1

Kernel Fisher's Discriminant with Heterogeneous Kernels

M. Murat Dundar[1] and Glenn Fung[2]

[1] Indiana University - Purdue University (IUPUI), Indianapolis, IN 46022
[2] Siemens Medical Solutions Inc. USA, Malvern, PA 19355

1.1 Introduction

In hyperspectral data analysis, materials of practical interest, such as agricultural crops, forest plantations, natural vegetation, minerals, and fields of interest in urban areas exist in a variety of states and are usually observed in a number of conditions of illumination. That is, most land-cover types do not have a single spectral response. For example a crop type will show different spectral characteristics at different times of the day and year. Similarly, roof tops are usually made of a variety of different materials including concrete, tile, bricks, glass etc. all of which have different spectral responses. The number of such examples can easily be augmented.

One possible way to deal with this problem is to model each class distribution data using Finite Mixture models (McLachlan and Peel (2004)). Finite Mixture Models usually leads to competitive performance when there is enough labeled data to reveal the underlying structure of the class distributions. However, in most real world settings, this may not be the case. The price one must pay for labeled data is usually prohibitively expensive, as acquiring labeled data requires a tedious and time consuming process of human labeling.

Kernel Methods for Remote Sensing Data Analysis. Edited by Gustavo Camps-Valls and Lorenzo Bruzzone

Another alternative for modeling multi-modal class distributions is kernel machines. Kernel machines was first introduced with Support Vector Machines (Vapnik (2000)) but later adapted to several other classification algorithms including Fisher's Discriminant (Mika et al. (1999)). Kernel concept provides the flexibility required to model complex data structures that originate from a wide range of class conditional distributions. Earlier studies show that Fisher's discriminant when implemented with kernel machines yields favorable results for the analysis of hyperspectral data with multimodal class distributions and limited training data (Dundar and Landgrebe (2004a)), (Dundar and Landgrebe (2004b)).

In the KFD algorithm the type of the kernel function and its parameters are usually estimated from a designated set of kernel models by cross-validation. With this approach the tuning procedure becomes quite computational for training set size larger than few hundred samples. We propose an iterative classification algorithm for Kernel Fisher's discriminant (KFD) using heterogeneous kernel models. In contrast with the standard KFD that requires the user to predefine a kernel function, we incorporate the task of choosing an appropriate kernel into the optimization problem to be solved. The choice of kernel is defined as a linear combination of kernels belonging to a potentially large family of different kernels.

Preliminary results with some benchmark datasets were presented earlier (Fung et al. (2004)). In this study additional experimental results on a hyperspectral dataset are presented to further validate the effectiveness of the proposed algorithm in *learning* the optimal combination of kernel functions. The results demonstrate that the prediction accuracy of the proposed algorithm is not significantly different than that achieved by the standard KFD in which the kernel parameters have been tuned using cross validation yet the training with the proposed algorithm is multiple folds faster than that of standard KFD.

This chapter is organized as follows. In the next section we will briefly review the Linear Fisher's Discriminant (LFD). Then, we will present a mathematical formulation of the Fisher's Discriminant algorithm that will form the basis for the Kernel Fisher's Discriminant (KFD). Next, we will discuss the implementation of KFD with heterogeneous kernel models and present an iterative algorithm for automatically selecting the kernels. Finally we will present results to validate the applicability of the proposed approach on a real-world problem with a hyperspectral dataset.

1.2 Linear Fisher's Discriminant

It is well known that in supervised classification problems the probability of error due to a Bayes classifier is the best that can be achieved. The Bayes classifier compares the a posteriori probabilities of all classes, and assigns the sample to the class with the highest probability. However for most classes of distributions, designing an optimum Bayes classifier is very difficult if not impractical. The primary problem stems from the finite size of the training set, leading to an imperfect estimate of the class probability density functions. The most common way to mitigate this problem is to assume normal distributions for all classes. Under this hypothesis standard classifiers using quadratic and linear discriminant functions can be designed.

The well-known Linear Fisher's Discriminant (LFD) (Fukunaga (1990)), arises in the special case when the considered information classes have a common covariance matrix. LFD is a classification method that projects the high dimensional data onto a line and performs

classification in this one-dimensional space. This projection is chosen such that the ratio of the scatter matrices (between and within classes) or the so called *Rayleigh quotient* is maximized. Even though LFD is mainly designed for binary classification problems it is extension to multiclass classification problems is also possible. For multiclass problems the ratio of between and within class scatter matrices can be maximized by solving a generalized eigenvalue problem. This leads to a projection matrix with $K-1$ eigenvectors where K is the number of classes in the dataset Fukunaga (1990).

In the rest of this chapter we will limit our discussion to binary classification problems. More specifically, we are given a training dataset $\{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \in \Re^d$ are input variables and $y_i \in \{-1, 1\}$ are class labels. Let $X \in R^{d \times n}$ be a matrix containing all the training samples and let $X_k \subseteq X \in R^{d \times n_k}$ be a matrix containing the n_k training samples for class ω_k, $k \in \{\pm\}$. Then, the LFD is the projection w, which maximizes,

$$J(w) = \frac{w^T S_B w}{w^T S_W w} \tag{1.1}$$

where

$$S_B = (\mu_+ - \mu_-)(\mu_+ - \mu_-)^T \tag{1.2}$$

$$S_W = \sum_{k \in \{\pm\}} \frac{1}{n_k} \left(X_k - \mu_k e_{n_k}^T\right)\left(X_k - \mu_k e_{n_k}^T\right)^T \tag{1.3}$$

are the between and within class scatter matrices respectively and

$$\mu_k = \frac{1}{n_k} X_k e_{n_k} \tag{1.4}$$

is the mean of class ω_k and e_{n_k} is an n_k dimensional vector of ones.

