Sparse Kernel Feature Analysis

Alex J. Smola,* Olvi L. Mangasarian,† and Bernhard Schölkopf‡

October 3, 1999

Abstract

Kernel Principal Component Analysis (KPCA) has proven to be a versatile tool for unsupervised learning, however at a high computational cost due to the dense expansions in terms of kernel functions. We overcome this problem by proposing a new class of feature extractors employing ℓ_1 norms in coefficient space instead of the reproducing kernel Hilbert space in which KPCA was originally formulated in. Moreover, the modified setting allows us to efficiently extract features maximizing criteria other than the variance much in a projection pursuit fashion.

1 Introduction

The problems in unsupervised learning are by far less precisely defined than in the supervised counterpart. Usually no explicit cost function exists with desired outputs or anything similar. Instead, one has to make certain assumptions on the data, with respect to which several questions may be asked.

A possible problem is: "Which properties of the data can be extracted with high confidence?" Or in other words, which *simple* feature extracting functions can be found among a given class with, say, unit variance and zero mean. This leads to a *feature extracting* approach of unsupervised learning such as Principal Component Analysis (PCA) [Jolliffe, 1986] for linear feature extractors or Kernel-PCA [Schölkopf et al., 1998] in the case of nonlinear functions.

Computational cost is not a major issue in PCA, since the matrix whose eigensystem has to be computed is at most of the size of the dimensionality of the data. Feature extraction, finally, is done by computing a simple dot product.

*Australian National University, Department of Engineering and RSISE, Canberra, 0200 ACT, Australia; Alex.Smola@anu.edu.au

†Computer Sciences Department, University of Wisconsin 53705, Madison, USA; olvi@cs.wisc.edu

‡Microsoft Research, St. George House, 1 Guildhall Street, Cambridge CB2 3NH, UK; bsc@microsoft.com

Things change dramatically in Kernel-PCA. Since the directions of projection can only be described in terms of a (typically completely dense) expansion of patterns mapped into some feature space, generation of the Kernel PCA basis involves finding the eigensystem of an $m \times m$ matrix, where m denotes the number of samples. The same problems arise for feature extractions, which is approximately m times as expensive as for PCA.

Several approaches tried to alleviate this problem, either by computation of a compact representation [Schölkopf et al., 1999, Burges and Schölkopf, 1997] or by the heuristic of selecting a subset of basis functions at random. Whilst leading to computationally more favourable expansions in the end, the process of obtaining the latter is still rather time consuming.

This leads to the question whether not a modified setting could provide comparable performance in extracting nonlinear features from data, whilst preserving sparsity, i.e. whilst having a compact functional representation. The latter is often achieved in supervised settings by using an ℓ_1 penalty on the expansion coefficients [Mangasarian, 1965, Chen et al., 1999, Saunders et al., 1998, Bennett, 1999, Smola, 1998, Bradley and Mangasarian, 1998b, Mangasarian, 1998]. Hence it appears promising to use the same approach for feature extraction, too. We derive an explicit algorithm for such a setting that was proposed by Smola [1998] which exhibits a high degree of sparsity (only k basis functions are required to compute the first k features), is simple to compute, and scales approximately one order of magnitude better on large datasets than standard Kernel-PCA.

The paper is organized as follows. In section 2 we state the problems for Kernel-PCA and more generally Kernel Feature Analysis. In Section 3 we devise an algorithm for sparse feature extraction. A theoretical analysis follows in Section 4 and the paper concludes with an experimental demonstration of the flexibility of the new approach.

2 Kernel Feature Analysis

Denote by $X := \{x_1, \ldots x_m\} \subset \mathcal{X}$ a set of patterns drawn independently identically distributed from an underlying probability distribution $p(x)$ and $\mathcal{X}$ the corresponding embedding space. Our goal is to compute feature extractors $f_i(x)$ that satisfy certain criteria of simplicity (e.g. small RKHS norm [Wahba, 1990, Smola and Schölkopf, 1998] or ℓ_1 norm [Mangasarian, 1965, Chen et al., 1999, Bennett, 1999, Smola, 1998, Bradley and Mangasarian, 1998a, Mangasarian, 1998]) and optimality (e.g. maximum variance [Hotelling, 1933, Karhunen, 1946]).

2.1 Principal Component Analysis (PCA)

Let us start with a (slightly nonstandard) formulation of Principal Component Analysis (PCA) [Schölkopf et al., 1999]. The first principal component of a sample is given by the direction of projection with maximum variance. More formally (see e.g. [Jolliffe, 1986]), for centered data

$$\tilde{X} := \left\{ \tilde{x}_i \,\middle|\, \tilde{x}_i = x_i - \frac{1}{m}\sum_{i=1}^{m} x_i \right\} \tag{1}$$

the first eigenvector v_1 can be obtained as

$$v_1 = \operatorname*{argmax}_{\|v\|^2 \le 1} \frac{1}{m}\sum_{i=1}^{m} |\langle v, \tilde{x}_i \rangle|^2. \tag{2}$$

The successive eigenvectors $v_2, \ldots v_d$ are chosen to be orthogonal to the previous ones, where each eigenvector v_i satisfies a similar property to (2) with respect to the remaining $(d - i + 1)$-dimensional subspace.

The solution of this optimization problem is normally obtained by performing a principal axis transform. Further below we shall show that there exist situations where it can be much easier than that. For instance, if we replaced the $\|v\| \le 1$ constraint by $\|v\|_1 \le 1$, then application of Rockafellar's theorem (Sec. 3.1) sequentially would give us the coordinate axes (ordered by decreasing values of the variance) as 1-norm principal components. We shall return to this point in a more general setting later.

For now, we first focus on a nonlinear generalization of PCA. This is particularly useful if data is abundant, i.e. if the sample size m is larger than the dimensionality n of the data: in this case, one may want to capture more information about the distribution of the data, i.e. more features than just n.

2.2 Kernel PCA

Schölkopf et al. [1998] proposed to map the data into some feature space $\mathcal{F}$ and perform PCA there. For this purpose define a map

$$\Phi : \mathcal{X} \to \mathcal{F} \tag{3}$$

whose purpose it is to "preprocess" data suitably such that nonlinear features can be obtained. Such a mapping may map x into its p–th order monomials or some other arbitrary dot product space. Problems arise, however, if the dimensionality of $\mathcal{F}$ is too large to be computed explicitly. Aizerman et al. [1964] introduced a method to circumvent this problem, and applied it in a convergence proof for the potential function method. It is applicable whenever an algorithm can be formulated in terms of dot products and does not require explicit knowledge of $\Phi(x)$. Just as Support Vector Machines [Boser et al., 1992], Kernel PCA satisfies this condition.

