An Improved Training Algorithm for Kernel Fisher Discriminants

Sebastian Mika[⋆‡], Alexander Smola[⋆†],

Bernhard Schölkopf[§]

[⋆] Work done while SM and AS were visiting MSR Cambridge

[‡] GMD FIRST.IDA, Kekuléstr. 7, 12489 Berlin, Germany

[†] Australian National University, Canberra, 0200 ACT, Australia

[§] Microsoft Research, 1 Guildhall Street, Cambridge CB2 3NH, UK

mika@first.gmd.de, alex.smola@anu.edu.au, bsc@scientist.com

5 November 2000

Technical Report
MSR-TR-2000-77

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

To be presented at AISTATS 2001

Abstract

We present a fast training algorithm for the kernel Fisher discriminant classifier. It uses a greedy approximation technique and has an empirical scaling behavior which improves upon the state of the art by more than an order of magnitude, thus rendering the kernel Fisher algorithm a viable option also for large datasets.

1 Introduction

Kernel Fisher Discriminant (KFD) [8, 9] is a nonlinear generalization of Fisher's Discriminant [2, 4]. The nonlinearity is introduced by the use of kernel functions [6], in analogy to Support Vector Machines (SVMs) [1], Kernel PCA [10] and various other techniques.
On a large number of benchmarks, KFD has shown classification accuracies on a par with SVMs. In addition, unlike SVMs, the outputs of KFD lend themselves to a probabilistic interpretation: empirically, the distributions of the two classes projected onto the Fisher direction of discrimination can be approximated very well by Gaussians, which allows the estimation of conditional class probabilities. In this sense, KFD can be considered a "probabilistic variant" of SVMs (cf. also [13]). Unfortunately, so far, there has been no efficient algorithm for KFD — all the known algorithms effectively scaled like $O(\ell^3)$, where ℓ is the sample size. In the current paper, we propose a much more efficient algorithm, utilizing sparse greedy approximation techniques [12, 11].

2 The Kernel Fisher Discriminant Revisited

For some set $\mathcal{X}$, let $\{\mathbf{x}_i \in \mathcal{X} | i = 1, \ldots, \ell\}$ be our training sample and $\mathbf{y} \in \{-1, 1\}^\ell$ be the vector of corresponding class labels. Furthermore define $\mathbf{1} \in \mathbb{R}^\ell$ as the vector of all ones, and let $\mathbf{1}_+, \mathbf{1}_- \in \mathbb{R}^\ell$ be the positive and negative parts of $\mathbf{1}$, i.e. $\mathbf{1}_+ = \max(\mathbf{y}, 0)$, $\mathbf{1}_- = \max(-\mathbf{y}, 0)$. In the linear case, it is known that Fisher's discriminant is computed by maximizing the coefficient

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top S_B \mathbf{w}}{\mathbf{w}^\top S_W \mathbf{w}} \tag{1}$$

of between and within class variance, i.e. $S_B = (\boldsymbol{m}_+ - \boldsymbol{m}_-)(\boldsymbol{m}_+ - \boldsymbol{m}_-)^\top$ and $S_W = \sum_{i,y_i=1} (\mathbf{x}_i - \boldsymbol{m}_+)(\mathbf{x}_i - \boldsymbol{m}_+)^\top + \sum_{i,y_i=-1} (\mathbf{x}_i - \boldsymbol{m}_-)(\mathbf{x}_i - \boldsymbol{m}_-)^\top$, where $\boldsymbol{m}_\pm$ denotes the sample mean for class ± 1.
It can be shown that this approach results in the optimal (in this case linear) decision for two Gaussian distributions with equal covariance structure. In spite of the fact that Fisher's discriminant often yields useful results even when this assumption is violated, its basic limitation is that the discriminating direction is linear. To overcome this limitation, [8] proposed to use the same approach as in Support Vector Machines [1] or Kernel PCA [10]: kernel functions. In a nutshell, the idea is to first apply a nonlinear mapping $\Phi : \mathcal{X} \to \mathcal{F}$ to the data