The above problem can be reformulated as follows. First notice that if w is a solution to (1.1), then so is any scalar multiple of it. Therefore, to avoid multiplicity of solutions, we impose an arbitrary constraint on $w^T S_B w = 4$, which is equivalent to $w^T(\mu_+ - \mu_-) = 2$. Then the optimization problem of (1.1) becomes

$$\begin{array}{cc} \min_{w \in R^d} & w^T S_W w \\ s.t. & w^T(\mu_+ - \mu_-) = 2 \end{array} \tag{1.5}$$

A closed-form solution w^* for (1.5) can be obtained by optimizing the Lagrange function associated with the above problem. This gives $w^* = \lambda S_W^{-1}(\mu_+ - \mu_-)$ where λ is the Lagrange multiplier obtained as $\lambda = \frac{1}{(\mu_+ - \mu_-)^T S_W^{-1}(\mu_+ - \mu_-)}$. For $d > n$, i.e. number of dimensionality is greater than the number of samples, S_W can be singular and thus the inverse does not exist. To avoid such ill-conditioned settings it is a common practive to replace S_W by $S_{W\nu} = S_W + \nu I$. Here ν acts as a regularizer over the classifier.

When classes are normally distributed with equal covariance, w^* is in the same direction as the discriminant in the corresponding Bayes classifier. Hence, for this special case LFD is equivalent to the Bayes optimal classifier. Although LFD relies heavily on assumptions that are not true in most real world problems, it has proven to be very powerful. In particular

when the distributions are unimodal and separated by the scatter of means, LFD becomes very effective. One reason why LFD may be preferred over more complex classifiers is that asa linear classifier it is less prone to overfitting.

For most real world data, a linear discriminant is clearly not complex enough. Classical techniques tackle these problems by using more sophisticated distributions in modeling the optimal Bayes classifier; however, these often sacrifice the closed form solution and are computationally more expensive. A relatively new approach in this domain is the kernel version of Fisher's Discriminant (Mika et al. (1999)) which is known as Kernel Fisher's Discriminant (KFD) in the literature. The main characteristic of this approach is the kernel concept, which was originally applied in Support Vector Machines and allows the efficient computation of Fisher's Discriminant in the kernel space. The linear discriminant in the kernel space corresponds to a powerful nonlinear decision function in the input space. Furthermore, different kernels can be used to accommodate the wide-range of nonlinearities that may occur in the data set. In the next section we derive the kernel version of the Fisher's Discriminant.

1.3 Kernel Fisher Discriminant

1.3.1 Mathematical programming formulation

The formulation in (1.5) is a parametric formulation of Fisher's Discriminant. The discriminative approach to the same problem can be obtained as follows. First, we define $\xi_i := w^T(x_i - \mu_k)$, $\forall i \in \omega_k$, $k \in \{\pm\}$ and impose $w^T\mu_+ = 1$, $w^T\mu_- = -1$. Let ξ^k be a vector containing all the ξ_i for class ω_k and y be the vector of class labels. Then the problem in (1.5) becomes

$$\begin{array}{rrcl} \min\limits_{(w,\gamma,\xi)\in R^{n+d+1}} & \frac{1}{2}\sum_{k\in\{\pm\}} \frac{1}{n_k\nu}{\xi^k}^T\xi^k & + & \frac{1}{2}w^Tw \\ \text{s.t.} & \xi & = & X^Tw - y \\ & e_{n_k}^T\xi^k & = & 0, k \in \{\pm\} \end{array} \tag{1.6}$$

The Lagrangian of (1.6) is given by

$$L(w,\xi,\lambda_1,\lambda_2) = \frac{1}{2}(\xi^T D\xi + w^Tw) + \lambda_1^T(\xi - X^Tw + y) + \lambda_2^T B^T\xi \tag{1.7}$$

where D is an $n \times n$ diagonal matrix with the first n_+ entries equal to $\frac{1}{n_+\nu}$ and the remaining n_- ones equal to $\frac{1}{n_-\nu}$, B is an $n \times 2$ indicator matrix with the first n_+ entries in the first column and the last n_- entries in the second column are set to one with all others being zero. Here $\lambda_1 \in R^n$ and $\lambda_2 \in R^2$ are the lagrange multipliers corresponding to $\xi = X^Tw - y$ and $e_{n_k}^T\xi^k = 0$, respectively. Solving for the gradient of (1.7) equal to zero, we obtain the Karush-Kuhn-Tucker (KKT) necessary and sufficient optimality conditions Mangasarian (1994) for the FLD problem with equality constraints given by

$$\begin{array}{rcl} w - X\lambda_1 & = & 0 \\ D\xi + \lambda_1 + B\lambda_2 & = & 0 \\ B^T\xi & = & 0 \\ X^Tw - y - \xi & = & 0 \end{array} \tag{1.8}$$

