Scaling up Kernel SVM on Limited Resources: A Low-rank Linearization Approach

Kai Zhang
Siemens Corporate Research & Technoplogy
kai-zhang@siemens.com

Liang Lan
Dept. Computer & Information Sciences
Temple University
lanliang@temple.edu

Zhuang Wang
Siemens Corporate Research & Technoplogy
zhuang.wang@siemens.com

Fabian Moerchen
Siemens Corporate Research & Technoplogy
fabian.moerchen@siemens.com

Abstract

Kernel Support Vector Machine delivers state-of-the-art results in non-linear classification, but the need to maintain a large number of support vectors poses a challenge in large scale training and testing. In contrast, linear SVM is much more scalable even on limited computing recourses (e.g. daily life PCs), but the learned model cannot capture non-linear concepts. To scale up kernel SVM on limited resources, we propose a low-rank linearization approach that transforms a non-linear SVM to a linear one via a novel, approximate empirical kernel map computed from efficient low-rank approximation of kernel matrices. We call it LLSVM (**L**ow-rank **L**inearized **SVM**). We theoretically study the gap between the solutions of the optimal and approximate kernel map, which is used in turn to provide important guidance on the sampling based kernel approximations. Our algorithm inherits high efficiency of linear SVMs and rich repesentability of kernel classifiers. Evaluation against large-scale linear and kernel SVMs on several truly large data sets shows that the proposed method achieves a better tradeoff between scalability and model representability.

Appearing in Proceedings of the 15^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume XX of JMLR: W&CP XX.

1 Introduction

Support Vector Machine (SVM) (Cortes and Vapnik, 1995) as the state-of-the-art classification algorithm has been widely applied in various scientific domains. The use of kernels allows the input samples to be mapped to a Reproducing Kernel Hilbert Space (RKHS), which is crucial to solving linearly non-separable problems. While kernel SVMs deliver the state-of-the-art results, the need to manipulate the kernel matrix imposes significant computational bottleneck, making it difficult to scale up on large data.

Many methods have been proposed for scaling up kernel SVM training: from the early work on solving the problem exactly through decomposition methods (Platt 1998; Chang et al., 2001), to the latter approximation algorithms using core vector set (Tsang et al., 2005) or incremental learning methods (Bordes et al., (2005)), to the more recent parallel computing techniques (Graf et al., 2005; Zhu et al., 2009). However, advances in kernel SVM training are still behind the steady growth of volumes of practical data sets. The latter leads to the curse of kernelization (Wang et al., 2010), where the number of support vectors that have to be explicitly maintained grows linearly with the sample size on noisy data (Steinwart, 2003). This induces huge memory cost and computational burden on model training. For example, on an OCR task with 8 million training samples, training a kernel SVM on 512 processors via parallel computing technique (Zhu et al., 2009) still takes 2 days' running time to finish. Moreover, the resulting model consists of hundreds of thousands of support vectors which brings in additional time and space burden in the prediction stage.

On the other hand, training linear SVMs on large data has regained tremendous attention recently. A key property of linear SVM is that the model weight can be

explicitly computed without having to maintain a large number of support vectors. A series of linear SVM solvers with both low memory and computational costs has been developed, including the stochastic gradient descent method (e.g. Zhang 2004, Shalev-Shwartz et al., 2007), the cutting plane method (Joachims, 2006), and the dual coordinate descent method (Hsieh et al., 2008). The linear time complexity of these algorithms, together with careful implementation and data management (Yu et al., 2011, Chang et al., 2011), allows linear SVM training on millions of samples in a matter of minutes on a regular PC.

Inspired by these successes, very recently, researchers began to explore the possibility of applying linear SVM solvers for efficient kernel SVM training. For example, Chang et al., (2010) and Sonnenburg et al., (2010) proposed to directly pre-compute the kernel mapping for training examples and then apply a linear SVM on them. While explicitly calculating the kernel map brings the benefits of computational efficiency in linear SVM, it only applies to certain types of kernels (e.g. low-degree polynomial kernel, string kernel) in which the dimensionality of the mapped feature space is low. In case of the dimensionality being blown up (e.g. using high-degree polynomial kernel, or the input space is already high-dimensional or dense), such methods lose efficiency. Rahimi and Recht (2007) proposed a novel way to approximate kernel map by using randomized features; however their methods are designed specifically for shift invariant kernels.

In this paper we pursue a general approach towards linearizing kernel SVM for large scale problems. This is achieved by efficient low-rank approximation of the kernel matrix, where the low-rank factors can be deemed as providing a novel, approximate empirical kernel map that explicitly transforms the kernel SVM into a linear space; the resultant linear SVM can then be solved efficiently using state-of-the-art linear solvers. This framework has several desirable properties. First, it can be applied to any (nonlinear) SVM variations and any PSD kernel; second, both the dimension of the approximate kernel map and the number of "basis" in the decision function can be freely controlled by the user, therefore guaranteeing efficient training and testing; third, theoretical bounds can established on the approximation, which in turn provides important guidance on sampling based low-rank approximation; last and most important, our approach inherits the rich repesentability of kernel SVM as well as the high efficiency of linear SVM, and ideally can be applied to arbitrarily large problems with limited computing resources via advanced incremental learning techniques (Yu et al., 2010).

The proposed linearization framework opens the door of tackling large scale kernel SVM by exploiting both theoretic and algorithmic advances in two rapidly developing areas: low-rank matrix decomposition and linear SVM optimization. Our preliminary empirical results have demonstrated that this is a promising direction towards building efficient yet accurate nonlinear SVM solvers on large amount of data.

The rest of this paper is organized as follows. Section 2 discusses how to transform a non-linear SVM into a linear one via matrix decomposition. In Section 3, we propose to apply Nyström low-rank approximation for efficient linearization of a non-linear SVM, and provide theoretical error analysis. In Section 4, we discuss related methods including Nyström method (Section 4.1), reduced SVM (Section 4.2), and dimension reduction (Section 4.3). Section 5 reports evaluation results. The last section concludes the paper.