and then to perform the same linear algorithm on the mapped data. If $\mathcal{F}$ is sufficiently rich, this increases the chance of finding a good linear, separating direction in the mapped space. This linear direction in the *feature* space $\mathcal{F}$ then implicitly yields a nonlinear direction in the input space. To avoid having to work in $\mathcal{F}$ explicitly, we use the *kernel trick*, i.e. we choose a feature space whose dot product can be computed by a kernel $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$,

$$\mathrm{k}(\mathbf{x}_i, \mathbf{x}_j) = (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)),$$

and formulate everything in terms of dot products in $\mathcal{F}$, i.e. in kernels on $\mathcal{X} \times \mathcal{X}$.

3 Fisher's Discriminant in Feature Space

To solve Fisher's problem in a kernel feature space $\mathcal{F}$ one needs a formulation which makes use of the training samples only in terms of dot products. One can prove [8] that the solution $\mathbf{w} \in \mathcal{F}$ of (1) can be expanded as

$$\mathbf{w} = \sum\nolimits_{i=1}^{\ell} \alpha_i \Phi(\mathbf{x}_i), \qquad \alpha_i \in \mathbb{R}. \tag{2}$$

It is straightforward to find an expression similar to (1) for the coefficients $\boldsymbol{\alpha}$ [8]. However, here we will use a different formulation for finding $\boldsymbol{\alpha}$, building on the following observation: the goal of Fisher's discriminant is to find a one dimensional projection on which the class means are far apart while the within class variance is small.
In [7] it was shown that KFD can be cast in a slightly generalized form as the following convex, quadratic optimization problem:

$$\min_{\boldsymbol{\alpha}, b, \boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{\xi}\|^2 + \frac{C}{2}\,\mathrm{P}(\boldsymbol{\alpha}) \tag{3}$$

subject to:

$$K\boldsymbol{\alpha} + \mathbf{1}b \quad = \quad \mathbf{y} + \boldsymbol{\xi} \tag{3a}$$

$$\mathbf{1}_+^\top \boldsymbol{\xi} = 0, \qquad \mathbf{1}_-^\top \boldsymbol{\xi} = 0. \tag{3b}$$

Here, C is a regularization constant, and P a regularization functional [8, 3] which we assume to be quadratic in the following, e.g. $\mathrm{P}(\boldsymbol{\alpha}) = \|\boldsymbol{\alpha}\|^2$ or $\mathrm{P}(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top K \boldsymbol{\alpha}$. The projection of a test point onto the discriminant is computed by $(\mathbf{w} \cdot \Phi(\mathbf{x})) = \sum_i \alpha_i \,\mathrm{k}(\mathbf{x}_i, \mathbf{x})$. The program essentially states that the output for each training sample should be close to its label, where we penalize the squared error of the deviation (constraint (3a)), and that the average deviation from the label should be zero, separately for each class (constraints (3b)).
However, for large data sets, solving (3) is expensive in terms of time and memory. Contrary to SVMs, the solutions are not sparse and deriving efficient decomposition techniques for the programming problem is difficult. In [7], it was proposed to use a ℓ_1 regularizer in (3) as an approximation to a ℓ_0 regularizer which would just count the number of non zero elements in $\boldsymbol{\alpha}$. While this solved

the problem of non-sparsity and was a promising candidate to use chunking techniques there was no really efficient algorithm yet.
Using a ℓ_0 regularizer or to add a nonlinear constraint of the form: *Find a solution* $\boldsymbol{\alpha}$ *with at most* m *non–zero elements*, would in principle be optimal. Unfortunately, such an approach is impossible to deal with analytically. Finding the true optimal solution would require to search the space of all possible solutions which make use of m possible α_i, i.e. one had to solve $\binom{\ell}{m}$ problems. We will presently derive an algorithm which might be viewed as a greedy approximation to such a solution. Along the lines of [11, 12], we will iteratively approximate the solution to (3) with as few non-zero α_i as possible.

