - \mu_X) + \vec{1}\right] \tag{37}$$

Given a target vector $t$, the vector $z^*$ that minimizes $J(z)$ can be considered as an approximate reconstruction of $t$ from its (nonlinear) principal components. In low dimensional input spaces, efficient numerical optimization can be performed over the whole input space. In general, (37) can be used to find the point in a given database (e.g. the sample data) whose image is closest to any desired vector. Fig. 4 illustrates this kind of nonlinear denoising in a number of 2D toy data distributions.



a) 8 comp. b) 2 comp. c) 4 comp. d) 3 comp.

**Figure 4**: Illustration of data denoising through reconstruction from the principal nonlinear components induced by the gaussian kernel. Note that this procedure is able to find structure in data.

## 7 Kernel Minimum Squared Error (KL)

A function $f$: $\mathbb{R}^l \to \mathbb{R}$ is a discriminant (decision) rule for a class of objects $C \subset \mathbb{R}^l$ if $f(x) > 0$ for $x \in C$ and $f(x) \leq 0$ otherwise. Linear discriminant functions, defined by $f(x) = w \cdot x$, are frequently used because of their mathematical simplicity. A popular method to learn (find) a good weight vector $w$ from examples is based on the solution of the following system of linear equations:

$$X\, w = b \tag{38}$$

where $X$ denotes the sample matrix and $b$ is the vector of associated class labels: $b_i = +1$ if $x_i \in C$ and $b_i = -1$ otherwise. When the sample size is reasonably large ($m>n$) the system is overconstrained and the minimum squared error (MSE) solution is sometimes recommended [12]. It provides the best linear asymptotic approximation to the Bayes rule and obtains acceptable separation of classes with simple distributions.

The vector $w^*$ that minimizes the MSE cost

$$J(w) = \|Xw - b\|^2 \tag{39}$$

can be written as

$$w^* = X^+ b \tag{40}$$

in terms of the pseudoinverse $X^+$ of the rectangular matrix $X$:

$$X^+ = (X^T X)^+ X^T \tag{41}$$

A kernel version of the MSE linear discriminant can be directly obtained from (14). The discriminant function on a vector $t$ can be written as

$$t^T w^* = t \cdot (X^T X)^+ X^T b = \tag{42}$$

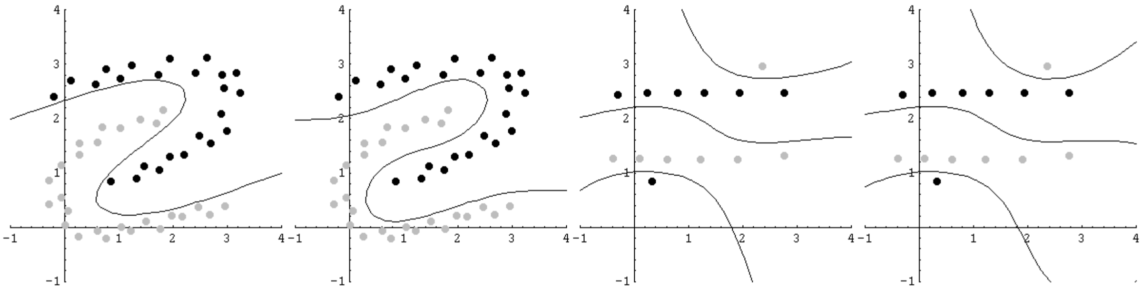
$$= t^T (X^T \alpha^T \Lambda^{-2} \alpha \, X) X^T b = \tag{43}$$

$$= t^T X^T \alpha^T \Lambda^{-2} \alpha \, K \, b = \tag{44}$$

$$= t_X^T \; K^+ \, b \tag{45}$$

where, as usual, $t_X = X\,t$ denotes the vector of projections of the input vector along the sample and $K^+$ is the pseudoinverse of the matrix of inner products of the sample. When $t$ is projected over the sample, the role of $X^+$ in (41) is played by $K^+$.

Expression (45) is particularly elegant and appealing, since it immediately provides nonlinear versions of the discriminant function just by redefinition of $t_X$ and $K$ using any desired kernel. This algorithm will be denoted by KL (*K*ernel MSE *L*inear machine). Figure 5 shows classification boundaries in illustrative toy problems.

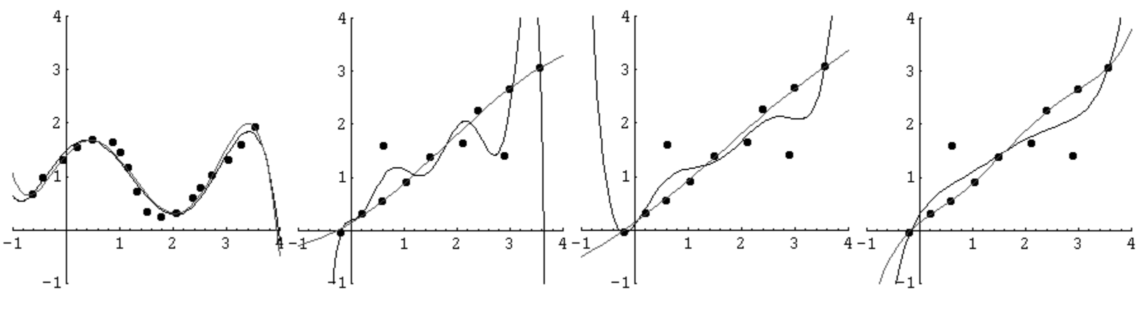


a) KL $\mathcal{P}_3$ b) SVM $\mathcal{P}_3$ c) KL $\mathcal{G}_1$ d) SVM $\mathcal{G}_1$

**Figure 5**. Classification boundaries of Kernel MSE linear machine (KL) in toy problems, compared with SVM ($C$=1000). Note similar artifacts (inductive bias) on both methods.

The KL machine has exactly the same structure (except sparsity) that the Support Vector Machine. Squared error minimization does not explicitly favor low VC-dimension boundaries, but KL solutions are still acceptable and frequently similar to those obtained by SVMs. Utilization of global statistical properties of the sample prevents overfitting and controls machine capacity as long as the kernel function has an adequate regularization effect. Learning is based on standard matrix diagonalization (instead of specific quadratic optimization required by SVM). Neither explicit misclassification cost nor tolerance margin in regression must be specified. Despite the nonlinear nature of KL solutions, the average error function (39) is a convex bowl with a single minimum, generating deterministic solutions which do not depend on random initialization of weights, frequent in descent/iterative procedures (e.g. backpropagation).

Furthermore, MSE approximation can be directly applied to regression problems, just by setting the desired vector $b$ in (38) and (45) to the observed values of the function (as in standard linear regression). The shape of the regression line is again very close to the one obtained by regression SVMs (see Fig. 6.a), although SVM is more resistant to outliers (Fig 6.b-d).



a) $\mathcal{P}_5$ b) $\mathcal{G}_3$ c) $\mathcal{G}_5$ d) $\mathcal{P}_5$

**Figure 6**. KL regression (black lines) compared with SVM (C=1000, ε=0.01) (gray lines). Note the robustness of SVM against outliers.

