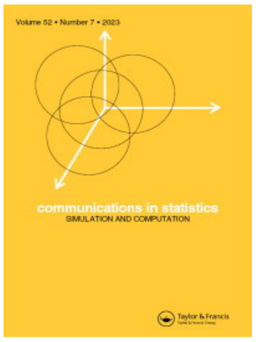
A novel algorithm for classification using a low rank approximation of kernel-based support vector machines with applications

Chatrabgoun, O., Esmaeilbeigi, M., Daneshkhah, A., Kamandi, A. & Salimi, N.

DOI 10.1080/03610918.2023.2236816
ISSN 0361-0918
ESSN 1532-4141

Publisher: Taylor and Francis



Communications in Statistics - Simulation and Computation

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/lssp20

A novel algorithm for classification using a low rank approximation of kernel-based support vector machines with applications

O. Chatrabgoun, M. Esmaeilbeigi, A. Daneshkhah, A. Kamandi & N. Salimi

To cite this article: O. Chatrabgoun, M. Esmaeilbeigi, A. Daneshkhah, A. Kamandi & N. Salimi (2023): A novel algorithm for classification using a low rank approximation of kernel-based support vector machines with applications, Communications in Statistics - Simulation and Computation, DOI: 10.1080/03610918.2023.2236816

To link to this article: https://doi.org/10.1080/03610918.2023.2236816

COMMUNICATIONS IN STATISTICS - SIMULATION AND COMPUTATION®
https://doi.org/10.1080/03610918.2023.2236816

Taylor & Francis
Taylor & Francis Group

A novel algorithm for classification using a low rank approximation of kernel-based support vector machines with applications

O. Chatrabgoun[a,b], M. Esmaeilbeigi[c], A. Daneshkhah[a,d], A. Kamandi[e], and N. Salimi[f]

[a]School of Computing, Electronics and Mathematics, Coventry University, Coventry, UK; [b]Department of Statistics, Faculty of Mathematical Sciences & Statistics, Malayer University, Malayer, Iran; [c]Department of Mathematics, Faculty of Mathematical Sciences & Statistics, Malayer University, Malayer, Iran; [d]Research Centre for Computational Science and Mathematical Modelling, Coventry University, Coventry, UK; [e]Department of Mathematics, University of Science and Technology of Mazandaran, Behshahr, Iran; [f]Department of Computer Engineering, Faculty of Engineering, Arak University, Arak, Iran

ABSTRACT
Support vector machines (SVMs), as a powerful technique for classification, are becoming increasingly popular in a wide range of applications. This is simply due to their robustness against several types of model assumptions violations and outliers. The Kernel-based SVM are very useful to capture non-linear patterns in the data, and for classification. However, this kernel-based method could become computationally very challenging because it increases the required time to train data. This increase in computational time is mainly due to the appearance of the kernel in solving the quadratic optimization problem (QOP). In order to tackle this computational complexity, we propose a novel method based on the low-rank approximation, by adapting a truncated Mercer series to the kernels. The quadratic optimization problem in the structure of kernel-based SVM will then be replaced with a much simpler optimization problem. In the proposed approach, the required time for the vector computations and matrix decompositions will be much faster such that these changes lead to efficiently resolve the QOP and ultimately increase efficiency in classification. We finally present some numerical illustrations based on the ROC curves and other classification performance benchmarks considered in this paper to assess the performance of the proposed low-rank approximation to the kernel in SVM structure. The results suggest considerable efficiency improvement has been observed in classification with significant reduction in computational time required to train and forecast the stock market index (S&P 500 index) and promoter recognition in DNA sequences.

ARTICLE HISTORY
Received 28 June 2022
Accepted 6 July 2023

KEYWORDS
Kernel-based SVMs; Machine learning; Promoter recognition; Quadratic optimization problem; S&P 500 index; Truncated Mercer series

1. Introduction

In recent years the extraction of knowledge from existing data (data mining) has become as one of the major goals in many applied sciences (Christianini and Shawe-Taylor 2000; Hastie, Tibshirani, and Friedman 2009). The main attraction of data mining methods, which can be divided into two categories of the supervised and unsupervised learning, is to extract important information from the existing data without needing to any presumptions (Han, Kamber, and Pei

CONTACT O. Chatrabgoun ad8337@coventry.ac.uk Department of Statistics, Faculty of Mathematical Sciences & Statistics, Malayer University, Malayer, Iran.

2012). The supervised learning methods cover classification in a discrete and regression in a continuous form. There are many data-driven techniques for implementing classification, including Gaussian processes (Rasmussen and Williams 2006; Nickisch and Rasmussen 2008), neural networks and SVM (Araújo, Oliveira, and Meira 2017; Qian and Zhang 2018) as the widely used methods for this purpose. The SVMs have recently gained considerable popularity in classification of real world application because of its promising performance in reducing the operational risk in classification which is more demanding than reducing the classification error only. The details of different SVMs which have been used in recent years can be found in (Steinwart and Christmann, 2008; Erdogan 2013; Vapnik 2013; Gupta and Richhariya 2018; Gupta, Richhariya, and Borah 2019; Borah and Gupta 2021; Hazarika and Gupta 2021, 2022).

The linear SVM is the simplest pattern recognition task which uses an optimal linear separating (possibly, multi-dimensional) hyperplane for classification task with maximum margin. In order to achieve this objective, learning problem for the SVM becomes a constrained non-linear optimization problem under quadratic cost function and the linear constraints. However, for more complex cases, the linear SVM would be ineffective in determining the classes using the optimal linear separating hyperplane to be derived on the initial input space. A more efficient way to tackle this issue for the complex cases is to use the kernel-based SVM at which the classification can be achieved by non-linearly mapping data to a high dimensional space (known as feature space) and employ the kernel trick. In other words, when the classes among which it is intended to discriminate cannot be linearly separated in the initial input space, the plausible way is to transform the original data (or input) space into a higher dimensional feature space by using different various non-linear mappings, such that the data would become linearly separable in the generated new higher dimensional feature space. Despite, the usefulness of transforming the observed data to a non-linearly separable data in the feature space using a linear SVM, the computational burden become infeasible. The alternative and more efficient way is to use the kernel SVM which prevents the increase of computational complexity (Scholkopf and Smola 2002; Steinwart and Christmann 2008). It should be noted that the computational performance of the kernel-based SVM generally refers to the useful properties of the kernels, such as the possibility of interacting with high-dimensional data, and encountering with non-linear data. In other words, the computation in the feature space can be expressed in terms of computations in the initial space, that is the computational complexity is not increased. In addition, the existence of some customizable parameters in the kernels makes the kernel-based SVM more flexible in classification (Fasshauer and McCourt 2016; Lan et al. 2019; Chatrabgoun et al. 2022).

However, the kernel SVM discussed above is known to achieve state-of-the art performances for many classification problems, but training and prediction are still slow for the complex cases because kernel methods non-linearly map data to a high dimensional space and employ the kernel trick could increase computational complexity and ultimately increase the required time for classification (Chih-Jen 2013). This computational complexity is due to the presence of kernel in the quadratic optimization problem (QOP) which makes the Hessian matrix required in QOP to be very dense and thus ill-conditioned.

It is thus very essential to develop an efficient approach to solve the above optimization problem for the kernel-based SVMs such that it can overcome the computational complexity discussed above, and also reduce the training time for classification. In the last decade, several efforts have been made in this regard, for example, data reduction methods have been used to reduce the computational complexity and costs in the kernel-based SVMs by reducing the dimensionality of the original data (Nguyen, Huang, and Joseph 2008). Here, some valuable information exits in the original data could be lost through this dimensionality reduction. An alternative method to resolve this issue is to randomly select some columns of the kernel in the SVMs structure, which could increase the computational complexity (Bach 2013). Selecting these columns are not always straightforward, and this could create another challenging problem. Andersen and Vandenberghe (2014) suggest a methodology by transforming the kernel matrix into a sparse matrix.

In this paper, we propose a novel method by using the low-rank approximation of the kernel terms, derived based on the truncated Mercer series expansion of the underlying kernel. Using the proposed method, the complex quadratic optimization problem described above will be reduced to a considerably simpler optimization problem. This low-rank approximation enables us to work with computationally more tractable low-rank kernel structure, instead of the full-rank kernel. The SVM structure is thus construed using the truncated Mercer series expansion terms which are function of eigenvalues and eigenfunctions of Hilbert-Schmitt operator of the selected kernel. Since the eigenvalues of the considered series expansion terms are severely decreasing, the kernel approximation can be easily derived using the first several terms of the expansions. However, Mercer series expansion is truncated by several initial terms, the corresponding low-rank approximation of the kernel in SVM can be made as accurate as is desired. In order to maintain this accuracy and make the low rank approximation computational more tractable, a new Hessian matrix required for the quadratic optimization of the kernel-based SVM will be derived which is sparse, well-conditioned and very straightforward to compute. It should be noted that the Hessian matrix computed in the full-rank form of the SVM kernel is highly dense which will cause the computation become very complex. However, vector computation and matrix decomposition based on the new Hessian matrix can be swiftly implemented with considerable less computational cost. This is evident because the computation time to solve the QOP and training time for classification are considerably decreased.