2 Transforming Non-linear SVM into Linear SVM

In this section, we show that a nonlinear SVM can be cast exactly as a linear SVM via symmetric decomposition of kernel matrices. Suppose we are given a set of training pairs $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathbf{R}^{d\times 1}$'s are concatenated as rows in the $n \times d$ training data matrix $\mathbf{X}_r$ and $y_i \in \pm 1$'s are stored in the training label $\mathbf{y}_r \in \mathbf{R}^{n\times 1}$. Similarly we have m testing samples in $\mathbf{X}_e \in \mathbf{R}^{m\times d}$. Assume we use a positive semi-definite (PSD) kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \psi(\cdot), \psi(\cdot) \rangle : \mathbf{R}^d \times \mathbf{R}^d \to \mathbf{R}$, where $\psi(\mathbf{x}_i)$ is the associated mapping function that implicitly maps the data point from the input space to the feature space. Define the kernel matrix on the training and testing data in blocks as $\mathbf{K} = \begin{bmatrix} \mathbf{K}_{rr} & \mathbf{K}_{re} \\ \mathbf{K}_{er} & \mathbf{K}_{ee} \end{bmatrix}$, where $\mathbf{K}_{rr} \in \mathbf{R}^{n\times n}$ is the kernel matrix defined on $\mathbf{X}_r$, $\mathbf{K}_{ee} \in \mathbf{R}^{m\times m}$ is defined on $\mathbf{X}_e$, and $\mathbf{K}_{er} \in \mathbf{R}^{m\times n}$ is defined on $\mathbf{X}_e$ and $\mathbf{X}_r$. Training a kernel SVM is to find the classifier $f(\mathbf{x}) = sign(\mathbf{w}^\top \psi(\mathbf{x}) + b)$ by solving the optimization

$$\min_{\mathbf{w},\xi,b} \quad \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum \xi_i \qquad (1)$$
$$y_i(\mathbf{w}^\top \psi(\mathbf{x}_i) + b) \geq 1 - \xi_i,$$

where $C > 0$ is the regularization parameter. In the following we discuss how to transform the non-linear (kernel) SVM (1) into a linear SVM via decomposition of the PSD kernel matrix.

Proposition 1 *Given training data $\mathbf{X}_r$ and label $\mathbf{y}_r$, and test data $\mathbf{X}_e$. A kernel SVM (1) trained on $\mathbf{X}_r$, $\mathbf{y}_r$, and tested on $\mathbf{X}_e$ is equivalent to a linear SVM trained on $\mathbf{F}_r$, $\mathbf{y}_r$ and tested on $\mathbf{F}_e$, where*

$$\mathbf{K} = \begin{bmatrix} \mathbf{F}_r \\ \mathbf{F}_e \end{bmatrix} \begin{bmatrix} \mathbf{F}_r^\top & \mathbf{F}_e^\top \end{bmatrix} \qquad (2)$$

is any decomposition of the psd kernel matrix $\mathbf{K}$ *evaluated on* $(\mathbf{X}_r, \mathbf{X}_e)$*, and the factor* $\mathbf{F}_r \in \mathbf{R}^{n\times p}$ *and* $\mathbf{F}_e \in \mathbf{R}^{m\times p}$ *can be deemed as "virtual samples" whose dimensionality* p *is the rank of* $\mathbf{K}$*.*

Proof 1 *The dual of the kernel SVM optimization (1) can be written as*

$$\begin{aligned} \min_{\alpha} \quad & \tfrac{1}{2}\alpha^\top \mathbf{Q}_{rr}\alpha - \textstyle\sum \alpha_i \\ s.t. \quad & 0 \le \alpha_i \le C, \textstyle\sum \alpha_i y_i = 0 \\ & \mathbf{Q}_{rr} = \mathbf{K}_{rr} \odot (\mathbf{y}_r \mathbf{y}_r^\top), \end{aligned} \tag{3}$$

where α *is the Lagrangian multipliers and* $\odot$ *is the entry-wise product between matrices. The prediction on the testing data can be written as*

$$\hat{\mathbf{y}}_e = \mathbf{K}_{er} \cdot (\alpha \odot \mathbf{y}_r). \tag{4}$$

Let $\mathbf{F} = \begin{bmatrix} \mathbf{F}_r \\ \mathbf{F}_e \end{bmatrix}$*, and* $\mathbf{F}_i$ *be the* i*th column in* $\mathbf{F}^\top$*. Assume we train a linear SVM using* $\mathbf{F}_r$ *and* $\mathbf{y}_r$*, with the primal form*

$$\begin{aligned} \min_{\overline{\mathbf{w}},\overline{\xi}_i,b} \quad & \tfrac{1}{2}\|\overline{\mathbf{w}}\|^2 + C\textstyle\sum \overline{\xi}_i \\ & y_i(\overline{\mathbf{w}}^\top \mathbf{F}_i + b) \ge 1 - \overline{\xi}_i \end{aligned} \tag{5}$$

The dual can then be written as

$$\begin{aligned} \min_{\overline{\alpha}} \quad & \tfrac{1}{2}\overline{\alpha}^\top \overline{\mathbf{Q}}_{rr}\overline{\alpha} - \textstyle\sum_i \overline{\alpha}_i \\ s.t. \quad & 0 \le \overline{\alpha}_i \le C, \textstyle\sum \overline{\alpha}_i y_i = 0 \\ & \overline{\mathbf{Q}}_{rr} = (\mathbf{F}_r \mathbf{F}_r^\top) \odot (\mathbf{y}_r \mathbf{y}_r^\top). \end{aligned} \tag{6}$$

Then the prediction on $\mathbf{F}_e$ *is*

$$\bar{\hat{\mathbf{y}}}_e = \mathbf{F}_e^\top \mathbf{F}_r \cdot (\overline{\alpha} \odot \mathbf{y}_r) \tag{7}$$

Comparing (3) and (6), we can see these two problems are equivalent and lead to the identical optimal solution $\alpha^* = \overline{\alpha}^*$ *since* $\mathbf{F}_r\mathbf{F}_r^\top = \mathbf{K}_{rr}$ *(2). Plugging the optimal solutions into (4) and (7), and noting the fact* $\mathbf{F}_e^\top \mathbf{F}_r = \mathbf{K}_{er}$ *(2), we can see that the prediction in (4) and (7) are identical, i.e.,* $\hat{\mathbf{y}}_e = \bar{\hat{\mathbf{y}}}_e$*. Namely, the kernel SVM (1) and the linear SVM (5) are equivalent.*

Proposition 1 shows that any kernel SVM can be cast as an equivalent linear SVM by decomposition of the kernel matrix $\mathbf{K} = \mathbf{F}\mathbf{F}^\top$ (2), where $\mathbf{F}$ serves as an empirical kernel map or *virtual samples*. The positive semi-definiteness of the kernel matrix guarantees that decomposition (2) always exists. When only training data is used, the decomposition

$$\mathbf{K}_{rr} = \mathbf{F}_r \mathbf{F}_r^\top \tag{8}$$

allows us to recover the Lagrangian multipliers in the original nonlinear decision function (3). In the remainder of the paper, we will be mostly interested in this case (8).

Motivated by this observation, we consider learning large scale kernel SVM in two stages: first, transform it to a linear SVM using kernel eigenvalue decomposition; second, solve a linear SVM efficiently. Obviously, the key to the success of such linearization is an efficient decomposition of the PSD kernel matrix to obtain the empirical kernel map $\mathbf{F}_r$ (8). How to efficiently perform the decomposition and how it would affect the quality of the resultant classifier are the questions we will answer in the next section.

3 SVM Low-rank Linearization

The kernel matrix is the key building block of kernel methods: its entries recover the inner product of the samples in the kernel induced feature space. This avoids explicit computation of the mapping $\psi(\mathbf{x})$ (which can be potentially infinite dimensional) but instead one only needs to perform kernel evaluations in the input space. Such "kernel trick" allows the model to capture highly non-linear classification concepts, but at the cost of manipulating the $n \times n$ kernel matrix. In comparison, linear SVM assumes a simple and explicit mapping (i.e., $\psi(\mathbf{x}) = \mathbf{x}$) which renders great potential computational efficiency.