Denote by $\mu := \frac{1}{m}\sum_{i=1}^{m}\Phi(x_i)$ the empirical mean of the mapped sample X in feature space $\mathcal{F}$. Then the covariance 'matrix' C can be formulated as

$$C = \frac{1}{m}\sum_{i=1}^{m}(\Phi(x_i) - \mu)(\Phi(x_i) - \mu)^\top \tag{4}$$

and we want to compute the eigensystem (v_i, λ_i) satisfying

$$Cv_i = \lambda_i v_i \text{ for all } 1 \leq i \leq m. \tag{5}$$

However, since we may not want to (or even be unable to) compute the covariance matrix C explicitly, we resort to an implicit representation of v_i in terms of the mapped patterns $\Phi(x_i)$. This can always be done since the image of C is contained in the span of $\{\Phi(x_1), \ldots \Phi(x_m)\}$ (cf. Schölkopf et al. [1998]). Moreover, by the same reasoning, we only have to check that (5) is satisfied for projections onto any $\Phi(x_i)$, which leads to the following equivalent formulation.

$$\frac{1}{m}\sum_{j,j'}^{m}\langle\tilde{\Phi}(x_i), \tilde{\Phi}(x_j)\rangle\langle\tilde{\Phi}(x_j), \tilde{\Phi}(x_{j'})\rangle\alpha_{j'} = \lambda\frac{1}{m}\sum_{j}^{m}\langle\tilde{\Phi}(x_i), \tilde{\Phi}(x_j)\rangle\alpha_j \tag{6}$$

Eq. (6) has to hold for any $1 \leq i \leq m$. Here we defined $\tilde{\Phi}(x) := \Phi(x) - \mu$. Note that the new formulation does not require explicit computation of $\Phi(x)$ any more. Instead, only dot products of the mapped patterns into $\mathcal{F}$ appear.

This allows us to introduce kernels. Denote by

$$k(x, x') := \langle\Phi(x), \Phi(x')\rangle \tag{7}$$

a kernel corresponding to Φ. In many cases, k is much cheaper to compute than Φ. For instance, the dot product of all monomial features of order d generated from x can be written as

$$k(x, x') = \langle x, x'\rangle^d. \tag{8}$$

Similar relations hold for inhomogeneous polynomials

$$k(x, x') = (\langle x, x'\rangle + 1\rangle^d. \tag{9}$$

In general, any kernel corresponding to a positive integral operator, i.e.

$$\int k(x, x')f(x)f(x')\, dx\, dx' \geq 0 \text{ for all } f \in L_2, \tag{10}$$

corresponds to dot products in some feature space $\mathcal{F}$ (see [Smola et al., 1998] for details). A popular example of this class are Gaussian radial basis functions

$$k(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}} \text{ for some } \sigma > 0. \tag{11}$$

It turns out [Schölkopf et al., 1998] that (6) can be rewritten as

$$\tilde{K}\tilde{K}\alpha = \lambda\tilde{K}\alpha \tag{12}$$

where $\tilde{K} = (\mathbf{1} - 1_m)K(\mathbf{1} - 1_m)$, $K_{ij} = k(x_i, x_j)$, $\mathbf{1}$ denotes the identity matrix, and 1_m the matrix of all entries set to $\frac{1}{m}$. Hence the eigensystem λ_i, α^i can be found by computing the eigenvalues and eigenvectors of $\tilde{K}$. The final eigensystem of C is then given by

$$v_i = \lambda_i^{-\frac{1}{2}} \sum_{j=1}^{m} \alpha_j^i \Phi(x_j). \tag{13}$$

2.3 Sparse Kernel PCA

By the reasoning we exploited just before to describe PCA, one may reformulate Kernel PCA as follows. Define the set of admissible weight vectors as

$$V_{\mathrm{RKHS}} := \left\{ w \,\middle|\, w = \sum_{i=1}^{m} \alpha_i \Phi(x_i) \text{ with } \|w\|^2 = \sum_{i,j}^{m} \alpha_i \alpha_j k(x_i, x_j) \leq 1 \right\}. \tag{14}$$

Now the problem of finding the first principal component (5) (i.e. the direction of projection with maximum variance) may be transformed into

$$v_1 = \operatorname*{argmax}_{v \in V_{\mathrm{RKHS}}} \frac{1}{m} \sum_{i=1}^{m} \left| \langle v, \Phi(x_i) \rangle - \frac{1}{m} \sum_{i=1}^{m} \langle v, \Phi(x_j) \rangle \right|^2. \tag{15}$$

This immediately leads to the question whether other (possibly more general) sets might lead to useful feature extractors, too. In the following, unless specified otherwise, V will denote the space of possible directions. In particular one could choose

$$V_{\mathrm{LP}} := \left\{ w \,\middle|\, w = \sum_{i=1}^{m} \alpha_i \Phi(x_i) \text{ with } \sum_{i}^{m} |\alpha_i| \leq 1 \right\}. \tag{16}$$

This is equivalent to a sparsity regularizer used in supervised learning (cf. e.g. [Mangasarian, 1965, Chen et al., 1999]) and leads to the following definition of the first "principal vector" in the ℓ_1 context, hence a new way of kernel *feature* analysis.

$$v_1 = \operatorname*{argmax}_{v \in V_{\mathrm{LP}}} \frac{1}{m} \sum_{i=1}^{m} \left| \langle v, \Phi(x_i) - \frac{1}{m} \sum_{i=1}^{m} \langle v, \Phi(x_j) \rangle \rangle \right|^2 \tag{17}$$

Again, subsequent "principal vectors" can be defined by enforcing optimality with respect to the remaining orthogonal subspaces. The solution of (17) has the nice property of being sparse in terms of the coefficients α_i, due to the ℓ_1 constraint (the coefficients may be chosen from the "hyperdiamond-shaped" ℓ_1 ball).[1] In fact, as we shall show in section 3, the optimal solution is found by picking the direction $\Phi(x_i)$ corresponding to a single pattern, i.e. the solution lies on one of the vertices of the ℓ_1 ball.

Moreover, instead of ℓ_1 constraints, one could as well seek optimal solutions with respect to ℓ_p balls, i.e.

$$V_p := \left\{ w \,\middle|\, w = \sum_{i=1}^{m} \alpha_i \Phi(x_i) \text{ with } \sum_{i}^{m} |\alpha_i|^p \leq 1 \right\}. \tag{18}$$

2.4 Kernel Projection Pursuit

The second modification regarding (15) is to choose a *contrast* function other than the variance, that should be optimized. Hence one obtains solutions of the following type of problems

$$v_1 = \operatorname*{argmax}_{v \in V} \frac{1}{m} \sum_{i=1}^{m} q(\langle v, \Phi(x_i) \rangle) \tag{19}$$

or more generally

$$v_1 = \operatorname*{argmax}_{v \in V} Q\left(\{\langle v, \Phi(x_1)\rangle, \ldots, \langle v, \Phi(x_m)\rangle\}\right) \tag{20}$$

where $q(\cdot)$ and $Q(\cdot)$ are functions which are maximized for a property of the resulting function $\langle v_1, \Phi(x) \rangle$ that might be of interest. This leads to methods which are quite similar to projection pursuit, however with the novelty that they act in feature space rather than in input space.[2]

Common contrast functions define directions as interesting, if they extract features which are least Gaussian, have several modalities, maximize the Fisher Information, the negative Shannon entropy, or other quantities of interest. For a detailed account on these issues see the work of Friedman and Tukey [1974], Friedman and Stuetzle [1981], Huber [1985], Jones and Sibson [1987], Friedman [1987], Härdle [1991]. A possible function q would be $q(\xi) = \xi^4$, thus a function that favours projections with non–Gaussian distributions (and high kurtosis).