The first two equations of (1.8) give the following expressions for the original problem variables (w, ξ) in terms of the Lagrange multipliers λ_1 and λ_2:

$$w = X\lambda_1, \;\; \xi = -D^{-1}(\lambda_1 + B\lambda_2) \tag{1.9}$$

Substituting these in the last two equalities of (1.8) gives us an explicit expression for λ_1 and λ_2 in terms of X and y as follows

$$\begin{bmatrix} X^TX + D^{-1} & D^{-1}B \\ B^TD^{-1} & B^TD^{-1}B^T \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} y \\ B^Ty - B^TX^TX \end{bmatrix} \tag{1.10}$$

Solving the linear system of equations (1.10) gives us the solution $\begin{bmatrix} \lambda_1^* \\ \lambda_2^* \end{bmatrix}$ which in turn yields $w^* = X\lambda_1^*$ from (1.9). Here * denotes optimal solutions. In the rest of this chapter we drop the subscript from λ_1 for notational simplicity.

The "kernelized" version of the Fisher's Discriminant can be obtained in this framework by replacing the primal variable w by its dual equivalent $w = X\lambda$ in (1.6) to obtain:

$$\begin{array}{rrcl} \min\limits_{(\lambda,\xi)\in R^{n+d}} & \frac{1}{2}\sum_{k\in\{\pm\}} \frac{1}{n_k\nu}{\xi^k}^T\xi^k & + & \frac{1}{2}\lambda^T\lambda \\ \text{s.t.} & \xi & = & X^TX\lambda - y \\ & e_{n_k}^T\xi^k & = & 0, k \in \{\pm\} \end{array} \tag{1.11}$$

where the objective function has also been modified to minimize weighted 2-norm sums of the dual variables. If we now replace the X^TX by a nonlinear kernel $K(X^T, X)$, we obtain a formulation that is equivalent to the Kernel Fisher Discriminant described in (Mika et al. (2000)).

$$\begin{array}{rrcl} \min\limits_{(\lambda,\xi)\in R^{n+d}} & \frac{1}{2}\sum_{k\in\{\pm\}} \frac{1}{n_k\nu}{\xi^k}^T\xi^k & + & \frac{1}{2}\lambda^T\lambda \\ \text{s.t.} & \xi & = & K(X^T, X)\lambda - y \\ & e_{n_k}^T\xi^k & = & 0, k \in \{\pm\} \end{array} \tag{1.12}$$

Recent SVM formulations with least squares loss (Suykens and Vandewalle (1999)) are much the same in spirit as the problem in (1.6). Using a similar duality analysis to the one presented here, and then "kernelizing", the authors obtain the objective function

$$\nu\frac{1}{2}\|\xi\|^2 + \frac{1}{2}\lambda^T K(X^T, X)\lambda. \tag{1.13}$$

The regularization term $\lambda^T K(X^T, X)\lambda$ determines that the model complexity is regularized in a reproducing kernel Hilbert space (RKHS) associated with the specific kernel K where the kernel function K has to satisfy Mercer's conditions and $K(X^T, X)$ has to be positive semidefinite.

By comparing the objective function (1.13) to problem (1.12), we can see that problem (1.12) does not regularize in terms of RKHS. Instead, the columns in a kernel matrix are simply regarded as new features $K(X^T, X)$ of the classification task in addition to the original features X. We can then construct classifiers based on the features introduced by a kernel in the same way we build classifiers using original features X.

1.4 Kernel Fisher's Discriminant with Heterogeneous Kernels

The kernelized version of the Fisher's Discriminant (KFD) gives us the flexibility required to model complex data structures that originate from a wide-range of class conditional distributions. Like its linear space counterpart, the statistics estimation is performed at full dimensionality (no feature extraction is needed) allowing us to exploit all the separability the data provides without having to deal with severe numerical issues.

Like most other kernel-based approaches KFD also suffers from the computational complexity of working with kernel functions to a greater extent. The computational complexity of the algorithm is on the order of $O(n^3)$ making its use impractical for large datasets. A common way around this problem is to expand the kernel matrix in terms of a random subset of the training samples. However, the problem of selecting the best kernel function type and parameters still remains. Usually, cross-validation is used to optimize the algorithm over a large number of kernel parameters and the parameter set that maximizes the cross-validation performance is chosen as the optimum set.

Cross-validation is a method for estimating predictive error of the classifier with the training data. It splits the training dataset into k equal-sized pieces called folds. At each stage one fold is left out as testing data and the classifier is trained with the remaining $k-1$ folds. This process is repeated until all k folds are tested and aggregate test error is recorded as the $k-fold$ cross validation performance.

In this section, we propose a methodology for selecting the optimum kernel function as a weighted summation of several other kernel functions where the weights of the kernel functions are learned automatically through an alternating optimization technique. To be more specific, let us suppose that instead of the kernel K being defined by a single kernel mapping (i.e., Gaussian, polynomial, etc.), the kernel K is composed of a linear combination of kernel functions $K_j, j = 1, \ldots, k$, as below

$$K(X^T, X) = \sum_{j=1}^{k} a_j K_j(X^T, X), \tag{1.14}$$

where $a_j \geq 0$.