Proposition 1 provides a new perspective on the kernel map embodied through the empirical kernel matrix $\mathbf{K}$. It shows that any exact decomposition of the kernel matrix can preserve the dot products among feature induced kernel mapping $\psi(\mathbf{x}_i)$'s via a new, *empirical kernel map* $\mathbf{F}_i$'s, as

$$\mathbf{K}_{ij} = \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle = \langle \mathbf{F}_i, \mathbf{F}_j \rangle.$$

This is the key to transforming a non-linear SVM into an explicit linear counterpart. It bridges the gap between non-linear and linear SVMs and opens the possibility of training large scale non-linear SVM by advanced linear solvers.

The central question therefore is to find a suitable decomposition (8). Given an $n \times n$ kernel matrix on the training set, with the eigenvalue decomposition

$$\mathbf{K}_{rr} = \mathbf{U}_r \mathbf{\Lambda}_r \mathbf{U}_r^\top \tag{9}$$

where $\mathbf{U}_r \in \mathbf{R}^{n\times n}$ contain orthogonal eigenvectors such that $\mathbf{U}_r^\top \mathbf{U}_r = \mathbf{I}_{n\times n}$, and $\mathbf{\Lambda}_r$ is a diagonal matrix whose diagonal entries are eigenvalues $\mathbf{\Lambda}_r(ii) = \lambda_i$ in an descending order. Then the empirical kernel map on the training data (8) can be chosen as

$$\mathbf{F}_r = \mathbf{U}_r \mathbf{\Lambda}_r^{1/2}. \tag{10}$$

Theoretically, the eigenvalue decomposition provides the optimal rank-k approximation of the kernel matrix

$$\min_{rank(\widetilde{K}_{rr})=k} \|\mathbf{K}_{rr} - \widetilde{\mathbf{K}}_{rr}\|_F^2 = \sum_{i=k+1}^{n} \lambda_i^2, \tag{11}$$

where $\widetilde{\mathbf{K}}_{rr}$ is the rank-k approximated matrix. In other words, given a dimension, k, the feature map

$$\mathbf{F}_r^{(k)} = \mathbf{U}_r^{(k)}(\mathbf{\Lambda}_r^{(k)})^{1/2} \tag{12}$$

composed of top eigenvectors/values is the optimal since the inner products it recovers is the closest to $\mathbf{K}_{rr}$ among all rank-k kernel maps, which equals sum of squared minimum n–k eigenvalues as shown in (11).

However, exact computation of the top k eigenvectors requires $O(n^2k)$ time and $O(n^2)$ space, which is not suitable for large problems. So we seek an approximate decomposition here. We are interested in the Nyström method that has gained great popularity recently in scaling up kernel based algorithms (Williams and Seeger, 2001; Kumar et al., 2009). Given a set of training samples $\mathbf{X}_r$ and the kernel matrix $\mathbf{K}_{rr}$, the Nyström method chooses a subset of k samples $\mathcal{Z}$ and provides a rank-k approximation of the kernel matrix as

$$\widetilde{\mathbf{K}}_{rr} = \mathbf{K}_{rz}\mathbf{K}_{zz}^{-1}\mathbf{K}_{rz}^\top, \tag{13}$$

where $\mathbf{K}_{rz} \in \mathbf{R}^{n\times k}$ is the kernel matrix on $\mathbf{X}_r$ and $\mathbf{X}_z$, and $\mathbf{K}_{zz} \in \mathbf{R}^{k\times k}$ is the kernel matrix on $\mathcal{Z}$.

Next we show how to approximate the optimal kernel map (12) using the Nyström low-rank approximation (13). Let the eigenvalue decomposition of $\mathbf{K}_{zz}$ be $\mathbf{U}_z\mathbf{\Lambda}_z\mathbf{U}_z^\top$, then (13) can be written as

$$\begin{aligned}\widetilde{\mathbf{K}}_{rr} &= \widetilde{\mathbf{F}}_r\widetilde{\mathbf{F}}_r^\top \\ \widetilde{\mathbf{F}}_r &= \mathbf{K}_{rz}\mathbf{U}_z\mathbf{\Lambda}_z^{-1/2}.\end{aligned} \tag{14}$$

As can be seen, the rank-k approximation by the Nyström method (13) provides a natural approximation to the optimal kernel map $\mathbf{F}_r$ (12). To see this, consider the extreme case where the landmark set $\mathcal{Z}$ in the Nyström method is chosen as the whole data set: then $\mathbf{U}_z \to \mathbf{U}_r$, $\mathbf{\Lambda}_z \to \mathbf{\Lambda}_r$, $\mathbf{K}_{rz} \to \mathbf{K}_{rr}$ and as a result we have, when $|\mathcal{Z}| \to n$

$$\begin{aligned}\mathbf{K}_{rz}\mathbf{U}_z\mathbf{\Lambda}_z^{-1/2} &= \mathbf{K}_{rr}\mathbf{U}_r\mathbf{\Lambda}_r^{-1/2} \\ &= \mathbf{U}_r\mathbf{\Lambda}_r\mathbf{U}_r^\top\mathbf{U}_r\mathbf{\Lambda}_r^{-1/2} \\ &= \mathbf{U}_r\mathbf{\Lambda}_r^{1/2}.\end{aligned}$$

Namely, $\widetilde{\mathbf{F}}_r \to \mathbf{F}_r$.

Extensive work has been devoted to bound Nyström's approximation error in the form of the Frobenius norm (Drineas and Mahoney, 2005; Kumar et al., 2009)

$$\mathcal{E}_f = \left\|\mathbf{K}_{rr} - \widetilde{\mathbf{K}}_{rr}\right\|_F, \tag{15}$$

or the spectral norm (Cortes et al., 2010)

$$\mathcal{E}_2 = \left\|\mathbf{K}_{rr} - \widetilde{\mathbf{K}}_{rr}\right\|_2. \tag{16}$$

Usually these bounds are expressed as the sum of two terms: one is the rank-k approximation error via the optimal rank-k approximation (i.e. standard eigenvalue decomposition), and the other is the slackness of the approximation to the optimal rank-k approximation.

Next we analyze how Nyström low-rank approximation affects quality of the resultant SVM classifier. More specifically, we focus on the gap between the SVM model weights computed via the optimal and the Nyström based kernel maps.