Note that overfitting in the MSE approach can be controlled by trading-off approximation error and flatness in the solution, as in *ridge* regression or weight decay techniques. To do this we minimize the combined cost function:

$$J(w, w_0) = \|Xw - w_0 - b\|^2 + C\|w\|^2 \qquad (46)$$

where the threshold $w_0$ is now made explicit since in general it should not be minimized[5] (it was previously generated automatically by the usual kernel expansions), and the regularization constant $C$ specifies the cost of "large" weights. Finding the optimum solution for this new cost function is straightforward and, analogously to (45), this kind of regularized solutions can be immediately extended to the kernel case. However, although $C$ values yielding quite satisfactory solutions can be found by cross-validation [18], it is not easy to develop a theoretically sound method for selecting the best degree of regularization. In contrast, SVM solutions are more robust against the optimization parameters and recently a theoretical method has been proposed [37] for setting them in terms of the noise level. In this context the interpretation of SVM operation as a regularization machine is also interesting [44] [16]. In the experimental section we will evaluate the basic MSE solution, without weight penalization, with the understanding that accuracy may improve by empirical choice of the regularization constant $C$.

Based on pseudoinversion, KL has also the numerical stability problem already mentioned in the context of KM. Fortunately, setting to zero the smallest eigenvalues has beneficial regularization effects, avoiding unjustified interpolation components. We have also observed that discarding the small eigenvalues makes the algorithm resistant to inadequate selection of kernel parameters (for instance, too small RBF scales are prone to overfitting). Again, except when explicitly mentioned, in the experimental section we will always perform pseudoinversion discarding the proportion $P = 10^{-5}$ of the whole spectral mass, corresponding to the smallest eigenvalues, although fine tuning of this proportion may sometimes improve classification accuracy.

KL corresponds to a generalized linear machine [29] where the input vector is expanded using the whole empirical kernel map. This structure is that of the method of Potential Functions, origin of the kernel method, introduced and studied by [3] [1] [2]. They used iterative learning rules (the Perceptron rule for separable problems and stochastic approximation for nonseparable ones) [12] [14] [52]. Instead, KL weights are found by direct (off-line) minimization of the average squared error (MSE criterion) using pseudoinversion. The structure of KL is also closely related to the Radial Basis Function (RBF) networks [7] [25] [30]. In this case learning is frequently based on unsupervised heuristics for specifying location and extent (centers and widths) of a reduced number of receptive fields [28]. KL is a more direct, brute-force approach, allowing any kind of kernel function.

Working implicitly in the high dimensional feature space defined by the kernel amounts to working explicitly in the space of kernel projections, with dimension equal to sample size[6]. In consequence, sparsity in the solutions becomes an essential issue, especially in large databases [34] [15]. There is an increasing interest in this problem, and several approaches such as the Reduced Set Method [8] and the approaches proposed in [31] could be relevant in our context.

An empirical approach to sparse representations for KL is suggested by its interpretation as standard linear MSE machine working on the kernel expansion: satisfactory solutions can be also obtained when the expansion is made only along a random sample of the training input vectors and the error is minimized over the whole training data. Algorithmically, this corresponds just to random elimination of a number of columns in the matrix $K$ in expression (45) and computation of the pseudoinverse of the resultant rectangular matrix (corresponding to an overconstrained system of equations). Note that this is not at all equivalent to working only with a fraction of the dataset. Partial expansion has even more regularization effects than KL with complete kernel expansion,

---

[5] For instance, in the linear regression case, this cost favors horizontal (low derivative) lines, but the offset is not penalized. When used with kernel expansions, the regularized nonlinear solutions are usually very similar whether or not the explicit threshold is taken into account.

[6] In some cases, kernel versions of standard algorithms can be obtained just by applying the ordinary algorithm on data preprocessed through the empirical kernel map. For instance, in the case of KL, expression (45) corresponds exactly to the ordinary MSE solution when the original data $X$ has been transformed into $K$, by means of the empirical kernel map. Also, Kernel PCA and standard PCA on the kernel expansion are closely related. In contrast, nonlinear SVMs cannot be designed in this way, since distances in the feature space do not depend on the statistical distribution of the samples, which affects the kernel expansion.

reducing overfitting. In fact, this strategy was proposed as an early method to locate centers in RBF networks [7]. In theoretical terms, expanding the optimal (high dimensional) weight vector $w^*$ as a linear combination of the selected subsample possibly deteriorates the approximation, but this is more than compensated by the regularization effect of solving an overconstrained system of equations. In contrast with ordinary MSE problems, where the pseudoinverse $X^+$ can be computed by ordinary inversion of the lower rank matrix $X^TX$, the design matrix of a partial kernel expansion may still be severely ill-conditioned, requiring again pseudoinversion with fine-tuning of the spectral mass assumed to be zero. This version of the algorithm will be denoted by KLR (*KL* with *R*educed expansion).

The experiments (Section 8) show that extraordinary savings in learning (and execution) time in certain large databases can be achieved without loss of accuracy (and often with significant improvements). Unfortunately, the optimal proportion of samples to be selected for the kernel expansion depends on the nature of databases. In any case, to illustrate the strength of this approach, in the experimental section we will evaluate KL both in the simple form (45) and with kernel expansion restricted to 10% of the samples, randomly chosen[7] (KLR-10%). Again, fine tuning of this proportion may improve the results in practical situations.

We conclude this section with some comments on kernel versions of alternative learning rules for linear discriminant functions. The weight vector $w$ for a linear discriminant function can be obtained by maximization of Fishers's separability criterion:

$$F(w) = \frac{|\mu_1 - \mu_2|}{\sigma_1^2 + \sigma_2^2} \tag{47}$$

where $\mu_1$, $\mu_2$, $\sigma_1$ and $\sigma_2$ denote respectively the mean values and standard deviations of the projections $w \cdot x$ of the elements in each class (i.e., the objects in $C$ and the objects not in $C$). We look for a direction in which the two projected subsamples become as much separated and concentrated as possible, as measured by their first and second order statistics. It can be shown [14] that the weight vector $w^*$ which maximizes this criterion is proportional to the most dominant eigenvector of $\Sigma^{-1}\Sigma_\mu$ , where $\Sigma_\mu$ denotes the covariance matrix of the mean values of all classes. In our simple two-class situation the desired direction is:

$$w^* \propto \Sigma^{-1}(\mu_2 - \mu_1) \tag{48}$$

A kernel version of Fisher's discriminant has been proposed in [27]. Using (15) we obtain a straightforward derivation for this machine:

$$w^* \cdot t = m\, b'\, K\, \beta^T\, \Omega^{-2}\, \beta\, t_X \tag{49}$$

where $b'$ is a modified vector of desired outputs in which values +1 and –1 are divided by the sample size of each class.

However, it is known that the Fisher optimum solution can be obtained from the MSE / pseudoinverse procedure (40) if the components $b_i$ in the vector of desired outputs in (38) are changed to the inverse of the proportion of samples in each class [12]. The computation of the MSE solution is easier and, in addition, it also provides the optimum threshold value, which Fisher's criterion itself is unable to specify. Yet another linear discriminant function that can be directly extended into a kernel version is the optimum classifier for gaussian classes with a common covariance matrix. It turns out that the discriminant direction in this case is defined again by Fisher's linear discriminant, with a threshold value which depends on the location of the mean values and on the *a priori* probabilities of each class.