As it is pointed out in Lanckriet et al. (2003), the set $\{K_1(A, A'), \ldots, K_k(X^T, X)\}$ can be seen as a predefined set of initial "guesses" of the kernel matrix and it could contain very different kernel matrix models, (e.g., linear, Gaussian, polynomial) with different parameter values. In this formulation parameters specific to each kernel are fixed a priori. Instead of fine tuning the kernel parameters for a predetermined kernel via cross-validation, we can optimize the set of values $a_i \geq 0$ in order to obtain a positive semi-definite linear combination $K(X^T, X) = \sum_{j=1}^{k} a_j K_j(X^T, X)$ suitable for the specific classification problem. Substituting equation (1.14) in equation (1.11) we obtain the KFD formulation with heterogeneous linear combinations of kernels as follows

$$\begin{array}{rll} \min\limits_{(\lambda,\xi,a)\in R^{n+d+k}} & \frac{1}{2}\sum_{k\in\{\pm\}}\frac{1}{n_k\nu}{\xi^k}^T\xi^k & + \quad \frac{1}{2}\lambda^T\lambda \\ \text{s.t.} & \xi & = \quad \sum_{j=1}^{k} a_j K_j\lambda - y \\ & e_{n_k}^T\xi^k & = \quad 0, k\in\{\pm\} \\ & a_j & \geq \quad 0, j\in\{1,\ldots,k\} \end{array} \tag{1.15}$$

where $K_j = K_j(X^T, X)$. When considering linear combinations of kernels, the hypothesis space may become larger, making the issue of capacity control an important one. It is known that if two classifiers have similar training error, a smaller capacity may lead to better generalization on future unseen data (Cherkassky and Mulier (1998); Vapnik (2000)). In order to reduce the size of the hypothesis and model space and to gain strong convexity in all variables, an additional regularization term $\frac{1}{2}a'a$ is added to the objective function of problem (1.15). The problem then becomes

$$\begin{array}{rll} \min\limits_{(\lambda,\xi,a)\in R^{n+d+k}} & \frac{1}{2}\sum_{k\in\{\pm\}}\frac{1}{n_k\nu}{\xi^k}^T\xi^k & + \quad \frac{1}{2}\lambda^T\lambda + \frac{1}{2}a^Ta \\ \text{s.t.} & \xi & = \quad \sum_{j=1}^{k} a_j K_j\lambda - y \\ & e_{n_k}^T\xi^k & = \quad 0, k\in\{\pm\} \\ & a_j & \geq \quad 0, j\in\{1,\ldots,k\} \end{array} \tag{1.16}$$

A new sample x is then classified by the following classifier:

$$\left(\sum_{j=1}^{k}(a_j K_j(x, X))\right)\lambda = \left\{ \begin{array}{ll} \geq b, & x\in\omega_+, \\ < b, & x\in\omega_- \end{array} \right). \tag{1.17}$$

where b is a predefined threshold that adjusts the tradeoff between incorrectly classifying a positive sample as negative, i.e. false negative, and incorrectly classifying a negative sample as positive, i.e., false positive.

Even though the objective function in (1.16) is strictly convex in terms of the problem variables ξ, λ and a, the problem itself is not convex due to the nonconvex equality constraint $\xi = \sum_{j=1}^{k} a_j K_j\lambda - y$. However, the problem in (1.16) can be treated as a biconvex programming problem first by fixing $a = a^*$ and solving (1.16) for ξ and λ^* and then fixing $\lambda = \lambda^*$ and solving for ξ and a^*. More specifically when we fix $a = a^*$ we obtain the following subproblem

$$\begin{array}{rll} \min\limits_{(\lambda,\xi)\in R^{n+d}} & \frac{1}{2}\sum_{k\in\{\pm\}}\frac{1}{n_k\nu}{\xi^k}^T\xi^k & + \quad \frac{1}{2}\lambda^T\lambda \\ \text{s.t.} & \xi & = \quad \sum_{j=1}^{k} a_j^* K_j\lambda - y \\ & e_{n_k}^T\xi^k & = \quad 0, k\in\{\pm\} \end{array} \tag{1.18}$$

and similarly when we fix $\lambda = \lambda^*$ we obtain the subproblem

$$\begin{array}{rll} \min\limits_{(\xi,a)\in R^{n+k}} & \frac{1}{2}\sum_{k\in\{\pm\}}\frac{1}{n_k\nu}{\xi^k}^T\xi^k & + \quad \frac{1}{2}a^Ta \\ \text{s.t.} & \xi & = \quad \sum_{j=1}^{k} a_j K_j\lambda^* - y \\ & e_{n_k}^T\xi^k & = \quad 0, k\in\{\pm\} \\ & a_j & \geq \quad 0, j\in\{1,\ldots,k\} \end{array} \tag{1.19}$$

Note that both problems in (1.18) and (1.19) are strongly convex with a unique optimizer, which implies that the original objective function is guaranteed to improve at each iteration. We are now ready to describe our proposed algorithm.

1.5 Automatic kernel selection KFD Algorithm

Algorithm 1.5.1 Automatic kernel selection KFD Algorithm (AKFD)
Given n data points in R^d represented by the $d \times n$ matrix X and vector y of ± 1 labels denoting the class of each data point (i.e., each column of X), the parameter ν and an initial $a^0 \in R^k$, we generate the nonlinear classifier (1.17) as follows:

(0) Calculate $K_1, \ldots, K_k$, the k kernels on the kernel family, where for each j, $K_j = K_j(X^T, X)$.

For each iteration i do:

(i) Given $a^{(i-1)}$ calculate the linear combination $K = \sum_{j=1}^{k} a_j^{(i-1)} K_j$.