Theorem 1 *Let* $\mathbf{w}$ *and* $\mathbf{w}'$ *be the solutions of the problem (1) by using the optimal and approximate rank-k empirical kernel maps,* $\mathbf{F}_r^{(k)}$ *and* $\widetilde{\mathbf{F}}_r^{(k)}$ *in (12) and (14), respectively. Then* $\|\mathbf{w} - \mathbf{w}'\|^2$ *is bounded by*

$$2C^2\rho^{\frac{1}{2}}\left[ke_f^{\frac{1}{4}} + \left(\lambda_1 tr(A) + ke_2 tr(\widetilde{\mathbf{\Lambda}}_{(k)}^{-1}))(e_f^{\frac{1}{4}} + tr(\mathbf{\Lambda}_{(k)}^2)^{1/4}\right)\right]$$

where $A \in \mathbf{R}^{k\times k}$ *is a diagonal matrix with entries* $A_{ii} = \max\left(\frac{1}{\mathbf{\Lambda}_{ii}} + \frac{1}{\widetilde{\mathbf{\Lambda}}_{ii}}, \frac{3}{\widetilde{\mathbf{\Lambda}}_{ii}} - \frac{1}{\mathbf{\Lambda}_{ii}}\right)$, $\mathbf{\Lambda} = diag(\lambda_1, ..., \lambda_n)$ *and* $\widetilde{\mathbf{\Lambda}} = diag(\widetilde{\lambda}_1, ..., \widetilde{\lambda}_k)$ *are the exact and approximate (by Nyström method) eigenvalues (sorted in descending order) of the kernel matrix,*

$$e_f = \left(\sum_{i=k+1}^{n}\lambda_i^2\right)^{1/2} + \mathcal{E}_f, \tag{17}$$

$$e_2 = \lambda_{k+1} + \mathcal{E}_2, \tag{18}$$

and $\mathcal{E}_f$ *and* $\mathcal{E}_2$ *are known error bounds ((15) and (16)) on the gap between the Nyström low-rank approximation and the original kernel matrix;* ρ *is a kernel dependent constant.*

Proof is in appendix. Theorem 1 shows the gap between solutions of the approximate and optimal rank-k kernel map is bounded by Nyström low-rank approximation error. The better the low-rank approximation, the better the potential classification performance.

We are therefore interested in an accurate low-rank approximation. The performance of Nyström method depends crucially on the sampling scheme. There are various choices: the column-norm sampling (Drineas et al., 2005) requires $O(n^2)$ time and is expensive; the ensemble Nyström method (Kumar et al., 2010) is suited for parallel processing but not on limited resources; random sampling is quite efficient but less accurate; the k-means based sampling (Zhang et al., 2008) significantly outperforms random sampling at the cost of reasonable amount of additional computations. In our experiments, we adopted a fast approximate k-means sampling using only a few iterations (which does not necessarily converge), which requires only linear time and leads to desired classification accuracy.

3.1 Algorithm and Implementation

After linearizing the kernel SVM, we apply the Selective Block Minimization framework (Chang and Ron, 2011) to handle the resultant linear SVM on large data sets that do not fit in main memory. The data is chunked into non-overlapping subsets and sequentially loaded in memory. For data in memory, the dual coordinate descent algorithm (Hsieh et al., 2008) is used to incrementally update the linear model and the dual variables. After the sub-problem (loosely) converges, a fixed portion $p\%$ of the less informative examples is removed and the next subset will be loaded. This procedure repeats until the stoping criteria is met (e.g. maximal passes of data). It has been shown that SBM will converge to the globally optimal SVM solution.

We name our approach LLSVM (low-rank linearized SVM) and summarize it in Algorithm 1. Our method has a constant space and linear time complexity in terms of the number of training examples. Testing is also quite efficient (see Table 4 for details).

Algorithm 1 LLSVM Algorithm.

Training Stage

Input: training data $\mathbf{X}_r$, label $\mathbf{y}_r$, kernel κ, regularization parameter C, landmark set size k.

Output: model $\hat{\mathbf{w}} \in \mathbf{R}^{k\times 1}$, mapping $\mathbf{M} \in \mathbf{R}^{k\times k}$.

1: select k landmark points $\mathcal{Z}$ from $\mathbf{X}_r$;
2: compute $\mathbf{K}_{zz}$ and eigendecomposition $\mathbf{U}_z\mathbf{\Lambda}_z\mathbf{U}_z^\top$;.
3: compute $\mathbf{M} = \mathbf{U}_z\mathbf{\Lambda}_z^{-1/2}$.
4: compute $\mathbf{K}_{rz}$, train linear SVM on $\mathbf{K}_{rz}\mathbf{M}$ by SBM.

Prediction Stage

Input: test data $\mathbf{X}_e$, model $\hat{\mathbf{w}}$, mapping matrix $\mathbf{M}$.

Output: predicted labels $\hat{\mathbf{y}}_e$

1: compute $\mathbf{K}_{ez}$;
2: predict by $\hat{\mathbf{y}}_e = \hat{\mathbf{w}}^\top\ \mathbf{K}_{ez}\mathbf{M}$.

4 Connections with Other Work

4.1 Nyström Related Methods

The Nyström method has been very popular in scaling up kernel based algorithms. Examples include the computation of large kernel matrix inverse (Williams et al., 2004), the eigenvectors of dominant eigenvalues (Fowlkes et al., 2005) and manifold learning (Kumar et al., 2008; Zhang et al., 2010). Recently, Cortes et al., (2010) study the impact of low-rank approximation of kernel matrix on kernel based classifiers (kernel regression, graph-based algorithms, and SVM), and also in the specific context of Nyström low-rank approximation. Their theoretical analysis is based on the following empirical kernel map

$$\Phi(\mathbf{x}) = (\mathbf{K}^\dagger)^{1/2} \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}) \\ k(\mathbf{x}_2, \mathbf{x}) \\ \vdots \\ k(\mathbf{x}_n, \mathbf{x}) \end{bmatrix} \tag{19}$$

where $\mathbf{K}^\dagger$ denotes the pseudo-inverse. As can be seen, this is equivalent to using $(\mathbf{K}^\dagger)^{1/2}\mathbf{K} = \mathbf{K}^{1/2}$ as virtual training samples in a linear SVM. Since $\mathbf{K}^{1/2} = \mathbf{U}\mathbf{\Lambda}^{1/2}\mathbf{U}^\top$ is an $n \times n$ matrix, this requires $n \times n$ space and may not directly improve the computational efficiency of SVM training. Similar problem appears when a low-rank approximated kernel matrix is considered, i.e., the resultant virtual samples $\widetilde{\mathbf{K}}^{1/2} = \widetilde{\mathbf{U}}\widetilde{\mathbf{\Lambda}}^{1/2}\widetilde{\mathbf{U}}^\top$ is still $n \times n$ matrix, where $\widetilde{\mathbf{U}} \in \mathbf{R}^{n\times k}$ are the approximate eigenvectors and $\widetilde{\mathbf{\Lambda}} \in \mathbf{R}^{k\times k}$ are approximate eigenvalues. In comparison, the empirical kernel map we proposed (14) has a much lower dimensionality $n \times k$ ($k \ll n$) and will directly benefit SVM training computationally.

Early work (Fine et al., 2001) on SVM training has studied using the low-rank approximation method (Smola and Schölkopf, 2000; Willams and Seeger, 2001) to speed up the inverse of the kernel matrix for the inner solver - the interior point method. Instead, our approach applies the low-rank approximation to transfer a non-linear SVM into a linear one that prevents the expensive matrix manipulation at the first place. Moreover, we study the impact of using such approximation on the quality of the resulting model.