3 Algorithms

Solving a general feature extraction problem as described in the previous section (Eq. (20)) is often prone to local minima and thus difficult to achieve. However, under certain conditions, which will be made more explicit, the optimal solution can be found easily in quadratic or even linear time.

[1] Note that the requirement of $\|v\|^2 = 1$ or the corresponding ℓ_1 constraint are necessary — the value of the target function could increase without bound otherwise simply by rescaling v.

[2] Both methods coincide when setting $\Phi(x) = x$.

3.1 Properties of the Optimal Solution

The key idea is based on a theorem due to Rockafellar [1970].

Theorem 1 (Rockafellar [1970], Cor. 32.3.4) *The problem of globally maximizing a convex function bounded above on a polyhedral set not containing straight lines that go to infinity in both directions has a vertex solution.*

This result can be used to obtain local or sometimes global solutions [Bradley et al., 1998, 1997] to convex maximization problems on polyhedral sets which are typically NP–hard. However, if the polyhedral set is suitably chosen, we can use the theorem to our advantage and construct an efficient algorithm. For this purpose we have to transform the result to a form more amenable to our analysis.

Corollary 2 (Vertex Solutions for Kernel Feature Analysis) *The optimization problem*

$$v_1 = \underset{v \in V_p}{\operatorname{argmax}} \; Q\left(\{\langle v, \Phi(x_1)\rangle, \ldots, \langle v, \Phi(x_m)\rangle\}\right) \tag{21}$$

with V_p defined as in (18) and $p \in (0,1]$ has a vertex solution in $V_1 = V_{\mathrm{LP}}$ if Q is convex and bounded above on V_1.

This means that all problems on $V = V_p$ with ($p \in (0,1]$) have identical solutions.
Proof We start with $p = 1$. Clearly the ℓ_1 ball (and thus V_1) is a compact polyhedral set and Q is convex and bounded on V by construction. Hence Theorem 1 applies and an optimal solution can be found at one of the vertices $(0, \ldots, 0, 1, 0, \ldots, 0) \in \mathbb{R}^m$ of the ℓ_1 ball. In other words, the solution can be found by simply picking the direction of projection $\Phi(x_i)$ with largest Q value.

Next we observe that the vertices of $U(\ell_1)$ are contained in $U(\ell_p)$, and that moreover $U(\ell_p) \subset U(\ell_1)$ for $0 < p < 1$ where $U(\cdot)$ denotes the unit ball with respect to the corresponding metric. Since the maximum over a subset (i.e. $U(\ell_p)$) cannot exceed the maximum over a superset ($U(\ell_1)$), it follows directly that the vertices of $U(\ell_1)$ contain also a solution for $V = V_p (p \in (0,1])$. ∎

This allows us to compute the first component of interest v_1 by simply evaluating the terms $Q\left(\{k(x_i, x_1), \ldots, k(x_i, x_m)\}\right)$ for all $1 \leq i \leq m$ and picking the one with the largest value. We assume that projections onto v and $-v$ yield the same Q value (i.e. are equally interesting), thus we may limit ourselves to the positive orthant only.

3.2 Sequential Decompositions

The obvious question is how to proceed once the first direction of interest has been found. In the following denote by V^i the space of directions to select from in the ith round; in particular, $V^1 := V$. The vectors spanning that space will be denoted by Φ^i_j. We start with $\Phi^1_j = \Phi(x_j)$. Unless stated otherwise we will limit ourselves to the case of a 1–norm on the coefficients (i.e. V_1^i). Recall that we proved that p–norms with $0 < p < 1$ lead to identical solutions on the corresponding set V_p^1. Hence we have

$$V^i := \left\{ w \,\middle|\, w = \sum_{j=1}^{m} \alpha_j^i \Phi_{ij} \text{ with } \sum_{j=1}^{m} |\alpha_j^i| \leq 1 \right\}. \tag{22}$$

We discuss the following three possible choices.

1. One might, for instance, simply remove the corresponding vector $\Phi_j^i = \Phi(x_j)$ from the set of possible directions (and keep all other directions unchanged), i.e. $\Phi_j^{i+1} = \Phi_j^i$ and repeat the proposed procedure to find the next vector v_{i+1}. However, this may lead to many very similar 'principal' directions (each of which might be interesting on its own) which do not convey much more information than just a single previous direction (cf. Figure 1). This is definitely not desirable, even though the computational cost for subsequent computations is very low (a simple sorting operation), once all Q values have been computed.

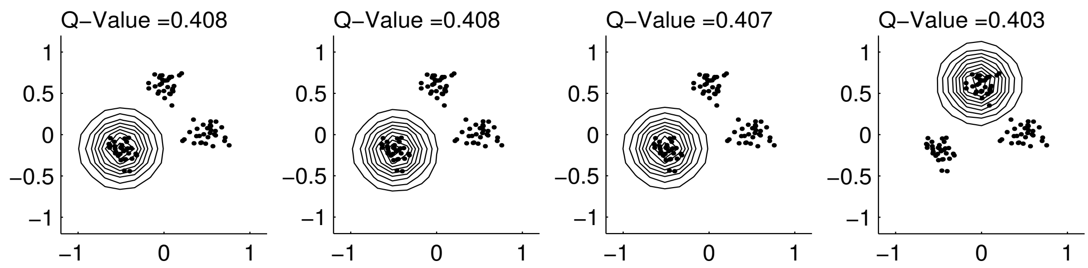

Figure 1: Contour plots of first 4 feature extractors of sparse Kernel Feature Analysis with the selection criterion of 1 for Gaussian RBF Kernels ($\sigma^2 = 0.05$) for a dataset of 120 samples. The small dots denote data. Note that many feature extractors are very similar.

2. A second alternative is to require that each direction v_i be orthogonal to all previous directions, i.e. $\langle v_i, v_j \rangle = 0$ for all $i > j$. The easiest way to achieve this is by orthogonalizing all the vectors spanning V^{i+1} with respect to v_i, i.e.
$$\Phi_j^{i+1} := \Phi_j^i - \frac{v_i}{\|v_i\|^2} \langle \Phi_j^i, v_i \rangle. \tag{23}$$
As in the previous case, also this approach reduces the set of vectors spanning V^{i+1} by one, since v_i is chosen among Φ_j^i. Computation of the first k principal features involves only k kernel functions. This is so since Φ_j^i is a linear combination of i images of patterns, of which $i-1$ already were used in the computation of the previous $i-1$ features.

