Training set approximation for kernel methods

Vojtěch Franc, Václav Hlaváč

Czech Technical University, Faculty of Electrical Engineering
Department of Cybernetics, Center for Machine Perception
121 35 Prague 2, Karlovo náměstí 13, Czech Republic
xfrancv@cmp.felk.cvut.cz

Abstract *We propose a technique for a training set approximation and its usage in kernel methods. The approach aims to represent data in a low dimensional space with possibly minimal representation error which is similar to the Principal Component Analysis (PCA). In contrast to the PCA, the basis vectors of the low dimensional space used for data representation are properly selected vectors from the training set and not as their linear combinations. The basis vectors can be selected by a simple algorithm which has low computational requirements and allows on-line processing of huge data sets. The proposed method was used to approximate training sets of the Support Vector Machines and Kernel Fisher Linear Discriminant which are known method for learning classifiers. The experiments show that the proposed approximation can significantly reduce the complexity of the found classifiers (the number of the support vectors) while retaining their accuracy.*

1 Introduction

The kernel methods have become a fast developing branch of machine learning and pattern recognition in several past years. The kernel methods use kernel functions to perform the feature space straightening effectively. This technique allows to exploit established theory behind the linear algorithms to design their non-linear counterparts. The representatives of these methods are for instance the Support Vector Machines (SVM) [12] and the Kernel Fisher Linear Discriminant (KFLD) [5] which serve as classifier design or Kernel Principal Component Analysis (KPCA) [11] useful for non-linear feature extraction. The kernel learning methods are generally characterized by the following properties:

- The training data $\mathbf{X} = [\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}]$, $\boldsymbol{x_i} \in \mathcal{X}$ are transformed by a function $\phi: \mathcal{X} \rightarrow \mathcal{F}$ to a new high dimensional feature space $\mathcal{F}$. We denote the set of training data transformed to the high dimensional space $\mathcal{F}$ as $\mathbf{F} = [\phi(\boldsymbol{x_1}), \ldots, \phi(\boldsymbol{x_n})]$.

- The solution found is linear in the feature space $\mathcal{F}$, i.e. the function $f(\boldsymbol{x}) = \boldsymbol{w}^T \cdot \phi(\boldsymbol{x}) + b$ we search for is characterized by a vector $\boldsymbol{w} \in \mathcal{F}$ and a scalar $b \in \Re$.

- The kernel functions $k: \mathcal{X} \times \mathcal{X} \rightarrow \Re$ which corresponds to the dot products of non-linearly mapped data, i.e., $k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \phi(\boldsymbol{x_i})^T \cdot \phi(\boldsymbol{x_j})$, are used to avoid problems of high dimensionality of the space $\mathcal{F}$. This implies that the algorithm must use the dot products of training data only. The matrix of all dot products in the space $\mathcal{F}$ is denoted as the kernel matrix $\mathbf{K}(i,j) = k(\boldsymbol{x_i}, \boldsymbol{x_j})$ and it is of size $[n \times n]$.

- The solution found is expressed in terms of kernel expansion $f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x_i}) + b$, where α_i, $i = 1, ..., n$ are real coefficients which determine the vector $\boldsymbol{w} \in \mathcal{F}$ as a linear combination of transformed training data, i.e., $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x_i})$.

When using the kernel method the following problems can be encountered:

- The storage of the training data in terms of the dot products is too expensive since the size of kernel matrix $\mathbf{K}$ increases quadratically with the number of training data.

- The computation of the kernel functions $k(\boldsymbol{x_i}, \boldsymbol{x_j})$ is expensive.

- The solution which is expressed as a kernel expansion $f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x_i}, \boldsymbol{x}) + b$ is not sparse, i.e., many coefficients α_i are nonzero. This situation can occurs for instance in the SVM when the number of support vectors is huge or in the KFLD and the KPCA, since there is the solution expressed using all training data. The non-sparse solution implies an expensive evaluation of the function $f(\boldsymbol{x})$ (e.g., slow classification or feature extraction).

Several approaches to the problem of non-sparse kernel expansion were proposed. These methods are based on approximating the found solution, e.g., the reduced set method [2, 10] or the method by Osuna et. al [7].

We propose a new solution to the problems mentioned above which is based on approximating the training set in the non-linear kernel space $\mathcal{F}$.

The novel approach will be described in Section 2. The application of the proposed approach to the SVM and KLFD is described in Section 4. The experiments performed are mentioned in Section 5 and Section 6 concludes the paper.

2 Training set approximation

The transformed training data $\mathbf{F} = [\phi(\boldsymbol{x_1}), \ldots, \phi(\boldsymbol{x_n})]$ live in a subspace $\mathrm{span}(\mathbf{F}) \subseteq \mathcal{F}$. Let us suppose that we have a