4.2 Reduced SVM

Lee and Mangasarian (2000), Lin and Lin (2003) found that non-linear SVMs can be (approximately) transferred to linear SVMs in the context of Reduced SVM, where the goal is to reduce the number of support vectors in the learned model. To achieve this, the decision function is restricted to be spanned by only a small subset of samples $\mathcal{Z}$,

$$\mathbf{w} = \sum_{i\in\mathcal{Z}} y_i\alpha_i\psi(\mathbf{x}_i). \tag{20}$$

After adding a penalty term b^2 in the primal objective the reduced SVM optimization problem is written as

$$\begin{aligned} \min_{\mu_z,\xi} \quad & \tfrac{1}{2}(\mu_z^\top \mathbf{K}_{zz}\mu_z) + \tfrac{1}{2}b^2 + C\textstyle\sum_{i=1}^n \xi_i \\ s.t. \quad & \mathbf{K}_{rz}\mu_z + b\mathbf{y}_r \geq e - \xi \end{aligned} \tag{21}$$

where $\mu_z = \mathbf{y}_z \odot \alpha_z$, $\mathbf{y}_z$ and α_z are the training labels and Lagrangian multipliers corresponding to the subset $\mathcal{Z}$, $\mathbf{K}_{zz}$ is the kernel matrix evaluated on $\mathcal{Z}$, and $\mathbf{K}_{rz}$ is the kernel matrix between the training data $\mathbf{X}_r$ and the subset $\mathcal{Z}$, and e is a vector of all 1's.

Due to the Hessian matrix $\mathbf{K}_{zz}$, it is not easy to solve the QP (21) for large problems. Therefore, Lee and Mangasarian (2000) proposed to remove $\mathbf{K}_{zz}$ from (21) which reduces it to a linear SVM and can therefore be solved efficiently. However, empirically, removing $\mathbf{K}_{zz}$ will lead to inferior performance than solving (21) exactly. In the following, we show that an exact solution can be obtained efficiently using our low-rank linearization approach.

Proposition 2 *Consider the reduced SVM (21). Perform eigenvalue decomposition* $\mathbf{K}_{zz} = \mathbf{U}_z\mathbf{\Lambda}_z\mathbf{U}_z^\top$. *Then (21) can be solved exactly by a linear SVM using* $\mathbf{K}_{rz}\mathbf{U}_z\mathbf{\Lambda}_z^{-1/2}$ *as training samples. Testing on* $\mathbf{X}_e$ *can be performed by applying the learned linear model on* $\mathbf{K}_{ez}\mathbf{U}_z\mathbf{\Lambda}_z^{-1/2}$.

Proof 2 *Given the eigenvalue decomposition of* $\mathbf{K}_{zz}$, *define a linear transform*

$$\beta_z = \mathbf{\Lambda}_z^{1/2}\mathbf{U}_z^\top\mu_z. \tag{22}$$

Notice that $\mathbf{\Lambda}_z$ *is a diagonal matrix whose diagonal entries are the eigenvalues. If the eigenvalues are all strictly positive, then* $\mathbf{\Lambda}_z$ *is invertible. By noting that* $\mathbf{U}_z\mathbf{U}_z^\top = \mathbf{U}_z^\top\mathbf{U}_z = \mathbf{I}$, *we can invert the above transform as*

$$\mu_z = \mathbf{U}_z\mathbf{\Lambda}_z^{-1/2}\beta_z. \tag{23}$$

Plugging (22) and (23) into (21), we arrive at the following problem

$$\begin{aligned} \min_{\beta_z,\xi} \quad & \tfrac{1}{2}\|\beta_z\|^2 + \tfrac{1}{2}b^2 + C\textstyle\sum_{i=1}^n \xi_i \\ s.t. \quad & (\mathbf{K}_{rz}\mathbf{U}_z\mathbf{\Lambda}_z^{-1/2})\beta_z + b\mathbf{y}_r \geq e - \xi \end{aligned} \tag{24}$$

which is a linear SVM trained on $\mathbf{K}_{rz}\mathbf{U}_z\mathbf{\Lambda}_z^{-1/2}$.

Using (20) the prediction of RSVM (21) on testing set will be $\mathbf{K}_{ez}\mu_z$. *Since* $\mu_z = \mathbf{U}_z\mathbf{\Lambda}_z^{-1/2}\beta_z$ *(23), the prediction will be equivalent to* $(\mathbf{K}_{ez}\mathbf{U}_z\mathbf{\Lambda}_z^{-1/2})\beta_z$. *Namely, it can be deemed as applying the learned model* β_z *on "transformed" test samples* $\mathbf{K}_{ez}\mathbf{U}_z\mathbf{\Lambda}_z^{-1/2}$.

If $\mathcal{Z}$ *has no duplicate samples, the kernel matrix* $\mathbf{K}_{zz}$ *will be positive definite for Gaussian kernel (Micchelli 1986). Namely eigenvalues in* $\mathbf{\Lambda}_z$ *are strictly positive and exact solution can be recovered. In case other kernel function is adopted and zero eigenvalues might appear, one can simply replace diminishing eigenvalues with a small constant when using (23).*

Proposition 2 provides a new way for efficiently solving RSVM by transforming it to an equivalent linear SVM. In case the sub-kernel matrix $\mathbf{K}_{zz}$ is strictly positive definite, the solution is exact, and does not lose any information in comparison with those used by Lee and Mangasarian (2000) and Lin and Lin (2003). Another advantage is that our method applies to any SVM formulation using PSD kernel, while the RSVM is designed for a certain SVM variant.

Table 1: The empirical kernel map used in different methods ($|\mathcal{Z}| = k \ll n$).

Method	kernel map	dimension
(Cortes *et al.*, 2010)	$\mathbf{K}_{rr}^{1/2}$	$n \times n$
RSVM	$\mathbf{K}_{rz}$	$n \times k$
LLSVM (ours)	$\mathbf{K}_{rz}\mathbf{U}_z\mathbf{\Lambda}_z^{-1/2}$	$n \times k$

As can be seen, the new way for solving the RSVM (Proposition 2) has the same form as the low-rank linearized SVM proposed in Section 3. The important consequence of this connection is that the tremendous work on landmark selection in the Nyström method (Williams and Seeger, 2001; Zhang and Kwok, 2008) method can be borrowed directly for efficient RSVM solutions. In comparison, the original RSVM simply uses random sampling, or couples the subset selection with sparse SVM training which can be quite expensive on large data sets (Wu et al., 2006).

Another interesting observation about RSVM is that after dropping the Hessian $\mathbf{K}_{zz}$ in (21), the resultant problem is equivalent to a linear SVM trained on $\mathbf{K}_{rz}$, namely, each training sample is re-expressed by its similarity with k landmark points in $\mathcal{Z}$. In Table 1, we list the empirical kernel maps (and their dimensions) used in related methods (Cortes et al., 2010; Lee and Mangasarian, 2000; Lin and Lin, 2003).

