Greedy Algorithm for a Training Set Reduction in the 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,hlavac}@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) [11] and the Kernel Fisher Linear Discriminant (KFLD) [5] which serve as classifier design or Kernel Principal Component Analysis (KPCA) [10] 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 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. The authors would like to thank to the anonymous reviewers for their useful comments.

- The kernel functions $k: \mathcal{X} \times \mathcal{X} \rightarrow \Re$ are used to avoid problems of high dimensionality of the space $\mathcal{F}$. The value of kernel function corresponds to the dot product of the non-linearly mapped data, i.e., $k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \phi(\boldsymbol{x_i})^T \cdot \phi(\boldsymbol{x_j})$. This implies that the algorithm must use the dot products of training data only. The matrix of all the 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 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 function $f(\boldsymbol{x})$ 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. The vector $\boldsymbol{w} \in \mathcal{F}$ is determined 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 solution $f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x_i}, \boldsymbol{x}) + b$ is not sparse, i.e., many coefficients α_i are nonzero. It can occur 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).

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 [2,9] 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 is 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 $\text{span}(\mathbf{F}) \subseteq \mathcal{F}$. 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 $\text{span}(\mathbf{F_r}) \subseteq \mathcal{F}$. We aim to express the transformed training data $\mathbf{F}$ in a 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 $\text{span}(\mathbf{F_r})$. 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

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

As the kernel matrix $\mathbf{K_r}$ is positive definite 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 $\operatorname{span}(\mathbf{F})$ 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 $\operatorname{span}(\mathbf{F_r}) = \operatorname{span}(\mathbf{F})$, 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 $\operatorname{span}(\mathbf{F_r}) \neq \operatorname{span}(\mathbf{F})$ (actually we always select $\mathbf{F_r}$ such that $\operatorname{span}(\mathbf{F_r}) \subseteq \operatorname{span}(\mathbf{F})$ 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 $\operatorname{span}(\mathbf{F_r})$. The next section describes an approach how to select vectors of the set $\mathbf{X_r}$ used for approximation.

3 Algorithm

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

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

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 $\boldsymbol{r} = \operatorname*{argmax}_{\boldsymbol{x} \in \mathbf{X}} k(x, x)$.
2. Iterate while the size of $\mathbf{X_r}$ is less than m:
   (a) 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 $\mathbf{X_r}$.
   (b) 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 since

$$\mathrm{mse} = \sum_{i=1}^{n} (\phi(\boldsymbol{x_i}) - \phi(\boldsymbol{x_i})')^T \cdot (\phi(\boldsymbol{x_i}) - \phi(\boldsymbol{x_i})') = \sum_{i=1}^{n} \mathrm{se}(\boldsymbol{x_i}) \leq (n-m) \max_{\boldsymbol{x} \in \mathbf{X} \setminus \mathbf{X_r}} \mathrm{se}(\boldsymbol{x}).$$

Step 1 can be seen as a selection of the training vector with worst approximation error when the subset $\mathbf{X_r}$ is empty, i.e., all vectors are projected onto the origin. Note, that all vectors $\phi(\boldsymbol{r})$, $\boldsymbol{r} \in X_r$ selected by the Algorithm 1 are vertices of the convex hull of the non-linearly mapped training data [1].

Let us mention the connection to the classical Principal Component Analysis (PCA) or Kernel Principal Component Analysis (KPCA) [10], 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).

[1] Thanks to J. Matas who pointed out this fact.

4 Applications of training set approximation

In this section we will describe the use of the proposed approximation approach with connection to the Support Vector Machines (SVM) and Kernel Fisher Linear Discriminant (KFLD). The SVM and the KFLD are important representatives of the methods learning the kernel classifiers.

