Constructing Descriptive and Discriminative Nonlinear Features: Rayleigh Coefficients in Kernel Feature Spaces

Sebastian Mika, Gunnar Rätsch, *Member*, *IEEE*, Jason Weston, Bernhard Schölkopf, Alex Smola, and Klaus-Robert Müller

Abstract—We incorporate prior knowledge to construct nonlinear algorithms for invariant feature extraction and discrimination. Employing a unified framework in terms of a nonlinearized variant of the Rayleigh coefficient, we propose nonlinear generalizations of Fisher's discriminant and oriented PCA using support vector kernel functions. Extensive simulations show the utility of our approach.

Index Terms—Fisher's discriminant, nonlinear feature extraction, support vector machine, kernel functions, Rayleigh coefficient, oriented PCA.

1 Introduction

It is common practice to preprocess data by extracting linear or nonlinear features. In many such feature extraction techniques, one has a criterion assessing the quality of a single feature which ought to be optimized. For additional features, one often uses the same quality criterion, but enforces additional constraints with respect to the formerly found features. The most well-known feature extraction technique in this framework is principal component analysis, PCA (e.g., [1]). However, PCA is a linear technique and cannot capture nonlinear structure in a data set. Therefore, nonlinear generalizations have been proposed, among them, kernel PCA [2], which computes the principal components of the data set mapped nonlinearly into some high-dimensional feature space $\mathcal{F}$. This work generalizes what has been done for kernel PCA to a more general setting. We will first recall how to use prior information for extracting meaningful features in a linear setting leading us to the Rayleigh coefficient. In a second step, which is the main contribution of this work, we propose a nonlinear variant of the Rayleigh coefficient and discuss regularization approaches and implementation issues.

1.1 Linear Features and Rayleigh Coefficients

Often, one has prior information available that can be used to formulate quality criteria or, probably even more common, the features are extracted for a certain purpose, e.g., for subsequently training some classifier. For instance, we might know that the examples are corrupted by noise or that there are invariance transformations under which a classification should not change. The concepts of known noise or transformation invariance are closely related: both can be interpreted as causing a change in a feature, which should be avoided. Clearly, invariance alone is never a sufficient condition for a good feature, as we could simply take the constant feature. What one would like to obtain is a feature, which is as invariant as possible while still covering as much of the information necessary for describing the data's properties of interest.

A classical and well-known technique that solves this type of problem, considering only one linear feature, is the maximization of the so called *Rayleigh* coefficient (e.g., [3])

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top S_I \mathbf{w}}{\mathbf{w}^\top S_N \mathbf{w}}. \tag{1}$$

Here, $\mathbf{w}$ denotes the weight vector of a linear feature extractor (i.e., for an example $\mathbf{x}$, the feature is given by the projections $(\mathbf{w} \cdot \mathbf{x})$) and S_I, S_N are symmetric matrices designed such that they measure the desired information and the undesired noise along the direction of $\mathbf{w}$. The ratio in (1) is maximized when one covers as much as possible of the desired information while avoiding the undesired.

1.2 Choices for Matrices

In the following, we give some common choices for the matrices S_I and S_N leading to well-known algorithms. To this end, let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$ be our training sample and, where appropriate, $\mathcal{X}_1 \cup \mathcal{X}_2 = \mathcal{X}, \mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$, two subclasses (with $|\mathcal{X}_i| = \ell_i$).

Principal Component Analysis. If S_I is the covariance matrix

$$S_I = \frac{1}{\ell}\sum_{\mathbf{x}\in\mathcal{X}}(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^\top, \quad \mathbf{m} = \frac{1}{\ell}\sum_{\mathbf{x}\in\mathcal{X}}\mathbf{x}, \tag{2}$$

and S_N identity, we recover standard PCA.

Oriented PCA. If S_I is the data covariance and S_N the noise covariance (which can be estimated analogous to (2), but over examples sampled from the assumed noise distribution), we obtain oriented PCA [1], which aims at finding a direction that describes most variance in the data while avoiding known noise as well as possible.

Fisher's discriminant. If we look for discriminating directions for classification, we can choose S_I to measure the separability of class centers (between class variance) and S_N to measure the within class variance. In that case, we recover the well known Fisher discriminant (e.g., [4]) where S_I and S_N are given by

$$S_I = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^\top, \quad \text{and}$$
$$S_N = \sum_{i=1,2}\sum_{\mathbf{x}\in\mathcal{X}_i}(\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^\top,$$

$\mathbf{m}_i$ denoting the sample mean for class i.

Invariances. To incorporate a known invariance, e.g., in oriented PCA, one can use a matrix similar to the tangent covariance matrix [5], [6] in the denominator of (1):

$$S_N = \sum_{\mathbf{x}\in\mathcal{X}}(\mathbf{x}-\mathcal{L}_t\mathbf{x})(\mathbf{x}-\mathcal{L}_t\mathbf{x})^\top, \tag{3}$$

for some small $t > 0$. Here, $\mathcal{L}_t$ is a 1-parameter transformation, e.g., a rotation. This choice for S_N can be seen as a finite difference approximation of the covariance of the tangent of $\mathcal{L}_t$ at point $\mathbf{x}$ (details, e.g., in [7]). Using S_I as in (2) and S_N as in (3) in oriented kernel PCA, we impose invariance under the local transformation $\mathcal{L}_t$ (see Section 5.1 and Fig. 1 for an illustration).

Of course, we are free to combine any of these matrices to get both, say invariance and discrimination, by setting S_N to a weighted sum of the tangent covariance matrix and the within class scatter. Alike, we can also add several tangent covariance matrices to impose invariances under more than just one transformation or add different matrices describing desired properties to compute S_I.