Since Φ_j^i is not necessarily in V_1 (the sum of the expansion coefficients α will not be contained inside the ℓ_p unit ball any more) we will call this approach *Unnormalized* Kernel Feature Analysis.

3. This immediately suggests a *normalized* version of the previous algorithm. In addition to (23) one applies a normalization step such that

$$\Phi_j^i = \sum_l \alpha_{jl}^i \Phi(x_l) \text{ with } \sum_l |\alpha_{jl}^i|^p = 1. \tag{24}$$

In other words, one projects the directions back onto V_p. One could ask why not to solve the problem of finding a successive orthogonal decomposition of V_p (at least for $p = 1$) directly. This is undesirable since the intersection between an ℓ_1 ball and a cutting plane through the origin (i.e. the set of vectors orthogonal to v_1) could create an exponential number of vertices.

This can be seen by considering two arbitrary directions $\Phi(x_i), \Phi(x_{i'})$ not identical to v_1. The edges of the absolute convex combination of these directions intersect the plane normal to v_1 at at least two points. These constitute vertices of the polyhedral subset which would have to be taken into consideration when solving the next subproblem. Since this would be computationally too expensive we will not pursue this option any further.

3.3 A Simple Algorithm

The previous considerations lead to the following simple algorithm to compute sparse kernel feature analyzers (we omit the trivial Choice 1) of Section 3.2:

1. Compute the matrix

$$k_{ij} := k(x_i, x_j) \tag{25}$$

This costs $O(d \cdot m^2)$ operations, where d is the dimensionality of input space $\mathcal{X}$.

2. Initialize $\alpha_{01}, \ldots, \alpha_{0m} = 1$, and $\mathrm{idx}(\cdot)$ as the empty list (we have dropped the i, indicating that in the implementation, the same variables can be used in each round i). These are the initial scalings for the directions of projection. The cost is $O(m)$.

3. For $i = 1$ to I repeat (I denotes the number of features to be extracted)

   (a) Compute the Q values based on $\Phi(x_1), \ldots, \Phi(x_m)$ for all directions $\Phi_{i1}, \ldots \Phi_{i,m-i+1}$. This is done by computing

$$\langle \Phi_j^i, \Phi(x_l) \rangle = \alpha_{0j} k_{jl} + \sum_{t=1}^{i-1} \alpha_{tl} k_{\mathrm{idx}(t),l} \tag{26}$$

These operations cost $O(i \cdot m^2)$ steps since we need i operations per dot product.

(b) Compute the Q value for each direction Φ_j^i. This again costs $O(m^2)$ operations.

(c) Perform a maximum search over all Q values $(O(m))$ and pick the corresponding Φ_j^i. This is the i–th principal direction v_i. Store the corresponding coefficients $\alpha_{1j}, \ldots \alpha_{ij}$. Do not use this row any further. Finally set $\mathrm{idx}(i) = j$.

(d) Compute the new search space. In any case we have to perform the orthogonalization step according to (23). Since we already computed (26) before, the values of projection can be obtained in $O(i)$ steps each, i.e. in total an $O(i \cdot m)$ operation.

Finally all coefficients have to be stored into α_{ij}. All entries concerning $\Phi(x_j)$ are sorted into α_{il} with $1 \leq l \leq m$ respectively. The other coefficients are assigned to α_{tl} with $1 \leq t < i$ and $1 \leq l \leq m$.

(e) If we are dealing with a normalized variant of Kernel Feature Analysis, the coefficients α_{tl} with $0 \leq l \leq m$ are rescaled by

$$\beta_l := \sum_{t=0}^{i} |\alpha_{tl}^{\mathrm{old}}|^p \text{ such that } \alpha_{tl}^{\mathrm{new}} = \beta_l^{-1/p} \alpha_{tl}^{\mathrm{old}} \tag{27}$$

This again is an $O(i \cdot m)$ operation.

END

END

Whilst the computation of i features on new data requires only $O(i \cdot m)$ operations, extraction of the principal directions themselves is still an $O(m^2)$ operation. The constants, however, are significantly smaller than when computing eigenvectors of a matrix (the latter requires several passes over the matrix K_{ij}). This is the case since finding the direction with maximum Q value still requires computation of all dot products between all possible directions of projection and the actual patterns to be analyzed.

3.4 A Probabilistic Speedup

It is quite unsatisfactory having to compute all possible Q values for all directions, if in the end, only one of those directions is chosen. Intuitively one might suggest bailing out earlier from the calculation of those directions that do not seem to be promising in the first place. This approach is rather risky (at least if done without care), since one may easily ignore important directions if too many Φ_j^i are prematurely discarded.

What we need is a tool to quantify maximum deviations, such that, at least with high confidence we can assume that the proposed algorithm found a desirable direction. Hoeffding's theorem is very useful in this regard.

Theorem 3 (Hoeffding [1963]) *Let $\xi_1, \ldots, \xi_m$ be independent bounded random variables such that ξ_i falls in the interval $[a_i, a_i + b_i]$ with probability one. Denote their average by $S_m = \frac{1}{m}\sum_i \xi_i$. Then for any $\varepsilon > 0$ one has*

$$\left.\begin{array}{l} \Pr\{S_m - E[S_m] \geq \varepsilon\} \\ \Pr\{E[S_m] - S_m \geq \varepsilon\} \end{array}\right\} \leq e^{-\frac{2m^2\varepsilon^2}{\sum_{i=1}^m b_i^2}} \tag{28}$$

Unfortunately, this theorem only states maximum deviations between empirical and true means of distributions. What we need is a statement regarding sample averages over different sample size. This can be readily constructed as follows.

Corollary 4 (Maximum Deviation Bounds for Empirical Means) *Let $\xi_1, \ldots, \xi_m$ be independent identically distributed bounded random variables, falling into the interval $[a, a + b]$ with probability one. Denote their average by $S_m = \frac{1}{m}\sum_i \xi_i$. Moreover denote by $\xi_{s(1)}, \ldots, \xi_{s(\tilde{m})}$ with $\tilde{m} < m$ a subset of the same random variables (with $s : \{1, \ldots, \tilde{m}\} \to \{1, \ldots, m\}$ being an injective map), and $S_{\tilde{m}} = \frac{1}{\tilde{m}}\sum_i \xi_{s(i)}$. Then for any $\varepsilon > 0$ one has*

$$\left.\begin{array}{l} \Pr\{S_m - S_{\tilde{m}} \geq \varepsilon\} \\ \Pr\{S_{\tilde{m}} - S_m \geq \varepsilon\} \end{array}\right\} \leq e^{-\frac{2m\tilde{m}\varepsilon^2}{(m-\tilde{m})b^2}} \tag{29}$$