Forecasting the trend of the stock market, particularly the S&P 500 index is a long-time attractive topic (Liu et al. 2016). This index is influenced by other important financial indexes across the world, including commodity price and financial technical indicators. In this paper, we systematically investigate the proposed low-rank approximation for the kernel-based SVM for predicting the S&P 500 index, and then compare the predictive performance of the proposed method with the one derived based on the full-rank SVM. Also, we use the 2002 collection of data of drosophila (D. melanogaster) promoter regions (Berkeley Drosophila Genome Project, 2002) (Damaševicius 2008). The dataset has three parts: promoter, intron and CDS. The main goal with such dataset is to classify promoters.

The structure of this paper is organized as follows. In Sec. 2, we first describe the kernel-based SVM structure, and then discuss the proper properties of these kernels and their computational challenges for classification. We introduce the low-rank kernel approximation and resolve the computational challenges of the kernel-based SVMs in Sec. 3. We present the numerical results for the proposed low rank approximation in Sec. 4 and show the advantages of our method in comparison of the existing methods. We then evaluate the proposed methodology in analyzing the stock market data, including forecasting S&P 500 index and promoter classification. Finally, Sec. 6 is dedicated to results and discussion and Sec. 7 is devoted to brief conclusions.

2. SVM classification

First, we try to make a preliminary introduction to the SVM technique and especially the linear SVM for unfamiliar readers. Thereafter, kernel-based SVM will be described.

2.1. Preliminaries

Suppose that we are given a set of training data $\{(\mathbf{x}_i, y_i) | i = 1, \ldots, N\}$ with measurements $\mathbf{x_i} \in \mathbb{R}^d$ and data values in the form of labels $y_i \in \{-1, +1\}$. We can use SVM when our data has exactly two classes as a standard (binary) classification problem which consists of finding a predictor that will allow us to assign an appropriate label, either -1 or $+1$, to a future measurement $\mathbf{x}$. An SVM classifies data by finding the best hyperplane that separates all data points of one class from those of the other class. Also, the support vectors are the data points that are closest to the separating hyperplane; these points are on the boundary of the slab. Such a predictor might be of the form $(h(\mathbf{x}))$, where h denotes a hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$, such that the weight $\mathbf{w}$ serve as the unit normal vector to the hyperplane and b denotes the interception or bias term. In fact, the rule of classification is that

$$\text{if} \qquad y_i = 1 \quad \Rightarrow \quad \mathbf{w}^T\mathbf{x}_i + b > 1, \tag{1}$$

$$\text{if} \qquad y_i = -1 \quad \Rightarrow \quad \mathbf{w}^T\mathbf{x}_i + b < -1 \tag{2}$$

In the simplest case, our predictor is given by $(h(\mathbf{x}))$, where h is of the form

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b, \qquad \mathbf{x} \in \mathbb{R}^d, \tag{3}$$

that separates the given measurements with label -1 from those with label $+1$. The weight vector $\boldsymbol{w}$ and the bias b can be determined by maximizing the margin or gap to both sides of the hyperplane (2). In fact, the best hyperplane for an SVM means the one with the largest margin between the two classes. Margin means the maximal width of the slab parallel to the hyperplane that has no interior data points. Since the size of this margin equals to $||\mathbf{w}||^{-1}$, and we would ideally like to have this margin as large as possible, we want to minimize $||\mathbf{w}||$, the norm of the coefficients of h. Thus, we would have an unconstrained minimization problem of the form

$$\min ||\mathbf{w}||,$$

or equally

$$\min \frac{1}{2}||\mathbf{w}||^2 = \frac{1}{2}\mathbf{w}^T\mathbf{w}, \tag{4}$$

by combining both conditions (1) and (2), we can create the following constraint

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) > 1, \tag{5}$$

such that we are faced with a constrained optimization problem. Instead of this formulation, the following constrained optimization with slack variables ε_i, and box constraint C as a free parameter will be used. Using ε_i and the penalty coefficient C is more common since they also allows us to deal with the case where the given measurements are not perfectly separable by h. Therefore, we rewrite our optimization problem (4) as

$$\min \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\varepsilon_i, \tag{6}$$

such that with slack variables ε_i and box constraint C, the constraint (5) will be modified to

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) > 1 - \varepsilon_i. \tag{7}$$

In these formulations, we can see that increasing C places more weight on the slack variables, meaning the optimization attempts to make a stricter separation between classes. Equivalently, reducing C toward 0 makes misclassification less important. The considered formulation (6) and (7) can be combined together *via* Lagrange multipliers α_i and is known as the primal problem in the SVM literature, i.e.

$$L_P = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\varepsilon_i - \sum_{i=1}^{N}\alpha_i\left[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \varepsilon_i\right].$$

This corresponding primal problem can be changed to the Dual problem which derived from setting the $\mathbf{w}$-gradient of the primal Lagrange multiplier functional equals to zero. This Dual problem is of the form

$$L_D = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{N}\alpha_i, \tag{8}$$

subject to $\sum_{i=1}^{N}\alpha_i y_i = 0$, and $0 \le \alpha_i \le C$, such that

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i.$$

The bias b_i, $i = 1, 2, \ldots, N$, is given by $b_i = y_i - \mathbf{w}^T\mathbf{x}_i$ for any i and the optimal value $\alpha_i > 0$. For stability purposes, we actually consider all qualifying indices and find b using the mean. Note that C needs to be either set by the user or determined by an additional parameter optimization methods such as cross validation.

2.2. Kernel-based classification

Sometimes it is not possible to classify existing data with one hyperplane, and separate them using a linear SVM. For this reason, we consider a feature space in place of the data itself or input space, and try to separate data in feature space by linear SVM or hyperplane. For example, the feature space can be the distance of data from each other or a function of this distance. If we show the feature space of the measurements with $\phi_\mathbf{x}$, as shown in Figure 1), they will not be linearly separable within the input space, but they will be in the feature space.

Note that this feature space is potentially infinite-dimensional and therefore offers much more flexibility for separating the data than the finite dimensional input space. This fact has a theoretical foundation in the form of Cover's theorem (Cover 1965), which ensures that data which cannot be separated by a hyperplane in input space most likely will be linearly separable after being transformed to feature space by a suitable feature map. Thus, SVMs - in terms of feature space, in particular- are a proper technique to use in order to tackle with intricate data classification problems.

The algorithms for non-linear classification are now more or less the same as before; simply replace the measurements $\mathbf{x}_i$ in input space by their features $\phi_{\mathbf{x}_i}$ in feature space. Therefore, the separating hyperplane (3) can be expressed in the form

$$h(\mathbf{x}) = \phi_\mathbf{x}^T\mathbf{w} + b = 0, \qquad \mathbf{x} \in \mathbb{R}^d, \tag{9}$$

and the Dual problem (8) for SVM classification using the transformed input data is given by

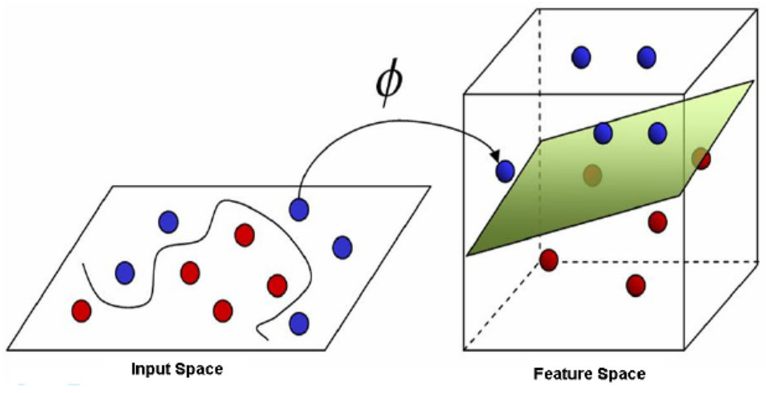


Figure 1. The observed data points are not linearly separable in the input space, but they are in the feature space (Sosa 2016).

$$\begin{aligned} &\min \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \phi_{\mathbf{x}_i}^T \phi_{\mathbf{x}_j} \quad - \sum_{i=1}^{N}\alpha_i \\ &\text{s.t} \quad \sum_{i=1}^{N}\alpha_i y_i = 0 \quad \text{and} \qquad 0 \le \alpha_i \le C. \end{aligned} \tag{10}$$