Note that there are no problem-independent reasons to prefer either of the above inductive principles; nonlinear deformation of the metrics in the feature space may render their assumptions useless in the input space.

---

[7] Other heuristic strategies can be conceived, as for instance, elimination of nearly collinear samples, as measured by the inner products in matrix $K$. However, this takes some computing effort and our experiments showed no statistically significant improvement in accuracy with respect to random selection.

# 8 Experimental Comparison with Support Vector Machines

In general terms, matrix pseudoinversion (i.e., finding eigenvalues and eigenvectors) is simpler (and consequently faster) than the quadratic programming (QP) algorithms [4] required by Support Vector Machines in nontrivial problems [45]. Nevertheless, significant improvements to traditional off-the-self quadratic optimization algorithms have been developed recently, based on a completely different approach to that of classical pivoting methods (in the style of the Simplex method for lineal programming). The so-called *interior point algorithms* [20] came up as a response of the operations-research community to the worst-case exponential time of the pivoting methods for lineal and quadratic programming. This is a very active research field, where new variations of the method are frequently proposed, specifically adapted to optimize performance in different problem conditions (such as sparsity, possible non-feasibility, and so on).

Moreover, some of these methods have been specifically adapted to solve SVM learning problems, perhaps being SVM$^{light}$ [23] and SMO (Sequential Minimal Optimization) [33] the most remarkable. These implementations have proved to be very efficient on a number of large-scale databases, but both of them are based on successive decomposition of the original QP problem into smaller ones, which makes them highly data-dependent. In particular, in SMO the smaller optimization problems are taken to the extreme, analytically solvable case of only two variables, thus not requiring the use of an underlying numerical QP library. These decomposition techniques have shown their advantages in many databases in terms of both storage and computation time, but also have some drawbacks. More precisely, the run-time of the methods is very sensitive to overlapped input classes, originating many support vectors. In this kind of situations, the steepest feasible direction used to update the so called *working set* in SVM$^{light}$ –or the heuristic for choosing the two Lagrange multipliers to optimize in SMO– can result in an exponential growing of the decomposition iterations needed to find the correct solution. Especially in these problems, matrix pseudoinversion using spectral decomposition of matrices (underlying all of the methods proposed in this paper) can be a reasonable alternative. The run-time of efficient methods for finding eigenvalues and eigenvectors of a symmetric matrix is not data-dependent (except in low-rank or sparse matrices, in which they can be even faster than in the general case). There exist algorithms with worst-case complexity $O(m^3)$ [19], and efficient serial and parallel implementations have been developed [11][8].

In this section we first present an extensive comparison of the first and second order kernel methods described in this paper –kernel mean (KD) (equivalent to Parzen estimation), kernel Mahalanobis (KM), kernel minimum squared expanding input vectors with 100% of the database (KL) and only with 10% (KLR)– and support vector machines (SVM). The comparison will be made in terms of both computation time and prediction accuracy over a synthetic nonlinear classification task with controllable class overlapping. Then, in a subsequent subsection, we will compare these methods and SVM over some standard real world datasets from the UCI benchmark repository [5].

For simplicity, we used the RBF kernel $\mathcal{G}_\gamma$ in all the experiments, with different width values $\gamma$ as a way to control capacity. Generally, the wider $\gamma$, the lower the capacity and, hopefully, the better the results on test data, particularly when classes overlap (rote memorization of the training set is avoided). However, if the kernel width is too large the learning machine loses its required nonlinear character. In practice, the optimal kernel parameter, as well as the other free parameters, such as the $C$ penalty in SVMs or the percentage of spectral mass discarded in matrix pseudoinversion in KM, KL and KLR, are chosen by cross-validation, trying different combinations of values. Consequently, fast learning times are very convenient to speed up this first stage.

Rather than trying to outperform the accuracy of SVM, we will provide evidence that the proposed methods may constitute sometimes a reasonable and efficient alternative in nonlinear problems with large intrinsic error, where the optimal solution must misclassify a number of input samples to avoid overfitting. In these situations, samples near the separating hypersurface are not particularly informative about structure of the classes. Our results also illustrate the influence of the distribution of the input data in the performance of SVM learning QP algorithms, while KL or KM achieve similar error rates requiring fixed computation times that depend only on the size of the matrix $K$. As we will see, in many situations KLR also shows excellent accuracy with negligible learning effort and with the additional advantage of sparsity.

---

[8] In fact, polynomial time methods for convex quadratic programming are also available [54]. But they are based on iteration of $O(n^3)$ single steps to find successive approximations to the optimal solution, with the number of iterations depending also on a precision factor $\varepsilon$. This makes them slower than the direct methods available for spectral decomposition of matrices.

## 8.1 Synthetic Data

We have prepared a synthetic two-attribute nonlinear classification problem with controllable class overlapping. Figure 7 shows the optimum classification boundaries and the level curves of the class-conditional densities in four classification tasks with a common underlying structure (each class is a mixture of four equiprobable spherical gaussians), with increasing degrees of overlapping $\sigma$ = 0.1, 0.2, 0.4 and 0.8. We also indicate the Bayes error probabilities of each classification task. Figure 8 shows some examples of boundaries obtained by the different kernel algorithms, with different values of noise $\sigma$ and RBF *localization* parameter[9] $\gamma'=1/(2\gamma^2)$.

| $\sigma = 0.1$ | $\sigma = 0.2$ | $\sigma = 0.4$ | $\sigma = 0.8$ |
|---|---|---|---|
| (Bayes err. = 0.54%) | (Bayes err. = 8.51%) | (Bayes err. = 24.11%) | (Bayes err. = 34.34%) |

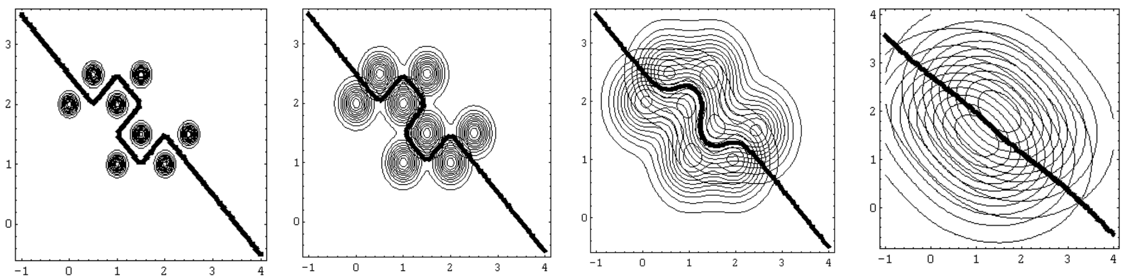
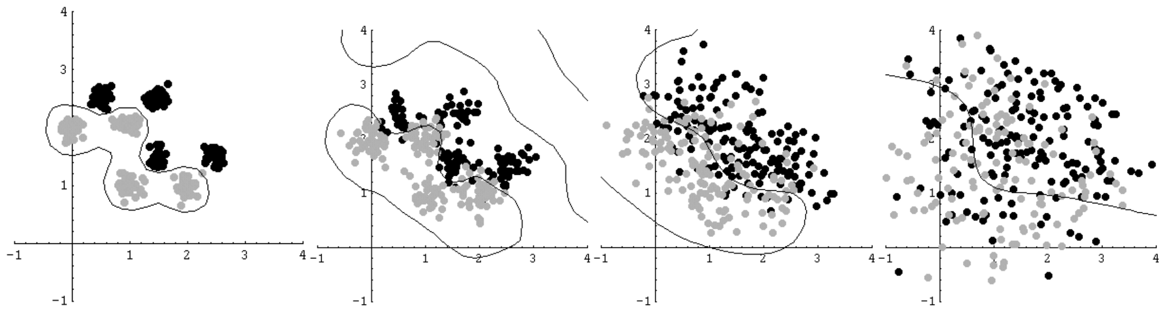
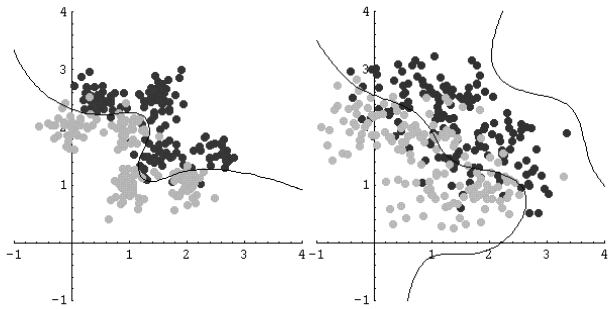