4.3 Non-linear Dimension Reduction

Our kernel map (11) can be deemed as approximately projecting samples onto principal components as in Kernel PCA (Schölkopf et al., 1998). However, instead of using KPCA for pre-processing, our contribution is on rigorously studying the linearization of kernel SVM, performing theoretical analysis and designing large scale solvers, in which we believe as a new contribution to bridge non-linear and linear SVMs via kernel eigenvalue decompositions.

5 Experiments

This section reports evaluations on the performance of the proposed method and competing algorithms.

Competing algorithms:

- SBM: the state-of-art linear SVM solver when data cannot fit into memory (Chang and Ron, 2011);

Table 2: Summary of the data sets used in our evaluations.

dataset	training size	testing size	dimension	number of classes	file size
IJCNN	49,990	91,701	22	2	20.9M
webspam	280,000	70,000	254	2	327M
covType	464,810	116,202	54	2	68.1M
Ncheckerboard	800,000	20,000	2	2	22.7M
mnist8m-b	8,000,000	100,000	784	2	18G

Table 3: Training time (in seconds/hours) and test error rate of different algorithms (exclude I/O time; '–' marks the algorithms cannot finish within 24h).

Datasets	IJCNN		webspam		covtype		Ncheckerboard		mnist8m-b	
Algorithms	time	err(%)	time	err(%)	time	err(%)	time	err(%)	time	err(%)
Linear[c]	2s	7.87±0.01	47s	6.96±0.01	16s	23.75±0.00	9s	48.90±0.54	12h	24.18±0.39
Poly2[a]	11s	2.16	0.9h	1.56	1.4h	19.91	17s	44.02	–	–
Libsvm[a]	27s	1.31	4.3h	0.80	15h	3.92	–	–	–	–
CVM[c]	80s	**1.20±0.00**	4.7h	0.78±0.00	9h	**2.50**	–	–	–	–
Lasvm[c]	24s	1.47±0.00	6.3h	**0.75±0.00**	–	–	–	–	–	–
AMM[b]	2s	2.4±0.11	80s	4.50±0.24	84s	24.02±0.31	41s	35.98±4.09	5.7h	3.40±0.33
LLSVM[c]	49s	1.24±0.01	0.8h	2.04±0.05	1.3h	14.95±0.13	429s	**0.59±0.00**	22h	**2.59**±0.01
#landmark	($k = 3000$)		($k = 4000$)		($k = 4000$)		($k = 1000$)		($k = 3000$)	

[a]Results from (Chang et al., 2010); machine configuration: 2.5G Hz Xeon L5420 processor, 16G RAM.
[b]Results from (Wang et al ., 2011); machine configuration: 3.0G Hz Intel Xeon processor, 16G RAM.
[c]Results obtained on an Intel(R) Q9400 PC with 2.66G Hz processor and 4G RAM.

- Libsvm: a popular kernel SVM solver by the decomposition method (Chang and Lin, 2011);
- Poly2: a fast solver for polynomial degree 2 SVM by Liblinear (Chang et al., 2010);
- CVM: A popular approximate solver for large-scale kernel SVMs (Tsang et al., 2005) ;
- Lasvm: a large-scale approximate online SVM solver (Bordes et al., 2005);
- AMM: a recent SVM-like algorithm which captures non-linear concepts through multiple linear weights (Wang et al., 2011).

Datasets: we use 5 large benchmark data sets[1] which is summarized in Table 2. We transfer the original multi-class mnist8m into binary mnist8m-b by using round digits (3, 6, 8, 9, 0) versus non-round digits (1, 2, 4, 5, 7). NcheckerBoard is a noisy version of Checkerboard (4×4 XOR problem) with 20% randomly swapped labels (but we still use the noisy-free Checkerboard as test data).

Setup: we use the RBF kernel for Libsvm, CVM, Lasvm and our algorithm (Algorithm 1). In our algorithm, we set $p\% = 75\%$ and the size of subset equal to 20,000 for SBM for all the experiments. For efficiency, we only run k-means clustering approximately using the first 20,000 data points. Due to all the studied datasets are large, we set our algorithm scan the data in a single pass. Our algorithm is implemented in C. Due to the huge computational cost, results on some method/dataset were taken directly from some recent publications.

For non-deterministic algorithms with variations in the results, we repeat experiments five times and report the mean and standard deviation of the error rate. For our algorithm, instead of fixing a constant number of landmark points k for all the data sets, we explore k in the range $[50, 4000]$. The training time and error rate as a function of k is plotted in Figure 1 for all the data sets, and the results with the best balance between training time and accuracy are reported in Table 2. We select the hyper-parameters for all the algorithms through 10-cross-validation.

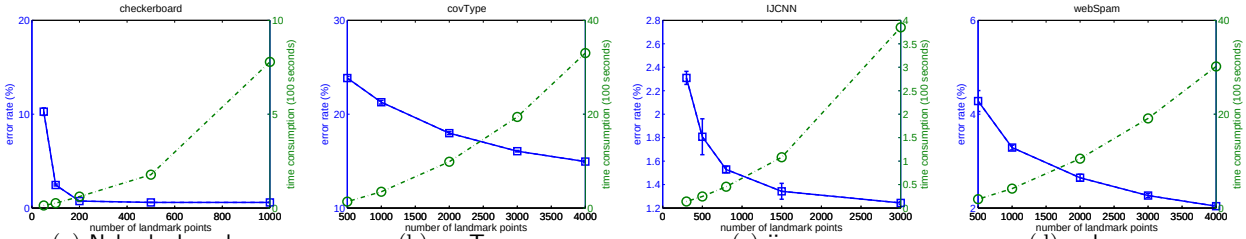
Results: the error rate and training time (excluding I/O time) of all the algorithms are summarized in Table 3. We use '–' to mark the results that cannot be finished within 24 hours. In terms of accuracy, we can see that our algorithm significantly outperforms linear SVM on all the data sets. Regarding the training time, our method is slower than the linear SVM but is still affordable considering our machine configuration. Compared with the kernel SVM solvers (Lib-

[1]ijcnn, webspam,covtype and mnist8m are available at http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

Table 4: prediction complexity on an instance

Method	LinearSVM	RBFSVM	Poly2SVM	AMM	LLSVM (ours)
Complexity[a]	$O(c_1 d)$	$O(c_2 \|S\|)$	$O(c_1 d^2)$	$O(c_1 md)$	$O(c_1 k^2 + c_2 k)$

[a] d is the feature dimensionality; $|S|$ is the number of support vectors; k is the number of landmark points in our method where $k \ll n$ with n being the training size; m is the number of linear weights in AMM; c_1 is the time for a scalar product; c_2 is the time for evaluating a RBF kernel function, where $c_1 \ll c_2$.