Now, obtaining feature space of the data is possible within the framework of Reproducing Kernel Hilbert Space (RKHS). In other words, the mapping of the introduced feature is considered as $\phi : \Omega \to H_K(\Omega)$ under the map

$$\mathbf{X} \mapsto \phi_{\mathbf{X}} = K(.,\mathbf{X}).$$

The data set is transferred from space Ω to feature space $H_K(\Omega)$, where $H_K(\Omega)$ is RKHS corresponded to the kernel K (Fasshauer and McCourt 2016). Regarding RKHS characterizations, the inner product in the feature space, $H_K(\Omega)$, using the kernel K is simply possible, namely:

$$K(\mathbf{X},\mathbf{Z}) = \phi_{\mathbf{X}}^T \phi_{\mathbf{Z}},$$

Now, using the above relation, a more general form of the Dual problem (10) in the feature space can be defined as follows:

$$\begin{aligned} &\min \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) \quad - \sum_{i=1}^{N}\alpha_i, \\ &\text{s.t} \qquad \sum_{i=1}^{N}\alpha_i y_i = 0 \qquad \text{and,} \qquad 0 \le \alpha_i \le C. \end{aligned} \tag{11}$$

where,

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i K(\mathbf{x},\mathbf{x}_i).$$

The classifier $h(\mathbf{x})$ given in (9) can now be modified based on the kernel K as

$$sign(h(\mathbf{x})) = sign(\sum_{i=1}^{N}\alpha_i y_i K(\mathbf{x}_i \ \mathbf{x}) + b), \tag{12}$$

where, in a similar way, b_i for $i = 1, 2, \ldots, N$ can be obtained as before, i.e. $b_i = y_i - \sum_{j=1}^{N}\alpha_j y_j K(\mathbf{x}_j,\mathbf{x}_i)$, and $0 \le \alpha_i \le C$. For stability purposes, we can again average over all such candidates. For positive definite kernels, it is also possible to formulate the separating hyperplane without the bias term b (Poggio et al. 2001).

Given the important role of the kernels, the above approach is named as a kernel-based SVM and provides the possibility of non-linear classification for the observed data. It should be noted that the capacity in the kernel-based SVM generally refers to the useful features of the kernels such as the possibility of interacting with high-dimensional data and the possibility of encountering data that are classified non-linearly. Additionally, the existence of some customizable parameters in the kernels (such as shape or scale parameter) makes the kernel-based SVM more flexible. Generally, the use of some known kernels in the kernel-based SVM is more common than other kernels. These kernels are the polynomial kernel, the Gaussian kernel and the sigmoid kernel, as their structures are shown in Table 1. Each of these kernels has a shape parameter that makes them more flexible. Moreover, kernels may be defined through their feature map (instead of their closed form), and this feature map can be picked depending on the specific application at hand (e.g. as a string kernel for text mining). The ubiquitous Gaussian kernel does not need more introductions, although it is curious to note that it seems to have been first mentioned in its role as a probability density function in the context of least squares approximation and maximum

Table 1. Polynomial, Gaussian and sigmoid kernel structure to use in kernel-based SVM.

Kernel	structure $(K(\boldsymbol{x}, \boldsymbol{z}))$
Polynomial	$(1+\boldsymbol{x}^T\boldsymbol{z})^{\epsilon}$
Gaussian	$e^{\epsilon\|\boldsymbol{x}-\boldsymbol{z}\|}$
Sigmoid	$tanh(1+\epsilon\boldsymbol{x}^T\boldsymbol{z})$

likelihood estimation. The Gaussian kernel is infinitely differentiable and positive definite. It will be used in this paper for the kernel-based SVM. In the literature on statistics and machine learning the Gaussian kernel is sometimes referred to as the squared exponential kernel.

Despite the advantages, unfortunately using the technique of kernel-based SVMs for classification increase computational complexity and ultimately increase the required time for classification. This computational complexity is due to the presence of kernel in QOP in the structure of kernel-based SVMs. In fact, the Hessian matrix in QOP is very dense and ill-condition. On the other hand, as far as the kernels in the SVMs structure has a high capacity for the classification of data with the non-linear property, their computational challenges will mainly increase. In addition, the involved optimization problem in the kernel-based SVMs will increase the computational cost and plays a major role in the training time and performance of this technique of classification. Therefore, providing an efficient way that leads to solve the optimization problem in the kernel-based SVMs such that it can reduce the computational complexity and especially the training time is very important. In the next section, we will show how by using the low-rank approximation of the kernel, based on the Mercer series expansion of the kernel, it is possible to replace the QOP with a much simpler problem.

3. Low-rank approximation for the kernel-based SVMs

According to the Mercer's theorem (Cover 1965; Yang, Duraiswami, and Davis 2004), each positive definite kernel can be shown using an infinite series expansion as follows:

$$K(\mathbf{x},\mathbf{z}) = \sum_{n=1}^{\infty} \lambda_n \varphi_n(\mathbf{x})\varphi_n(\mathbf{z}). \tag{13}$$

In this expansion, λ_n and φ_n, respectively, are positive eigenvalues and orthogonal eigenfunctions corresponded to the Hilbert-Schmidt operator of the kernel K. It should be noted that the Hilbert Schmidt operator, $\kappa : L_2(\Omega) \to L_2(\Omega)$, is defined as:

$$(\kappa f) = \int_{\Omega} K(\mathbf{x},\mathbf{z}) f(\mathbf{z}) d\mathbf{z},$$

such that the eigenvalues and eigenvectors used in the Mercer expansion are obtained by solving the following eigenvalue problem:

$$(\kappa\varphi)(\mathbf{X}) = \lambda\varphi(\mathbf{X}).$$

Note that it's not possible to work with the number of infinite terms from the Mercer series in computational systems. Now given the available decreasing rate in special eigenvalues (only the initial terms of this extension have a fundamental role in the approximation of kernel K), we will use the truncated low-rank approximation presented in the following equation for kernel approximation. In fact, if N is the number of observed data, then for M much smaller than N, we have:

$$K(\mathbf{x},\mathbf{z}) = \sum_{n=1}^{M} \lambda_n \varphi_n(\mathbf{x})\varphi_n(\mathbf{z}). \tag{14}$$

L_2-orthonormality of the eigenfunctions plays an important role in Mercer's theorem using the infinite series representation of a positive definite kernel in terms of the eigenvalues and eigenfunctions with uniform convergence guarantee. According to the truncated Mercer series theorem (14), if K is a positive definite kernel with the presented expansion in Eq. (13), and the truncated Mercer series with M terms defined as (14) then this truncated series is the best approximation with M terms from the perspective of least squares in $L_2(\Omega)$ for the kernel K. Now, by taking into account these conditions, it is possible to approximate the matrix $\mathbf{K}$ in the Dual problem (11) To this end, we can write

$$\mathbf{K} \simeq \boldsymbol{\phi}\boldsymbol{\Lambda}\boldsymbol{\phi}^T, \tag{15}$$

such that $N \times M$ matrix $\boldsymbol{\phi}$ will be defined as

$$\boldsymbol{\phi} = \begin{bmatrix} \boldsymbol{\varphi}(\mathbf{X}_1)^T \\ \vdots \\ \boldsymbol{\varphi}(\mathbf{X}_N)^T \end{bmatrix},$$

such that $\boldsymbol{\varphi}(\mathbf{X})$ is as

$$\boldsymbol{\varphi}(\mathbf{X}) = \begin{bmatrix} \varphi_1(\mathbf{X}) \\ \vdots \\ \varphi_{M+}(\mathbf{X}) \end{bmatrix},$$

the diagonal matrix $\boldsymbol{\Lambda}_{M\times M}$ is also given by

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_M \end{bmatrix}.$$

By taking into account the approximation in (15), using the truncated Mercer series theorem, we approximate matrix $\mathbf{K}$ in the second-order Dual optimization problem defined in (11). Since only the first M terms of the expansion are used, we use the symbol $\simeq$ instead of the symbol $=$. But fortunately, in many kernels for very small amounts of M, we can obtain very accurate approximations of the kernel K, so we will use the symbol $=$ instead of the symbol $\simeq$ in the sequel.