**Figure 7.** Synthetic two-attribute nonlinear classification problems, with different degrees of overlapping.



a) $\sigma$=0.1, KD b) $\sigma$=0.2, KM c) $\sigma$=0.4, KL d) $\sigma$=0.8, KLR-10%



e) $\sigma$=0.2, SVM f) $\sigma$=0.4, SVM

**Figure 8.** Illustration of the boundaries typically obtained by KD (Parzen), KM, KL, KLR-10% and SVM in the synthetic problems, with appropriate parameters. In all cases the size of the training set is 300.

In order to obtain statistically meaningful conclusions, we compared KD, KM, KL, KLR and SVM learning times and success rates in the above four classification problems with varying RBF kernel parameters $\gamma'$= 0.2, 1 and 5, and sample sizes $m$ = 100, 300, 600 and 1200. In all the methods involving pseudoinversion of matrices (KM, KL, and KLR), a proportion $P=10^{-5}$ of the spectral mass was discarded. That is, only the largest eigenval-

---

[9] Throughout this section, for computational convenience, we will use $k(\vec{x}_1,\vec{x}_2)=e^{-\gamma'\|\vec{x}_1-\vec{x}_2\|^2}=G_{\frac{1}{\sqrt{2\gamma'}}}(\vec{x}_1,\vec{x}_2)$. Consequently, more regularized machines are obtained with lower $\gamma'$, corresponding to wider radial basis functions.

ues that added up to 99.999% of the total trace of *K* were considered. In the same way, the *C* parameter for SVMs was fixed to 1000. Ten repetitions of the learning stage with independent training sets were executed in each configuration. The error probabilities were estimated using independent test sets of size 3000. Figure 9 summarizes the results[10]. The graphs show average success rate and average execution time of each learning configuration. SVM learning time is represented with a confidence interval (±2 standard deviations) to show its variability even for experiments with identical σ and γ' values. The rest of magnitudes have insignificant variability.

The experiments show that the approximation to the optimal classifier deteriorates as the degree of overlapping σ increases, but the quality of the solutions obtained by all the methods remains comparable, with a very slight advantage of SVMs (particularly for small sample sizes, due to better capacity control). There are also clearly bad results (see, for example, KD (Parzen) with σ=0.1 and γ'=0.2, KM with σ=0.4 and γ'=5.0, or KLR-10% with σ=0.1 and γ'=0.2). The main reason is that a bad choice of the γ' parameter can affect seriously the performance, and that a fixed proportion *P* of the spectral mass is not always adequate for pseudoinversion. This suggests the necessity of fine-tuning of these parameters in real problems. Anyway, the choice of C and γ' affects SVM too (see, for example, σ=0.8 and γ'=5.0), so some form of cross-validation is needed in all the methods to adjust parameters.

Regarding computational efficiency, the experiments clearly confirm that the learning time of SVM, based on constrained quadratic optimization, eventually grows faster, especially for high degrees of overlapping. In contrast, KD and KLR-10% have negligible learning times, while KM and KL grow approximately 10 and 3 times slower, respectively. Compare also the variability of SVM run-time (due to the high data dependence of the $SVM^{light}$ underlying QP algorithm) with the run-time predictability of matrix pseudoinversion. SVM learning is frequently slower for wider RBFs (with a higher regularization effect) required to achieve generalization, trying to avoid rote learning. This has the important consequence that fine tuning of parameters using cross-validation may suppose a heavy computational load. The run-time of methods based in pseudoinversion is more predictable, as it depends only on sample size, and not on the particular distribution of the data or the selected parameters.

Finally, observe that for large sample sizes, KLR-10% obtains similar success rates to those of KL, close to the Bayes optimum, requiring negligible learning effort. In some cases, the reduction of the number of vectors for the kernel expansion has also beneficial regularization effects, even outperforming KL with full 100% expansion.

---

[10] The implementations of the learning algorithms used in the tests were $SVM^{light}$ (Joachims 1998) in the case of SVM's, and a set of simple programs for KD, KM, KL and KLR-10% written by the authors using the standard linear algebra libraries BLAS and LAPACK (Dongarra and Walker 1995). In the case of $SVM^{light}$, we disallowed the heuristic of considering an active set smaller than the whole database. Although this technique was faster in databases with low noise levels, its performance was unacceptably poor in databases with higher degrees of overlapping (growing exponentially). In consequence, the optimization problem was solved entirely by one execution the underlying QP algorithm (LOQO [53]), taking into account the whole database. All the implementations were written in C and run in a Pentium III 450 MHz under the Linux O.S.