- *S. Mika and K.-R. Müller are with Fraunhofer FIRST, Kekuléstr. 7, 12489 Berlin, Germany. E-mail: {Sebastian.Mika, Klaus-Robert.Mueller}@first.fhg.de.*
- *K.-R. Müller is also with the University of Potsdam, Neues Palais 10, 14469 Potsdam, Germany.*
- *G. Rätsch and A. Smola are with the Australian National University, Canberra, 0200 ACT, Australia. E-mail: {Gunnar.Raetsch, Alex.Smola}@anu.edu.au.*
- *J. Weston and B. Schölkopf are with Max Planck Institut für biologische Kybernetik, Spemannstr. 38, 72076 Tübingen, Germany. E-mail: {Jason.Weston, Bernhard.Schoelkopf}@tuebingen.mpg.de.*

Manuscript received 20 July 2001; revised 18 July 2002; accepted 19 Aug. 2002.
Recommended for acceptance by B. Frey.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 114570.

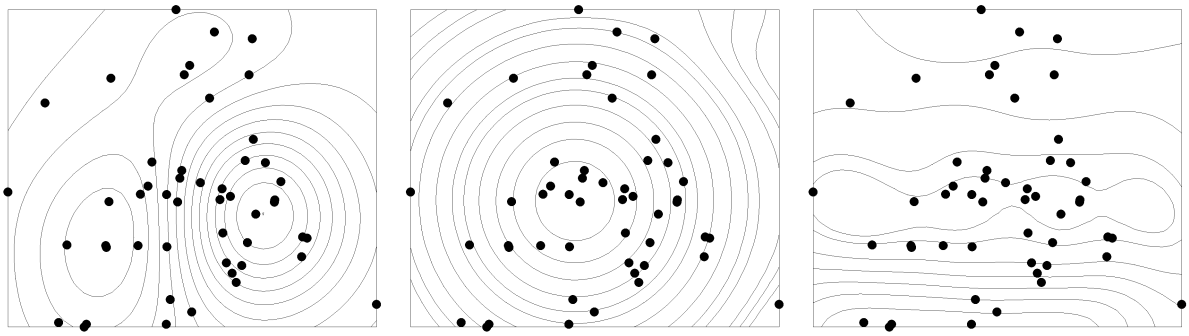

Fig. 1. Comparison of first features found by kernel PCA and oriented kernel PCA (see text); from left to right: KPCA, OKPCA with rotation, and OKPCA with translation invariance; all with Gaussian kernel. Although the data set does not contain any specific structure itself, the desired invariance can be imposed by using the appropriate tangent matrix T.

1.3 The Kernel Trick

In this paper, we generalize the above linear setting to a nonlinear one. In analogy to [8], [2], [9], [10], we first map the data via a nonlinear mapping Φ into some high, or even infinite dimensional feature space $\mathcal{F}$ and, then, optimize (1) in $\mathcal{F}$ (see also [11]). To avoid working with the mapped data explicitly (being impossible if $\mathcal{F}$ is of an infinite dimension), we introduce support vector kernel functions: we use the well-known *kernel trick* [12], [13], [7]. The kernel functions $\mathrm{k}(\mathbf{x},\mathbf{z})$ compute a dot product in a feature space $\mathcal{F}$: $\mathrm{k}(\mathbf{x},\mathbf{z}) = (\Phi(\mathbf{x})\cdot\Phi(\mathbf{z}))$. Formulating the algorithms in $\mathcal{F}$ using Φ only in dot products, we can replace any occurrence of a dot product by the kernel function k, which amounts to performing the same *linear* algorithm as before, but *implicitly* in a kernel feature space $\mathcal{F}$. Possible choices for k that have proven useful, e.g., in support vector machines [12], [13] or kernel PCA [2] are, e.g., Gaussian RBF, $\mathrm{k}(\mathbf{x},\mathbf{z}) = \exp(-\|\mathbf{x}-\mathbf{z}\|^2/c)$, or polynomial kernels, $\mathrm{k}(\mathbf{x},\mathbf{z}) = (\mathbf{x}\cdot\mathbf{z})^d$, for some positive constants $c \in \mathbb{R}$ and $d \in \mathbb{N}$, respectively. The latter have the interpretation of mapping the data to the space of all possible monomials of degree d [13]. For a choice of other kernel functions see, e.g., [13], [11] and the references therein.

The remainder of this article is organized as follows: Section 2 demonstrates in detail, how to formulate a linear Rayleigh coefficient such that one can obtain nonlinear generalizations. We exemplify this transformation for Fisher's discriminant leading to kernel Fisher discriminants (KFD) [8], [9], [10]—other variations are straightforward. Then, in Section 3, we introduce probabilistic interpretations of KFD leading to different regularization approaches. We discuss algorithmic issues in Section 4, concluding with the experiments in Section 5, and a discussion of our findings.

2 Kernelizing the Rayleigh Coefficient

We will now derive kernel Fisher discriminants as an example for the process of introducing kernels. We only consider discriminants for two classes, but our findings generalize easily to multiple classes (see [9], [10]).

2.1 Formulation Using Kernel Functions

Following the reasoning in Section 1.3, to optimize (1) in some kernel feature space $\mathcal{F}$, we need to find a formulation which uses only dot products of Φ-images. This can be done independently for numerator and denominator. The central observation is that, under some mild assumptions on S_I and S_N,[1] every solution of $\mathbf{w} \in \mathcal{F}$ can be written as an expansion in terms of mapped training data:

$$\mathbf{w} = \sum_{\mathbf{x}\in\mathcal{X}} \alpha_{\mathbf{x}} \Phi(\mathbf{x}), \tag{4}$$