Based on what we have obtained, we intend to use the new derived approximation for matrix $\mathbf{K}$ to reconstruct Dual optimization problem. To do this, at first, with some modifications, the optimization problem in (11) can be rewritten in the following matrix form:

$$\min_{\boldsymbol{\alpha}} \left(\frac{1}{2}\boldsymbol{\alpha}^T\mathbf{D}_y\mathbf{K}\mathbf{D}_y\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha}\right)$$
$$\text{Subjectto } \mathbf{y}^T\boldsymbol{\alpha} = 0, \quad \text{and} \quad \boldsymbol{\alpha} \in [0, C]^N,$$

where $\mathbf{D}_y$ is a diagonal matrix with elements $\{y_i\}_{i=1}^N$, on its main diagonal, and $\mathbf{e}$ contains a vector with elements 1. Now, according to the approximation obtained in relation (15), we can rewrite the matrix $\mathbf{K}$ as below:

$$\mathbf{K} = \left(\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{\phi}\right)^T\left(\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{\phi}\right).$$

Then, by considering $\mathbf{V} = \mathbf{D}_y\boldsymbol{\phi}\boldsymbol{\Lambda}^{\frac{1}{2}}$ and $\mathbf{V}\mathbf{V}^T = \mathbf{D}_y\mathbf{K}\mathbf{D}_y$, the Dual optimization problem (11) will be as follows:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\left((\mathbf{V}^T\boldsymbol{\alpha})^T(\mathbf{V}^T\boldsymbol{\alpha}) - \mathbf{e}^T\boldsymbol{\alpha}\right)$$
$$\text{Subject to } \mathbf{y}^T\boldsymbol{\alpha} = 0, \quad \text{and} \quad \boldsymbol{\alpha} \in [0, C]^N,$$

in the following, we consider the matrix $\mathbf{V}^T\boldsymbol{\alpha}$, which contains the unknown vector $\boldsymbol{\alpha}$, as follows:

$$\mathbf{V}^T\boldsymbol{\alpha} = \mathbf{I}_M\boldsymbol{\beta},$$

where $\mathbf{I}_M$ is an identity matrix of the order M and $\boldsymbol{\beta} \in \mathbb{R}^M$. With these considerations, the second-order Dual optimization problem presented in (11) can be rewritten as follows:

$$\begin{aligned} &\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \frac{1}{2}\begin{pmatrix} \boldsymbol{\beta}^T & \boldsymbol{\alpha}^T \end{pmatrix}\begin{pmatrix} \mathbf{I}_M & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \boldsymbol{\beta} \\ \boldsymbol{\alpha} \end{pmatrix} - \begin{pmatrix} \mathbf{0} & \mathbf{e}^T \end{pmatrix}\begin{pmatrix} \boldsymbol{\beta} \\ \boldsymbol{\alpha} \end{pmatrix} \\ &\text{Subject to } \begin{pmatrix} \mathbf{0} & \mathbf{y}^T \\ \mathbf{I}_M & \mathbf{V}^T \end{pmatrix}\begin{pmatrix} \boldsymbol{\beta} \\ \boldsymbol{\alpha} \end{pmatrix} = 0, \;\; \boldsymbol{\beta} \in \mathbb{R}^M, \;\; \text{and } \boldsymbol{\alpha} \in [0, C]^N. \end{aligned}$$

It is noteworthy that, although the system is of a higher order of $M+N$ (while the main system is of the order N), but due to the much simpler structure, the computational cost of solving this system is far less than the original system in (11). In addition, the Hessian matrix obtained in the new structure is very sparse and well-conditioned. This is while the Hessian matrix for the kernel K is quite dense in the previous form, and hereafter we call it full rank form. Therefore, in the new structure, the vector computations and matrix analyses required in the solving process of the second-order optimization problem will be much easier. Both of these changes lead to a faster solution to the second-order optimization, and ultimately reduce the required computational time for classification by using a kernel-based SVM. For example, the eigenvalues in Mercer series expansion for Gaussian kernel is plotted in Figure 2. As can be seen, after the first five eigenvalues, which are also severely descending, the rest equal zero, i.e.

$$0.6667 \quad 0.0741 \quad 0.0082 \quad 0.0009 \quad 0.0001$$

Therefore, Mercer series expansion is truncated by five or less initial terms, the corresponding low-rank approximation of the kernel in SVM can be made as accurate as is desired. In order to maintain this accuracy and make the low rank approximation computational more tractable, a new Hessian matrix required for the quadratic optimization of the kernel-based SVM will be derived which is sparse and well-conditioned, and very straightforward to compute. In following we will compute the condition number (CN), sparsity and computational time of the low-rank Hessian matrix in comparison with correspond full-rank matrix. Note that finding eigenvalues and eigenvectors for many positive definite kernels (e.g. Gaussian kernel, Chebyshev kernel, Exponential kernel, …) are easy tasks such they can be easily obtained using GaussQR Repository

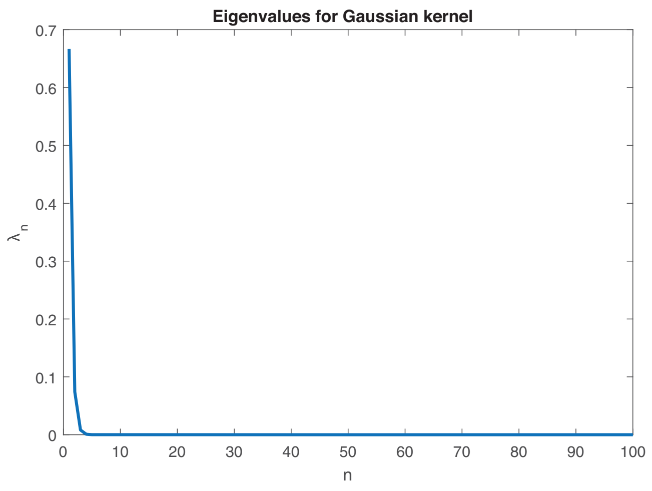


Figure 2. Eigenvalues in Mercer series expansion for Gaussian kernel.

in Matlab (Fasshauer and McCourt; 2016). This Repository provides demonstration of how kernels can be used in computation. In the sequel, we pursue these through a numerical example and real datasets.

4. Numerical results

In this section, in order to compare full-rank kernel-based SVM and it's low-rank approximation, we first introduce the following example. As we will see, this example uses a pattern which is not linearly separable and attempts to classify it based on the kernel SVM. Here, population 1 (denoted by blue $\square$ and green $+$) has centers at $\{(0,1),(1,0),(2,1)\}$ and population 2 (denoted by blue $\bigcirc$ and red $\times$) has centers at $\{(0,0),(1,1),(2,0)\}$. Test points are chosen from a normal distribution from those populations and training data is generated from the test points. These distribution centers (filled green $\square$ and red $\bigcirc$), test points (large green $+$ and red $\times$) and training points (small green $+$ and red $\times$) are displayed in Figure 3. This pattern is not linearly separable, and this fact is evident in Figure 3.

Here, we first classify these two populations using the low-rank approximation technique. This classification heavily depends on existing parameters C and ϵ (as a Gaussian shape parameter). The decision contours show the impact of the various Gaussian kernel parametrizations on the low-rank SVM. Figure 4a and b alternately fixes C and ϵ values and allows the other parameter to vary to illustrate the impact on the low-rank SVM. Clearly, the choice of ϵ plays a significant role, where larger ϵ encourages an SVM with more locality and smaller ϵ encourages less localized influence; this matches the standard localization behavior for Gaussian kernel in an interpolation setting. When $\epsilon = 1$ is fixed and different C values are considered, a similar impact occurs. It seems that smaller C values produce a less active decision contour, whereas large C encourages more local fluctuations.

Now, we consider fixing $C = 0.6$ with a variety of ϵ values. Figure 5a and b shows the number of missed classifications (out of 20 possible) as well as the margin $\frac{1}{||\mathbf{w}||}$ and the required number of support vectors, respectively. The margin does not appear to be a useful guide for determining an optimal ϵ value, as the margin grows unbounded as $\epsilon \rightarrow 0$; on the other hand, for a very large ϵ this example is perfectly classified. Minimizing the number of support vectors is optimal from a computational standpoint, and also seems to suggest a viable region for predictions.

When we fix $\epsilon = 1$ and consider a range of C values, as display in Figure 6a and b, the situation is no clearer. Good prediction results seem to occur for smaller C values, and very large values of C require few support vectors. It is worth noting that larger C values require more

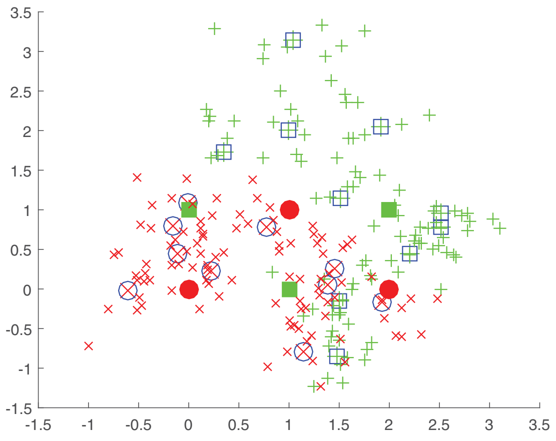


Figure 3. Scatter plot of input data and test points for population 1 versus population 2.

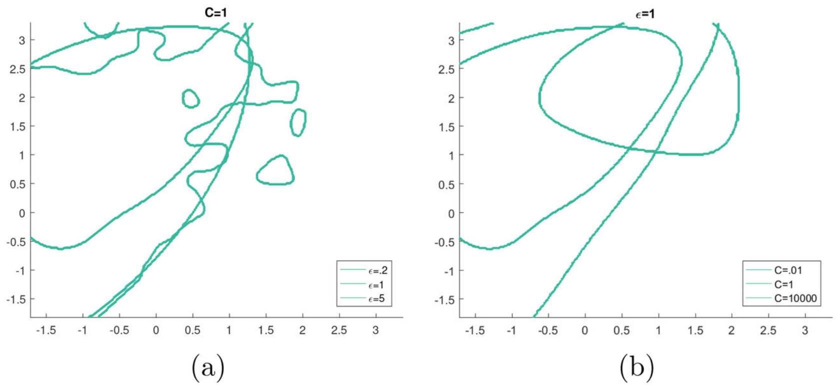


Figure 4. Fixes C (a) and ϵ (b) values and allows the other parameter to vary to show the impact on the SVM.