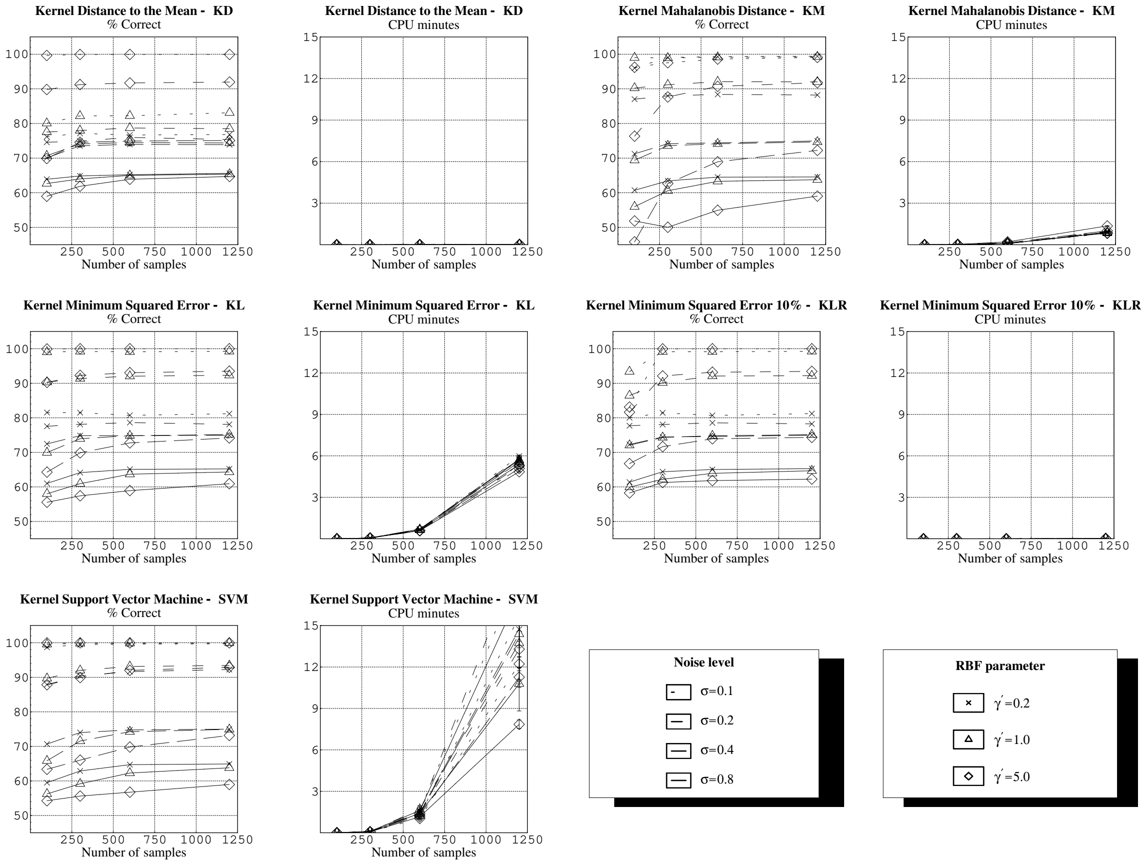



**Figure 9.** Comparison of test errors and execution times obtained by KD, KM, KL, KLR-10% and SVM in the synthetic problem. Line style indicates problem complexity (degree of overlapping): denser lines correspond to higher intrinsic error. Kernel width (amount of regularization) is indicated by symbols: diamond: $\gamma'$=0.2, triangle: $\gamma'$=1, and cross: $\gamma'$=5 (remember that the lower the $\gamma'$, the wider the RBF, see footnote 8). Observe how, as we increase the size of the database, the success rates of all the methods get close to the Bayes optimum (see Figure 7).

## 8.2 Natural Data

We have also tested the methods in some datasets obtained from the UCI benchmark repository [5]. We considered three real world problems: IONOSPHERE (classification of radar returns from the ionosphere), PIMA (classification of diabetes data in Pima Indians) and ABALONE (estimation of the age of a mollusk from weight, size and other measurements), as well as the artificial MONK's problems (widely used as benchmarks for inductive learning algorithms [47]). All the databases are used for binary classification[11].

The problems have different difficulty degrees. MONK1, MONK3 and IONOSPHERE are relatively easy, with a low degree of class overlapping in the input space. MONK2 is slightly harder, in the sense that the classification boundary depends on an abstract combination of features without easy geometrical interpretation. Overlapping is also apparent in ABALONE (samples with target values around the threshold are difficult to classify). Finally, PIMA is a very noisy dataset, in which extreme overlapping makes difficult to improve the success rate of optimal random decision (always choosing the majority class).

Every database was partitioned into three disjoint subsets, A (40% of the total size), B (40% also) and C (20% remainder). Given these partitions, the following procedure was applied to adjust the free parameters of the learning algorithms ($\gamma'$ for all methods, $C$ penalty in SVM, and the proportion $P$ of spectral mass discarded for

[11] ABALONE was originally designed to estimate a numeric value, so we transformed the problem into binary classification by setting a threshold in the target attribute that approximately divided the database into two classes of equal size. All the attributes in datasets IONOSPHERE, PIMA and ABALONE were first normalized to have zero mean and unit deviation. The MONK's datasets were used directly, without any preprocessing.

pseudoinversion, in KM, KL and KLR): The A subset of each database was always used to train the algorithms with all combinations of γ' in {0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1, 2.5, 5}, and $C$ in {$10^2$, $10^3$ ,$10^4$, $10^5$, $10^6$, $10^7$} for the SVM or $P$ in {$10^{-1}$, $10^{-2}$, $10^{-3}$ ,$10^{-4}$, $10^{-5}$, $10^{-6}$} for the methods needing matrix pseudoinversion. Then the value γ' for KD and the pair (γ',$C$) for SVM or (γ',$P$) for KM, KL and KLR-10% that achieved the lowest error rate when tested on the C partition were chosen as the optimal parameter set. The resulting machines were tested then on the B partition to obtain an objective measure of their quality in the binary classification problem.

Table 1 summarizes the results of our experiments, in terms of error rates and learning times. For each database, the number of samples and dimension of each vector is shown, as well as the sizes of the respective A, B and C subsets, and the proportion of the minority class (the error rate of optimal random classification, to be improved by the learning algorithms). Then, in each cell we show the optimal parameters and error rate for each method (obtained as explained above). The learning time (in CPU minutes) for each training algorithm execution is also shown. The computing environment is the same discussed in section 8.1 (see footnote 10).

| **Database (n. of vectors × dimension)**<br>(Master/cross valid./test sizes)<br>(Random rate) | **KD (Parzen)**<br>(Error rate %)<br>(CPU minutes)<br>(Opt. γ) | **KM (Mahalanobis)**<br>(Error rate %)<br>(CPU minutes)<br>(Opt. (γ,P) pair) | **KL (MSE)**<br>(Error rate %)<br>(CPU minutes)<br>(Opt. (γ,P) pair) | **KLR-10% (MSE Part. exp.)**<br>(Error rate %)<br>(CPU minutes)<br>(Opt. (γ,P) pair) | **SVM**<br>(Error rate %)<br>(CPU minutes)<br>(Opt. (γ,C) pair) |
|---|---|---|---|---|---|
| **IONO (351 × 33)**<br>(140/140/71) (37.86%) | (13.57%) (0.00)<br>(0.05) | (7.14%) (0.01)<br>(0.25, $10^{-4}$) | (9.29%) (0.00)<br>(0.05, $10^{-4}$) | (17.86%) (0.00)<br>(0.005, $10^{-6}$) | (5.71%) (0.01)<br>(0.25, $10^3$) |
| **ABALONE (4177 × 8)**<br>(1670/1670/837) (49.46%) | (22.40%) (0.05)<br>(0.025) | (23.05%) (2.34)<br>(0.005, $10^{-3}$) | (20.06%) (12.85)<br>(0.025, $10^{-6}$) | (19.94%) (0.05)<br>(0.05, $10^{-5}$) | (20.96%) (29.03)<br>(0.005, $10^3$) |
| **PIMA (768 × 8)**<br>(307/307/154) (36.81%) | (26.38%) (0.00)<br>(0.005) | (24.76%) (0.02)<br>(0.025, $10^{-2}$) | (22.80%) (0.06)<br>(0.01, $10^{-6}$) | (23.78%) (0.00)<br>(0.01, $10^{-6}$) | (21.5%) (0.06)<br>(0.005, $10^2$) |
| **MONK 1 (556 × 6)**<br>(222/222/112) (40.54%) | (14.41%) (0.00)<br>(2.5) | (9.81%) (0.01)<br>(1.0, $10^{-2}$) | (11.26%) (0.02)<br>(1.0, $10^{-6}$) | (30.63%) (0.00)<br>(0.1, $10^{-2}$) | (4.05%) (0.03)<br>(0.1, $10^3$) |
| **MONK 2 (601 × 6)**<br>(240/240/121) (35.42%) | (18.75%) (0.00)<br>(2.5) | (6.77%) (0.01)<br>(0.025, $10^{-3}$) | (27.92%) (0.03)<br>(1.0, $10^{-4}$) | (35.42%) (0.00)<br>(0.1, $10^{-2}$) | (9.58%) (0.05)<br>(0.1, $10^6$) |
| **MONK 3 (554 × 6)**<br>(221/221/112) (47.05%) | (13.57%) (0.00)<br>(1.0) | (3.62%) (0.01)<br>(0.005, $10^{-5}$) | (3.62%) (0.02)<br>(0.5, $10^{-3}$) | (5.43%) (0.00)<br>(0.1, $10^{-6}$) | (3.62%) (0.09)<br>(0.005, $10^3$) |