(i) Solve subproblem (1.18) to obtain $\lambda^{(i)}$.

(ii) Calculate $K_j \lambda^{(i)}$ for $j = 1, \ldots, k$.

(iii) Solve subproblem (1.19) to obtain a^i.

Stop when a predefined maximum number of iterations is reached or when the change in value of the objective function (1.16) evaluated in successive iterations is less than ϵ.

The most common cases arise when $k << n$ (i.e., the number of kernels functions considered on the kernel family is much smaller than the number of data points). In such situations, the complexity of the AKFD algorithm 1.5.1 is approximately $O(n^3)$.

Since each of the two optimization problems ((1.18) and (1.19)) that are required to be solved by the AKFD algorithm are strongly convex and thus each of them have a unique minimizer, the AKFD algorithm can also be interpreted as an Alternate Optimization (AO) problem (Bezdek and Hathaway (2003)). Alternate Optimization divides the entire variable space into a predefined number of subspaces and optimizes one group of variables at a time while the remaining variables are fixed. Classical instances of AO problems include fuzzy regression c-models and fuzzy c-means clustering.

The AKFD algorithm then, inherits the convergence properties and characteristics of AO problems. As stated in (Bezdek and Hathaway (2002)), the set of points for which Algorithm 1.5.1 can converge may include certain type of saddle points (i.e. points behaving like local minimizers only when projected along a subset of the variables, see Figure 1.1). However, it is also stated that is extremely difficult to find examples where convergence occurs to a saddle point rather than to a local minimizer. If the initial estimate is chosen sufficiently near a solution, AO is shown to converge linearly to a local minimizer (Bezdek and Hathaway (2002)). In practice, we found that Algorithm 1.5.1 typically converges in 3-4 iterations (3 or 4) to a local solution of problem (1.16).

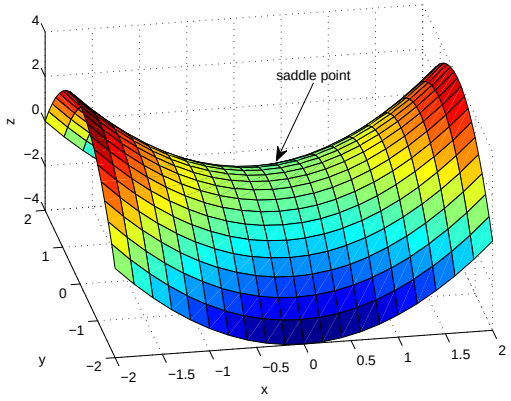

Figure 1.1 A saddle point for the function $z = x^2 - y^2$. It behaves like a local minimizer when projected along the x-axis.

1.6 Numerical Results

1.6.1 Dataset Used: Purdue Campus Data

This data set is a flightline over the Purdue University West Lafayette Campus. The hyperspectral data used was collected on September 30, 1999 with the airborne HYMAP system (Kruse et al. (2000)), providing image data in 126 spectral bands in the visible and IR regions (0.4μ-2.4μ). The system was flown at an altitude such that the pixel size is about 5 meters. The data set contains 358 scan lines with 390 pixels in each scan line. The list of classes and number of labeled samples for each class is given in Table 1.1. The image of the scene and the corresponding ground-truth regions of interest are shown in Figure 1.2 and Figure 1.3.

1.6.2 Classifier Design

We designed three different versions of Fishers discriminant. As a baseline classifier Linear Fishers Discriminant (LFD) is considered. To find out if the kernel version of the Fishers Discriminant improves our baseline, we implemented the Kernel Fishers Discriminant (KFD). Finally we design the automatic kernel selection algorithm for the Fishers discriminant (AKFD) to see how much we save from the training time and if this is achieved while maintaining the similar performance levels achieved by KFD.

The classifier parameters, namely, the regularization parameter, ν, and the type of the kernel function and the corresponding parameter are optimized using a 10-fold cross-validation approach. For the regularization parameter a discrete set of 10 values are considered, i.e. $\nu = [10^{-10}\ 10^{-8}\ 10^{-6}\ 10^{-5}\ 10^{-4}\ 10^{-3}\ 5\times10^{-2}\ 10^{-2}\ 5\times10^{-1}\ 10^{-1}\ 1]$. For the kernel function we considered linear and radial basis function (RBF Gaussian Functions) kernels. For the width of the rbf we considered $\sigma = [10^{-2}\ 10^{-1}\ 1\ 10]$. The data is normalized such

Table 1.1

Classes	Number of Samples
Roof Tops	10182
Streets	4571
Grass	1539
Trees	1743
Paths	907
Shadow	434
Cars	934
Fields	608
Total	20918

Number of labeled samples available for each class identified in the Purdue Campus Dataset.

that each feature is between -1 and 1. One-Agains-All multi-class strategy is used throughout the experiments Hsu and Lin (2002).

To be more specific for the KFD algorithm a single RBF Gaussian kernel is used. The width σ of this kernel is chosen from one of the 4 different values considered via cross-validation. However, for the AKFD algorithm a linear combination of 5 kernel functions (i.e., a linear and 4 RBF Gaussian functions one for each of the σ considered) is used.