4 The Algorithm

To proceed, let us rewrite (3). Define

$$\mathbf{a} = \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} \ell_+ - \ell_- \\ K^\top \mathbf{y} \end{bmatrix} \quad \mathbf{A}_\pm = \begin{bmatrix} \ell_\pm \\ K^\top \mathbf{1}_\pm \end{bmatrix} \quad H = \begin{bmatrix} \ell & \mathbf{1}^\top K \\ K^\top \mathbf{1} & K^\top K + C\,\mathrm{P} \end{bmatrix}. \tag{4}$$

Here, $\ell_\pm$ denotes the number of samples in class ± 1. Then the problem (3) can equivalently be rewritten as:

$$\min_{\mathbf{a}} \quad \frac{1}{2}\mathbf{a}^\top H \mathbf{a} - \mathbf{c}^\top \mathbf{a} + \frac{\ell}{2} \tag{5}$$

subject to:

$$\begin{aligned} \mathbf{A}_+^\top \mathbf{a} - \ell_+ &= 0 && \text{(5a)} \\ \mathbf{A}_-^\top \mathbf{a} + \ell_- &= 0. && \text{(5b)} \end{aligned}$$

Forming the Lagrangian of (5) with multipliers $\lambda_\pm$

$$L(\mathbf{a}, \lambda_+, \lambda_-) = \frac{1}{2}\mathbf{a}^\top H \mathbf{a} - \mathbf{c}^\top \mathbf{a} + \lambda_+(\mathbf{A}_+^\top \mathbf{a} - \ell_+) + \lambda_-(\mathbf{A}_-^\top \mathbf{a} + \ell_-) + \frac{\ell}{2}, \tag{6}$$

and taking derivatives with respect to the primal variables $\mathbf{a}$ one obtains the dual

$$\max_{\mathbf{a}, \lambda_+, \lambda_-} \quad -\frac{1}{2}\mathbf{a}^\top H \mathbf{a} - \lambda_+ \ell_+ + \lambda_- \ell_- + \frac{\ell}{2} \tag{7}$$

subject to:

$$H\mathbf{a} - \mathbf{c} + (\lambda_+ \mathbf{A}_+ + \lambda_- \mathbf{A}_-) = 0. \tag{8a}$$

Now we use the dual constraint (8a) to solve for $\mathbf{a}$, i.e.

$$\mathbf{a} = H^{-1}\left(\mathbf{c} - (\lambda_+ \mathbf{A}_+ + \lambda_- \mathbf{A}_-)\right). \tag{8}$$

This equation is well defined if H has full rank (see (4)). If not we can still perform this step as we will approximate H^{-1} instead of computing it directly.

Resubstituting (8) into the dual problem (which has no constraints left) yields the following problem in the two variables λ_+ and λ_-:

$$\begin{aligned}\max_{\lambda_+,\lambda_-} -\frac{1}{2}\begin{bmatrix}\lambda_+\\ \lambda_-\end{bmatrix}^\top \begin{bmatrix}\mathbf{A}_+^\top H^{-1}\mathbf{A}_+ & \mathbf{A}_+^\top H^{-1}\mathbf{A}_-\\ \mathbf{A}_-^\top H^{-1}\mathbf{A}_+ & \mathbf{A}_-^\top H^{-1}\mathbf{A}_-\end{bmatrix}\begin{bmatrix}\lambda_+\\ \lambda_-\end{bmatrix}\\ + \begin{bmatrix}-\ell_+ + \mathbf{c}^\top H^{-1}\mathbf{A}_+\\ \ell_- + \mathbf{c}^\top H^{-1}\mathbf{A}_-\end{bmatrix}\begin{bmatrix}\lambda_+\\ \lambda_-\end{bmatrix} - \frac{1}{2}\mathbf{c}^\top H^{-1}\mathbf{c} + \frac{\ell}{2}.\end{aligned} \tag{9}$$

This problem can be solved analytically, yielding values for λ_+ and λ_- which substituted into (8) yield values for $\mathbf{a}$ or $\boldsymbol{\alpha}$ and b, respectively.