**Table 1.** Error rates and training times of KD, KM, KL, KLR-10% and SVM on standard benchmarks from the UCI repository. For each database and method, we show the size of the training set, the dimension of the examples and the RBF width used, the main variables influencing the learning time.

The results clearly confirm the remarkable generalization power of SVM, demonstrated in learning situations with small class overlapping (MONK1, IONOSPHERE, MONK3) as well as in harder problems (PIMA). However, the rest of the methods can obtain very acceptable test error rates, comparable to SVM in the same problems (see, for example, KL and KM in MONK3, KM in IONO, or KL and KLR-10% in the difficult problem PIMA). Some methods even outperform SVM in certain cases, especially in those with high class overlapping (KL and KLR-10% in ABALONE, or KM in MONK2). In these cases, the regularization effect of average error minimization in KL methods, or the ability to capture the geometry of the classes in KM (combined with elimination of small eigenvalues in pseudoinversion) seem to be more appropriate than finding large margin classifiers with a certain penalty $C$ for misclassifications.

Again, learning times of SVMs are always longer, due to the greater complexity of the underlying QP solver. Nevertheless, note that they are faster in the classification stage, due to sparsity in the solution. This is a great advantage of SVMs in situations where classification time is critical. Of all the methods discussed in this paper, only KLR is sparse in the classification machine generated; in the rest of methods a kernel expansion of the input sample must be computed with each sample of the database, which can be slow. In KM, two additional matrix-vector multiplication operations must be computed, making the classification even slower. But this last method has the additional advantage that it operates in half sized matrices (one for each class), so learning times get divided approximately by four, with respect to those of KL[12].

Finally, note that KLR-10% often obtains similar error rates requiring extremely short learning and classifying times, especially for large databases (PIMA, ABALONE). Here, 90% reduction for the input vector expansion can even have beneficial regularization effects. In very small databases (e.g. MONK's problems), the loss of performance of KLR may be sometimes unacceptable, but in these small problems KL itself is very fast, so reduction of the input vector expansion is not advised. KD (equivalent to Parzen estimation) has virtually null learning times (only the constant $\mathbf{1}K\mathbf{1}$ has to be computed for non-density kernels). But its error rates seem to be worse and the method also lacks sparsity. Anyway, it is interesting to notice how a simple mean-distance classifier in feature space often obtains very acceptable results.

# 9 Conclusion

Inspired by [39], we have presented a convenient toolkit for immediate derivation of kernel versions of classical statistical pattern analysis algorithms. The main contributions developed under this framework can be summarized as follows:

1) We have shown how this setting provides simple derivations for other previously proposed kernel variants of standard methods such as PCA and Fisher's discriminant.

2) We have illustrated some connections between first order kernel methods and nonparametric density estimation: Parzen's approximation is equivalent to distance to the mean in the feature space induced by the smoothing function.

3) We have introduced the kernel version of Mahalanobis distance, a new kind of nonparametric modeling technique with interesting properties caused by variance normalization in feature space. KM automatically adapts the local effective degree of smoothing using a fixed-width gaussian kernel. It exhibits an "overshooting" effect which emphasizes both presence and absence of samples in input space. Unlike Parzen estimation, correct nonparametric models can be also built using arbitrary (e.g. polynomial) interpolating functions.

4) Finally, we have proposed a particularly simple kernel version of the Minimum Squared Error linear machine (KL), which includes a number of generalized linear models, and shows very satisfactory results in experiments with synthetic and natural data. In large databases, a sparse version of KL based on partial kernel expansion (KLR) has proved to be highly accurate and extremely efficient.

The SVM, based on Vapnik's remarkable Learning Theory, combines low VC-dimension solutions (through maximization of margin) and kernel-based nonlinearity. The machines proposed in this paper are especially useful to ascertain the relative responsibility of kernel feature spaces and inductive bias in the extraordinary generalization ability of SVMs. Our experiments indicate that simple algorithms take advantage of kernel-based nonlinearity to obtain very acceptable, although typically suboptimal, solutions. Best generalization in nonlinear problems with low intrinsic error can only be obtained by the superior inductive bias of the SVM.

The learning algorithms proposed in this paper are essentially based on pseudoinversion, computed through eigendecomposition [19], with computational complexity $O(m^3)$. This is significantly more efficient than off-the-self quadratic programming algorithms [4]. In problems with high class overlapping requiring a large number of support vectors it is even faster than algorithms especially designed for SVM optimization [23][13]. Such hard optimization process is unavoidable for the powerful inductive bias of SVM, which, at the same time, finds sparse solutions with low VC dimension.

Our experiments confirm that simple kernel machines such as KM and KL are competitive with SVMs in problems with appreciable statistical overlap, where location and covariance structure of the populations carry more information about optimal estimates than the situation of somewhat atypical vectors supporting a large margin classifier.

---

[12] Supposing 50% of samples in each class, as pseudoinversion is $O(m^3)$, inverting two matrices $m/2 \times m/2$ will be four times faster.

[13] Differences in learning time between KL and SVM will be still larger in regression tasks, since SVM require two Lagrange multipliers for each data point, while KL regression and classification only differ in the vector of desired outputs.

We believe that conceptual simplicity and learning efficiency make the approach defended in this paper a useful alternative in certain situations. Its limitations only emphasize the outstanding importance of Vapnik's learning theory.