where we use $\mathbf{x}$ to index the expansion coefficients α. The importance of this observation for *any* kernel-based learning technique is that, in general, there is no other way to express the solution $\mathbf{w} \in \mathcal{F}$, either because $\mathcal{F}$ is too high or infinite dimensional, or because we do not even know the actual feature space $\mathcal{F}$ connected to a certain kernel. The use of expansion (4) makes things tractable.[2] To see that this expansion is valid, consider symmetric operators S on the at most ℓ-dimensional subspace spanned by the $\Phi(\mathbf{x}_i)$ in $\mathcal{F}$, e.g., any matrix S, which is exclusively constructed from the $\Phi(\mathbf{x}_i)$ by means of linear operations. Furthermore, let $\mathbf{w} = \mathbf{v}_1 + \mathbf{v}_2$, where $\mathbf{v}_1 \in \mathrm{span}\{\Phi(\mathbf{x}_i) : i = 1,\ldots,\ell\}$ and $\mathbf{v}_2 \perp \mathrm{span}\{\Phi(\mathbf{x}_i) : i = 1,\ldots,\ell\}$. Then, for any such S,

$$\langle \mathbf{w}, S\mathbf{w}\rangle = \langle (\mathbf{v}_1+\mathbf{v}_2), S(\mathbf{v}_1+\mathbf{v}_2)\rangle = \langle \mathbf{v}_1, S\mathbf{v}_1\rangle,$$

using $S\mathbf{v}_2 = 0$ and $\langle \mathbf{v}, S\mathbf{v}\rangle = \langle S\mathbf{v}, \mathbf{v}\rangle$ for symmetric matrices. As $\mathbf{v}_1$ lies in the span of the $\Phi(\mathbf{x}_i)$ and, by construction, the operator S only operates on this subspace, $S\mathbf{v}_1$ lies in $\mathrm{span}\{\Phi(\mathbf{x}_i)\}$ as well. Thus, for any such quadratic form $\mathbf{w}^\top S\mathbf{w}$, it is sufficient to consider that part of $\mathbf{w}$, which lies in the span of the examples, i.e., there exists an expansion of the form (4) for $\mathbf{w}$, maximizing (1) (in $\mathcal{F}$) (see [15]).[3] Mulitplying either of the matrices proposed for S_1 and S_N from the left and right with the expansion (4), we can find a formulation using only dot products.

2.2 Derivation of Kernel Fisher Discriminants

Let Φ be the non-linear mapping to the feature space $\mathcal{F}$. To find the linear discriminant in $\mathcal{F}$ (which is then nonlinear in the input space) we need to maximize (1), but now with $\mathbf{w} \in \mathcal{F}$ and S_I and S_N being the corresponding matrices in $\mathcal{F}$, i.e.,

$$\begin{aligned} S_I &:= (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^\top, \\ S_N &:= \sum_{i=1,2}\sum_{\mathbf{x}\in\mathcal{X}_i} (\Phi(\mathbf{x}) - \mathbf{m}_i)(\Phi(\mathbf{x}) - \mathbf{m}_i)^\top \quad \text{and} \\ \mathbf{m}_i &:= \frac{1}{\ell_i}\sum_{\mathbf{z}\in\mathcal{X}_i} \Phi(\mathbf{z}). \end{aligned}$$

We define $K_{\mathbf{x}}$ as the column corresponding to the element $\mathbf{x}$ in the kernel matrix $(K)_{\mathbf{xz}} = \mathrm{k}(\mathbf{x},\mathbf{z})$, $\mathbf{x},\mathbf{z} \in \mathcal{X}$. To find a formulation using only dot-products or kernel functions $\mathrm{k}(\mathbf{x},\mathbf{z}) = (\Phi(\mathbf{x})\cdot\Phi(\mathbf{z}))$, respectively, we use the expansion (4). First, using the definition of $\mathbf{m}_i$ and (4), we get

1. These assumptions are detailed below.

2. In Boosting, one explicitly uses this mapping by employing different techniques to deal with the dimensionality of $\mathcal{F}$. See [14] for a detailed discussion.

3. Note that the tangent covariance matrix (3) is not constructed exclusively from the training examples in $\mathcal{X}$, but also from the transformations generated by $\mathcal{L}_t\mathbf{x}$. Thus, one would have to include all $\mathcal{L}_t\mathbf{x}$ into the expansion (4). In practice, however, it might suffice to restrict the solution to the elements in $\mathcal{X}$ (see also Section 4.1).

$$\begin{aligned}\mathbf{w}^\top \mathbf{m}_i &= \frac{1}{\ell_i}\sum_{\mathbf{x}\in\mathcal{X}}\sum_{\mathbf{z}\in\mathcal{X}_i}\alpha_{\mathbf{x}}(\Phi(\mathbf{x})\cdot\Phi(\mathbf{z}))\\ &= \frac{1}{\ell_i}\sum_{\mathbf{x}\in\mathcal{X}}\sum_{\mathbf{z}\in\mathcal{X}_i}\alpha_{\mathbf{x}}\mathrm{k}(\mathbf{x},\mathbf{z})\\ &= \alpha^\top\mu_i,\end{aligned} \tag{5}$$

where we replaced the dot products of Φ-images by the kernel function and defined $\mu_i := \frac{1}{\ell_i}\sum_{\mathbf{x}\in\mathcal{X}_i} K_x$. Now consider the numerator of (1): by definition of S_I and using (5), it can be rewritten as

$$\begin{aligned}\mathbf{w}^\top S_I\mathbf{w} &= \mathbf{w}^\top(\mathbf{m}_2-\mathbf{m}_1)(\mathbf{m}_2-\mathbf{m}_1)^\top\mathbf{w}\\ &= \alpha^\top(\mu_2-\mu_1)(\mu_2-\mu_1)^\top\alpha\\ &= \alpha^\top M\alpha,\quad M := (\mu_2-\mu_1)(\mu_2-\mu_1)^\top.\end{aligned} \tag{6}$$