finite set $\mathbf{X_r} = [\boldsymbol{r_1}, \ldots, \boldsymbol{r_m}]$, $\boldsymbol{r_i} \in \mathcal{X}$, $m < n$, and its image in the feature space $\mathcal{F}$, i.e, the set $\mathbf{F_r} = [\phi(\boldsymbol{r_1}), \ldots, \phi(\boldsymbol{r_m})]$. Let us also suppose that the vectors $\phi(\boldsymbol{r_i})$ are linearly independent and so that they form a basis of linear subspace $\mathrm{span}(\mathbf{F_r}) \subseteq \mathcal{F}$. We aim to express the transformed training data $\mathbf{F}$ using linear basis defined by the set $\mathbf{F_r}$. A method how to properly select the set $\mathbf{X_r}$ is described in the sequel. The $\mathbf{F}' = [\phi(\boldsymbol{x_1})', \ldots, \phi(\boldsymbol{x_n})']$ will denote a set of approximations of vectors $\mathbf{F} = [\phi(\boldsymbol{x_1}), \ldots, \phi(\boldsymbol{x_n})]$ which will be computed as minimal square error projections on the subspace $\mathrm{span}(\mathbf{F_r})$. It means that the approximation $\phi(\boldsymbol{x})' \in \mathbf{F}'$ of a vector $\phi(\boldsymbol{x}) \in \mathbf{F}$ is expressed as a linear combination of vectors of $\mathbf{F_r}$, i.e., $\phi(\boldsymbol{x})' = \mathbf{F_r} \cdot \boldsymbol{\beta}$ (we used matrix notation). The vector $\boldsymbol{\beta}$ contains real coefficients of linear combination and it is computed as

$$\boldsymbol{\beta} = \underset{\boldsymbol{\beta}'}{\operatorname{argmin}} \left(\phi(\boldsymbol{x}) - \mathbf{F_r} \cdot \boldsymbol{\beta}'\right)^T \cdot \left(\phi(\boldsymbol{x}) - \mathbf{F_r} \cdot \boldsymbol{\beta}'\right) .$$

The well known analytical solution of this problem is

$$\boldsymbol{\beta} = (\mathbf{F_r}^T \cdot \mathbf{F_r})^{-1} \cdot \mathbf{F_r}^T \cdot \phi(\boldsymbol{x}). \tag{1}$$

The solution for $\boldsymbol{\beta}$ in the terms of dot product has form

$$\boldsymbol{\beta} = \mathbf{K_r}^{-1} \cdot \mathbf{k_r}(\boldsymbol{x}) , \tag{2}$$

where $\boldsymbol{x} \in \mathbf{X}$ is a vector to be approximated, $\mathbf{K_r} = \mathbf{F_r}^T \cdot \mathbf{F_r}$ is a kernel matrix $[m \times m]$ of vectors from the set $\mathbf{X_r}$ and $\boldsymbol{k_r}(\boldsymbol{x})$ is a vector $[m \times 1]$ containing values of kernel functions of $\boldsymbol{x}$ and $\boldsymbol{r} \in \mathbf{X_r}$, i.e., $\boldsymbol{k_r}(\boldsymbol{x}) = [k(\boldsymbol{x}, \boldsymbol{r_1}), \ldots, k(\boldsymbol{x}, \boldsymbol{r_m})]$. We denote $\boldsymbol{\beta_i}$ the vector which contains coefficients computed for a vector $\boldsymbol{x_i} \in \mathbf{X}$. Thus the approximated value of kernel function of training vectors $\boldsymbol{x_i}, \boldsymbol{x_j} \in \mathbf{X}$ is computed as

$$\begin{aligned} k'(\boldsymbol{x_i}, \boldsymbol{x_j}) &= (\phi(\boldsymbol{x_i})')^T \cdot \phi(\boldsymbol{x_j})' \\ &= (\mathbf{F_r} \cdot \boldsymbol{\beta_i})^T \cdot (\mathbf{F_r} \cdot \boldsymbol{\beta_j}) \\ &= \boldsymbol{\beta_i}^T \cdot \mathbf{K_r} \cdot \boldsymbol{\beta_j} . \end{aligned}$$

Since the kernel matrix $\mathbf{K_r}$ $[m \times m]$ is positive definite ($\mathbf{F_r}$ contains linearly independent vectors) it can be decomposed by the Choleski factorization as $\mathbf{K_r} = \mathbf{R}^T \cdot \mathbf{R}$, where matrix $\mathbf{R}$ is an upper triangular matrix. We can simplify the computation of the approximated kernel function as

$$k'(\boldsymbol{x_i}, \boldsymbol{x_j}) = \boldsymbol{\beta_i}^T \cdot \mathbf{R}^T \cdot \mathbf{R} \cdot \boldsymbol{\beta_j} = \boldsymbol{\gamma_i}^T \cdot \boldsymbol{\gamma_j} , \tag{3}$$

i.e., a dot product of vectors $\boldsymbol{\gamma_i}$ and $\boldsymbol{\gamma_j}$. Now, we can represent the training set $\mathbf{X}$ mapped to the non-linear space $\mathcal{F}$ by a matrix $\boldsymbol{\Gamma} = [\boldsymbol{\gamma_1}, \ldots, \boldsymbol{\gamma_n}]$ of size $[m \times n]$ instead of the kernel matrix $\mathbf{K}$ $[n \times n]$, where m is the number of vectors $\mathbf{F_r}$ used to approximate subspace $\mathrm{span}(\mathbf{F})$ and n is the number of training vectors. When we put $\mathbf{F_r} = \mathbf{F}$ then $m = n$ and we always obtain the perfect approximation without error, i.e., $k(\boldsymbol{x_i}, \boldsymbol{x_j}) = k'(\boldsymbol{x_i}, \boldsymbol{x_j})$. The perfect approximation is obtained if $\mathrm{span}(\mathbf{F_r}) = \mathrm{span}(\mathbf{F})$, which can occur even if $m < n$, for instance when the number of training data is high and the data are linearly dependent in the space $\mathcal{F}$. However, even if $\mathrm{span}(\mathbf{F_r}) \neq \mathrm{span}(\mathbf{F})$ (actually we always select $\mathbf{F_r}$ such that $\mathrm{span}(\mathbf{F_r}) \subseteq \mathrm{span}(\mathbf{F})$ as will be explained in the sequel) we can obtain a good approximation as experiments show (see below). Let us mention that $\boldsymbol{\gamma}$ is just expression of the vector $\phi(\boldsymbol{x})' = \mathbf{F_r} \cdot \boldsymbol{\beta}$ in the orthonormal basis (columns of matrix $\mathbf{R}$ are basis vectors) of the subspace $\mathrm{span}(\mathbf{F_r})$. The next section describes an approach how to select vectors of the set $\mathbf{X_r}$ used for approximation.