The AKFD algorithm is initialized with a set to all ones (i.e., initially all kernels are assumed to contribute equally). The algorithm is terminated when the improvement in the objective function at any iteration over the previous iteration is less than 0.1% or a maximum number of 20 iterations is reached. We ran our experiments for four different sizes of training sets, i.e. r=0.01 (n=209), r=0.02 (n=418), r=0.03 (n=627), r=0.04 (n=836) where r denotes the ratio of the training to labeled samples. Training samples are selected randomly and each experiment is repeated 10 times. The size of the expansion set S for the kernel matrices is limited to 250 randomly selected samples, i.e. kernel matrices are computed using $K(X^T, S)$ instead of $K(X^T, X)$ where $S \subset X$ and size of S is 250. The labeled samples that are not used for training are used for testing.

1.6.3 Analysis of the results

Table 1.2 and Table 1.3 show the percentage average classification accuracies averaged over 10 runs achieved and the total time taken by each algorithm respectively. The following conclusions can be drawn from these results. When the training size is small linear version performs as well as the kernel version. As the training size increases linear version is no longer competent yet the prediction accuracy for kernel version increases with increasing r. The proposed AKFD algorithm generates results comparable to the KFD algorithm yet it is multiple folds faster than the KFD algorithm. For $r = 0.04$ (roughly 800 samples) the entire training plus testing took roughly 21 hours of running time. The same task is completed in 7 hours by the AKFD algorithm. The computer used for these experiments was equipped with intel core 2 duo CPU with a 1.8GHz clock speed.

Figure 1.2 RGB image of Purdue Campus Dataset. Bands 32, 16 and 8 are used respectively for the R, G and B values.

The classification maps obtained for $r = 0.04$ (for one of the iterations) are displayed in Figures 1.4, 1.5, 1.6 for LFD, KFD and AKFD respectively. As the classification maps corresponding to KFD and AKFD suggest, the difference between the predictive accuracies of KFD and AKFD is quite negligible, (i.e., test accuracy for AKFD is 90.6% and for KFD is 90.5%). The areas of the image where LFD performs poorly are annotated by the red rectangles.

The optimized values of the kernel weights, a obtained by the AKFD algorithm and the value of the kernel parameter, σ selected by cross-validation for KFD are shown in Table 1.4 for each one-against-all classification task. The results suggest that AKFD favors a linear combination with mostly non-zero weights for the individual kernel functions as opposed to KFD, which uses only one kernel function selected from several others available. Despite seemingly different kernel models being used by the two algorithms, we observe almost identical predictive accuracies for the two classifiers.

Table 1.2

Classifiers	n=209	n=418	n=627	n=836
LFD	81.0 (5.2)	84.1 (1.0)	84.2 (0.4)	84.6 (0.8)
KFD	82.5 (3.2)	86.5 (2.9)	89.3 (1.9)	89.9 (1.5)
AKFD	80.6 (2.9)	85.8 (1.6)	87.9 (1.4)	90.1 (0.9)

Percentage overall classification accuracies averaged over 10 runs. Numbers in parenthesis are standard deviations.

Table 1.3

Classifiers	r=0.01	r=0.02	r=0.03	r=0.04
LFD	99	114	147	176
KFD	3967	26276	35667	74773
AKFD	2125	5902	11834	26133

Total computational time (in seconds) for 10 iterations of each algorithm.

Table 1.4

model, σ	linear	rbf, 0.01	rbf, 0.1	rbf, 1	rbf, 10	
Roof tops	1.3	0.3	0.2	0.3	0.3	$\sigma = 0.01$
Roads	0	0.8	0	1.1	1.2	$\sigma = 0.1$
Grass	0	0.7	0	0.5	0.9	$\sigma = 0.01$
Trees	0	0.3	0	0.1	0.5	$\sigma = 0.01$
Paths	0	1.1	0	0.8	1.0	$\sigma = 0.01$
Shadow	0.9	1.0	1.2	0.9	1.0	$\sigma = 0.1$
Cars	1.2	1.1	0	0.6	0.9	$\sigma = 0.1$
Fields	1.3	0.5	0.4	0.9	1.1	$\sigma = 0.1$

The optimized values of the kernel weights, a obtained by the AKFD algorithm. The last column shows the value of the σ selected by cross-validation for the KFD algorithm.