Similarly, noting that $\sum_{x\in\mathcal{X}_i} K_{\mathbf{x}} = \ell_i\mu_i$ and using again (4), (5), and the definition of $\mathbf{m}_i$ we find for the denominator of (2):

$$\begin{aligned}\mathbf{w}^\top S_N\mathbf{w} &= \mathbf{w}^\top\left[\sum_{i=1,2}\sum_{\mathbf{x}\in\mathcal{X}_i}(\Phi(\mathbf{x})-\mathbf{m}_i)(\Phi(\mathbf{x})-\mathbf{m}_i)^\top\right]\mathbf{w}\\ &= \alpha^\top\left[\sum_{i=1,2}\sum_{\mathbf{x}\in\mathcal{X}_i}(K_{\mathbf{x}}-\mu_i)(K_{\mathbf{x}}-\mu_i)^\top\right]\alpha\\ &= \alpha^\top(K(I-\mathbf{v}_1\mathbf{v}_1^\top-\mathbf{v}_2\mathbf{v}_2^\top)K^\top)\alpha\\ &= \alpha^\top N\alpha\\ &\quad\text{with}\ \ N := KDK^\top\\ &\quad\text{and}\ \ D := I-\mathbf{v}_1\mathbf{v}_1^\top-\mathbf{v}_2\mathbf{v}_2^\top,\end{aligned} \tag{7}$$

where I is the identity matrix and $\mathbf{v}_j$ is the vector with element $(\mathbf{v}_j)_i = 1/\sqrt{\ell_j}$ if the example i belongs to class j and zero otherwise. Combining (6) and (7), we can find the Fisher discriminant in $\mathcal{F}$ by maximizing

$$J(\alpha) = \frac{\alpha^\top M\alpha}{\alpha^\top N\alpha}. \tag{8}$$

Algorithms using different matrices for S_I or S_N are easily obtained along the same lines. For instance, if we choose S_I as the plain covariance matrix (2) as in kernel PCA, we get as well $\mathbf{w}^\top S_I\mathbf{w} = \alpha^\top N\alpha$, but now N simplifies to $N = K(I-\mathbf{v}\mathbf{v}^\top)K^\top$ with $\mathbf{v}$ being the $1/\sqrt{\ell}$ vector.

To perform Fisher's discriminant in a kernel feature space, we still have to maximize a Rayleigh coefficient. With our reformulation, it is now a quotient in terms of expansion coefficients α, and not in terms of $\mathbf{w}\in\mathcal{F}$. The projection of a new example $\mathbf{z}$ onto $\mathbf{w}$ in $\mathcal{F}$ can be computed by $(\mathbf{w}\cdot\Phi(\mathbf{z})) = \sum_{x\in\mathcal{X}}\alpha_{\mathbf{x}}\mathrm{k}(\mathbf{x},\mathbf{z})$. If we are looking for more than one feature (e.g., in oriented kernel PCA) it is straightforward to check that their expansions are given by the subsequent generalized Eigenvectors of (8), again in complete analogy to the original input space algorithms.

However, one problem remains: Since already the matrix $D = I-\mathbf{v}_1\mathbf{v}_1^\top-\mathbf{v}_2\mathbf{v}_2^\top$ has only rank $\ell-2$, so has N, regardless of the rank of the kernel matrix K. Hence, there exist a vector α with $\alpha^\top N\alpha = 0$ and (8) is not well defined anymore. While, in practice, one could deal with that problem (e.g., minimize $1/J(\alpha)$ instead), this shows a crucial problem with Fisher's discriminants: If the number of dimensions is large compared to the number of examples, the problem becomes ill-posed. Especially, in the case of kernel algorithms, we effectively work in the space spanned by all ℓ mapped training examples $\Phi(\mathbf{x})$ which are, in practice, often linearly independent. Thus, we are estimating covariances in an ℓ-dimensionsal space from ℓ examples, which is, in general, ill-posed as well. In Section 3, we will discuss how to overcome these problems by using regularization.

2.3 KFD, Least Squares, and Quadratic Optimization

It is a well-known fact that Fisher's discriminant for two classes is equivalent to a least squares regression to the class labels (e.g., [3]). It is straight forward to show that the same holds true in a feature space $\mathcal{F}$. Thus, for the two class KFD problem, one can optimize the following:

$$\min_{\alpha,b,\xi}\quad \|\xi\|^2 + C\Omega(\alpha)\quad\text{subject to :}\quad K\alpha + \mathbf{1}b = \mathbf{y}-\xi. \tag{9}$$

Here, $\mathbf{1}$ is the vector of all ones, $\mathbf{y}$ the vector of all labels, and $\Omega(\alpha)$ is an optional regularization operator which we will discuss in Section 3. For a suitable choice of Ω, namely linear or quadratic, as made in the following, this amounts to a quadratic programming (QP) problem similar to, e.g., SVM. The offset b is introduced during the transition from the Rayleigh coefficient to the QP problem. It can be used as a threshold in a final classification. However, in practice, it is advisable to estimate a better threshold (see (15)). From a classification perspective, the QP has an appealing interpretation. The constraints ensure that the average class distance, projected onto the direction of discrimination, is two (for ±1 labels), while the intraclass variance is minimized, i.e., we maximize the *average margin*. Contrarily, the SVM approach [12] optimizes for a large *minimal margin*. Furthermore, it is apparent, that KFD bears strong connection to a least squares (kernel-) regression. In fact, (9) has been proposed in [16], as a "kernelized" version of ridge regression (simply using real valued $\mathbf{y}$s). Not surprising anymore, it turns out that KFD is equivalent to a technique proposed in [17], called *Least Squares Support Vector Machines*.