3 Algorithm

Let $\mathrm{se}(\boldsymbol{x})$ denote an approximation error of non-linearly mapped the training vector $\phi(\boldsymbol{x})$ which is defined as

$$\begin{aligned} \mathrm{se}(\boldsymbol{x}) &= (\phi(\boldsymbol{x}) - \phi(\boldsymbol{x})')^T \cdot (\phi(\boldsymbol{x}) - \phi(\boldsymbol{x})') \\ &= (\phi(\boldsymbol{x}) - \mathbf{F_r} \cdot \boldsymbol{\beta})^T \cdot (\phi(\boldsymbol{x}) - \mathbf{F_r} \cdot \boldsymbol{\beta}) \\ &= k(\boldsymbol{x}, \boldsymbol{x}) - 2\boldsymbol{k_r}(\boldsymbol{x})^T \cdot \boldsymbol{\beta} + \boldsymbol{\beta}^T \cdot \mathbf{K_r} \cdot \boldsymbol{\beta} \end{aligned}$$

We propose to use a simple greedy algorithm which iteratively adds the vectors with the highest $\mathrm{se}(x)$ to the set $\mathbf{X_r}$ and iterates until the prescribed limit on the approximation error is achieved, i.e., $\mathrm{se}(\boldsymbol{x}) < \varepsilon$, $\forall \boldsymbol{x} \in \mathbf{X}$, or until allowed size m (our limitations on memory) of the set $\mathbf{X_r}$ is achieved. Such algorithm can look as follows:

Algorithm 1: **Training set approximation**

1. Initialize the $\mathbf{X_r} = [\boldsymbol{r}]$, where $\boldsymbol{r}$ is a randomly selected vector $\boldsymbol{x} \in \mathbf{X}$.
2. Iterate while the size of $\mathbf{X_r}$ is less than m:
   - Compute $\mathrm{se}(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - 2\boldsymbol{k_r}(\boldsymbol{x})^T \cdot \boldsymbol{\beta} + \boldsymbol{\beta}^T \cdot \mathbf{K_r} \cdot \boldsymbol{\beta}$ for all training vectors which are not yet included in $\mathbf{X_r}$, i.e., $\boldsymbol{x} \in \mathbf{X} \setminus \mathbf{X_r}$. It requires to compute $\boldsymbol{\beta} = \mathbf{K_r}^{-1} \cdot \boldsymbol{k_r}(\boldsymbol{x})$ where $\mathbf{K_r}$ is a kernel matrix of the current set $\mathbf{X_r}$.
   - If $\max_{\boldsymbol{x} \in \mathbf{X} \setminus \mathbf{X_r}} \mathrm{se}(\boldsymbol{x}) < \varepsilon$ then exit the algorithm else insert the $\boldsymbol{x} = \operatorname{argmax}_{\boldsymbol{x} \in \mathbf{X} \setminus \mathbf{X_r}} \mathrm{se}(\boldsymbol{x})$ to the set $\mathbf{X_r}$ and continue iterations.

The result of the Algorithm 1 is a subset $\mathbf{X_r} \subseteq \mathbf{X}$ which contains the basis vectors as well as the matrix $\mathbf{K_r}^{-1}$ useful to compute the new representation of data using (2).

When using Sherman-Woodbury formula [3] for matrix inverse $\mathbf{K_r}^{-1}$ then the computationally complexity of the algorithm is $O(nm^3)$. The Algorithm 1 does not only minimize the approximation error $\mathrm{se}(\boldsymbol{x})$ but it also minimizes the mean square error

$$\mathrm{mse} = \sum_{i=1}^{n} (\phi(\boldsymbol{x_i}) - \phi(\boldsymbol{x_i})')^T \cdot (\phi(\boldsymbol{x_i}) - \phi(\boldsymbol{x_i})') = \sum_{i=1}^{n} \mathrm{se}(\boldsymbol{x_i})$$

since it minimizes the upper bound

$$\mathrm{mse} \leq (n - m) \min_{\boldsymbol{x} \in \mathbf{X_r}} \mathrm{se}(\boldsymbol{x}) .$$

However, the approximation set found is not ensured to be the optimal one. To find the optimal approximation set one

would have to try all $\begin{pmatrix} n \\ m \end{pmatrix}$ possibilities which is computationally infeasible.

Let us mention the connection to the classical Principal Component Analysis (PCA) or Kernel Principal Component Analysis (KPCA) [11], which exactly minimizes the mean square error mse. However, the basis vectors are linear combinations of all the training data which means that the KPCA requires all training data to represent solution. The basis vectors found by the proposed method are selected vectors from the training set which is more convenient for kernel methods. Moreover, the proposed Algorithm 1 allows on-line processing of data. On the other hand, the Algorithm 1 finds only approximate solution.