A Sparse Greedy Approximation. Of course, this problem is no easier to solve than the original one nor does it yield a sparse solution: H^{-1} is an $(\ell+1)\times(\ell+1)$ matrix and for large datasets its inversion is not feasible, neither in terms of time nor memory cost. Now, the idea is to use the following, greedy approximation scheme (cf. [11]). Instead of trying to find a full set of ℓ α_i's for the solution (2), we approximate the optimal solution by a shorter expansion containing only $m \ll \ell$ terms.

Starting with an empty solution $m = 0$, select in each iteration a new sample $\mathbf{x}_i$ (or an index i) and resolve the problem for the expansion (2) containing this new index and all previously picked indices; stop as soon as a suitable criterion is satisfied. This approach would still be infeasible in terms of computational cost if we had to solve the full quadratic program (5) anew in each iteration or invert H in (8) and (9). But with the derivation made before it is possible to find a close approximation to the optimal solution in each iteration at a cost of $\mathcal{O}(\kappa\ell m^2)$ where κ is a user defined value (see below).
Writing down the quadratic program (3) for KFD when the expansion for the solution is restricted to an m element subset $\mathcal{I} \subset [\ell]$

$$\mathbf{w}_\mathcal{I} = \sum_{i\in\mathcal{I}} \alpha_i \Phi(\mathbf{x}_i) \tag{10}$$

of the training patterns amounts to replacing the $\ell\times\ell$ matrix K by the $\ell\times m$ matrix K^m, where $K^m_{ij} = \mathrm{k}(\mathbf{x}_i,\mathbf{x}_j)$, $i = 1,\ldots,\ell$ and $j \in \mathcal{I}$. Analogously, we can derive the formulation (5) using the matrix K^m in (4). The problem is of order $m\times m$ now. Assume we already know the optimal solution (and inverse of H) using m kernel-functions. Then H^{-1} for $m+1$ samples can be obtained by a rank one update of the previous H^{-1} using only m basis functions: The following Lemma (e.g. [5]) tells us how to obtain the new H^{-1}.

Lemma 1 (Sherman–Woodbury–Formula). *The inverse of a symmetric, positive matrix can be computed as:*

$$\begin{bmatrix}H & B\\ B^\top & C\end{bmatrix}^{-1} = \begin{bmatrix}H^{-1} + (H^{-1}B)\gamma(H^{-1}B)^\top & -\gamma(H^{-1}B)\\ -(\gamma(H^{-1}B))^\top & \gamma\end{bmatrix},$$

where $\gamma = (C - B^\top H^{-1}B)^{-1}$.

Note that for our case B is a vector and C a scalar. This is an operation of cost $\mathcal{O}(m^2)$ as we already know the inverse of the smaller system. The last major problem is to pick an index i in each iteration. Ideally one would choose the i for which we get the biggest decrease in the primal-objective (or equivalently as they are identical for the optimal coefficients $\mathbf{a}$, the dual-objective (7)). But this would mean that we have to compute the update H^{-1} for all $\ell - m$ indices which are unused so far — again, too expensive. One possible solution lies in a second approximation. Instead of choosing the *best* possible index it is usually sufficient to find an index for which with high probability we achieve something close to the optimal achievement. It turns out [12] that it can be enough to consider 59 randomly chosen indices from the remaining ones:

Lemma 2 (Maximum of Random Variables). *Denote by $\rho_1, \ldots, \rho_m$ identically distributed independent random variables with a common cumulative distribution function F. Then the cumulative distribution function of $\rho = \max_{i \in [m]} \rho_i$ is F^m.*

This means that for the uniform distribution on $[0, 1]$ $\max_{i \in [m]} \rho_i$ is distributed according to ρ^m. Thus, to obtain an estimate that is with probability 0.95 among the best 0.05 of all estimates, a random sample of size $\kappa := (\log 0.05 / \log 0.95) = 59$ is enough.