Proof By construction $E[S_m - S_{\tilde{m}}] = 0$ since S_m and $S_{\tilde{m}}$ are both averages over sums of random variables drawn from the same distribution. Hence we only have to rewrite $S_m - S_{\tilde{m}}$ as an average over (different) random variables itself for being able to apply Hoeffding's bound. Without loss of generality, we assume that $s(i) = i$ for $i = 1, \ldots, \tilde{m}$.

$$S_m - S_{\tilde{m}} = \frac{1}{m}\sum_{i=1}^{m} \xi_i - \frac{1}{\tilde{m}}\sum_{i=1}^{\tilde{m}} \xi_i = \frac{1}{m}\sum_{i=1}^{\tilde{m}} \left(1 - \tfrac{m}{\tilde{m}}\right)\xi_i + \frac{1}{m}\sum_{i=\tilde{m}+1}^{m} \xi_i \tag{30}$$

Thus we may split up $S_m - S_{\tilde{m}}$ into a sum of $\tilde{m}$ random variables with range $b_i = (\frac{m}{\tilde{m}} - 1)b$ and $m - \tilde{m}$ random variables with range $b_i = b$. Thus

$$\sum_{i=1}^{m} b_i^2 = b^2\tilde{m}\left(\frac{m}{\tilde{m}} - 1\right)^2 + (m - \tilde{m})b^2 = b^2(m - \tilde{m})\frac{m}{\tilde{m}} \tag{31}$$

Substituting this into (28) and noting that $S_m - S_{\tilde{m}} - E[S_m - S_{\tilde{m}}] = S_m - S_{\tilde{m}}$ completes the proof. ∎

For small $\tilde{m}/m$ the RHS in (29) reduces to $\exp(-\frac{2\tilde{m}\varepsilon^2}{b^2})$. In other words —

deviations on the subsample $\tilde{m}$ dominate the overall deviation of $S_m - S_{\tilde{m}}$ from 0. This allows us to compute a cutoff criterion. For convenience we compute this quantity for deviations between subsets of size m and $2m$ (and $\tilde{m}$ functions to be tested).

Corollary 5 (Cutoff Guarantee)
Choose some $\epsilon > 0$. Denote by $\tilde{m}$ the total number of directions of projection Φ_i, by m the size of the half sample and by Q_j^m, Q_j^{2m} the corresponding estimates. Then

$$\left.\begin{array}{l} \Pr\left\{\sup_{1\leq j\leq \tilde{m}} \left[Q_j^m[\Phi_i] - Q_j^{2m}[\Phi_i]\right] > \epsilon\right\} \\ \Pr\left\{\sup_{1\leq j\leq \tilde{m}} \left[Q_j^{2m}[\Phi_i] - Q_j^m[\Phi_i]\right] > \epsilon\right\} \end{array}\right\} \leq \tilde{m} e^{-\frac{4m\epsilon^2}{b^2}} \tag{32}$$

Moreover, the probability that the maximum of Q_j^{2m} will be contained in the set $\{j|Q_j^m < Q_{\bar{j}}^m - \varepsilon\}$ (where $Q_{\bar{j}}^m$ is the maximum over the two subsets) is bounded from above by

$$2\tilde{m} e^{-\frac{m\epsilon^2}{b^2}}. \tag{33}$$

Equivalently, for any $\delta > 0$, given $\tilde{m}$ and m, the probability that the maximum of Q_j^i is contained in set removed by the cutoff is less than δ if

$$\varepsilon^2 \geq \frac{b^2}{m}\left(\log 2\tilde{m} - \log \delta\right). \tag{34}$$

Proof Equation (32) follows directly from corollary 4 by upper bounding the *joint* probability of maximum deviation by the sum of the individual deviations (this leads to the factor $\tilde{m}$). Moreover functions excluded from the subset must not lead to Q-values larger than $Q_i^m + \epsilon/2$, functions included not lead to Q-values smaller than $Q_i^m - \epsilon/2$. Thus the modified exponent and the factor 2. Finally, (34) follows from solving (33) for ϵ^2. ∎

Assume that we can eliminate half of the basis functions at every single step, then the computational cost for computing a single function is $O(m \log m)$ which is a significant improvement over the $O(m^2)$ cost of Kernel-PCA. This result follows directly from properties of divide and conquer algorithms (e.g. Fast Fourier Transformation).

3.5 A Quantile Trick

In some cases one is not necessarily interested in the *best n* feature extractors but would already be content to obtain feature extractors that are *among the best* ones obtainable. For instance, it might be sufficient for preprocessing if each feature was among the best 5% obtainable. This leads to another approach of avoiding the maximum search operation over all m possible directions: compute a subsample of $\tilde{m}$ directions and choose the largest Q–value amongst them. One can show that such a subsampling approach leads on average to values in the $\frac{\tilde{m}}{\tilde{m}+1}$ quantile range.

Lemma 6 (Quantile Levels by Subsampling) *Taking the maximum element of an $\tilde{m}$ subsample of a set leads on average to an element within the $\frac{\tilde{m}}{\tilde{m}+1}$ quantile range of the whole set and with probability $1-\eta$ to a quantile of at least $\eta^{\frac{1}{m}}$.*

Proof For large sample size (ignoring finite sample size effects) the probability that none of the elements of the subsample has a quantile ranking higher than $q \in (0,1)$ is simply $q^{\tilde{m}}$ (since it is q for each sample and we compute the joint probability which is assumed to be independent, ignoring finite sample size effects). Solving for the quantiles leads to the $\eta^{\frac{1}{m}}$ lower bound with confidence $1-\eta$. Moreover, the density function is given by $p(q) = \tilde{m}q^{\tilde{m}-1}$, and the expected quantile is thus $\hat{q} = \int_0^1 q\, p(q)\, dq = \frac{\tilde{m}}{\tilde{m}+1}$. ∎

For instance, in order to obtain a 95% quantile with 95% percent probability one has to consider a subset of only 59 samples. This leads to a considerable speedup in computing feature extractors since in general m is several orders of magnitude larger than that. Note that this only ensures that we will obtain *one* of the best feature extractors, but no guarantee can be given in terms of its *absolute distance* to the largest best extractor (since this depends on the tails of the distribution). For this type of reasoning, the strategy of the previous section should be used.

Overall, computational complexity for extraction of a single feature is reduced to $O(cm)$ rather than $O(m^2)$. The same applies to the memory requirements. Thus unless the *best* feature extractors are needed this should be the method of choice.

4 Theory

Despite the rather simple algorithmic rule — (1) find the function with the maximum Q–value; (2) project $\Phi(x_i)$ out; (3) repeat — kernel feature analysis has interesting theoretical properties such as the possibility to state uniform convergence bounds and a close resemblance with wavelet techniques proposed by Mallat and Zhang [1993] and reduced set algorithms for Support Vector machines as the one by Schölkopf et al. [1999].