3 A Probabilistic Interpretation

Fisher's discriminant is known to be the Bayes optimal classifier for two normal distributions with equal covariance (i.e., KFD is optimal for two Gaussians in feature space). This allows to draw an interesting connection to the Relevance Vector Machine (RVM, [18]) and Gaussian processes [19], [7], [20]. Consider a regression onto the labels of the form $(\mathbf{w}\cdot\Phi(\mathbf{x}))+b$, where $\mathbf{w}$ is given by (4). Assuming a Gaussian noise model with variance σ, the likelihood in a latent variable model can be written as

$$p(\xi|\mathbf{y},\alpha,\sigma^2) \equiv \exp\left(-\frac{1}{2\sigma^2}\sum_i((\mathbf{w}\cdot\Phi(\mathbf{x}_i))+b-y_i)^2\right).$$

Assuming a prior $p(\alpha|\mathbf{C})$ over the weights with hyper-parameters $\mathbf{C}$ (here, $\mathbf{C}$ could be a vector of, e.g., length ℓ), then computing the posterior would yield the RVM. An advantage of the RVM approach is that all hyperparameters σ and $\mathbf{C}$ are estimated automatically. The drawback, however, is that one has to solve a difficult optimization problem prone to local minima. However, assuming the noise variance σ is known (i.e., dropping terms depending solely on σ) and taking the logarithm of the posterior $p(\xi|\mathbf{y},\alpha,\sigma^2)p(\alpha|\mathbf{C})$, yields

$$\min_{\alpha,b,\xi}\|\xi\|^2 - \log(p(\alpha|\mathbf{C})), \tag{10}$$

subject to the constraint in (9). Interpreting the prior as a regularization operator Ω, and introducing an appropriate common weighting factor C yields the KFD problem (9) (which has a global solution that can be relatively easily found). This probabilistic interpretation has some appealing properties.

Interpretation of Outputs. The probabilistic framework reflects the fact that the outputs produced by KFD can be interpreted as probabilities, thus making it possible to assign a confidence to the final classification. This is in contrast to, e.g., SVMs whose outputs cannot directly be seen as probabilities (see [21] and references therein).

Noise Models. In the above discussion, we assumed a Gaussian noise model and some yet unspecified prior that corresponds to a regularizer [22]. But one is not limited to Gaussian models. Assuming a Laplacian noise model we would get $\|\xi\|_1$ instead of $\|\xi\|_2^2$ in the objective of (10) or (9), respectively (see also, (14)). Other typical and useful choices are the ε-insensitive loss, Huber's robust

loss, or higher order polynomial loss functions, all inducing a density model (see [11], [7]).

Regularization. What we have not considered yet is the question of regularization. For instance, for KFD, a solution with zero within class variance (i.e., $\alpha^\top N\alpha = 0$) is very likely due to overfitting and corresponds in terms of (9) to a zero error regression to the labels. The way we regularize, the solution bears strong connections to the prior $p(\alpha|\mathbf{C})$ chosen in the probabilistic model developed above: e.g., choosing a zero-mean Gaussian as in RVM and assuming that this Gaussians' variance $\mathbf{C}$ is known and a multiple of the identity would lead to a regularizer of the form $\Omega(\alpha) = \|\alpha\|^2$, i.e., in (8) we would replace N with

$$N_C := N + CI \quad (C \geq 0). \tag{11}$$

Crucially, choosing a single, fixed variance parameter for all α, we do not achieve sparsity as in RVM anymore. Particularly interesting, is the alternative of a Laplacian prior on α, which would correspond to a l_1-norm penalty on the α's, i.e., $\Omega(\alpha) = \|\alpha\|_1$ and the optimization problem becomes

$$\min_{\alpha,b,\xi} \quad \|\xi\|^2 + C\|\alpha\|_1 \quad \text{subject to:} \quad K\alpha + \mathbf{1}b = \mathbf{y} - \xi, \tag{12}$$

leading to sparse solutions in the KFD.[4] We call this particular setting *sparse KFD* (SKFD) [5] (see also Laplacian Processes in [7]). Finally, one can use other regularization type additives, e.g., penalizing $\|\mathbf{w}\|^2 = \alpha^\top K\alpha$ in analogy to SVM, inducing a regularity dependent on the chosen kernel function [22].

4 Algorithms

Having successfully formulated a Rayleigh coefficient in feature space, we still have one major problem: while we could avoid working explicitly in the extremely high or infinite dimensional space $\mathcal{F}$, we are now facing a problem in ℓ variables, a number which in many practical applications would not allow to store or manipulate $\ell \times \ell$ matrices on a computer anymore. Furthermore, solving, e.g., an Eigenproblem or a QP of this size is very time consuming ($\mathcal{O}(\ell^3)$). In the following, we will propose several ways to deal with this general problem when optimizing the Rayleigh coefficient in feature space. The method proposed in Section 4.1 is applicable to any choice made for the matrices S_I and S_N, i.e., to any Rayleigh coefficient-based algorithm in feature space. The other methods do only apply to KFD. While the proposed schemes already allow to solve fairly large problems, we expect that further improvements in terms of memory requirements and speed are possible by applying ideas from, e.g., the literature on Gaussian processes (see [7]). Moreover, recently a SMO style algorithm as for SVM [23] has been proposed to solve the KFD problem [24].

4.1 Training on a Subset

To maximize (8) (or (9) for KFD), we need to solve an $\ell \times \ell$ Eigen- or mathematical programming problem, which might be intractable for large ℓ. As the solutions are not sparse, one cannot directly use techniques like chunking as it has been done for support vector machines (see [7]). One possible solution, which is applicable to any choice of matrices S_I and S_N, is to restrict the feature extractor $\mathbf{w}$ to lie in a subspace, i.e., instead of expanding $\mathbf{w}$ by (4), we write

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \Phi(\mathbf{z}_i), \tag{13}$$

4. A reason for the induced sparseness is the fact that vectors far from the coordinate axes are "larger" with respect to the l_1-norm than with respect to l_p-norms ($p > 1$). Consider the vectors $(1, 0)$ and $(1/\sqrt{2}, 1/\sqrt{2})$. For the two norm, $\|(1,0)\|_2 = \|(1/\sqrt{2}, 1/\sqrt{2})\|_2 = 1$, but for the l_1-norm, $1 = \|(1,0)\|_1 < \|(1/\sqrt{2}, 1/\sqrt{2})\|_2 = \sqrt{2}$. Using an l_1 regularizer, the optimal solution is always a vertex solution (or can be expressed as such) and tends to be very sparse.

with $m \ll \ell$ (analogous to the reduced set method of SVM, see [25]). The examples $\mathbf{z}_i$, $i = 1, \ldots, m$, could either be a subset of the training examples $\mathcal{X}$ or, e.g., be estimated by some clustering algorithm. The derivation of (8) does not change, only K is now $m \times \ell$ and we end up with $m \times m$ matrices for N and M.