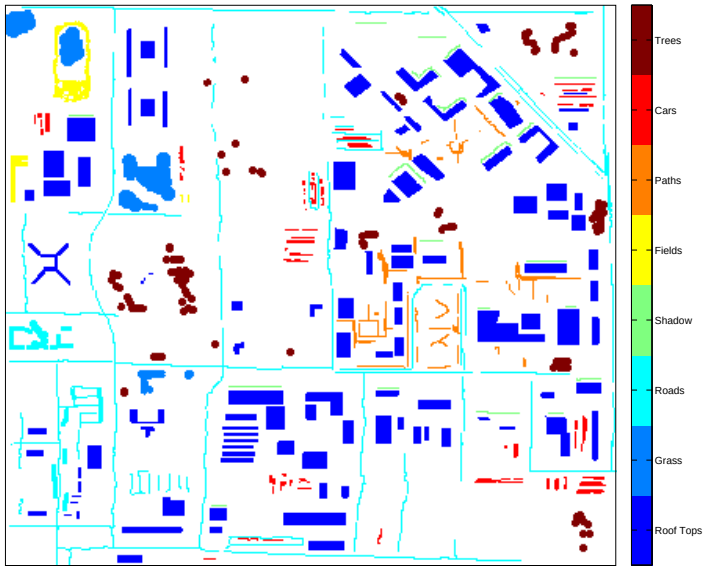

Figure 1.3 Ground Truth Fields for the Purdue Campus Data

1.7 Conclusion

In this chapter we first reviewed the basics of Fisher's Discriminant, then presented a mathematical programming approach for kernelizing the algorithm so as to obtain nonlinear classifiers and finally proposed an alternating optimization algorithm for automatically learning the kernel function. Unlike LFD, KFD has the potential to deal with data of complex structures such as multimodal data. However, this comes at the cost of increased computational time. The computational time for training increases, at the order of $O(n^3)$ for KFD. Moreover, to optimize the algorithm for different kernel functions and parameters, a cross-validation scheme is required. More specifically, if one is considering p different parameters for the kernel function and using a $k-fold$ cross-validation approach, the algorithm needs to run for $p \times k$ times to find the optimum kernel parameter. The proposed AKFD algorithm on the other hand eliminates the need for the cross-validation by automatically learning the weights of different kernel functions considered. Let the number of iterations before convergence be N; then, for $N << p \times k$ the computational gain could be significant.

In our experimental setting the AKFD algorithm usually converged in less than 5 iterations. We considered 5 different values for the kernel width and adopted a 10-fold cross validation framework. So instead of running the KFD algorithm $5 \times 10 = 50$ times to select the optimum kernel parameter, we just run the AKFD algorithm once. Each iteration of the AKFD algorithm roughly takes the same amount of time as KFD. Thus in the AKFD algorithm the optimum kernel is selected in around 5 iterations whereas in KFD this task takes 50 iterations. As for the online testing the AKFD algorithm is slower because all of the kernel

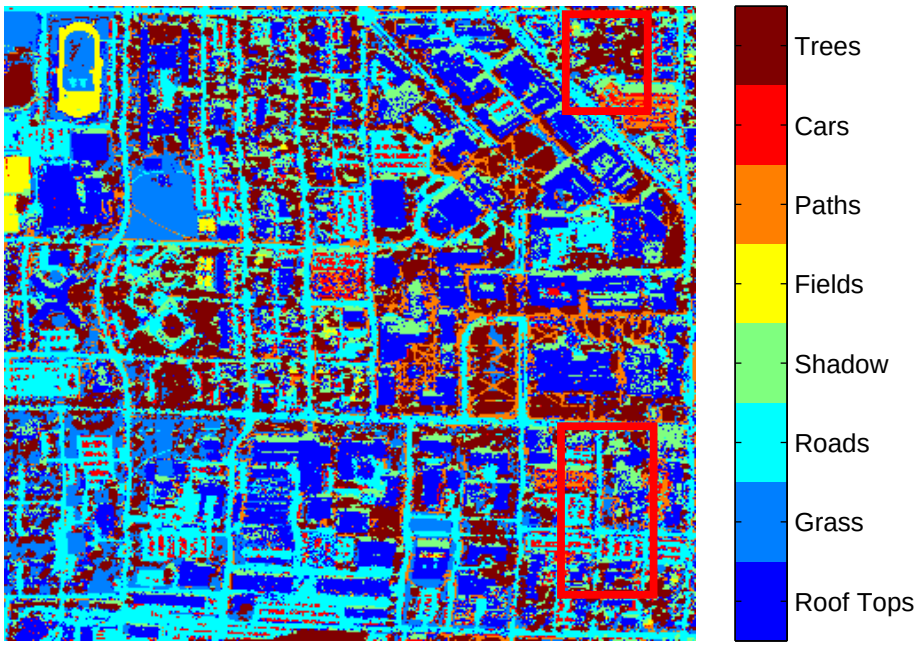

Figure 1.4 Classification map obtained for LFD for r=0.04. Test accuracy 86.5%

matrices needs to be computed during testing which takes more time than computing just one kernel matrix as in KFD. As the numbers in Table 1.3 suggest even when the testing times are included, the overall computational times clearly favors AKFD over KFD.

As the experimental results suggest the predictive accuracy of the proposed algorithm is not significantly different than that obtained by the KFD algorithm. That is, the computational gain is achieved while maintaining similar predictive performance.

The results in Table 1.4 indicate that the kernel functions obtained by KFD and AKFD could be significantly different yet both algorithms yield similar predictive performance. We believe further research is required to investigate how the kernel selected by the KFD algorithm correlates with that obtained by AKFD. Another area that needs attention is the initialization of the weights, a in Section 1.5. In this study we assumed all kernel models are a priori likely and thus assigned equal weights for each of them. However it is worthwhile to analyze the impact of initialization on the final weights optimized and how this in turn affects the predictive performance of the algorithm.