4.1 Greedy m–target Approximation

Mallat and Zhang [1993], Schölkopf et al. [1999] deal with the problem of a greedy 1–target approximation, i.e. given a target function f and a set of basis functions S the aim is to find an approximation of f with as few terms as possible in an iterative fashion. Ignoring differences in the consideration of Hilbert spaces used (Mallat and Zhang [1993] use $L_2(\mathbb{R}^n)$ or rather $[a,b]^n$, Schölkopf et al. [1999] deal with Reproducing Kernel Hilbert Spaces, implicitly given by the kernel $k(x,x')$) and in the choice of basis functions that make up S, both algorithms proceed as follows, starting with $f_1 := f$:

1. At stage i find the best approximating function $s_i \in S$ for f_i, i.e.

$$s_i := \operatorname*{argmin}_{s \in S} \min_{\alpha \in \mathbb{R}} \|f_i - \alpha s\|^2 \tag{35}$$

2. Set $f_{i+1} := f_i - \|s_i\|^{-2} \langle f_i, s_i \rangle s_i$, $f_1 := f$, and continue with step (1).

In a more general feature extraction problem one has no knowledge about the target concept f beforehand. Thus it would be desirable to construct a general feature extractor in order to represent the expressive power of the set $S := \{k(x_1, x), \ldots, k(x_m, x)\}$ by a subset $\tilde{S} = \{k(x_{i(1)}, x), \ldots, k(x_{i(\tilde{m})}, x)\}$. A standard approach in machine learning is to perform clustering on the dataset (cf. e.g. [Bishop, 1995, Schölkopf et al., 1997]) and uses the cluster centers as initialization for the centers of the basis functions. In analogy to above, however, one could think of a different algorithm:

1. At stage i find the best approximating function $s_i \in S$ for $f^1, \ldots, f^n$, i.e.

$$s_i := \operatorname*{argmin}_{s \in S} \sum_{j=1}^{m} \min_{\alpha \in \mathbb{R}} \|f_i^j - \alpha s\|^2 \tag{36}$$

2. Set $f_{i+1}^j := f_i^j - \|s_i\|^{-2} \langle f_i^j, s_i \rangle s_i$, $f_1^j := f^j$, and continue with step (1).

Recalling from Principal Component Analysis (cf. e.g. [Jolliffe, 1986]) that the direction which minimizes the remaining variance is also the one maximizing the variance projected onto it, this setting becomes very similar to the formulation of sparse Kernel Feature Analysis as proposed in Section 3. The main difference however is that the norm $\|\cdot\|$ and the dot product $\langle \cdot, \cdot \rangle$ do not match as required for an optimal strategy.

4.2 Uniform Convergence Properties

It is immediately obvious why uniform convergence properties are useful in the context of supervised learning since they give performance guarantees of the classifier (or regressor) on (unseen) test data, based on its complexity and its behaviour on the (seen) training data. The situation is slightly more subtle in the unsupervised learning setting and the immediate question is which question to ask.

Here we propose a quantity which is related to the support sets of a distribution as already exploited in one-class Support Vector machines by Schölkopf et al. [1999]. A necessary requirement for a feature extractor f to be reliable is that the probability $\Pr\{f(x) > c\}$ does not differ too much from the relative frequency of occurrence of such an event on the training set, since otherwise any classifier built by using the feature extractors f would share the (bad) statistical properties of f itself.

Introducing the scale parameter γ which will play the role of a margin, uniform convergence bounds for classification with a margin (see [Smola et al., 2000a] for a survey on large margin related bounds and techniques) can be rewritten to suit our needs. They take on the following form:

$$Pr\left\{\sup_{f\in\mathcal{F}}\left|\Pr\{f(x)>c\}-\sum_{f(x_i)>c-\gamma}\frac{1}{m}\right|>\epsilon\right\}\leq c_1(m)E\left[\mathcal{N}(\gamma,\mathcal{F},\ell_p^m)\right]e^{-\epsilon^\beta m/c_2}. \tag{37}$$

This can be seen by replacing in the classical classification bounds c by the threshold of the classifier and γ by the width of the margin. Here $\mathcal{N}$ denotes the *covering number* of a set of hypotheses $\mathcal{F}$, ℓ_p^m the metric with respect to which this quantity is determined. Moreover $c_1(m)$ depends at most linearly on m and c_2 is a positive constant. The expectation on $\mathcal{N}$ is taken over the distribution of the data and is usually replaced by the sup over all possible datasets for technical reasons.

Without going into further technical details we note that the important quantity to be determined is $\mathcal{N}(\gamma,\mathcal{F},\ell_p^m)$ which depends on the size of the class of functions. Hence, if the capacity $\mathcal{F}$ is limited (corresponding to a small class V in (19)) the features become more reliable. In particular, for the function classes considered in sparse kernel feature analysis (convex combinations of basis functions), bounds on the covering numbers can be given explicitly [Smola et al., 2000b]. This shows that the implicit regularization (by choosing V accordingly) leads to reliable feature extractors. Moreover (37) gives a rationale why one should look for functions with large variance from the same model class, since then γ will be small with respect to the overall range of the feature values $f(x_i)$.

As it is not the central topic of the the present paper, we refrain from actually assembling all the above ingredients to obtain kernel-dependent bounds on the reliability of feature extractors, and instead move on to experiments.

5 Experiments

We presently show that on 2D toy examples, Kernel Feature Analysis provides features qualitatively similar to Kernel Principal Component Analysis. Figures 2 and 3 show features extracted by both types of algorithms.

Note that the main difference lies in the first few features. For instance, KFA is only using one basis function for the first feature (due to the built-in sparsity), which enforces a feature extractor sitting on one of the three clusters. Kernel PCA, on the other hand, will typically have contributions of all basis functions already for the first feature.

Figure 4 displays features extracted by the quantile version of sparse KFA which is significantly faster to compute; note the close resemblance to figure 3.