(a) Ncheckerboard (b) covType (c) ijcnn (d) webspam

Figure 1: Error rate and training time as a function of k (the number of landmark points).

svm, CVM, Lasvm), our method delivers comparable error rates to the optimum achievable on each data set (expect covType) but consumes much less training time. Considering that in covType task nearly half of the training examples (around 230 thousands) become support vectors for a non-linear SVM, our error rate achieved by using only 4 thousands landmark points (or basis) is quite acceptable. More importantly, neither the exact solver (Libsvm) nor the two popular approximate large-scale SVM solvers (CVM and Lasvm) can finish the training on the largest data set mnist8m-b or the noisiest data set NcheckBoard within 24 hours due to the curse of kernelization; while our method can easily scale up on such data (actually even on arbitrarily large data sets) using a daily life PC. Comparing with the most scalable non-linear algorithm AMM, our method has significantly lower error rate and is only several times slower. It is worth to note that our results are produced on a machine that is less powerful than those used by Chang et al., (2010) for Libsvm, Poly2 and by Wang et al., (2011) for AMM.

From Figure 1, we can observe that the error rate converges as the number of landmark points k grows. For some data sets, a few hundreds landmark points are sufficient to achieve satisfactory accuracy; while on some difficult tasks more landmark points are needed for a good accuracy. As expected, the training time grows quadratically as k increases.

In table 4 we list the prediction complexity of different algorithms. The SVM using RBF kernel is the most expensive, which involves extensive kernel evaluations that in turn grows linearly with the number of support vectors which can be quite large on noisy data. On the other hand, the linear models such as Linear, Poly2 and AMM have much lighter prediction burden which only takes a number of scalar product operations. The prediction complexity of our method is somewhere in between and can be freely controlled by user-defined parameter k. When k is small the kernel evaluation dominates the cost; when k is large the transformation (multiplication with the $k \times k$ mapping matrix $\mathbf{M}$ in Algorithm 1) dominates.

6 Conclusion and Future Work

In this paper, we proposed a novel approach for linearizing kernel SVM via symmetric kernel low-rank decomposition for large scale learning problems. This allows us to fully exploit algorithmic advances in the rapidly developing areas of linear SVM optimization and low-rank approximation, and to build highly efficient solvers for large scale non-linear SVM on limited computing resources. Encouraging results are observed on large scale benchmark data sets.

In the future, we will apply general non-linear dimension reduction schemes to linearize non-linear SVMs, and extend the proposed idea to semi-supervised learning. Theoretically, we will resort to numerical perturbation analysis to provide tighter bounds on the solution of the QP problem with low-rank kernels. We will also incorporate metric learning in the linearization step for more robust nonlinear classification.

Parallelization. Though working on single PC now, our approach is very parallelization friendly, since both low-rank approximation and linear SVM can have parallel solutions. This brings interesting future topics.

Appendix

Proof 3 *Define the difference matrix* $\mathbf{E}^{(k)} = \mathbf{K}_{rr}^{(k)} - \widetilde{\mathbf{K}}_{rr} = \mathbf{F}_r^{(k)}(\mathbf{F}^{(k)})^\top - \widetilde{\mathbf{F}}_r^{(k)}(\widetilde{\mathbf{F}}_r^{(k)})^\top$, *where* $\mathbf{K}_{rr}^{(k)}$ *is the optimal rank-k reconstruction of* $\mathbf{K}_{rr}$ *using top k eigenvectors and eigenvalues, and* $\widetilde{\mathbf{K}}_{rr}$ *is the Nyström rank-k approximation. Let* $e_f = \|\mathbf{E}^{(k)}\|_F$, $e_2 = \|\mathbf{E}^{(k)}\|_2$.

Note that by definition and properties of matrix Frobenius norm, $\|\mathbf{K}_{rr}^{(k)} - \mathbf{K}_{rr}\|_F^2 = tr((\mathbf{\Lambda}^2)^{>k})$, *where* $\mathbf{\Lambda}$ *is diagonal eigenvalue matrix of* $\mathbf{K}_{rr}$ *and* $tr((\mathbf{\Lambda}^2)^{>k}) = \sum_{p=k+1}^{n} \lambda_p^2$. *Also recall that in the Nyström method we have* $\|\mathbf{K}_{rr} - \widetilde{\mathbf{K}}_{rr}\|_F \leq \mathcal{E}_f$. *By simple triangular inequality, we have (17). Similarly,* $\|\mathbf{K}_{rr}^{(k)} - \mathbf{K}_{rr}\|_2 = \lambda_{k+1}$ *by definition of the spectral norm, and* $\|\mathbf{K}_{rr} - \widetilde{\mathbf{K}}_{rr}\|_2 \leq \mathcal{E}_2$ *from the Nyström method, so using triangular inequality we have (18).*

Next we use e_2 *and* e_f *to bound* $\|\mathbf{w} - \mathbf{w}'\|_2$. *Note that the approximate kernel map (14) can be written as* $\widetilde{\mathbf{F}}_r^{(k)} = \widetilde{\mathbf{U}}^{(k)}(\widetilde{\mathbf{\Lambda}}^{(k)})^{1/2}$, *where* $\widetilde{\mathbf{U}}^{(k)} = \mathbf{K}_{rz}\mathbf{U}_z\mathbf{\Lambda}_z^{-1}$ *are Nyström approximate eigenvectors, and* $\widetilde{\mathbf{\Lambda}}^{(k)}$ *are approximate eigenvalues. Then we have*

$$\|\Delta_u\|_F = \|\mathbf{U}^{(k)} - \widetilde{\mathbf{U}}^{(k)}\|_F \leq \lambda_1 tr(A) + e_2 tr(\widetilde{\mathbf{\Lambda}}_{(t)}^{-1}),$$

$$\|\Delta_\lambda\|_F = \|(\mathbf{\Lambda}^{(k)})^{1/2} - (\widetilde{\mathbf{\Lambda}}^{(k)})^{1/2}\|_F \leq e_f^{\frac{1}{4}}.$$

where A is as defdiagonal matrix with entries $A_{ii} = \max\left(\frac{1}{\mathbf{\Lambda}_{ii}} + \frac{1}{\widetilde{\mathbf{\Lambda}}_{ii}}, \frac{3}{\widetilde{\mathbf{\Lambda}}_{ii}} - \frac{1}{\mathbf{\Lambda}_{ii}}\right)$, $tr(\widetilde{\mathbf{\Lambda}}_{(k)}^{-1}) = \sum_{i=1}^{k} 1/\widetilde{\mathbf{\Lambda}}_{ii}$. *The first inequality is based on the equation (7) of (Zhang 2006); the second is based on the well known Hoffman-Wielandt inequality as well as the lemma 1 in (Cortes 2010). With the above bounds, and note that* $\|\mathbf{U}^{(k)}\|_F = k$, $\|\mathbf{\Lambda}^{(k)}\|_F = \sqrt{\sum_{i=1}^{k} \mathbf{\Lambda}_{ii}^2} = \sqrt{tr(\mathbf{\Lambda}_{(k)}^2)}$, *we have*

$$\begin{aligned} &\|\mathbf{F}_r^{(k)} - \widetilde{\mathbf{F}}_r^{(k)}\|_F \qquad (25)\\ &= \left\|\mathbf{U}^{(k)}(\mathbf{\Lambda}^{(k)})^{1/2} - (\mathbf{U}^{(k)} - \Delta_u)((\mathbf{\Lambda}^{(k)})^{1/2} - \Delta_\lambda)\right\|_F \\ &\leq \|\mathbf{U}^{(k)}\Delta_\lambda\|_F + \|\Delta_u(\mathbf{\Lambda}^{(k)})^{1/2}\|_F + \|\Delta_u\Delta_\lambda\|_F \\ &\leq k e_f^{\frac{1}{4}} + \left(\lambda_1 tr(A) + k e_2 tr(\widetilde{\mathbf{\Lambda}}_{(k)}^{-1})(e_f^{\frac{1}{4}} + tr(\mathbf{\Lambda}_{(k)}^2)^{1/4}\right) \end{aligned}$$