Let us note that the found basis vectors can be orthogonalized on-line using well known Gram-Schmidt procedure or using the Choleski factorization which we used as described above (3).

4 Applications of training set approximation

In the following subsections we show how to use the proposed method for learning the Support Vector Machines (SVM) and Kernel Fisher Linear Discriminant (KFLD).

4.1 Approximation of SVM

The Sequential Minimal Optimizer [8] is a representative of the algorithms solving the SVM quadratic optimization problem. The advantage of this algorithm is that it does not require to store the whole kernel matrix $\mathbf{K}$ $[n \times n]$ but evaluates kernel function $k(\boldsymbol{x_i}, \boldsymbol{x_j})$ when needed. The evaluation of the kernel function $k(\boldsymbol{x_i}, \boldsymbol{x_j})$ is the bottleneck of this iterative algorithms. We use approximation of the training set by the set $\mathbf{\Gamma} = [\boldsymbol{\gamma_1}, \ldots, \boldsymbol{\gamma_n}]$ which requires $O(mn)$ memory. The approximation of value of the kernel function can be computed as $k(\boldsymbol{x_i}, \boldsymbol{x_j})' = \boldsymbol{\gamma_i}^T \cdot \boldsymbol{\gamma_j}$, which is the dot product of two m-dimensional vectors requiring $O(m)$ operations. By selecting m we can trade-off the precision of approximation and the computational/memory demands.

The SVM algorithms in general find the discriminant function in the form

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x_i}) + b = \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x})^T \cdot \phi(\boldsymbol{x_i}) + b\,,$$

where coefficients α_i are non-zero only for a subset of training patterns. The patterns with non-zero α_i are called Support Vectors (SVs). The SVM itself yields the sparse solution so that after training SVM using the training set approximation two cases can occur:

1. The number of SVs is less than the number of vectors used for training set approximation, i.e., $\mathrm{n_{SV}} < m$. In this case, it is of advantage to use the found SVs to represent the discriminant function $f(\boldsymbol{x})$.

2. The number of SVs is higher than the number of vectors used for training set approximation, i.e., $\mathrm{n_{SV}} > m$. In this case, we take the approximations of the SVs to represent the discriminant function $f(\boldsymbol{x})$ and recomputed the multipliers α_i as described bellow.

The SVs (training vectors with $\alpha_i \neq 0$) are expressed as linear combinations of the vectors $\mathbf{X_r}$ used in the approximation. The discriminant function $f(\boldsymbol{x})$ is expressed using the vectors $\mathbf{X_r}$ instead of using the SVs, i.e.,

$$\begin{aligned} f'(\boldsymbol{x}) &= \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x})^T \cdot \phi'(\boldsymbol{x_i}) + b \\ &= \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x})^T \cdot \sum_{j=1}^{m} \beta_{ij} \phi(\boldsymbol{r_j}) + b \\ &= \sum_{j=1}^{m} k(\boldsymbol{x}, \boldsymbol{r_j}) \sum_{i=1}^{n} \alpha_i \beta_{i,j} + b \\ &= \sum_{j=1}^{m} \alpha'_j k(\boldsymbol{x}, \mathbf{r_j}) + b \end{aligned}$$

where we used β_{ij} to denote the j-th component of the vector $\boldsymbol{\beta_i}$. Now we can see the vectors of the set $\mathbf{X_r}$ as a new support vectors and α'_i as the new multipliers. Using strategy described above we always have the number $\min(\mathrm{n_{SV}}, m)$ of vectors to represent the discriminant function.

Let us note that it would be of advantage to use the set of SVs as the vectors approximating the training set. The reason is that the SVs determine the SVM classifier and consequently these vectors should be well approximated. The possible solution is: (i) to find the SVM classifier, (ii) to make $\mathbf{X_r}$ from the found SVs and (iii) to train the SVM classifier again.

4.2 Approximation of KFLD

The KFLD [4, 5, 6] is a non-linear extension of the classical Fisher Linear Discriminant (FLD) using the kernel trick. Let $[(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_n}, y_n)]$ be a training set of patterns $\boldsymbol{x} \in \mathcal{X}$ and corresponding labels be $y \in [-1, +1]$. Let $\boldsymbol{\mu_1} = \frac{1}{|\boldsymbol{i_+}|} \sum_{i \in \boldsymbol{i_+}} \boldsymbol{x_i}$, $\boldsymbol{\mu_2} = \frac{1}{|\boldsymbol{i_-}|} \sum_{i \in \boldsymbol{i_-}} \boldsymbol{x_i}$ denote the mean values of patterns from the first and the second class, respectively. The $\boldsymbol{i_+}$ denotes set of indices of training patterns set with label $y = +1$, $\boldsymbol{i_-}$ with labels $y = -1$, respectively. Likewise, let $\mathbf{C_1} = \frac{1}{|\boldsymbol{i_+}|} \sum_{i \in \boldsymbol{i_+}} (\boldsymbol{x_i} - \boldsymbol{\mu_1}) \cdot (\boldsymbol{x_i} - \boldsymbol{\mu_1})^T$, $\mathbf{C_2} = \frac{1}{|\boldsymbol{i_-}|} \sum_{i \in \boldsymbol{i_-}} (\boldsymbol{x_i} - \boldsymbol{\mu_2}) \cdot (\boldsymbol{x_i} - \boldsymbol{\mu_2})^T$ be covariation matrices of the patterns from the first and the second class. The FLD aims to find such a normal vector $\boldsymbol{w}$ of the discriminant function $f(\boldsymbol{x}) = \boldsymbol{w}^T \cdot \boldsymbol{x} + b$ that maximizes the ratio