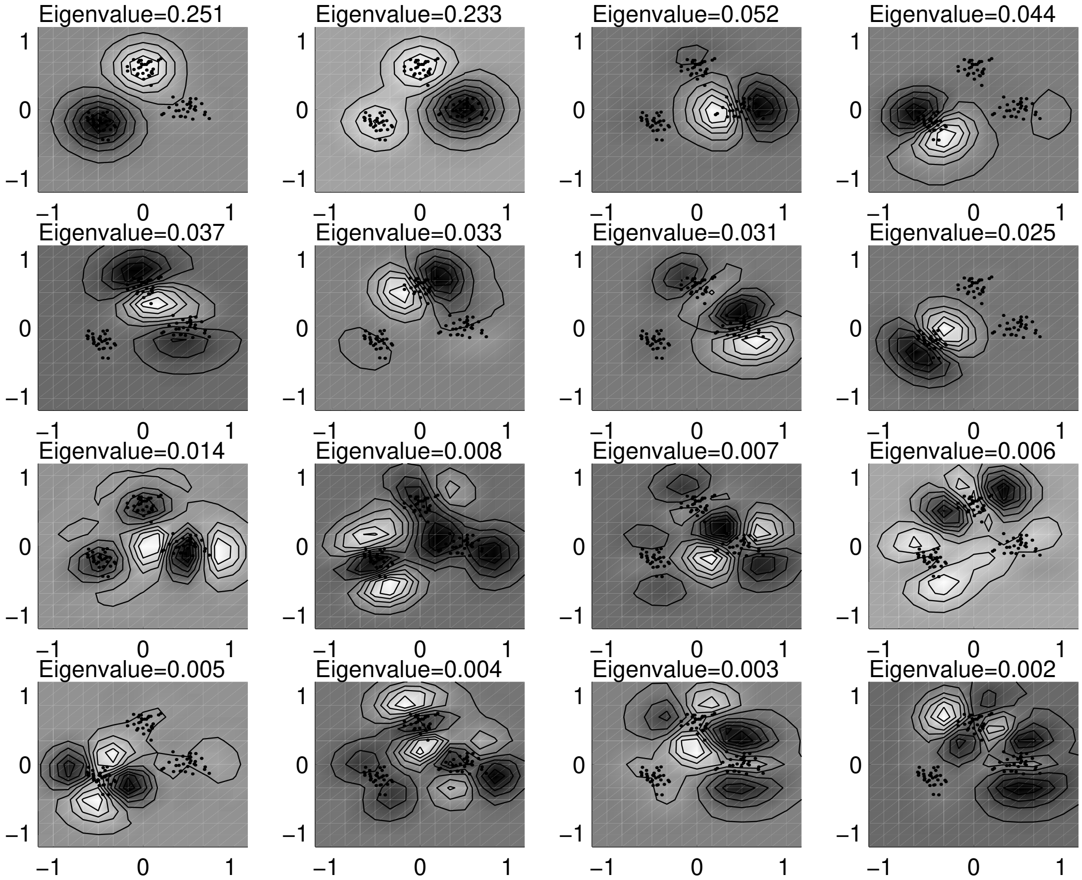

Figure 2: Contour plots of first 16 feature extractors of Kernel-PCA for Gaussian RBF Kernels ($\sigma^2 = 0.05$) for a dataset of 120 samples. The small dots denote data.

In all cases, it can be seen that the features are meaningful in that they discover nontrivial structure. The first features identify the cluster structure in the data set, while the higher order ones analyze the individual clusters in more detail.

In conclusion, we note that the present paper has proposed a new algorithm for feature extraction which is fast and leads to sparse representations. In addition, we have given a theoretical analysis of the algorithm and some toy examples. Future work will include the application to real-world feature extraction problems.

Acknowledgements Parts of this Data Mining Institute Report 99-04, October 1999, were done while AJS was visiting the Department of Computer Sciences, University of Wisconsin, Madison. AS is supported by a grant of the Deutsche Forschungsgesellschaft (Sm 62/1-1). Research of OLM is supported by National Science Foundation Grants CCR-9729842, CDA-9623632, by Air Force Office of

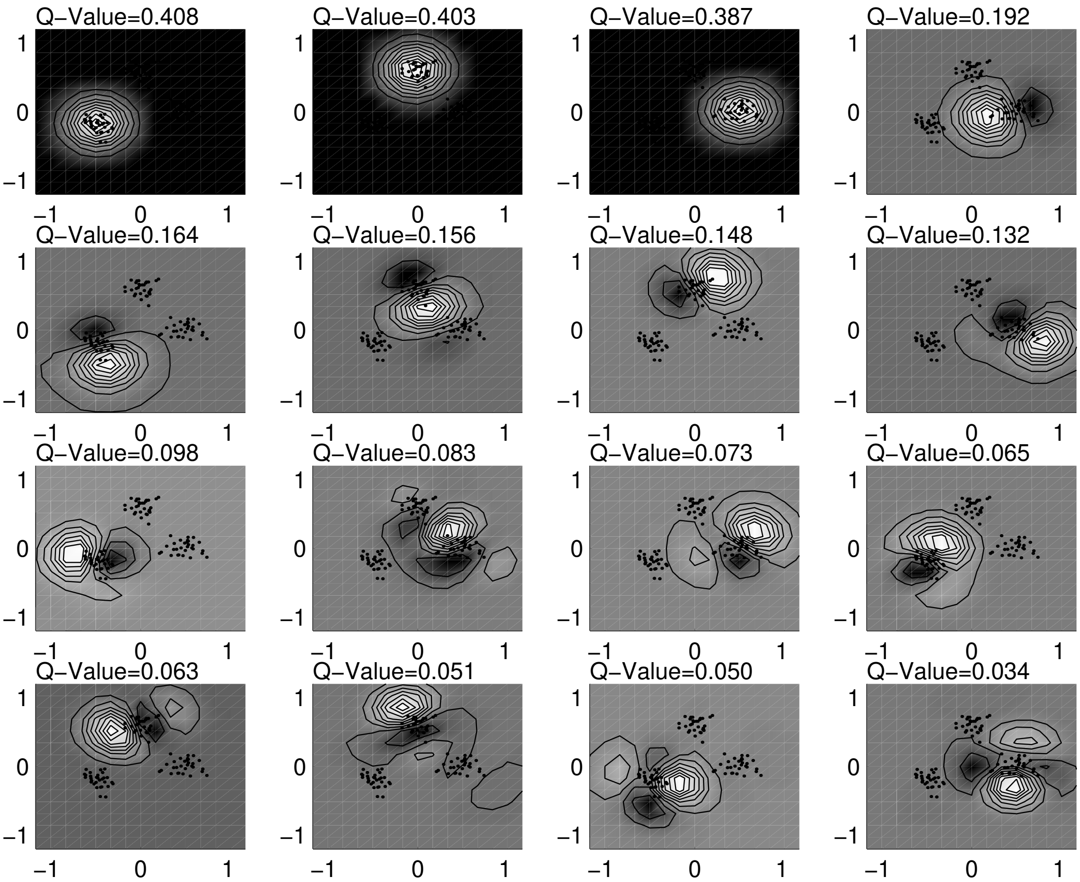

Figure 3: The first 16 features of sparse Kernel Feature Analysis for the same kernel as in Figure 2. Note that every additional feature only needs one more kernel function to be computed. Observe how KFA first selects the cluster centers, then starts splitting the clusters, next splits them orthogonally, with much fewer kernel functions than Kernel-PCA. The Q-value denotes the variance of the projected-out feature.