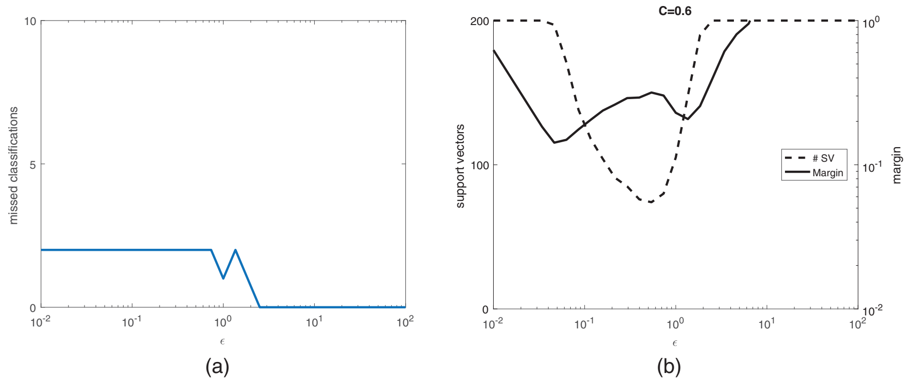


Figure 5. Fixes $C = 0.6$ with a variety of ϵ which shows the number of missed classifications (a) as well as the margin and the required number of support vectors (b).

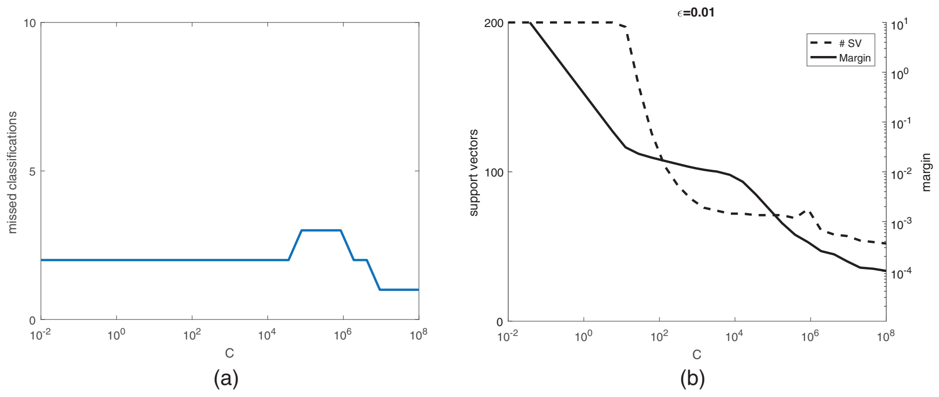


Figure 6. Fixes $\epsilon = 0.01$ with a variety of C which shows the number of missed classifications (a) as well as the margin and the required number of support vectors (b).

computing time to solve the quadratic optimization program because a larger search space is under consideration.

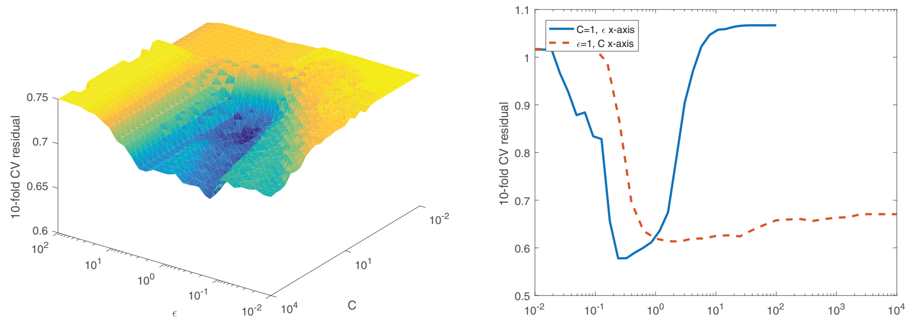
The numerical results demonstrate that finding an optimal SVM parametrization using either the margin or the number of support vectors is not always a useful strategy. Therefore, one technique should be used to estimate the optimal parameters. A more common technique in the machine learning community is to use cross-validation (CV).

4.1. Parameter estimation in low-rank kernel-based SVM with cross-validation

Cross validation is a popular technique in statistics which uses the given data (instead of the-usually unknown solution) to predict optimal values of model parameters for data fitting. The main idea is to split the data into a training set τ and a validation set ν and to then use some form of error norm obtained by gauging the accuracy of the fit built from information on the training set at points in the validation set. Cross validation (CV) is an especially popular version of cross-validation and corresponds to using a training set consisting of all but some of the data points, which in turn are the sole member of the validation set (Hickernell and Hon 1999; Rippa 1999; Fasshauer and Zhang 2007; Scheuerer 2011). Often times, cross-validation is conducted in one of two ways: 1) Leave-one-out cross-validation (LOOCV): All the data except a single point is used to compute in kernel-based SVM classification, and the residual is judged at that point. In this setting, $V = \left\{\nu^{(1)}, \nu^{(2)}, \ldots, \nu^{(N)}\right\} = \{x_1, x_2, \ldots, x_N\}$ and the errors at each of those points are added up to find total error. As explained in (Fasshauer 2007), it is most likely the preferred form to compute. Thus the entire LOOCV computation can be performed with little overhead compared to the computation of the classification. 2) Leave-half-out cross-validation: Half of the data is omitted to create an classification and the residual is judged on the other half; then the process is flipped and both results are combined to compute total error. In this setting, $V = \left\{\nu^{(1)}, \nu^{(2)}\right\}$ and $|\nu^{(1)}| = |\nu^{(2)}|$, or as close as possible. Sometimes instead of Leave-half-out cross-validation, some part of the data preserve for cross-validation. This part of data is called fold. Here, a 10-fold cross-validation scheme is used to measure the effectiveness of each of the ϵ and C parameters. In Figure 7a, we show the associated cross-validation residuals in terms of two parameter ϵ and C. Also, in Figure 7b, we show the associated cross-validation residuals with one parameter fixed and the other varied over $[10^{-2}, 10^{2}]$. We also show that there is an optimal region for the 10-fold cross-validation residual, where decreases in ϵ are matched by increases in C. In Figure 8, the number of missed classification items in terms of two parameter ϵ and C depicted. Also, in Figure 9, the training time for classification in terms of two parameter ϵ and C have been shown.

4.2. Challenge of determining M, the truncation value

After finding an optimal way for parameter estimation using 10-fold CV, one of the challenges in using the low-rank approximation has always been the selection of the truncation value, M, in (14).



(a) (b)

Figure 7. Fixes $\epsilon = 0.01$ with a variety of C which shows the number of missed classifications (a) as well as the margin and the required number of support vectors (b).

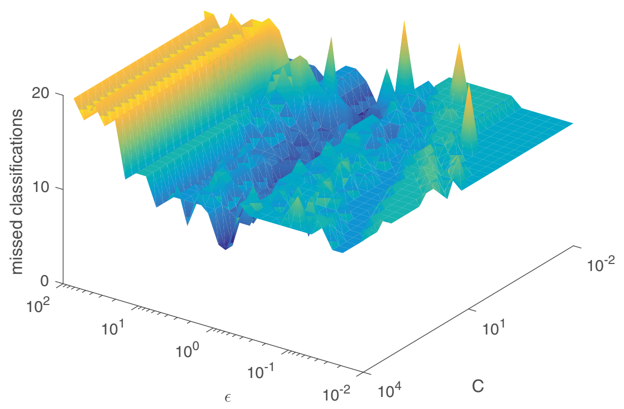


Figure 8. The number of missed classification items in terms of two parameter ϵ and C.

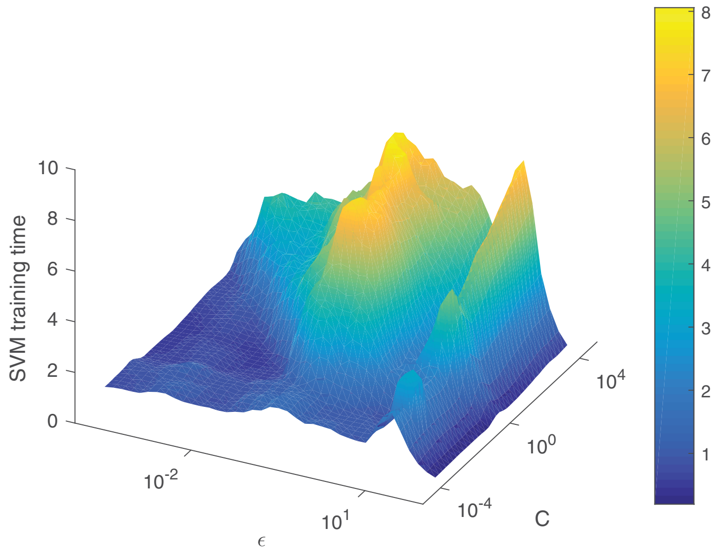


Figure 9. Training time for classification in terms of two parameter ϵ and C.