## Appendix

We present here a complete derivation of the relationship between $XX^T$ and $X^TX$ based on [14](p.39) and [39]. We will focus on the more complex case of centered data (the analogous decomposition of the autocorrelation matrix $S$ can be derived in a similar fashion). Our goal is to prove (12) and (13). Consider the decomposition of the covariance matrix given by (3) and (7):

$$C = \frac{1}{m} X^T X - X \vec{1}\,\vec{1}^T X^T = V^T E\, V \tag{50}$$

Since the eigenvectors necessarily lie in the span of the centered data, they can be written as the following linear combination:

$$V = \gamma\,(X\text{-}\ \mathbf{1}X) \tag{51}$$

We must find the coefficient matrix $\gamma$ in terms of the matrix $K$-$\mathbf{1}K$-$K\mathbf{1}$-$\mathbf{1}K\mathbf{1}$ (built with the inner products of the samples $K$=$XX^T$). Multiplying (50) by $X$-$\mathbf{1}X$ from the left side, and by $V^T$ =($X$- $\mathbf{1}X$)$^T$ $\gamma^T$ from the right side, we obtain (after substituting $XX^T$ by $K$):

$$\begin{gathered} \frac{1}{m}\left(K^2 - K\mathbf{1}K - \mathbf{1}K^2 + \mathbf{1}K\mathbf{1}K - K^2\mathbf{1} + K\mathbf{1}K\mathbf{1} + \mathbf{1}K^2\mathbf{1} - \mathbf{1}K\mathbf{1}K\mathbf{1}\right)\gamma^T = \\ \left(K - \mathbf{1}K - K\mathbf{1} + \mathbf{1}K\mathbf{1}\right)\gamma^T E \end{gathered} \tag{52}$$

Now, using the fact that

$$K^2 - K\mathbf{1}K - \mathbf{1}K^2 + \mathbf{1}K\mathbf{1}K - K^2\mathbf{1} + K\mathbf{1}K\mathbf{1} + \mathbf{1}K^2\mathbf{1} - \mathbf{1}K\mathbf{1}K\mathbf{1} = \left(K - \mathbf{1}K - K\mathbf{1} + \mathbf{1}K\mathbf{1}\right)^2 \tag{53}$$

and multiplying (52) by the pseudoinverse ($K$-$\mathbf{1}K$-$K\mathbf{1}$+$\mathbf{1}K\mathbf{1}$)$^+$ from the left side, we get the following condition for $\gamma$:

$$\frac{1}{m}\left(K - \mathbf{1}K - K\mathbf{1} + \mathbf{1}K\mathbf{1}\right)\gamma^T = \gamma^T E \tag{54}$$

Since $E$ is diagonal, the elements in $\gamma$ are proportional to the eigenvectors of $K$-$\mathbf{1}K$-$K\mathbf{1}$-$\mathbf{1}K$. For the same reason the eigenvalues of $K$-$\mathbf{1}K$-$K\mathbf{1}$-$\mathbf{1}K\mathbf{1}$ are $\Omega$=$mE$, thus justifying (12). Using the notation in (9) we can write:

$$\gamma = D\,\beta \tag{55}$$

where $D$ is some diagonal matrix. We obtain $D$ from the orthogonality of the eigenvectors of $C$:

$$I = VV^T = \gamma\,(X\text{-}\mathbf{1}X)(X\text{-}\mathbf{1}X)^T\,\gamma^T = D\,\beta\,(K\text{-}\mathbf{1}K\text{-}K\mathbf{1}\text{-}\mathbf{1}K\mathbf{1})\,\beta^T D = D\,\beta\,\beta^T\,\Omega\,\beta\,\beta^T D = D\,\Omega\,D \tag{56}$$

therefore:

$$D = \Omega^{-1/2} \tag{57}$$

Now, from (51), using (55) and (57), we have

$$V = \Omega^{-1/2}\beta(X\text{-}\ \mathbf{1}X) \tag{58}$$

But the term $\Omega^{-1/2}\beta\mathbf{1}X$ cancels out, since we can show that $\beta\mathbf{1}$=0:

$$\begin{gathered} K_c = \beta^T\Omega\beta \Rightarrow \beta K_c = \Omega\beta \Rightarrow \Omega^{-1}\beta K_c\mathbf{1} = \beta\mathbf{1} \Rightarrow \Omega^{-1}\beta\left(K\mathbf{1}\text{-}\mathbf{1}K\mathbf{1}\text{-}K\mathbf{1}\mathbf{1}\text{-}\mathbf{1}K\mathbf{1}\mathbf{1}\right) = \beta\mathbf{1} \Rightarrow \\ \Rightarrow \Omega^{-1}\beta\left(K\mathbf{1}\text{-}\mathbf{1}K\mathbf{1}\text{-}K\mathbf{1}\text{-}\mathbf{1}K\mathbf{1}\right) = \beta\mathbf{1} \Rightarrow \Omega^{-1}\beta 0 = \beta\mathbf{1} \Rightarrow 0 = \beta\mathbf{1} \end{gathered} \tag{59}$$

Using (59), the equality (58) is equivalent to (13), thus completing the proof. Finally, to prove the general expression (17) it suffices to show that

$$(X^T X)^+ = X^T \left(XX^T\right)^+ \left(XX^T\right)^+ X \tag{60}$$

since (17) immediately follows from (60) when $X$ is replaced by $A^{1/2}X$, for any symmetric matrix $A^{1/2}$. Using the notation in (8), (10) and (11), we have

$$\begin{aligned} X^T \left(XX^T\right)^+ \left(XX^T\right)^+ X &= \\ X^T \left(\alpha^T \Lambda \alpha\right)^+ \left(\alpha^T \Lambda \alpha\right)^+ X &= \\ X^T \left(\alpha^T \Lambda^{-1} \alpha\right)\left(\alpha^T \Lambda^{-1} \alpha\right) X &= \\ X^T \alpha^T \Lambda^{-1/2} \Lambda^{-1} \Lambda^{-1/2} \alpha X &= \\ U^T \Lambda^{-1} U = (mS)^+ &= \left(X^T X\right)^+ \end{aligned} \tag{61}$$

Which justifies (60).

## Acknowledgements

We would like to thank the associated editor and the referees for their careful reading and useful suggestions. This work has been supported by the Spanish CICYT grant TIC98-0559.

## References

[1] M.A. Aizerman, E.M. Braverman, and L.I. Rozonoer, “Theoretical foundations of the potential function method in pattern recognition learning”. *Automation and Remote Control* **25**, pp.821-837, 1964.

[2] M.A. Aizerman, E.M. Braverman, and L.I. Rozonoer, “The Robbins-Monro process and the method of potential functions”. *Automation and Remote Control* **26**, pp.1882-1885, 1965.

[3] O.A. Bashkirov, E.M. Braverman, and I.B. Muchnik, "Potential function algorithms for pattern recognition learning machines", *Automation and Remote Control*, **25**, 629-631, 1964.

[4] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty, *Nonlinear Programming (Theory and Algorithms)*. John Wiley & Sons, 1993.

[5] C. Blake, E. Keogh, and C.J. Merz, UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[6] B.E. Boser, I.M. Guyon, and V.N. Vapnik, “A training algorithm for optimal margin classifiers”, In D. Haussler, (ed.), *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144-152, Pittsburgh, PA, ACM Press, 1992.

[7] D.S. Broomhead and D. Lowe, "Multivariate functional interpolation and adaptive networks", *Complex Systems* **2**, 321-355.

[8] C. Burges, “Simplified Support Vector Decision Rules”. In Saitta, L. (ed.). *Proc. 13th Int. Conf. on Machine Learning,* pp.71-77, Morgan Kaufmann, 1996.