$$\boldsymbol{w} = \operatorname*{argmax}_{\boldsymbol{w'}} \frac{(\boldsymbol{w'}^T \cdot (\boldsymbol{\mu_1} - \boldsymbol{\mu_2}))^2}{\boldsymbol{w'}^T \cdot (\mathbf{C_1} + \mathbf{C_2}) \cdot \boldsymbol{w'}}\,. \tag{4}$$

The bias b can be found by one-dimensional search. When assuming normally distributed data then the FLD yields optimal Bayesian decision strategy. Moreover, when we assume equal a priori class probabilities then the bias b can be determined directly. This is used in the experiments described below.

The extension of the FLD to the non-linear discriminant function analysis can be done by the use of the kernel functions. We search for the discriminant function of the form

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x_i}) + b = \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x})^T \cdot \phi(\boldsymbol{x_i}) + b\,,$$

which is linear in the space $\mathcal{F}$. It can be shown [4] that the optimization criterion (4) can be rewritten as the quadratic optimization problem

$$\boldsymbol{\alpha} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}}\, \boldsymbol{\alpha}^T \cdot \mathbf{K} \cdot \mathbf{Z} \cdot \mathbf{K} \cdot \boldsymbol{\alpha} + C\boldsymbol{\alpha}^T \cdot \boldsymbol{\alpha} \qquad (5)$$

subject to

$$\boldsymbol{\alpha}^T \cdot \mathbf{K} \cdot \boldsymbol{e} = 2\,, \qquad (6)$$

where $\mathbf{K}$ is the kernel matrix of the set $\mathbf{X} = [\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}]$ of training patterns, C is a regularization constant, $\mathbf{Z} = \mathbf{Z}_+ + \mathbf{Z}_-$ is a matrix of size $[n \times n]$ and $\boldsymbol{e} = \boldsymbol{e}_+ + \boldsymbol{e}_-$ is a vector $[n \times 1]$. Matrices $\mathbf{Z}_+$, $\mathbf{Z}_-$ and vectors $\boldsymbol{e}_+$, $\boldsymbol{e}_-$ have a simple structure and are defined as follows:

$$\boldsymbol{e_k}(i) = \begin{cases} \frac{1}{|\boldsymbol{i_k}|} & \text{if} \quad y_i = k \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{Z_k}(i,j) = \begin{cases} 0 & \text{if} \quad (y_i \neq k) \vee (y_j \neq k) \\ -\frac{1}{|\boldsymbol{i_k}|^2} & \text{if} \quad (y_i = y_j = k) \wedge (i \neq j) \\ \frac{1}{|\boldsymbol{i_k}|} - \frac{1}{|\boldsymbol{i_k}|^2} & \text{if} \quad (y_i = y_j = k) \wedge (i = j) \end{cases}$$

Solving the optimization problem (5) yields the kernel FLD (KFLD). Now we apply the training set approximation method to this problem. The idea is to adopt an additional constrain on the solution which ensures that the solution will lie in $\operatorname{span}(\mathbf{F_r})$. Firstly, we express the discriminant function in terms of the vectors $\mathbf{X_r} = [\boldsymbol{r_1}, \ldots, \boldsymbol{r_m}]$ used for approximation as

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i' k(\boldsymbol{x}, \boldsymbol{r_i}) + b = \sum_{i=1}^{m} \alpha_i' \phi(\boldsymbol{x})^T \cdot \phi(\boldsymbol{r_i}) + b\,.$$

Secondly, we approximate the condition (6) as

$$\boldsymbol{\alpha'}^T \cdot {\mathbf{F_r}}^T \cdot \mathbf{F'} \cdot \boldsymbol{e} = \boldsymbol{\alpha'}^T \cdot \mathbf{K_r} \cdot \mathbf{B} \cdot \boldsymbol{e} = \boldsymbol{\alpha'}^T \cdot \boldsymbol{m'} = 2\,,$$

where the matrix $\mathbf{B} = [\boldsymbol{\beta_1}, \ldots, \boldsymbol{\beta_n}]$ contains approximation coefficients of the training data and $\boldsymbol{m'}$ is a vector $[m \times 1]$. Finally, we approximate the quadratic objective function of the criterion (5) as

$$\begin{aligned} & \boldsymbol{\alpha'}^T \cdot {\mathbf{F_r}}^T \cdot \mathbf{F'} \cdot \mathbf{Z} \cdot \mathbf{F'}^T \cdot \mathbf{F_r} \cdot \boldsymbol{\alpha'} \\ = \; & \boldsymbol{\alpha'}^T \cdot \mathbf{K_r} \cdot \mathbf{B} \cdot \mathbf{Z} \cdot \mathbf{B}^T \cdot \mathbf{K_r} \cdot \boldsymbol{\alpha'} \\ = \; & \boldsymbol{\alpha'}^T \cdot \mathbf{N'} \cdot \boldsymbol{\alpha'}\,, \end{aligned}$$