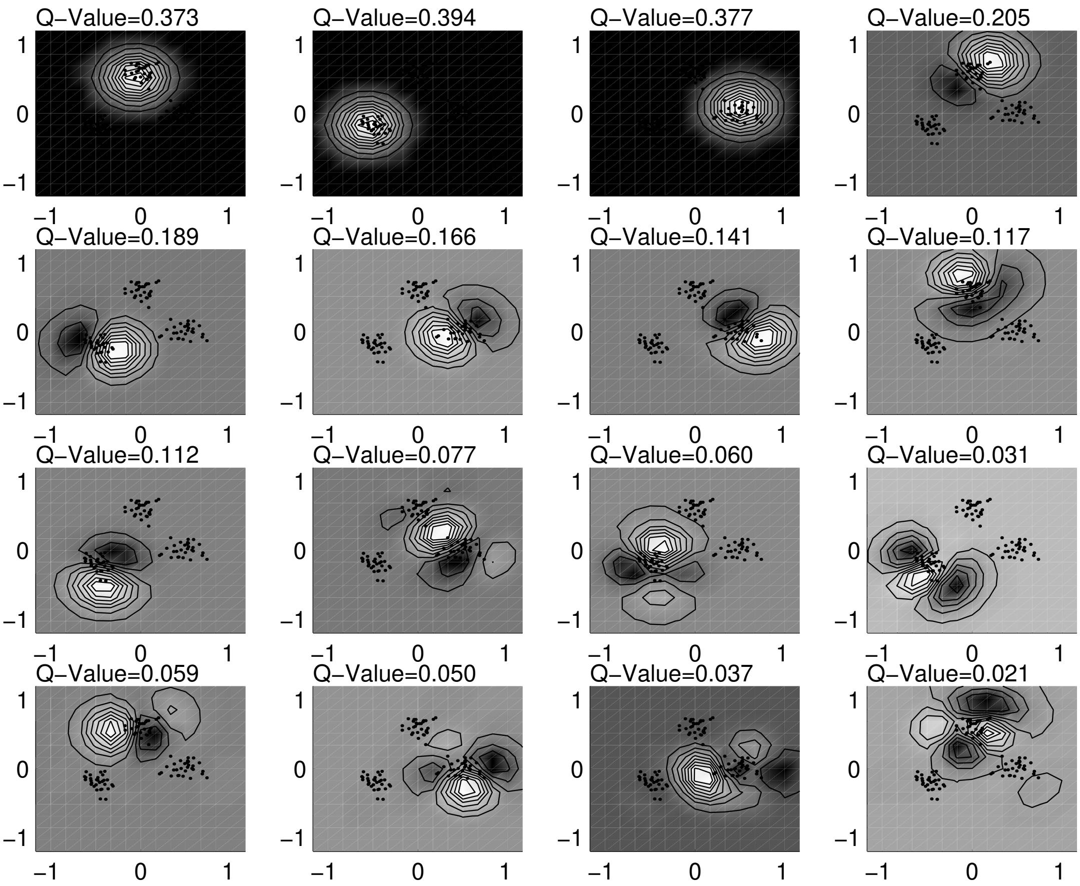

Figure 4: Randomized Version of sparse Kernel Feature Analysis. Only a subset of 10 features per iteration was used (leading to an average quantile of over 95%. Note the similarity to Figure 2 which is an $O(m^2)$ rather than $O(km)$ algorithm.

Scientific Research Grant F49620-97-1-0326 and by the Microsoft Corporation. AJS and BS gratefully acknowledge support of GMD FIRST where parts of this work were done.

References

M. A. Aizerman, E. M. Braverman, and L. I. Rozonoér. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

K. Bennett. Combining support vector and mathematical programming methods for induction. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - SV Learning*, pages 307–326, Cambridge, MA, 1999. MIT Press.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

P. Bradley and O. Mangasarian. k–plane clustering. Mathematical Programming Technical Report 98-08, University of Wisconsin Madison, 1998a.

P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Machine Learning Proceedings of the Fifteenth International Conference(ICML '98)*, pages 82–90, San Francisco, California, 1998b. Morgan Kaufmann. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.Z.

P. S. Bradley, O. L. Mangasarian, and W. N. Street. Clustering via concave minimization. In *Advances in Neural Information Processing Systems*, volume 9, pages 368–374, Cambridge, MA, 1997. MIT Press.

P. S. Bradley, O. L. Mangasarian, and W. N. Street. Feature selection via mathematical programming. *INFORMS Journal on Computing*, 10(2):209–217, 1998. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/95-21.ps.Z.

C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector learning machines. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 375–381, Cambridge, MA, 1997. MIT Press.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *Siam Journal of Scientific Computing*, 20(1):33–61, 1999.

J. H. Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82:249–266, 1987.

J. H. Friedman and W. Stuetzle. Projection pursuit regression. *J. American Statistical Association*, 76(376):817–823, 1981.

J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, C-23(9):881–890, 1974.

W. Härdle. *Smoothing Techniques, With Implementations in S.* Springer, New York, 1991.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441 and 498–520, 1933.

P. J. Huber. Projection pursuit. *The Annals of Statistics*, 13(2):435–475, 1985.

I. T. Jolliffe. *Principal Component Analysis.* Springer-Verlag, New York, New York, 1986.

M. C. Jones and R. Sibson. What is projection pursuit? *Journal of the Royal Statistical Society, Series A*, 150(1):1–36, 1987.

K. Karhunen. Zur Spektraltheorie stochastischer Prozesse. *Ann. Acad. Sci. Fenn.*, 34, 1946.

S. Mallat and Z. Zhang. Matching Pursuit in a time-frequency dictionary. *IEEE Transactions on Signal Processing*, 41:3397–3415, 1993.

O. L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.

O. L. Mangasarian. Generalized support vector machines. Technical Report 98-xx, University of Wisconsin, Computer Sciences Department, Madison, 1998.

R.T. Rockafellar. *Convex Analysis*, volume 28 of *Princeton Mathematics Series.* Princeton University Press, 1970.

C. Saunders, M. O. Stitson, J. Weston, L. Bottou, B. Schölkopf, and A. Smola. Support vector machine — reference manual. Technical Report CSD-TR-98-03, Department of Computer Science, Royal Holloway, University of London, Egham, UK, 1998.

B. Schölkopf, S. Mika, C.J.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*, 10(5):1000–1017, 1999.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - SV Learning*, pages 327–352. MIT Press, Cambridge, MA, 1999.

B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE Trans. Sign. Processing*, 45:2758 – 2765, 1997.

B. Schölkopf, R.C. Williamson, A.J. Smola, and J. Shawe-Taylor. SV estimation of a distribution's support. In *Neural Information Processing Systems*, 1999. forthcoming.

A. J. Smola. *Learning with Kernels.* PhD thesis, Technische Universität Berlin, 1998.

A. J. Smola and B. Schölkopf. From regularization operators to support vector kernels. In *Advances in Neural information processings systems 10*, pages 343–349, San Mateo, CA, 1998.

A. J. Smola, B. Schölkopf, and K.-R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11:637–649, 1998.

A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans. *Advances in Large Margin Classifiers.* MIT Press, Cambridge, MA, 2000a. forthcoming.

A.J. Smola, A. Elisseeff, B. Schölkopf, and R.C. Williamson. Entropy numbers for convex combinations and MLPs. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, Cambridge, MA, 2000b. MIT Press. forthcoming.

G. Wahba. *Splines Models for Observational Data.* Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.