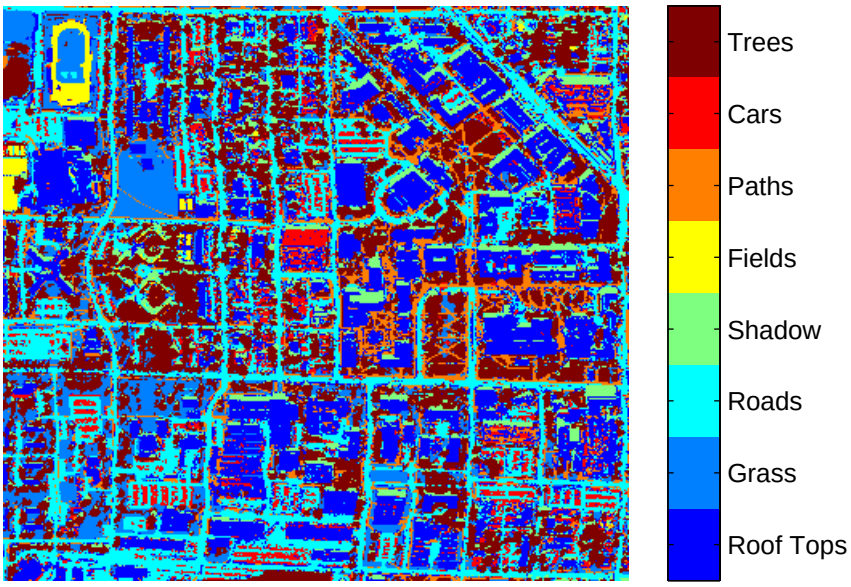

Figure 1.5 Classification map obtained for KFD for r=0.04. Test accuracy 90.5%

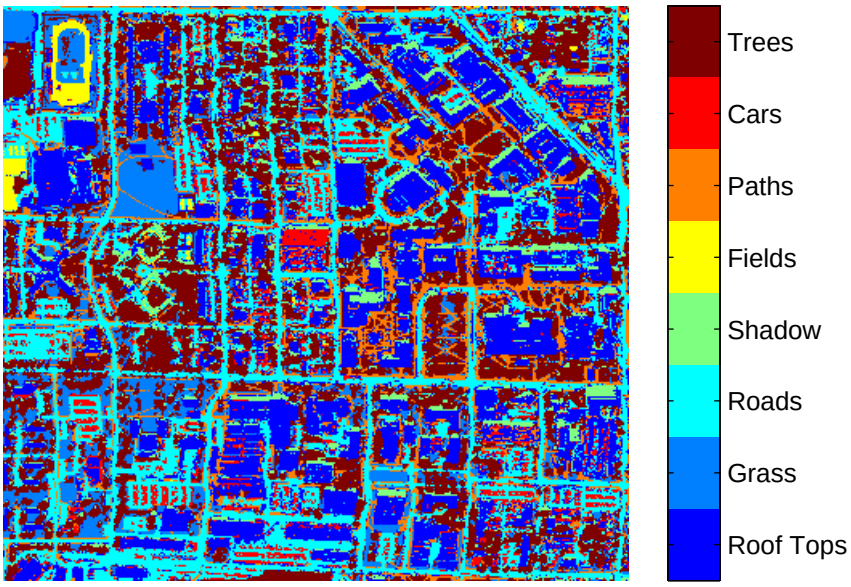

Figure 1.6 Classification map obtained for KFD for r=0.04. Test accuracy 90.6%

Bibliography

Bezdek J and Hathaway R 2002 Some notes on alternating optimization *Proceedings of the 2002 AFSS International Conference on Fuzzy Systems*, pp. 288–300. Springer-Verlag.

Bezdek J and Hathaway R 2003 Convergence of alternating optimization. *Neural, Parallel Sci. Comput.* **11**(4), 351–368.

Cherkassky V and Mulier F 1998 *Learning from Data - Concepts, Theory and Methods*. John Wiley & Sons, New York.

Dundar M and Landgrebe D 2004a A cost-effective semi-supervised classifier approach with kernels. *IEEE Transactions on Geoscience and Remote Sensing* **42**(1), 264–270.

Dundar M and Landgrebe D 2004b Toward an optimal supervised classifier for the analysis of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing* **42**(1), 271–277.

Fukunaga K 1990 *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, CA.

Fung G, Dundar M, Bi J and Rao RB 2004 A fast iterative algorithm for fisher discriminant using heterogeneous kernels *In Proceedings of the Twenty-First international Conference on Machine Learning (ICML '04)*.

Hsu CW and Lin CJ 2002 A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks* **13**(2), 415–425.

Kruse FA, Boardman JW, Lefkoff AB, Young JM, Kierein-Young KS, Cocks TD, Jensen R and Cocks PA 2000 Hymap: An australian hyperspectral sensor solving global problems - results from usa hymap data acquisitions *Proceedings of the 10th Australian Remote Sensing and Photogrammetry Conference*.

Lanckriet G, Cristianini N, Bartlett P, Ghaoui LE and Jordan M 2003 Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research* **5**, 27–72.

Mangasarian OL 1994 *Nonlinear Programming*. SIAM, Philadelphia, PA.

McLachlan G and Peel D 2004 *Finite Mixture Models*. Wiley-Interscience.

Mika S, Rätsch G and Müller KR 2000 A mathematical programming approach to the kernel fisher algorithm *NIPS*, pp. 591–597.

Mika S, Rätsch G, Weston J, Schölkopf B and Müller KR 1999 Fisher discriminant analysis with kernels In *Neural Networks for Signal Processing IX* (ed. Hu YH, Larsen J, Wilson E and Douglas S), pp. 41–48. IEEE.

Suykens JAK and Vandewalle J 1999 Least squares support vector machine classifiers. *Neural Processing Letters* **9**(3), 293–300.

Vapnik VN 2000 *The Nature of Statistical Learning Theory* second edn. Springer, New York.