In fact, the truncation in the Mercer's expansion, affects on the kernel approximation and ultimately affects on the low-rank kernel-based SVM. Therefore, we should use a truncation criterion to determine an appropriate M-value for our low-rank approximation of the kernel such that to be accurate in terms of classification errors. To determine M in the Gaussian kernel (Fornberg and Piret 2008) stated different truncation criterion where this criterion considers only the magnitude of the eigenvalues. This truncation design is somewhat unsatisfying in the kernel-based SVM, because it requires the actual construction of eigenfunctions to make a decision, and thus the entire construction process cannot be planned in advance. In a practical setting, it is a bit displeasing.

Another careful analysis of truncation lengths for general kernels, especially Gaussian kernel, is presented in (Griebel, Rieger, and Zwicknagl 2015). There it was shown that the truncation length should be chosen in dependence on N and the smallest eigenvalue of the kernel matrix $\boldsymbol{K}$. In fact, one should have

$$\sum_{n=M+1}^{\infty} \lambda_n \le \frac{\lambda_{\min}(\mathbf{K})}{N}.$$

We can show if the truncation length chooses according to this criterion then training in kernel-based SVM with the truncated kernel will have the same approximation order as with the full kernel. But this criterion appears to have only limited practical applicability, especially if we are dealing with a very ill-conditioned matrix $\boldsymbol{K}$. In that case, it is impossible to accurately compute the smallest eigenvalue of $\boldsymbol{K}$. Therefore, given the constraints of the mentioned techniques, we use an approach dependent on the existing problem to select the value M. In this technique, we consider the value of M as a coefficient of the total sample (N) and compute at each time the missed classification number. As shown in Figure 10, we can see that there is a jump in the number of missed classifications after $M = 0.1N$, and after that, no change was made. From now on, this strategy will be used.

4.3. Comparison between full-rank and low-rank kernel-based SVM

After viewing the performance of the low-rank kernel-based SVM, we will examine the capability of this new structure in reducing the classification time and training time of the observed data in comparison with the full rank method. In fact, in this new structure, the quadratic optimization equation in the SVM will be much simpler to solve. This will reduce the training time in the classification structure of low-rank kernel-based SVM compared to the full-rank mode. This is illustrated in Figure 11. In other words, to show this, the number of different points is considered, varying from 10 to 10000. For a small number of points, this difference is negligible, although the low-order approximation is better than the full-rank mode, while for a large number of points, this difference is very high, and this shows very good performance for the low-rank kernel compared to the full-rank. In this calculation, the parameter $\epsilon = 0.01$, and $C = 1$ is considered. The number of terms used in the low-rank kernel approximation (M) is considered to equal 0.1 training points, i.e. $M = 0.1N$. Also, based on the truncated Mercer series expansion, the corresponding low-rank approximation of the kernel in SVM can be made as accurate as is desired. In order to maintain this accuracy and make the low rank approximation computational more tractable, a new Hessian matrix required for the quadratic optimization of the kernel-based SVM will be

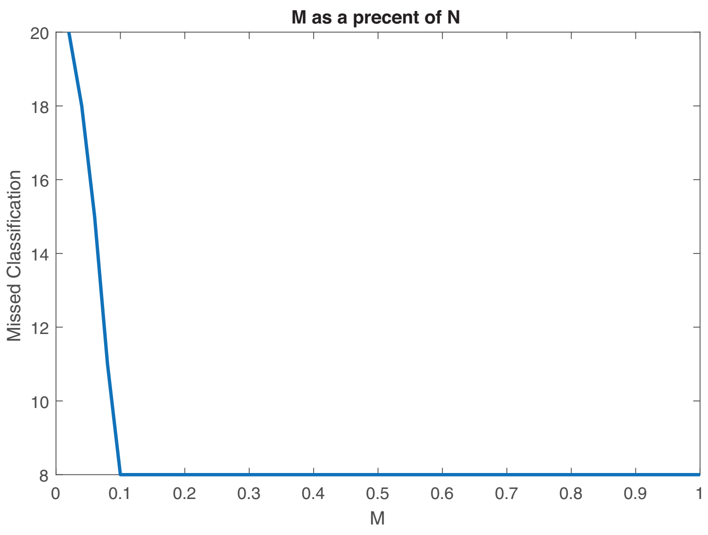


Figure 10. The number of missed classification items in terms of M as a percent of N.

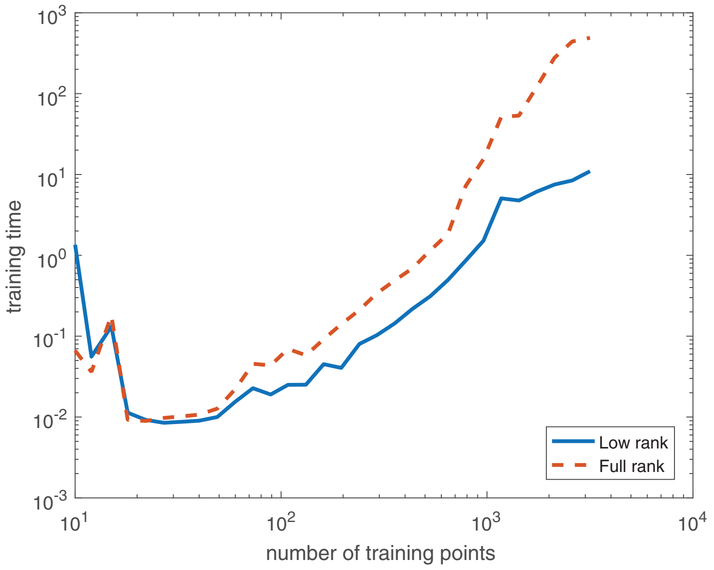


Figure 11. The training time in terms of the number of points in the low-rank and full-rank kernel-based SVM.

Table 2. Comparison CN, sparsity and computational time between low rank approximation and full rank SVM.

Method	CN	Sparsity	Time
Full-rank	1.3153×10^6	0.9460	80.16
Low-rank	3.1850×10^3	0.1502	5.31

derived which is sparse and well-conditioned, and very straightforward to compute. Table 2 give us the condition number (CN), sparsity and computational time of the Hessian matrix. These results point out essentially that low-rank Hessian matrix is sparser and well-conditioned than Hessian matrix of the full-rank method. Also, the low-rank Hessian matrix can be swiftly implemented with considerable less computational cost (In minutes).

4.4. The results of roc curve

Receiver operating characteristic (ROC) curves is a useful way to evaluate the performance of binary classification schemes. In this subsection, we compare the performance of low-rank kernel-based SVM with full-rank using ROC curves. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings for binary classification. The true-positive rate is also known as sensitivity, and the false-positive rate is also known as the fall-out which can be calculated as (1-specificity). The ROC curve for low-rank kernel-based SVM against full-rank is depicted in Figure 12. There are many criteria for evaluating these ROC curves. One of the most important of these criteria is the area under the curve (often referred to as simply the AUC). AUC equals to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. Both methods of classification, low-rank and full-rank, have almost the same AUC such that this amount for low-rank equals to 0.713 and compared to 0.745 for full-rank has a negligible difference. Although the ROC and AUC introduced as a useful way to evaluate the performance of binary classification schemes but more measurements Accuracy, F_1 and Matthews correlation coefficient (MCC) is used to check the performance of our method. The F_1 score is a number between 0 and 1 and is the harmonic mean of precision and recall i.e.

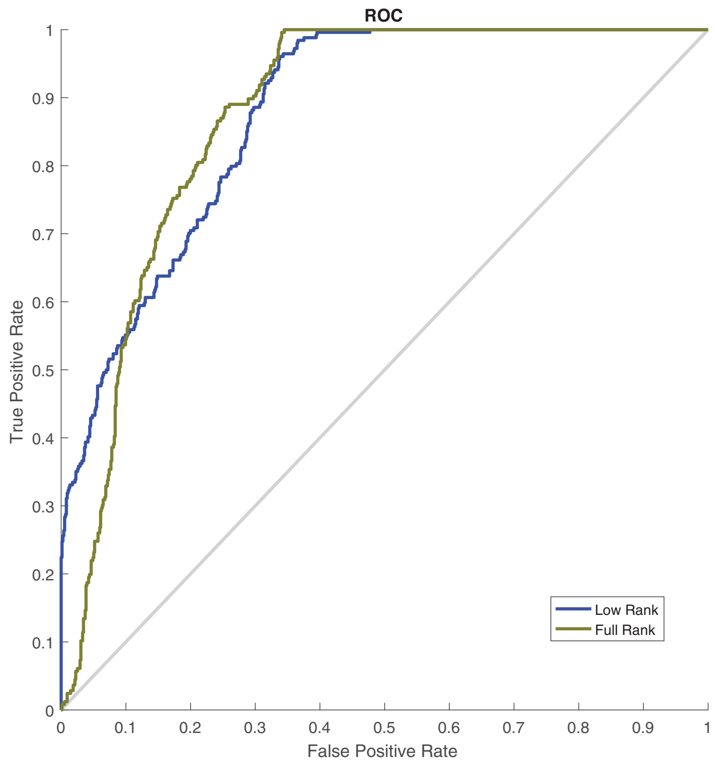


Figure 12. The true positive rate (TPR) against the false positive rate (FPR) for low-rank kernel-based SVM and full-rank.