where $\mathbf{N'}$ is a matrix $[m \times m]$. Now we can write the approximated optimization problem

$$\boldsymbol{\alpha'} = \underset{\boldsymbol{\alpha'}}{\operatorname{argmin}}\, \boldsymbol{\alpha'}^T \cdot \mathbf{N'} \cdot \boldsymbol{\alpha'} + C\boldsymbol{\alpha'}^T \cdot \boldsymbol{\alpha'} \qquad (7)$$

subject to

$$\boldsymbol{\alpha'} \cdot \boldsymbol{m'} = 2\,,$$

where we optimize with respect to the vector $\boldsymbol{\alpha'}$ of size $[m \times 1]$. Let us note, that the approximated KFLD problem (7) requires the computation with matrix of size $[m \times m]$ which can be considerably smaller than the matrix $[n \times n]$ of the original problem (without approximation) (5).

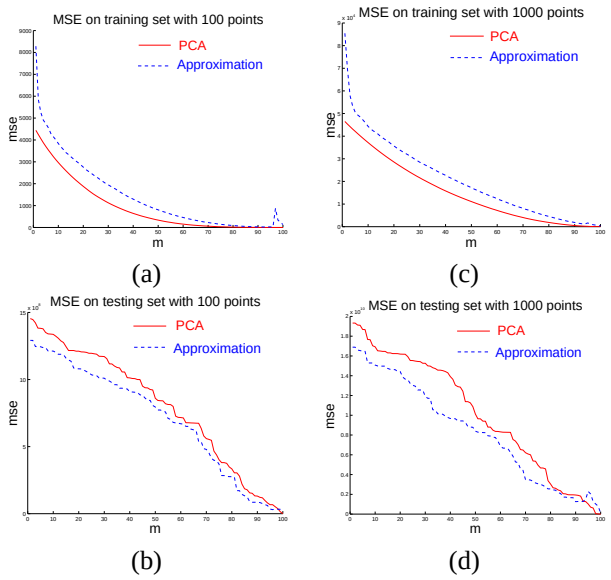

Figure 1: Comparison between the PCA and its approximation computed by the proposed approach. The figures show the mean square error mse of representation with respect to the number of basis vectors m. The assessment is done on synthetically generated training (a)(c) and testing (b)(d) data sets with $n = 100$ points (a)(b) and $n = 1000$ points (c)(d).

5 Experiments

The next subsections describe three experiments we performed to assess the proposed approach. Firstly, we tested how well the proposed Algorithm 1 can approximate the training set compared to the PCA which finds the optimal solution. Secondly, we tested usage of the training set approximation with the SVM and the KFLD learning methods on a simple synthetic problem which allows to visualize the found solution. Thirdly, we tested the proposed approach on the standard benchmark data sets.

5.1 Comparison with the PCA on synthetic data

We used the proposed approach for data representation and compare it to the standard PCA. We used synthetic data randomly generated according to the Gaussian distribution. We generated training and testing data sets containing $n = 100$ and $n = 1000$ points in 100-dimensional space. We measured the mean square error mse between the original data and their representation computed by the PCA and by the proposed approach with respect to the number of used basis vectors. The results can be seen in Figure 1. It can be seen from the results than the proposed approach is a slightly suboptimal on the training but fully comparable on the testing data in terms of the mean square error mse.

5.2 Training set approximation of SVM and KFLD tested on Riply data set

We used the approach described in Section 4 to find the SVM and KFLD classifiers on the Riply [9] data set or, more precisely, on its approximation. The Riply data sets consists of testing and training set of 2-dimensional data which can be simply visualized. We used two kernel functions: (i) RBF with $\sigma = 1$ and (ii) polynomial of degree 2. The number of

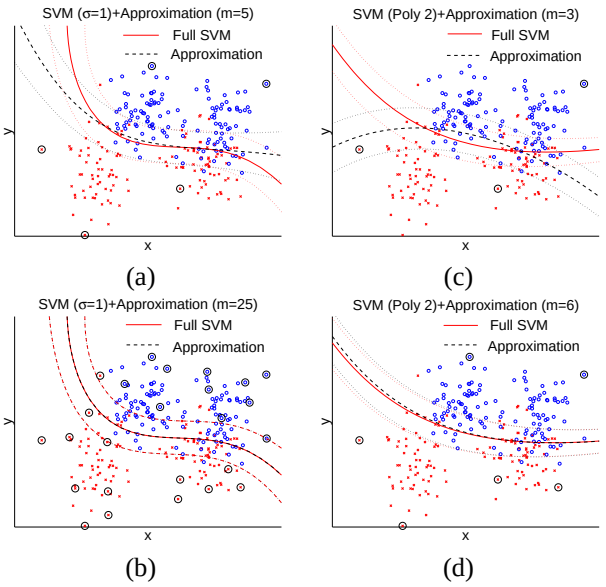

Figure 2: Decision boundaries of the SVM classifiers found on the full (solid line) and on the approximated (dashed line) training set. The big circles denote the data points used for approximation. For detailed description see Table 1 and the text.