4.2 A Sparse, Greedy Approach

Another choice would be to greedily construct a solution to (8) by forward selection. Starting with all α's equal to zero (i.e., an empty expansion), one iteratively selects new examples to add to the expansion such that the gain in the objective is maximal. In practice, one does not need to add all examples but could stop if a good approximate solution is reached. Clearly, such an approach is only sensible if there is an *efficient* way of choosing an optimal or close to optimal example in each iteration. For KFD, this is possible and has been described in detail in [21], [7].

4.3 Linear Optimization

Finally, if we optimize (9) using an l_1-norm regularizer (i.e., we are doing sparse KFD (12)), and furthermore, replace the loss function on ξ by an l_1-norm as well, we end up with a linear program:

$$\min_{\alpha,b,\xi} \quad \|\xi\|_1 + C\|\alpha\|_1 \quad \text{subject to:} \quad K\alpha + \mathbf{1}b = \mathbf{y} - \xi. \tag{14}$$

This is in analogy to linear programming machines as they emerged in the context of SVM [26]. Such a linear program will be very sparse in α and can be optimized extremely efficiently using a technique called column generation (e.g., [27] and references therein). We call this approach *linear sparse KFD* (LSKFD) [5].

5 Experiments

Here, we present some experimental results to give a proof of concept for the proposed oriented kernel PCA and the incorporation of invariances into KFD. Furthermore, we show that KFD and some of its variants are capable of producing state-of-the-art classification results.[5]

5.1 Using Prior Knowledge

A toy example (Fig. 1) shows a comparison of kernel PCA and oriented kernel PCA, where we used the full covariance (2) as S_I and the tangent covariance (3) of 1) rotated examples and 2) examples translated along the x-axis as noise matrix S_N. The toy example shows how imposing the desired invariance yields meaningful, invariant features.

In another experiment, we incorporated prior knowledge in KFD using the USPS database of handwritten digits, which consists of $7,291$ training and $2,007$ test patterns, each $D = 256$ dimensional gray-scale images of the digits $0 \ldots 9$. We chose the regularized within-class scatter (11) ($C = 10^{-3}$, which is the optimal value found by cross validation for the problem without invariances) as S_N and added to it a multiple λ of the tangent covariance (3). As invariance transformations, we have chosen horizontal and vertical translation, rotation, and line thickening (see [7]), simply averaging the matrices corresponding to each transformation. The feature was extracted by using the restricted expansion (13), where the examples $\mathbf{x}_i$ were chosen to be the first $3,000$ training examples. We have chosen a Gaussian kernel of width $0.3 \cdot D$, which is optimal for SVMs [7]. For each class, we computed the KFD, which classifies one class against the rest. Then, we determine the 10-class error by the winner-takes-all scheme choosing the threshold such that the distance of each example to the decision boundary (i.e., the margin) is maximal, while still allowing for some errors, i.e., we solved the following linear, simple to solve problem:

$$\max_{\rho,b} \rho - C\sum_{i=1}^{\ell} xi_i \tag{15}$$

5. Basic implementations and more detailed results will be made available on http://www.first.fraunhofer.de/~mika.

TABLE 1
Comparison between KFD, Sparse KFD (SKFD), Sparse KFD with Linear Loss on ξ (LSKFD), and SVMs (Taken from [15])

	SVM		KFD	SKFD		LSKFD	
Banana	11.5±0.07	(78%)	*10.8±0.05*	11.2±0.48	(86%)	**10.6±0.04**	(92%)
B.Cancer	26.0±0.47	(42%)	*25.8±0.46*	**25.2±0.44**	(88%)	*25.8±0.47*	(88%)
Diabetes	23.5±0.17	(57%)	*23.2±0.16*	**23.1±0.18**	(97%)	23.6±0.18	(97%)
German	**23.6±0.21**	(58%)	*23.7±0.22*	**23.6±0.23**	(96%)	24.1±0.23	(98%)
Heart	**16.0±0.33**	(51%)	*16.1±0.34*	16.4±0.31	(88%)	**16.0±0.36**	(96%)
Ringnorm	1.7±0.01	(62%)	**1.5±0.01**	*1.6±0.01*	(85%)	**1.5±0.01**	(94%)
F.Sonar	**32.4±0.18**	(9%)	*33.2±0.17*	33.4±0.17	(67%)	34.4±0.23	(99%)
Thyroid	4.8±0.22	(79%)	**4.2±0.21**	*4.3±0.18*	(88%)	4.7±0.22	(89%)
Titanic	**22.4±0.10**	(10%)	23.2±0.20	22.6±0.17	(8%)	*22.5±0.20*	(95%)
Waveform	**9.9±0.04**	(60%)	**9.9±0.04**	*10.1±0.04*	(81%)	10.2±0.04	(96%)

All experiments were carried out with RBF-kernels $\exp(-\|\mathbf{x}-\mathbf{z}\|^2/c)$. Best result in bold face, second best in italics. The numbers in parentheses denote the fraction of expansions coefficients, which were zero (for KFD all coefficients are nonzero).

subject to:

$$y_i(\zeta_i + b) \geq \rho - \xi_i \text{ and } \rho, \xi_i \geq 0 \quad \forall i = 1, \ldots, \ell,$$

where $\zeta_i = (\mathbf{w} \cdot \mathbf{x}_i)$ is the projection of the ith training example onto the direction of discrimination $\mathbf{w}$ and C trades of the size of ρ against the constraint violations ξ_i. Without invariances, i.e., $\lambda = 0$, we achieved a test error of 3.7 percent, slightly better than a plain SVM with the same kernel (4.2 percent) [7]. For $\lambda = 10^{-3}$, using the averaged tangent covariance matrices as described above, led to a very slight improvement to 3.6 percent. The performance did not significantly improve, which can be attributed to the fact that we used the same expansion coefficients in both cases. The tangent covariance matrix, however, lives in a larger subspace, and indeed, a subsequent experiment where we used $3,000$ vectors, which were obtained by k-means clustering from a larger data set, including the original training examples and $12,000$ virtual examples generated by appropriate invariance transformations (as described before), led to an error rate of 3.1 percent. We conjecture that, with the development of optimization techniques that allow to use a larger number of expansion coefficients, this result can be improved by expanding into all samples, the original training examples and the virtual examples.