Let exists $\rho \geq 0$ *such that* $\kappa(\mathbf{x}, \mathbf{x}) \leq \rho$. *Then it can be shown that (Cortes et al., 2010)*

$$\begin{aligned} \|\mathbf{w} - \mathbf{w}'\|^2 &\leq 2C^2\rho^{\frac{1}{2}} \sum \|\Phi(\mathbf{x}_i) - \Phi'(\mathbf{x}_i)\| \\ &\leq 2C^2\rho^{\frac{1}{2}} \|\mathbf{F}_r^{(k)} - \widetilde{\mathbf{F}}_r^{(k)}\|_F \qquad (26) \end{aligned}$$

By plugging (25) in (26), we complete the proof.

References

M. Belkin and P. Niyogi (2003). Laplacian Eigenmaps for Dimensionality Reduction and Data Representation, *Neural Computation* 15(6):1373 – 1396.

Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard and C.-J. Lin (2010). Training and Testing Low-degree Polynomial Data Mappings via Linear SVM. *Journal of Machine Learning Research* 11:1471 – 1490.

C.-C. Chang and C.-J. Lin (2011). LIBSVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2(27):1–27.

K.-W. Chang and D. Roth (2011). Selective Block Minimization for Faster Convergence of Limited Memory Large-Scale Linear Models, *ACM SIGKDD Conference on Knowledge Discovery and Data Mining.*

O. Chapelle, J. Weston and B. Schölkopf (2003). Cluster Kernels for Semi-Supervised Learning, *Advances in Neural Information Processing Systems*, 585-592.

C. Cortes, M. Mohri and A. Talwalkar (2010). On the Impact of Kernel Approximation on Learning Accuracy, *International Conference on Artificial Intelligence and Statistics.*

C. Cortes and V. Vapnik (1995). Support-vector networks. *Machine Learning.*

P. Drineas and M. Mahoney (2005). On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-based Learning. *Journal of Machine Learning Research*, 6, 2153–2175.

S. Fine, K. Scheinberg, N. Cristianini, J. Shawe-taylor and B. Williamson (2001). Efficient SVM Training Using Low-rank Kernel Representations. *Journal of Machine Learning Research* 2:243 – 264.

C. Fowlkes, S. Belongie, F. Chung and J. Malik (2004). Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:214–225.

H. P. Graf, E. Cosatto, L. Bottou, I. Durdanovic and V. Vapnik (2005). Parallel Support Vector Machines: The Cascade SVM, *Advances in Neural Information Processing Systems*, 521–528.

C-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi and S. Sundararajan (2008). A Dual Coordinate Descent Method for Large-scale Linear SVM, *International Conference on Machine Learning.*

T. Joachims (2006). Training linear SVMs in linear time. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 217-226.

S. Kumar, M. Mohri, and A. Talwalkar (2010). Ensemble Nystrm̈ Method. *Advances in Neural Information Processing Systems.*

S. Kumar, M. Mohri, and A. Talwalkar (2009). Sampling techniques for the Nyström method. *International Conference on Artificial Intelligence and Statis-*

tics.

Y. J. Lee and O. L. Mangasarian (2000). RSVM: Reduced Support Vector Machines, *Technical Report* Computer Science Department, University of Wisconsin, Madison.

K.-M. Lin and C.-J. Lin (2003). A Study on Reduced Support Vector Machines *IEEE Transactions on Neural Networks*, 14(6):1449–1459.

C. A. Micchelli (1986). Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions. *Constructive Approximation* 2, 11–22.

A.Y. Ng, M.I. Jordan and Y. Weiss (2001). On Spectral Clustering: Analysis and an algorithm, *Advances in Neural Information Processing Systems*, 849–856.

J. C. Platt (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, Advances in Kernel Methods: Support Vector Learning, pages 185 – 208. MIT Press, Cambridge, MA.

A. Rahimi and B. Recht (2007). Random features for large-scale kernel machines. *In Advances in Nueral Information Processing Systems*.

B. Schölkopf, J. Smola and K.R. Müller (1998). Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation* 10(5):1299–1319.

S. Shalev-Shwartz, Y. Singer and N. Srebro, (2007). Pegasos: Primal Estimated Sub-gradient Solver for SVM. *International Conference on Machine Learning*.

A. J. Smola and B. Schölkopf (2000). Sparse greedy matrix approximation for machine learning. *International Conference on Machine Learning* 911–918.

S. Sonnenburg and V. Franc (2010). COFFIN : A Computational Framework for Linear SVMs, *International Conference on Machine Learning*.

I.W. Tsang, J. Kwok and P.M. Cheung (2005). Core Vector Machines: Fast SVM Training on Very Large Data Sets, *Journal of Machine Learning Research* 6:363–392.

Z. Wang, N. Djuric, K. Crammer and S. Vucetic (2011). Trading Representability with Scalability: Adaptive Multi-Hyperplane Machine for Nonlinear Classification, *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Z. Wang, K. Crammer and S. Vucetic (2010). Multi-Class Pegasos on a Budget, *International Conference on Machine Learning*.

C. Williams and M. Seeger. (2001). Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems* 682–688.

M. Wu, B. Schölkopf and G. Bakir (2006). A Direct Method for Building Sparse Kernel Learning Algorithms. *Journal of Machine Learning Research* 7:603–624.

H.-F. Yu, C.-J. Hsieh, K.-W. Chang, and C.-J. Lin (2010). Large linear classification when data cannot fit in memory. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

T. Zhang (2004). Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms, *International Conference on Machine Learning*, 919–926.

K. Zhang and J. Kwok (2006). Block-quantized kernel matrix for Fast Spectral Embedding, *International Conference on Machine Learning*.

K. Zhang, I. Tsang, and J. Kwok (2008). Improved Nyström Low-rank Approximation and Error Analysis, *International Conference on Machine Learning*.

K. Zhang and J. Kwok (2010). Clustered Nyström Method for Large Scale Manifold Learning and Dimension Reduction. *IEEE Transactions on Neural Networks*, 21(10):1576 – 1587.

Z. A. Zhu, W. Chen, G. Wang, C. Zhu, and Z. Chen (2009). P-packsvm: parallel primal gradient descent kernel svm, *IEEE International Conference on Data Mining*.