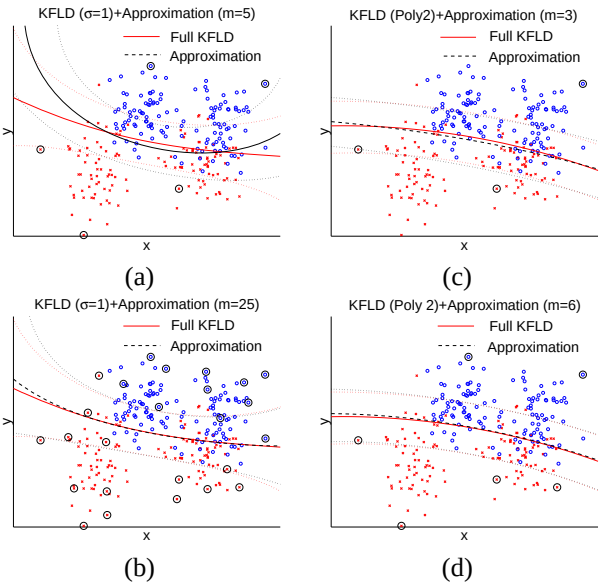

Figure 3: Decision boundaries of the KFLD classifiers found on the full (solid line) and on the approximated (dashed line) training set. The big circles denote the data points used for approximation. For detailed description see Table 1 and the text.

	RBF ($\sigma = 1$)			
	TrnErr	TestErr	n_{SV}	Fig
SVM	14.4	9.4	94	3(a)(b)
SVM+Approx ($m = 5$)	14.4	9.6	5	3(a)
SVM+Approx ($m = 25$)	14.4	9.4	25	3(b)
KLFD	14.8	10.4	250	4(a)(b)
KFLD+Approx ($m = 5$)	16.4	9.8	5	4(a)
KFLD+Approx ($m = 25$)	14.8	10.4	25	4(b)
	Polynomial ($d = 2$)			
SVM	14.0	9.9	90	3(c)(d)
SVM+Approx ($m = 3$)	14.4	13.5	3	3(c)
SVM+Approx ($m = 6$)	14.4	9.8	6	3(c)
KLFD	14.8	11.7	250	4(c)(d)
KFLD+Approx ($m = 3$)	13.6	10.8	3	4(c)
KFLD+Approx ($m = 6$)	14.8	11.8	6	4(d)

Table 1: Comparison of the SVM and the KFLD classifiers trained on full and the approximated Riply data set.

vectors used for approximation was $m = 5$ and $m = 25$ for the RBF kernel and $m = 3$ and $m = 6$ for the polynomial kernel. During tests we measured: (i) percentage of training error TrnErr, (ii) percentage of testing error TestErr and (iii) number of the support vectors n_{SV}. The overall results are enlisted in Table 1. The decision boundaries of found classifiers can be seen in Figures 2 and 3. It can be seen that the decision boundaries computed on the full training set and on its approximation with sufficient number of basis vectors almost coincide. The approximation approach preserved the classifiers accuracy but the number of support vectors n_{SV} is significantly less.

5.3 Training set approximation of SVM and KFLD tested on benchmark data sets

We tested the proposed approach described in Section 4 to find the SVM and KFLD classifier on selected problems from the IDA benchmark repository [1] to assess the proposed approach. The IDA repository contains both synthetic and real word binary problems. Each problem consists of 100 realizations of training and testing sets. The assessment is done on the all 100 realizations and all the measured values are computed as the mean values.

The Algorithm 1 used for approximation has two parameters: (i) the maximal allowed approximation error ε and (ii) the maximal number basis vectors m. In these initial experiments we set these parameters to fixed values. We set the parameter $\varepsilon = 0.001$ and $m = 0.1n$ (training set reduced to 10% of its original size) for the SVM approximation and $m = 0.25n$ (training set reduced to 25% of its original size) for the KFLD approximation.

Free parameters of both the SVM and KFLD algorithm are the argument of the used RBF kernel function $k(\boldsymbol{x_i}, \boldsymbol{x_j}) = exp(-\sigma||\boldsymbol{x_i} - \boldsymbol{x_j}||^2)$ and regularization constant C. We used first 5 realization of data (this protocol was adopted from [1]) to selected the best combination of parameters $\sigma = [2^{-8}, 2^{-7}, \ldots, 2^3]$ and $C = [2^0, 2^1, \ldots, 2^{12}]$. The pairs of arguments (σ, C) which yielded the smallest testing error were selected. During the experiments we measured the following values:

TrnErr Percentage of training errors.

TestErr Percentage of testing errors.

ker_eval The number of kernel evaluations used in the training stage which is, in fact, a measure of training time. In the case when the approximation was used that the ker_eval means the number of kernel evaluations used to compute the training set approximation by the Algorithm 1 and the number of kernel evaluations used by the training algorithm (SMO or KFLD) is enlisted in the brackets. The number in the brackets actually means the number of computations of dot products $\boldsymbol{\gamma_i}^T \cdot \boldsymbol{\gamma_j}$ which approximate the true kernel evaluations.

n_{SV} The number of the support vectors, i.e. the number of pattern which determine the classifier and directly influences the speed of classification.