5.2 Kernel Fisher Discriminant

To evaluate the performance of our KFD approaches, we performed an extensive comparison to SVMs on the IDA benchmark repository.[6] The results in Table 1 show the average test error and the standard deviation of the averages' estimation, over 100 runs with different realizations of the data sets. To estimate the necessary parameters, we ran 5-fold cross validation on the first five realizations of the training sets and took the model parameters to be the median over the five estimates (see [11] for details of the setup). Finally, to do classification, we estimated a threshold b using (15).[7] From Table 1, we see that both SVM and the KFD variants, on average, perform equally well. Noteworthy is the significantly higher degree of sparsity for SKFD and particularly for LSKFD than observed for SVMs, making, e.g., predictions much faster.

6. All data sets used in the experiments can be obtained via http://mlg.anu.edu.au/~raetsch. Thanks to M. Zwitter and M. Soklic for providing the `breast cancer` data. Data originally obtained from [28], [29].

7. Solving a program of the form (9) as in SKFD and LSKFD already yields a threshold; but also there we estimated an improved threshold by using (15).

6 Conclusion

In the task of learning from data, it is, to a certain extent, equivalent to have prior knowledge about, e.g., invariances or about specific sources of noise. In the case of feature extraction, we seek features, which are all sufficiently "noise"-invariant while still *describing* "interesting" structure. For classification, we compute *discriminating* features that are—at the same time—invariant with respect to certain invariance transformations. Oriented PCA and Fisher's discriminant are examples of two such linear techniques, both optimizing a Rayleigh coefficient (1). Since linear methods are often too restricted, we used support vector kernel functions to obtain *nonlinear* versions of these *linear* algorithms, namely, oriented kernel PCA and kernel Fisher discriminant analysis. In our experiments, we gave a proof of concept for the oriented kernel PCA and showed that the kernel Fisher discriminant algorithm is competitive to other state-of-the-art algorithms yielding sparser solutions.

Future research will focus 1) on further improvements on the algorithmic complexity of our algorithms, which in the case of KFD is lower than the one of RVMs, but so far larger than the one of the SVM algorithm, and 2) on further connections between KFD and support vector machines. Moreover, we are interested in the analysis of the generalization performance of such algorithms, e.g., along the lines of [30].

Acknowledgments

This work was partially supported by the DFG (JA 379/9-2, MU 987/1-1, AS 62/1-1), and EU BLISS (IST-1999-14190).

References

[1] K.I. Diamantaras and S.Y. Kung, *Principal Component Neural Networks*. New York: Wiley, 1996.
[2] B. Schölkopf, A.J. Smola, and K.-R. Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation*, vol. 10, pp. 1299-1319, 1998.
[3] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.
[4] R.A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, vol. 7, pp. 179-188, 1936.
[5] S. Mika, G. Rätsch, and K.-R. Müller, "A Mathematical Programming Approach to the Kernel Fisher Algorithm," *Proc. Conf. Neural Information Processing Systems*, T.K. Leen, T.G. Dietterich, and V. Tresp, eds., vol. 13, pp. 591-597, 2001.
[6] P.Y. Simard, Y.A. LeCun, J.S. Denker, and B. Victorri, "Transformation Invariance in Pattern Recognition—Tangent Distance and Tangent Propagation," *Neural Networks: Tricks of the Trade*, Springer, G. Orr and K.-R. Müller, eds., vol. 1524, pp. 239-274, 1998.