4.1 Approximation of Support Vector Machines

The SVM aim to learn the classifier $f(\boldsymbol{x})\colon \mathcal{X} \rightarrow \{-1,+1\}$ from training data $\mathbf{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n]$ and their hidden states $\boldsymbol{y} = [y_1, ..., y_n]^T, y_i \in \{-1,+1\}$. Learning of the linear SVM classifier $f(\boldsymbol{x}) = \boldsymbol{w}^T \cdot \boldsymbol{x} + b$ is equivalent of solving the following quadratic programming task

$$\boldsymbol{w}, b = \underset{\boldsymbol{w},b}{\operatorname{argmin}} \frac{1}{2}||\boldsymbol{w}||^2 + C\boldsymbol{\xi} \cdot \boldsymbol{e}\,, \quad \text{s.t.} \quad \mathbf{Y} \cdot (\mathbf{X}^T \cdot \boldsymbol{w} + b\boldsymbol{e}) \geq \boldsymbol{e} - \boldsymbol{\xi}\,, \tag{4}$$

where $\mathbf{Y}$ is a diagonal matrix made from the vector of hidden states $\boldsymbol{y}$, $\boldsymbol{e} = [1, 1, ..., 1]^T$ and $\boldsymbol{\xi} = [\xi_1, \xi_2, ..., \xi_n]$ is a vector of slack variables. The non-linear SVM corresponds to the linear SVM learned on the non-linearly transformed training data $\mathbf{F} = [\boldsymbol{\phi}(\boldsymbol{x}_1), ..., \boldsymbol{\phi}(\boldsymbol{x}_n)]$. The non-linear SVM classifier has the form $f(\boldsymbol{x}) = w^T \cdot \phi(\boldsymbol{x}) + b = (\mathbf{F}\boldsymbol{\alpha})^T \cdot \phi(\boldsymbol{x}) + b$. Using the kernel functions we can write $||\boldsymbol{w}||^2 = \boldsymbol{\alpha}^T \cdot \mathbf{K} \cdot \boldsymbol{\alpha}$ and $\mathbf{F}^T \cdot \boldsymbol{w} = \mathbf{K} \cdot \boldsymbol{\alpha}$ which can be substituted to the (4). It results to the quadratic programming task for the non-linear SVM of the form

$$\boldsymbol{\alpha}, b = \underset{\boldsymbol{\alpha},b}{\operatorname{argmin}} \frac{1}{2}\boldsymbol{\alpha}^T \cdot \mathbf{K} \cdot \boldsymbol{\alpha} + C\boldsymbol{\xi} \cdot \boldsymbol{e}\,, \quad \text{s.t.} \quad \mathbf{Y} \cdot (\mathbf{K} \cdot \boldsymbol{\alpha} + b\boldsymbol{e}) \geq \boldsymbol{e} - \boldsymbol{\xi}\,. \tag{5}$$

The proposed method allows to find approximation of the full kernel matrix in the form $\mathbf{K}' = \boldsymbol{\Gamma}^T \cdot \boldsymbol{\Gamma}$, where $\boldsymbol{\Gamma} = [\boldsymbol{\gamma}_1, ..., \boldsymbol{\gamma}]$ is a new representation of the training data (see (3)). It can be easily shown (substituting $\mathbf{K}'$ for $\mathbf{K}$ in (5)) that solving the linear SVM (4) for the data $\boldsymbol{\Gamma}$ is equivalent to solving the non-linear SVM (5) with the approximated kernel $\mathbf{K}'$. Let $\boldsymbol{w}, b$ be the solution of (4) computed for the data $\boldsymbol{\Gamma}$. The approximated non-linear SVM classifier can be expressed as

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i k(\boldsymbol{x}, \boldsymbol{r}_i) + b \quad \text{and} \quad \boldsymbol{\alpha} = \mathbf{R}^{-1} \cdot \boldsymbol{w}\,.$$

In other words, we are able to find the approximated non-linear SVM classifier by the use of any solver for the linear SVM. Moreover, we can control the complexity of the resulting classifier since the number of the vectors defining the classifier (virtual support vectors) can be prescribed beforehand by the parameter m of the Algorithm 1.

4.2 Approximation of Kernel Fisher Linear Discriminant

The KFLD [4,5,6] is a non-linear extension of the classical Fisher Linear Discriminant (FLD) using the kernel trick. The aim here is to learn the binary

non-linear classifier $f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b$ from the given training data. It can be shown [4] that the learning of the KFLD can be expressed as the quadratic programming task

$$\boldsymbol{\alpha} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}}\, \boldsymbol{\alpha}^T \cdot \mathbf{N} \cdot \boldsymbol{\alpha} + C\boldsymbol{\alpha}^T \cdot \boldsymbol{\alpha} \quad \text{s.t.} \quad \boldsymbol{\alpha}^T \cdot \mathbf{K} \cdot \boldsymbol{e} = 2\,. \tag{6}$$

The matrix $\mathbf{N}$ is of size $[n \times n]$ is computed from the kernel matrix $\mathbf{K}$. The vector $\boldsymbol{e}$ is of size $[n \times 1]$. Solving the quadratic programming problem (6) is infeasible for large training sets. Moreover, the solution of the problem (6) is not sparse so that all the training data must be stored which results to a slow classification.