Table 3. Accuracy, F_1 and MCC to check the performance of low rank approximation and full rank SVM.

Method	Accuracy	F_1	MCC
Full-rank	0.74	0.81	0.41
Low-rank	0.72	0.79	0.38

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall},$$

such that

$$precision = \frac{TP}{TP + FP},$$

and

$$recall = \frac{TP}{TP + FN}.$$

Also, the MCC which is between -1 and 1 is defined as

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}.$$

Table 3 gives us Accuracy, F_1 and MCC to check the performance of low rank approximation in comparison with full rank SVM. As can be seen, these results point out low-rank SVM has almost the same efficiency as full-rank SVM, although the computational time has been greatly reduced.

5. Applications

5.1. Financial application-S&P 500 stock index

One of the most important issues is to classify the S&P 500 stock index, since the prediction of the movement of the stock market is a long-time attractive topic to researchers from different fields. The S&P 500 Index is a free float-adjusted market capitalization-weighted stock market index in the United States. It is used to record and monitor daily changes of the largest companies of the American stock market and is the main indicator of the overall market performance in the United States. As US is one of the financial centers across the world, the S&P 500 is considered as one of the most important global financial indicators. The value of S&P 500 may be influenced by other major financial indexes across the world, like FTSE 100, NIKKEI 225, SSE, DJIA and NASDAQ, and exchange rates like USD/CNY, USD/JPY and USD/GBP. Besides, we have to consider the technical indicators of the S&P 500 itself, and the crude oil price, as well as the gold price, could also influence the stock price. Based on that, we try to apply low and full rank kernel-based SVM to classify the moving direction of the S&P 500 index. The higher accuracy rate we can get, the more confidence we will develop a profitable trading strategy.

Different financial features data from 01/01/2004 to 12/31/2014 are collected. This paper analyzes the features as shown in Table 4. There are three types of features. One type is other global financial market indexes in addition to the S&P 500, such as FTSE 100, NIKKEI 225 and SSE. The second one is some currency rates corresponding to different stock index for different stock exchanges and commodity price that may have an important impact on the financial market, particularly crude oil and gold. The third type is the technical indicator of S&P 500 itself, because these mathematics formulas give us clues about the trend of the market. Among many technical indicators, momentum and rate-of-change are chosen as inputs. All features may have possible impacts on the S&P 500 index. All feature data is extracted from the yahoo finance website and Bloomberg terminal.

Data obtained from the above resources is the absolute daily price information. However, the daily return data rather than the absolute daily price is more important from a financial perspective. So we transform the daily price into daily return using the formula as shown below:

$$r_n = \frac{P_n - P_{n-1}}{P_{n-1}} \times 100,$$

Note that P_n is the n^{th} daily price and P_{n-1} is the $(n-1)^{th}$ daily price and r_n is the n^{th} daily return. To simplify the prediction of the S&P 500 index, we classify S&P 500 daily return into two states, which are upward state and downward state. If the daily return of the S&P 500 index is negative, the output is a downward state, which will be presented as -1; otherwise, if the daily return of the S&P 500 index is non-negative, the output is a upward state, which will be presented as 1. Note that, in order to test the validation of those models, 2/3 data are used for training and 1/3 data for testing. Firstly low-rank and full-rank kernel-based SVMs are trained with all 16 features and their accuracies are examined using the testing dataset. The results gained show that the Accuracy for the full-rank case is 62.51% while this amount for low-rank is

Table 4. Different financial features with their category.

Feature	Category	Feature	Category
S&P 500 daily return lag one day	Stock Index	DJIA daily return	Stock Index
S&P 500 daily volume	Stock Index	NASDAQ daily return	Stock Index
FTSE daily return	Stock Index	Gold price daily return	Commodity
Nikkei daily return	Stock Index	Crude oil daily return	Commodity
SSE daily return	Stock Index	S&P 500 Momentum ($n=4$)	Stock Index
USDGBP daily return	Currency Rate	S&P 500 ROC ($n=4$)	Stock Index
USDCNY daily return	Currency Rate	S&P 500 Momentum ($n=3$)	Stock Index
USDJPY daily return	Currency Rate	S&P 500 ROC ($n=3$)	Stock Index

Table 5. Accuracy, F_1 and MCC to check the performance of low rank approximation and full rank SVM for S&P 500 index.

Method	Accuracy	F_1	MCC
Full-rank	0.83	0.89	0.46
Low-rank	0.80	0.87	0.43

61.13%, and there is a negligible difference between them. Also, the training time for low-rank SVM on a laptop with a 1.6 GHz Intel Core i7 processor was 12.1 min while this time for full-rank was 192.11 min. Finally, while the training time has been significantly reduced, there is no significant effect on classification accuracy. In this calculation, the value of the optimal parameter using 10-fold CV method for the considered Gaussian kernel is estimated as $\epsilon = 1.18$, and $C = 1.56$. Also, the number of terms used in the low-rank kernel approximation (M) is considered to equal 0.1 training points as stated before. Table 5 gives us Accuracy, F_1 and MCC to check the performance of low rank approximation in comparison with full rank SVM for the classification of S&P 500 index. As can be seen, these results show that low-rank SVM has almost the same efficiency as full-rank SVM, although the computational time has been reduced.

5.2. Biological application: recognition of promoters in drosophila sequence dataset

We use the 2002 collection of data of drosophila (D. melanogaster) promoter regions (Berkeley Drosophila Genome Project, 2002). The dataset has three parts. The promoter dataset contains 1842 sequences from −250 bp to +50 bp with regards to the gene transcription site location. The intron dataset contains 1799 sequences. The CDS (coding sequence) dataset contains 2859 sequences. The training file has 1260 examples (372 promoters, 361 introns, 527 CDS). The test file has 6500 examples (1842 promoters, 1799 introns, 2859 CDS). The difficulty with this dataset is that promoter sequences have both intron and CDS parts (from +1 bp to +50 bp), thus classification of promoters is much more difficult than classification of pure intron or CDS sequences. Furthermore, the dataset is unbalanced: there are 29.5% promoters against 70.5% nonpromoters in the training dataset, and 28.3% promoters against 71.7% non-promoters in the test dataset.

Before applying SVM classification to available dataset, first, we must define a mapping of sequences to the feature space. We use orthogonal encoding, where nucleotides in a DNA sequence are viewed as unordered categorical values and represented by 4-dimensional orthogonal binary vectors: $A \rightarrow 0001, C \rightarrow 0010, G \rightarrow 0100, T \rightarrow 1000$. This method allows achieving better classification results than a more compact mapping scheme, where each nucleotide is mapped into 2-dimensional binary vectors $A \rightarrow 00, C \rightarrow 01, G \rightarrow 10, T \rightarrow 11$, due to the identical Hamming distances between the nucleotide encodings (Demeler and Zhou 1991). Our aim is to classify DNA sequences as promoters and non-promoters. Therefore, here we have a binary classification problem in which the outcomes are labeled either as positive (P) or negative (N) class. Table 6 gives us Accuracy, F_1 and MCC to check the performance of low rank approximation in comparison with full rank SVM to classify DNA sequences as promoters and non-promoters. As can be seen, these results show that low-rank SVM has almost the same efficiency as full-rank SVM, although the computational cost has been significantly reduced.

6. Results and discussion

As mentioned in the Introduction to cope with the kernel-based SVMs, many researchers have been proposed different ways to produce different versions to speed up the computations and reduce the computational time. In this paper, we have developed an efficient approach to solve the Quadratic optimization problem in the kernel-based SVMs such that it can overcome the computational complexity, and also reduce the training time for classification. Several efforts have been made in this regard, for example, data reduction methods have been used to reduce the

Table 6. Accuracy, F_1 and MCC to check the performance of low rank approximation and full rank SVM to classify DNA sequences as promoters and non-promoters.

Method	Accuracy	F_1	MCC
Full-rank	0.88	0.93	0.34
Low-rank	0.84	0.91	0.33

Table 7. Accuracy, F_1 and MCC to check the performance of low rank approximation in comparison with previous methods.