[7] B. Schölkopf and A.J. Smola, *Learning with Kernels.* Mass.: MIT Press, 2002.
[8] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, "Fisher Discriminant Analysis with Kernels," *Neural Networks for Signal Processing IX,* Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, eds., pp. 41-48, 1999.
[9] V. Roth and V. Steinhage, "Nonlinear Discriminant Analysis Using Kernel Functions," *Proc. Conf. Advances in Neural Information Processing Systems,* S.A. Solla, T.K. Leen, and K.-R. Müller, eds., vol. 12, pp. 568-574, 2000.
[10] G. Baudat and F. Anouar, "Generalized Discriminant Analysis Using a Kernel Approach," *Neural Computation,* vol. 12, no. 10, pp. 2385-2404, 2000.
[11] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An Introduction to Kernel-Based Learning Algorithms," *IEEE Trans. Neural Networks,* vol. 12, no. 2, pp. 181-201, 2001.
[12] B. Boser, I. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," *Proc. COLT,* D. Haussler, ed., pp. 144-152, 1992.
[13] V.N. Vapnik, *Statistical Learning Theory.* New York: Wiley, 1998.
[14] G. Rätsch, S. Mika, B. Schölkopf, and K.-R. Müller, "Constructing Boosting Algorithms from SVMs: An Application to One-Class Classification," *IEEE Trans. Pattern and Machine Analysis,* vol. 24, no. 9, pp. 1184-1199, Sept. 2002. earlier version is GMD Technical Report no. 119, 2000.
[15] B. Schölkopf, R. Herbrich, and A.J. Smola, "A Generalized Representer Theorem," *Proc. COLT/EuroCOLT,* Springer, D.P. Helmbold and R.C. Williamson, eds., pp. 416-426, 2001.
[16] C. Saunders, A. Gammermann, and V. Vovk, "Ridge Regression Learning Algorithm in Dual Variables," *Proc. 15th Int'l Conf. Machine Learning,* pp. 515-521, 1998.
[17] J.A.K. Suykens and J. Vanderwalle, "Least Squares Support Vector Machine Classifiers," *Neural Processing Letters,* vol. 9, no. 3, pp. 293-300, 1999.
[18] M.E. Tipping, "The Relevance Vector Machine," *Proc. Conf. Advances in Neural Information Processing Systems,* S.A. Solla, T.K. Leen, and K.-R. Müller, eds., vol. 12, pp. 652-658, 2000.
[19] C.M. Bishop, *Neural Networks for Pattern Recognition.* Oxford Univ. Press, 1995.
[20] T.V. Gestel, J.A.K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vanderwalle, "Bayesian Framework for Least Squares Support Vector Machine Classifiers, Gaussian Processs and Kernel Fisher Discriminant Analysis," technical report, Katholieke Universiteit Leuven, Aug. 2001.
[21] S. Mika, A.J. Smola, and B. Schölkopf, "An Improved Training Algorithm for Kernel Fisher Discriminants," *Proc. AISTATS,* T. Jaakkola and T. Richardson, eds., pp. 98-104, 2001.
[22] A.J. Smola, B. Schölkopf, and K.-R. Müller, "The Connection between Regularization Operators and Support Vector Kernels," *Neural Networks,* vol. 11, pp. 637-649, 1998.
[23] J. Platt, "Fast Training of Support Vector Machines Using Sequential Minimal Optimization," *Advances in Kernel Methods—Support Vector Learning,* Mass.: MIT Press, B. Schölkopf, C.J.C. Burges, and A.J. Smola, eds., pp. 185-208, 1999.
[24] S.S. Keerthi and S.K. Shevade, "SMO Algorithm for Least Squares SVM Formulations," Technical Report CD-02-08, Nat'l Univ. of Singapore, 2002.
[25] B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A.J. Smola, "Input Space vs. Feature Space in Kernel-Based Methods," *IEEE Trans. Neural Networks,* vol. 10, no. 5, pp. 1000-1017, Sept. 1999.
[26] T. Graepel, R. Herbrich, B. Schölkopf, A.J. Smola, P.L. Bartlett, K.-R. Müller, K. Obermayer, and R.C. Williamson, "Classification on Proximity Data with LP-Machines," *Proc. ICANN '99,* D. Willshaw and A. Murray, eds., vol. 1, pp. 304-309, 1999.
[27] G. Rätsch, A. Demiriz, and K. Bennett, "Sparse Regression Ensembles in Infinite and Finite Hypothesis Spaces," *Machine Learning,* vol. 48, no. 1-3, pp. 193-221, 2002.
[28] C.L. Blake and C.J. Merz, "UCI Repository of Machine Learning Databases," http://www.ics.uci.edu/~mlearn/MLRepository.html, a huge collection of artificial and real-world data sets, 1998.
[29] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/statlog/, benchmark repository used for the STATLOG competition. 2002.
[30] O. Bousquet and A. Elisseeff, "Stability and Generalization," *J. Machine Learning Research,* vol. 2, pp. 499-526, Mar. 2002.

Automatic Textual Document Categorization Based on Generalized Instance Sets and a Metamodel

Wai Lam, *Member*, *IEEE*, and Yiqiu Han

Abstract—We propose a new approach to text categorization known as generalized instance set (GIS) algorithm under the framework of generalized instance patterns. Our GIS algorithm unifies the strengths of k-NN and linear classifiers and adapts to characteristics of text categorization problems. It focuses on refining the original instances and constructs a set of generalized instances. We also propose a metamodel framework based on category feature characteristics. It has a metalearning phase which discovers a relationship between category feature characteristics and each component algorithm. Extensive experiments have been conducted on two large-scale document corpora for both GIS and the metamodel. The results demonstrate that both approaches generally achieve promising text categorization performance.

Index Terms—Text classification, instance-based learning, metamodel learning.

1 Introduction

THE aim of document categorization is to assign a number of appropriate categories to a textual document based on the content. This categorization process has many applications such as document routing, dissemination, or filtering [1]. A large amount of human resources are required to carry out such categorization task manually. The goal of automatic text categorization is to learn a classification scheme from training examples so that the scheme can be used to categorize unseen textual documents.

Text categorization problems possess several characteristics different from other pattern recognition problems [4], [3]. First, text categorization problems normally involve an extremely high dimensional space (e.g., exceed 30,000). The features usually represent words/terms derived from the textual content of documents. Second, each document contains only a small number of features despite the high dimensional space leading to a very sparse data representation. Third, the number of potential relevant features is very large, but only a few occur in a particular document. Many features may simultaneously involve in several categories. Finally, in general, the overlap between features in documents is quite small.

We explore common properties in two families of text categorization techniques, namely, the k-nearest-neighbor (k-NN) algorithm and linear classifiers. We propose a new approach known as the generalized instance set (GIS) algorithm under the framework of *generalized instance* patterns [6]. Our GIS algorithm unifies the strengths of k-NN and linear classifiers and adapts to characteristics of text categorization problems. It focuses on refining the original instance and constructs a set of generalized instances. We also propose a metamodel framework based on *category feature characteristics*. Category feature characteristics, derived from the training document set, capture some inherent properties of a particular category. Our metamodel approach has a metalearning phase which discovers a relationship between category feature characteristics and each component algorithm.

• *The authors are with the Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong. E-mail: {wlam, yqhan}@se.cuhk.edu.hk.*

Manuscript received 24 Aug. 2001; revised 4 Mar. 2002; accepted 31 May 2002.
Recommended for acceptance by V. Govindaraju.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 114852.