Termination. Still open is the question when to stop. If one wanted to compute the full solution this approach would not be very efficient as it would take $\mathcal{O}(\kappa \ell^3)$ which is worse than the original problem. A principled stopping rule would be to measure the distance of $\mathbf{w}_{\mathcal{I}}$ to the solution of the full problem and then to stop when this falls below a certain threshold. Unfortunately the full solution is, for obvious reasons, not available. Instead one could try to bound the difference of the objective (3) for the current solution to the optimal value obtained for the full problem as done in [11]. But again, in our case such an approach would be infeasible. Instead we have chosen a very simple heuristic which turned out to work well in the experiments: Stop when the average improvement in the dual objective (7) over the last p iterations is less than some threshold θ. The longer the averaging process, the more confident we are that the current solution is not at a plateau. The smaller the threshold, the closer we are to the original solution (indeed, setting the threshold to zero forces the algorithm to take all training samples into account). Still, this rule is sub-optimal in that a good threshold for the problem at hand seems difficult to predict (see experimental section).
The complete algorithm for a sparse greedy solution to the KFD problem is schematized in Figure 1. It is easy to implement using a linear algebra package like BLAS and has the potential to be easily parallelized (the matrix update) and distributed.

```
arguments:
        Sample $X = \{\mathbf{x}_1, \dots, \mathbf{x}_\ell\}, \mathbf{y} = \{y_1, \dots, y_\ell\}$
        Maximum number of coefficients
        or parameters of other stopping criterion:  OPTS
        Regularization constant $C$
        $\kappa$ and kernel $\mathrm{k}$
returns:
        Set of indices $I$ and corresponding $\boldsymbol{\alpha}$'s.
        Threshold $b$.
function  SG-KFD($X, \mathbf{y}, C, \kappa, \mathrm{k}$, OPTS)
  $m \leftarrow 0$
  $I \leftarrow \emptyset$
  while termination criterion not satisfied do
    $S \leftarrow$ ($\kappa$ elements from $[\ell] \setminus I$)
    $obj_{\max} \leftarrow \infty$
    for $i \in S$ do
      Compute column $i$ of kernel matrix
      Update inverse adding the $i$-th kernel, compute optimal $\mathbf{a}$
      Compute new dual objective
      if dual objective $< obj_{\max}$ do
        $\mathtt{i}_{\mathtt{opt}} \leftarrow i$
        $obj_{\max} \leftarrow$ dual objective
      endif
    endfor
    Update inverse $H$ and solution $\mathbf{a}$ with kernel $\mathtt{i}_{\mathtt{opt}}$
    $I \leftarrow I \cup \{\mathtt{i}_{\mathtt{opt}}\}$
    Check termination criterion
  endwhile
```

Figure 1: The Sparse Greedy Kernel Fisher Algorithm

5 Obtaining Probabilities as Outputs

One of the advantages of KFD over e.g. SVM is that the outputs of KFD can be interpreted in a probabilistic manner. If one is interested in probabilities these are straightforward to obtain. Implicitly the optimization problem for KFD assumes Gaussian distribution for the likelihood functions, an assumption which is empirically supported by examining the output histograms on different datasets: they exhibit a strong Gaussianity [7]. If we estimate mean and variance of these Gaussians using the training data it turns out that the class label is the mean (i.e. $\mu_+ = 1$, $\mu_- = -1$), due to the constraints (3b), and that the variances of the class conditional densities are given by $\sigma_+^2 = \frac{1}{\ell_+ - 1} \sum_{y_i = 1} \xi_i^2$ and $\sigma_-^2 = \frac{1}{\ell_- - 1} \sum_{y_i = -1} \xi_i^2$. In some cases it might be advantageous to estimate $\mu_\pm$ and $\sigma_\pm$ from a separate validation set in which case they can simply be computed from the outputs

$$q(\mathbf{x}) := (\mathbf{w} \cdot \mathbf{x}) + b = \sum_i \alpha_i \, \mathrm{k}(\mathbf{x}, \mathbf{x}_i) + b$$

obtained by projecting this data set onto the direction $\mathbf{w}$. No matter which way the parameters are estimated, one obtains the following class-conditional densities:

$$p(\mathbf{x}|y = \pm 1) = p(q(\mathbf{x})|y = \pm 1) = \left(2\pi\sigma_\pm^2\right)^{-1/2} \exp\left(\frac{(q(\mathbf{x}) - \mu_\pm)^2}{2\sigma_\pm^2}\right)$$