The use of the approximated kernel matrix $\mathbf{K}' = \mathbf{\Gamma}^T \cdot \mathbf{\Gamma}$ leads to the essential simplification of the problem (6). Following the derivation of the KFLD from [4], but with approximated kernel matrix $\mathbf{K}'$, yields a new quadratic programming task of the approximated KFLD. This new task has the same form as the original (6) but the matrix $\mathbf{N}$ is now of size $[m \times m]$ and the vector $\boldsymbol{e}$ is of size $[m \times 1]$. Consequently the classifier is determined by m training data. Thus we can control the complexity of the learning as well as the complexity of the resulting classifier by the parameter m of the Algorithm 1.

5 Experiments

We tested the proposed approach described in Section 4 to find the approximated SVM and KFLD classifier on selected problems from the IDA benchmark repository [1]. We used the Sequential Minimal Optimizer [8] to solve the linear SVM and the Matlab Optimization Toolbox to solve the quadratic programming task of the KFLD.

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 of basis vectors m. 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 to select 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 (i) percentage of training errors TrnErr, (ii) percentage of testing errors TestErr, (iii) number of kernel evaluations used in the training stage ker_eval, and (iv) number of the support vectors $\mathrm{n_{SV}}$. In fact, the $\mathrm{n_{SV}}$ is a measure of classification speed and ker_eval is a measure of speed of the learning stage. In the case when the approximation was used 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.

Table 1. Comparison of the SVM and the KFLD classifiers trained on full and the approximated training sets.

Data set	Method	TrnErr	TestErr	ker_eval	n_{SV}
BREAST	SVM	20.69	25.36	2.7×10^6	116
dim = 9	SVM+Approx	20.93	26.51	$7.8 \times 10^3(264 \times 10^6)$	20
$n_{trn} = 200$	KFLD	28.04	29.52	40×10^3	200
$n_{tst} = 77$	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
dim = 9	SVM+Approx	32.48	32.33	$49 \times 10^3(7.5 \times 10^6)$	37
$n_{trn} = 666$	KFLD	33.19	33.09	443.6×10^6	666
$n_{tst} = 400$	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
dim = 13	SVM+Approx	13.95	15.44	$5.6 \times 10^3(242.8 \times 10^3)$	17
$n_{trn} = 170$	KFLD	14.43	16.31	28.9×10^3	170
$n_{tst} = 100$	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
dim = 20	SVM+Approx	1.11	1.91	$31.2 \times 10^3(1.7 \times 10^6)$	40
$n_{trn} = 400$	KFLD	1.43	1.49	160×10^3	400
$n_{tst} = 7000$	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
dim = 3	SVM+Approx	19.56	22.94	$3.4 \times 10^3(350 \times 10^3)$	11
$n_{trn} = 150$	KFLD	21.99	23.81	22.5×10^3	150
$n_{tst} = 2051$	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
dim = 21	SVM+Approx	7.14	10.47	$31.2 \times 10^3(4.4 \times 10^6)$	40
$n_{trn} = 400$	KLFD	6.34	10.39	160×10^3	400
$n_{tst} = 4600$	KFLD+Approx	7.26	10.80	$115 \times 10^3(75 \times 10^3)$	100

The overall results of the experiments can be seen in Table 1. 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 ${\boldsymbol{\gamma_i}}^T \cdot \boldsymbol{\gamma_j}$. 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 approximating the kernel methods. The proposed method allows to reduce complexity of the found solution as well as computational and memory demands of the learning algorithms.

The idea of this method is to represent data in a 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 from the training set and not as 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.

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.

References

1. Intelligent Data Analysis (IDA) repository. `http://ida.first.gmd.de/~raetsch`.
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 Runtime 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. 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.
10. B. Schölkopf, A. Smola, and K.R. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Technical report, Max-Planck-Institute fur biologische Kybernetik, 1996.
11. V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., 1998.