[9] C. Burges, “A Tutorial on Support Vector Machines for Pattern Recognition”. *Data Mining and Knowledge Discovery*, Vol 2 Number 2, 1998.

[10] C. Cortes, and V. N. Vapnik, “Support Vector Networks”. *Machine Learning*, **20**, 273-297, 1995.

[11] J. Dongarra, and D. Walker, “Software Libraries for Linear Algebra Computations on High Performance Computers”. SIAM Review, **37**, pp. 151-180, 1995.

[12] R.O. Duda, and P.E. Hart, *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[13] T.T. Frieβ, N. Cristianini, and C. Campbell, “The Kernel-Adatron Algorithm: a Fast and Simple Learning Procedure for Support Vector Machines”.*15th Intl. Conf. Machine Learning*, Morgan Kaufman Publishers, 1998.

[14] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[15] F. Girosi, “An Equivalence Between Sparse Approximation and Support Vector Machines”. *Neural Computation* **10**, 1455-1480, 1998.

[16] F. Girosi, M. Jones, and T. Poggio, “Regularization Theory and Neural Networks Architectures”. *Neural Computation* Vol.**7**, No.2, pp.219-269, 1995.

[17] T. Graepel and K. Obermayer, "Fuzzy topographic kernel clustering". Proceedings of the 5th GI Workshop Fuzzy Neurosystems '98. (W. Brauer, ed.), pp.90-97, 1998.

[18] G.H. Golub, M. Heath and G. Wahba, "Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter", *Technometrics*, Vol. 21, No. 2, pp215-223, 1979.

[19] G.H. Golub and C.F. Van Loan, *Matrix Computations*. The John Hopkins Univ. Press, 1984.

[20] D. den Hertog. “Interior Point Approach to Linear, Quadratic and Convex Programming”, Kluwer Academic Publishers, 1997.

[21] M.M.Van Hulle, “Kernel-Based Equiprobabilistic Topographic Map Formation”. *Neural Computation* **10**, 1847-1871, 1998.

[22] A.J. Izenman, "Recent Developments in Nonparametric Density Estimation", *Jour. Amer. Stat. Assoc.*, Vol. 86, No. 413, 205-224, 1991.

[23] T. Joachims, “Making Large-Scale Support Vector Machine Learning Practical”. Included in [38].

[24] G. Kimeldorf and G. Wahba, “Some results on Tchebycheffian spline functions”. *J. Math. Anal. Applic.* 33, pp. 82-95, 1971.

[25] D. Lowe, "Radial Basis Function Networks" in M.A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, 1995.

[26] J.S. Marron, "A comparison of cross-validation techniques in density estimation", *The Annals of Statistics*, Vol. 15, No. 1, 152-162, 1987.

[27] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, “Fisher Discriminant Analysis with Kernels”. *Proc. IEEE Neural Networks for Signal Processing Workshop, NNSP 99, 1999.*

[28] J.Moody and C. Darken, "Fast learning in networks of locally tuned processing units", *Neural Computation* **1**, 281-294, 1989.

[29] N.J. Nilsson, [*The Mathematical Foundations of*] *Learning Machines*, Morgan Kauffman, 1965, 1990.

[30] M.J.L. Orr, "Introduction to Radial Basis Functions Networks", *Tech. Rep. Centre for Cognitive Science, Univ. Edinburgh*, 1996.

[31] E.E. Osuna and F. Girosi, F. “Reducing the Run-time Complexity in Support Vector Machines”. Included in [38].

[32] E. Parzen, "On estimation of a probability density function and mode". *Ann. Math. Stat.*, **33**, 1065-1076, 1962.

[33] J.C. Platt, “Fast Training of SVMs Using Sequential Minimal Optimization”. Included in [38].

[34] T. Poggio and F. Girosi, “A Sparse Representation for Function Approximation”. *Neural Computation* **10**, 1445-1454, 1998.

[35] M. Pontil and A. Verri, "Properties of Support Vector Machines". *Neural Computation* **10**, 995-974, 1998.

[36] M. Pontil and A. Verri, "Support Vector Machines for 3D Object Recognition". *IEEE TPAMI* Vol. 20, No.6, pp.637-646, 1998.

[37] B. Schölkopf, P. Bartlett, A. Smola, and R. Williamson, “Shrinking the Tube: A New Support Vector Regression Algorithm”. To appear in: M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Adv. in Neural Inf. Processing Systems* 11, MIT Press, Cambridge, MA, 1998.

[38] B. Schölkopf, C. Burges, and A. Smola, (eds.), *Advances in Kernel Methods, Support Vector Learning*. MIT Press, 1998.

[39] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem”. *Neural Computation* **10**, 1299-1319, 1998.

[40] B. Schölkopf, A.J. Smola, R. Williamson, and P. Bartlett, “New Support Vector Algorithms”. *Neuro COLT Technical Report* TR-1998-031, Royal Holloway College, 1998.

[41] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. Smola, “Input Space vs. Feature Space in Kernel-Based Methods”. *IEEE Transactions on Neural Networks*, Vol. 10, No. 5, pp. 1000-1017, 1999.

[42] A.J. Smola, and B. Schölkopf, “A Tutorial on Support Vector Regression”. *Neuro COLT Technical Report* TR-1998-030, Royal Holloway College, 1998.

[43] A. Smola and B. Schölkopf (eds.), URL `http://www.kernel-machines.org`, 2000.

[44] A.J. Smola, B. Schölkopf, and K.-R. Müller, “The connection between Regularization Operators and Support Vector Kernels”. *Neural Networks* **11**, pp. 637-649, 1998.

[45] A.J. Smola and B. Schölkopf, “On a kernel-based Method for Pattern Recognition, Regression, Approximation, and Operator Inversion”. *Algorithmica* **22**, pp. 211-231, 1998.

[46] A.J. Smola, T.T. Frieβ, and B. Schölkopf, “Semiparametric Support Vector and Linear Programming Machines”. Neuro COLT Technical Report TR-1998-024, Royal Holloway College. (to appear in NIPS'98), 1998.

[47] S. Thrun et al., "The MONK's Problems. A performance Comparison of Different Learning Algorithms". *Technical Report* CMU-CS-91-197, 1991.

[48] E. Trucco and A. Verri, *Introductory Techniques for 3D Computer vision*. Prentice-Hall, 1999.

[49] V. Vapnik, *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, 1982.

[50] V. Vapnik, S.E. Golowich, and A. Smola, “Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing.”. In M. Mozer, M. Jordan and T. Petsche, (eds.) *Advances in Neural Information Processing Systems* 9, pp.281-287, MIT Press, 1997.

[51] V.N. Vapnik, *The Nature of Statistical Learning Theory*. Springer, 1995.

[52] V.N. Vapnik, *Statistical Learning Theory (Adaptive and Learning Systems for Signal Processing, Communications and Control).* John Wiley & Sons. 1998.

[53] R.J. Vanderbei, “LOQO: an interior point code for quadratic programming”, Tech. rep. Program in Statistics and Operations Research, Princeton University, 1994.

[54] S.A. Vavasis, “Nonlinear Optimization: Complexity Issues”, Oxford Science, New York, 1991.

[55] G. Wahba, “Support Vector Machines, Reproducing Kernel Hilbert Spaces, and Randomized GAVC”. Included in [38].