The prior, if it is unknown for the problem at hand, can be estimated from the training data, i.e. $P(y = \pm 1) = \ell_\pm/\ell$, $\ell_\pm$ denoting the number of samples from the respective class. Using Bayes' theorem the conditional probabilities are then given by:

$$P(y = \pm 1|\mathbf{x}) = P(y = \pm 1|q) = \frac{p(q|y = \pm 1)P(y = \pm 1)}{p(q|y = 1)P(y = 1) + p(q|y = -1)P(y = -1)},$$

where $q = q(\mathbf{x})$ is defined as above.

6 Experimental evaluation

We now report the result of a small pilot experiment, carried out on a Pentium III, 500 MHz and 512 MB memory running Linux. We illustrate that the runtime behavior of our new algorithm improves significantly over the full quadratic optimization of (3). Furthermore, we show that the approximation does not significantly degrade the quality of the solutions.
To implement the new approach we used a single threaded, optimized BLAS. Timings were measured with the system command `clock()`. We compared this to an implementation of the quadratic program given by (3), which is partly in matlab and calls external C functions for all time–consuming operations. Here timings were measured by the matlab command `cputime()`. The quadratic optimizer used was loqo [14].

Timing First we compare the runtime of the new algorithm to the previous implementation. We used a one-against-the-rest task constructed from the USPS handwritten digit data set. The data are $N = 256$ dimensional and the set contains 7291 samples, categorized into ten classes. All experiments were done with a Gaussian kernel $\exp(\|\mathbf{x} - \mathbf{y}\|^2/(0.3 \cdot N)$ and using a regularization constant $C = 1$. We compared against the program given by (3) with the regularizer $P = \|\boldsymbol{\alpha}\|^2$. The results of our findings are given in Figure 2. It is important to keep in mind that the sparse greedy approach only needs to store at most an $m \times m$ matrix, where m is the maximal number of kernel functions chosen before termination. In contrast, previous approaches needed to store $\ell \times \ell$ matrices.

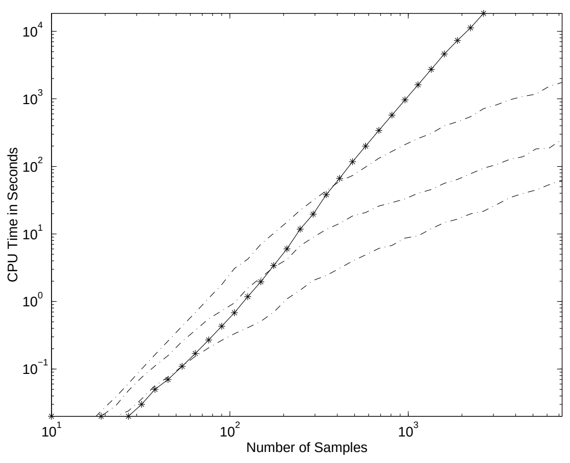

Figure 2: Runtime of sparse greedy KFD training. Depicted is the number of samples in the training set versus the CPU time of the proposed algorithm (dash dotted lines) and the QP formulation (3) (solid line). The estimates are averages over ten trials, one for each of the ten one–against–the–rest problems in the USPS database. The three lines for KFD are generated by requiring different accuracies on the dual error function in the stopping criterion, namely $10^{-a}, a = 1, \ldots, 3$ relative to the function value (in that order from bottom to top). There is a speed-accuracy tradeoff in that for large a, the algorithm converges more slowly. In the log-log plot it can be seen that the QP algorithm roughly scales cubic in the number of samples while the new algorithm scales with an exponent of about $\frac{3}{2}$ for large sample sizes.