Dataset	Method	Accuracy	F_1	MCC	Time
	Full-rank	0.83	0.89	0.46	12.1
S&P	Low-rank	0.80	0.87	0.43	192.11
500 index	Nguyen, Huang, and Joseph (2008)	0.79	0.86	0.44	50.16
	Bach (2013)	0.78	0.84	0.44	30.11
	Andersen and Vandenberghe (2014)	0.68	0.73	0.37	26.66
	Full-rank	0.88	0.93	0.34	27.31
DNA	Low-rank	0.84	0.91	0.33	430.45
sequence	Nguyen, Huang, and Joseph (2008)	0.83	0.90	0.32	73.11
	Bach (2013)	0.76	0.88	0.31	54.13
	Andersen and Vandenberghe (2014)	0.71	0.82	0.28	34.77

computational complexity and costs in the kernel-based SVMs by reducing the dimensionality of the original data (Nguyen, Huang, and Joseph 2008). Here, some valuable information exits in the original data could be lost through this dimensionality reduction. An alternative method to resolve this issue is to randomly select some columns of the kernel in the SVMs structure, which could increase the computational complexity (Bach 2013). Selecting these columns are not always straightforward, and this could create another challenging problem. Andersen and Vandenberghe (Andersen and Vandenberghe 2014) suggested a methodology by transforming the kernel matrix into a sparse matrix. Table 7 gives us Accuracy, F_1, MCC and computational time to check the performance of low rank kernel-based SVMs in comparison with previous methods to classify both datasets S&P 500 index and DNA sequence. As can be seen, in comparison with low-rank SVM other methods in this competition have been defeated. Also, although they have reduced the computational time compared to full-rank SVM, their computational time is very high compared to low-rank SVM.

7. Conclusions

In this paper, we introduce a novel low-rank approximation for the kernel-based SVM in order to classify binary data based on the truncated Mercer series expansion of the underlying kernel. In this methodology, the computational challenges due to the complex QOP associated with the standard kernel-based SVM are resolved by replacing the QOP with a much simpler and computationally efficient optimization problem. This would reduce the training time for the classification task using the approximated kernel-based SVM. The numerical results reveal that the accuracy and other measurements not much have changed for classification using the proposed low-rank approximation method in comparison with full-rank presentation. However, for a small number of training points, the computational time was negligible, but when the training data points increase, this difference also grows. In other words, it can be concluded that the approximated kernel-based SVM outperforms the full-rank kernel-based in reducing the computational cost without losing much accuracy and sensible changes to the other classiffication performance measurement. Similar results can be derived from analyzing the ROC curves of both techniques, the low-rank and full-rank presentation of the kernel-based SVM.

References

Andersen, M. S., and L. Vandenberghe. 2014. Support vector machine training using matrix completion techniques. *Journal of Machine Learning* 12:22–41.

Araújo, R. A., A. L. I. Oliveira, and S. Meira. 2017. A morphological neural network for binary classification problems. *Engineering Applications of Artificial Intelligence* 65:12–28. doi:10.1016/j.engappai.2017.07.014.

Bach, F. 2013. Sharp analysis of low-rank kernel matrix approximations. In *JMLR: Workshop and Conference Proceedings*. Vol. 30, 1–25.

Borah, P., and D. Gupta. 2021. Robust twin bounded support vector machines for outliers and imbalanced data. *Applied Intelligence* 51 (8):5314–43. doi:10.1007/s10489-020-01847-5.

Chatrabgoun, O., A. Daneshkhah, M. Esmaeilbeigi, N. Sohrabi Safa, A. H. Alenezi, and A. Rahman. 2022. Predicting primary sequence-based protein-protein interactions using a mercer series representation of nonlinear support vector machine. *IEEE Access* 10:124345–54. doi:10.1109/ACCESS.2022.3223994.

Chih-Jen, L. 2013. Support vector machines and kernel methods: Status and challenges. Talk at K. U. Leuven Optimization in Engineering Center. London: Cambridge University Press.

Christianini, N., and J. Shawe-Taylor. 2000. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge, UK: Cambridge University Press.

Cover, T. M. 1965. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers* EC-14 (3):326–34. doi:10.1109/PGEC.1965.264137.

Damaševicius, R. 2008. Optimization of SVM parameters for promoter recognition in DNA sequences. In *International Conference 20th EURO Mini Conference.*

Demeler, B., and G. W. Zhou. 1991. Neural network optimization for E. coli promoter prediction. *Nucleic Acids Research* 19 (7):1593–9. doi:10.1093/nar/19.7.1593.

Erdogan, B. E. 2013. Prediction of bankruptcy using support vector machines: An application to bank bankruptcy. *Journal of Statistical Computation and Simulation* 83 (8):1543–55. doi:10.1080/00949655.2012.666550.

Fasshauer, G. E. 2007. Meshfree approximation methods with Matlab. In *Interdisciplinary mathematical sciences*. Vol. 6. Singapore: World Scientific Publishing Co.

Fasshauer, G., and M. McCourt. 2016. *Kernel-based approximation method using Matlab*. Singapore: World Scientific Pub Co Inc.

Fasshauer, G. E., and J. G. Zhang. 2007. On choosing "optimal" shape parameters for RBF approximation. *Numerical Algorithms* 45 (1–4):345–68. doi:10.1007/s11075-007-9072-8.

Fornberg, B., and C. Piret. 2008. A stable algorithm for flat radial basis functions on a sphere. *SIAM Journal on Scientific Computing* 30 (1):60–80. doi:10.1137/060671991.

Griebel, M., C. Rieger, and B. Zwicknagl. 2015. Multiscale approximation and reproducing kernel Hilbert space methods. *SIAM Journal on Numerical Analysis* 53 (2):852–73. doi:10.1137/130932144.

Gupta, D., and B. Richhariya. 2018. Entropy based fuzzy least squares twin support vector machine for class imbalance learning. *Applied Intelligence* 48 (11):4212–31. doi:10.1007/s10489-018-1204-4.

Gupta, D., B. Richhariya, and P. Borah. 2019. A fuzzy twin support vector machine based on information entropy for class imbalance learning. *Neural Computing and Applications* 31 (11):7153–64. doi:10.1007/s00521-018-3551-9.

Han, J., M. Kamber, and J. Pei. 2012. *Data mining concepts and techniques*. 3rd ed. Waltham, MA: Elsevier Inc.

Hastie, T., R. Tibshirani, and J. Friedman. 2009. *Elements of statistical learning: data mining, inference, and prediction, 2nd ed., springer series in statistics*. New York: Springer-Verlag.

Hazarika, B. B., and D. Gupta. 2021. Density-weighted support vector machines for binary class imbalance learning. *Neural Computing and Applications* 33 (9):4243–61. doi:10.1007/s00521-020-05240-8.

Hazarika, B. B., and D. Gupta. 2022. Density weighted twin support vector machines for binary class imbalance learning. *Neural Processing Letters* 54 (2):1091–130. doi:10.1007/s11063-021-10671-y.

Hickernell, F. J., and Y. C. Hon. 1999. Radial basis function approximations as smoothing splines. *Journal of Applied Mathematics and Computing* 102 (1):1–24. doi:10.1016/S0096-3003(98)10012-7.

Lan, L., Z. Wang, S. Zhe, W. Cheng, J. Wang, and K. Zhang. 2019. Scaling up kernel SVM on limited resources: A low-rank linearization approach. *IEEE Transactions on Neural Networks and Learning Systems* 30 (2):369–78. doi:10.1109/TNNLS.2018.2838140.

Liu, C. D., J. H. Wang, D. Xiao, and Q. Liang. 2016. Forecasting S&P 500 stock index using statistical learning models. *Open Journal of Statistics* 6 (6):1067–75. doi:10.4236/ojs.2016.66086.

Nguyen, X., L. Huang, and A. D. Joseph. 2008. *Support vector machines, data reduction, and approximate kernel matrices*. Berlin Heidelberg: Springer-Verlag.

Nickisch, H., and C. E. Rasmussen. 2008. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research* 9:2035–78.

Poggio, T., S. Mukherjee, R. Rifkin, A. Rakhlin, and A. Verri. 2001. b, Tech. rep., MIT AI Memo 2001-011.

Qian, G., and L. Zhang. 2018. A simple feedforward convolutional conceptor neural network for classification. *Applied Soft Computing* 70:1034–41. doi:10.1016/j.asoc.2017.08.016.

Rasmussen, C. E., and C. K. I. Williams. 2006. *Gaussian processes for machine learning*. London: MIT Press.

Rippa, S. 1999. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Advances in Computational Mathematics* 11 (2–3):193–210. doi:10.1023/A:1018975909870.

Scheuerer, M. 2011. An alternative procedure for selecting a good value for the parameter c in RBF-interpolation. *Advances in Computational Mathematics* 34 (1):105–26. doi:10.1007/s10444-010-9146-3.

Scholkopf, B., and A. J. Smola. 2002. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. Cambridge: MIT Press.

Sosa, L. A. 2016. An overview of non-linear kernel functions for solving the human face recognition problem. PhD diss., University of California, Los Angeles.

Steinwart, I., and A. Christmann. 2008. *Support vector machines, information science and statistics*. Berlin: Springer.

Vapnik, V. 2013. *The nature of statistical learning theory*. New York: Springer-Verlag.

Yang, C., R. Duraiswami, and L. S. Davis. 2004. Efficient kernel machines using the improved fast Gauss transform. *Advances in Neural Information Processing Systems* 17:1561–8.