Data set	Method	TrnErr	TestErr	ker_eval	n_{SV}
BREAST	SVM	20.69	25.36	2.7×10^6	116
	SVM+Approx	20.93	26.51	$7.8 \times 10^3 (264 \times 10^6)$	20
	KFLD	28.04	29.52	40×10^3	200
	KFLD+Approx	20.82	29.82	$18.8 \times 10^3 (10 \times 10^3)$	50
FLARE	SVM	32.48	32.33	8.7×10^6	570
	SVM+Approx	32.48	32.33	$49 \times 10^3 (7.5 \times 10^6)$	37
	KFLD	33.19	33.09	443.6×10^6	666
	KFLD+Approx	33.34	33.97	$49 \times 10^3 (24.6 \times 10^3)$	37
HEART	SVM	13.82	15.31	291.3×10^3	100
	SVM+Approx	13.95	15.44	$5.6 \times 10^3 (242.8 \times 10^3)$	17
	KFLD	14.43	16.31	28.9×10^3	170
	KFLD+Approx	14.06	16.53	$13.4 \times 10^3 (7.3 \times 10^3)$	42
RINGNORM	SVM	0.07	1.60	1.7×10^6	218
	SVM+Approx	1.11	1.91	$31.2 \times 10^3 (1.7 \times 10^6)$	40
	KFLD	1.43	1.49	160×10^3	400
	KFLD+Approx	1.7	2.01	$75.1 \times 10^3 (40 \times 10^3)$	100
TITANIC	SVM	19.57	22.28	4.3×10^6	85
	SVM+Approx	19.56	22.94	$3.4 \times 10^3 (350 \times 10^3)$	11
	KFLD	21.99	23.81	22.5×10^3	150
	KFLD+Approx	22.47	24.26	$2.8 \times 10^3 (1.4 \times 10^3)$	9
WAVEFORM	SVM	2.68	9.92	1.0×10^6	175
	SVM+Approx	7.14	10.47	$31.2 \times 10^3 (4.4 \times 10^6)$	40
	KLFD	6.34	10.39	160×10^3	400
	KFLD+Approx	7.26	10.80	$115 \times 10^3 (75 \times 10^3)$	100

Table 2: Comparison of the SVM and the KFLD classifiers trained on full and the approximated training sets.

The overall results of the experiments can be seen in Table 2. The experiments show that the testing error TestErr of the classifiers found on the approximated training sets equals or is slightly worse than that of the full training set. The number of kernel evaluations ker_eval used for training set approximation is significantly smaller than that used by the learning algorithm. This can speed up the learning time when the kernel evaluation is significantly more expensive than the evaluation of the dot products ${\gamma_i}^T \cdot \gamma_j$. The number of the support vectors yielded by the approximation method is significantly smaller than that without approximation. This is especially apparent in the case of the KFLD where all the training data are used to represent decision rule.

6 Conclusions

We have proposed a simple method for data set approximation and its use for learning kernel methods. The proposed method allows to reduce complexity of the found solution (this solution is sparse) as well as computational and memory demands of the learning algorithms.

The idea of this method is to represent data in lower dimensional space with possibly minimal representation error which is similar to the Principal Component Analysis (PCA). In contrast to the PCA, the basis vectors used for data representation are selected vectors from the training set and not their linear combinations. These basis vectors can be selected by a simple greedy algorithm which does not require eigenvalue decomposition (as the PCA does) and its complexity is $O(nm^3)$ where n is size of training set and m the number of the basis vectors. The algorithm is on-line in nature and allows to process huge data sets.

We tested the proposed training set approximation in connection to the Support Vector Machines and Kernel Fisher Linear Discriminant. The results obtained show that the proposed approximation can significantly reduce the number of the support vectors while retaining the accuracy of the found classifiers.

Acknowledgement

The authors were supported by the European Union projects ICA 1-CT-2000- 70002, IST-2001-32184 ActIpret, by the Czech Ministry of Education under project MSM 212300013, by the Grant Agency of the Czech Republic project 102/03/0440.

References

[1] Intelligent Data Analysis (IDA) repository. `http://ida.firts.gmd.de`.

[2] C.J.C. Burges. Simplified Support Vector Decision Rule. In *13th Intl. Conf. on Machine Learning*, pages 71–77, San Mateo, 1996. Morgan Kaufmann.

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

[4] S.. Mika, G.. Rätsch, and K.R.. Müller. A Mathematical Programming Approach to the Kernel Fisher Algorithm. In *NIPS*, pages 591–597, 2000.

[5] 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.

[6] S.. Mika, A.. Smola, and B.. Scholkopf. An Improved Training Algorithm for Kernel Fisher Discriminants. In *AISTATS 2001*. Morgan Kaufmann, 2001.

[7] E.. Osuna and F.. Girosi. *Advances in Kernel Methods*, chapter Reducing the Run-time Complexity in Support Vector Machines, pages 271–284. MIT Press, 1998.

[8] J.C.. Platt. Sequential Minimal Optimizer: A Fast Algorithm for Training Support Vector Machines. Technical Report MSR-TR-98-14, Microsoft Research, 1998.

[9] B.D.. Riply. Neural Networks and Related Methods for Classification (with discusion). *J. Royal Statistical Soc. Series B*, 56:409–456, 1994.

[10] B.. Schölkopf, P.. Knirsch, and C.. Smola, A. Burges. Fast Approximation of Support Vector Kernel Expansions, and an Interpretation of Clustering as Approximation in Feature Spaces. In R.-J.Ahler. P.Levi, M.Schanz and F.May, editors, *Mustererkennung 1998-20. DAGM-Symp.*, pages 124–132, Berlin, Germany, 1998. Springer-Verlag.

[11] B.. Scholkopf, A.. Smola, and K.R.. Muller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Technical report, Max-Planck-Institute fur biologische Kybernetik, 1996.

[12] V.. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., 1998.