Performance As our new approach is an approximation to the original (theoretically exact) algorithm, the question arises how good the quality of this approximation is. To this end, we repeated the above experiment on the USPS database for different regularization constants $C = 10^{-3}, 10^{-4}, 10^{-5}$ and different kernel widths $c = 0.3 \cdot N, 0.4 \cdot N, 0.5 \cdot N$. The algorithm was terminated when the average achievement in the dual objective over the last five iterations was less than $10^{-1}, 10^{-2}, 5 \cdot 10^{-3}, 10^{-3}$, respectively, relative to the objective or when a maximum of 450 coefficients was found. As the purpose of this experiment is to show that our new approach is capable of producing results comparably to the full system no model selection was performed and just the best results on the *test set* are reported (cf. Table 1). A small improvement in the test error can be achieved using an optimized threshold b rather than the one given by the algorithm itself. This optimized b is found by training a linear support vector machine on the one dimensional outputs of the training date, i.e. we try to find a threshold which maximizes the smallest distance of the projections to the decision boundary (details e.g. in [8]).

Tolerance	10^{-1}	10^{-2}	$5 \cdot 10^{-3}$	10^{-3}
test error with QP threshold	10.4%	6.4%	5.3%	4.1%
test error with optimized threshold	10.3%	6.3%	5.3%	3.9%

Table 1: Minimal 10–class test error on the USPS dataset using the parameters described in the text. Shown is the threshold on the improvement in the dual objective used to terminate the algorithm (Tolerance), the test error using the threshold given by the algorithm itself, and the test error using an extra, optimized threshold b (see text). The best result of 3.9% is almost identical to the result of 3.7% obtained on the same dataset using an expansion fixed to the first 3000 training samples [9]. Note, moreover, that for our new algorithm, the number of samples in the expansion (2) is less than 450 in each single classifier.

So far the best result for KFD on the USPS dataset (without using prior knowledge) was 3.7% [9], using an expansion restricted to the first 3000 training patterns. From Table 1 it can be seen that our new approach produces results close to the QP solution, however, using a significantly smaller number of kernel functions (less than 450 vs. 3000). It can be observed that the chosen precision for the termination criterion is an important parameter. Still, although the high precision of 10^{-3} takes longer to train, the runtime of our new approach is more than ten times faster than solving the QP with 3000 patterns.

7 Conclusion

We presented a new algorithm for kernel Fisher discriminants. These algorithmic advances are crucial for the possibility of applying the KFD algorithms to problems that had previously been beyond its reach. First, since it trains significantly faster, and second, since it requires less memory.

References

[1] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

[2] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[3] J.H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

[4] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, 2nd edition, 1990.

[5] G.H. Golub and C.F. van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, London, 3rd edition, 1996.

[6] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A 209:415–446, 1909.

[7] S. Mika, G. Rätsch, and K.-R. Müller. A mathematical programming approach to the Kernel Fisher algorithm. In *Advances in Neural Information Processing Systems 13*, 2001. to appear.

[8] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999.

[9] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A.J. Smola, and K.-R. Müller. Invariant feature extraction and classification in kernel spaces. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 526–532. MIT Press, 2000.

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

[11] A.J. Smola and P.L. Bartlett. Sparse greedy gaussian process regression. In *Advances in Neural Information Processing Systems 13*, 2001. to appear.

[12] A.J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In P. Langley, editor, *Proc. ICML'00*, pages 911–918, San Francisco, 2000. Morgan Kaufmann.

[13] M.E. Tipping. The relevance vector machine. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 652–658. MIT Press, 2000.

[14] R.J. Vanderbei. LOQO user's manual – version 3.10. Technical Report SOR-97-08, Princeton University, Statistics and Operations Research, 1997. Code available at http://www.princeton.edu/˜rvdb/.