情報論的学習理論テクニカルレポート 2009
Technical Report on Information-Based Induction Sciences 2009 (IBIS2009)

楕円型サポートベクターマシン
Ellipsoidal Support Vector Machines

門馬道也*
Michinari Momma

Abstract: This paper proposes the ellipsoidal SVM (e-SVM) that uses an ellipsoid center, in the version space, to approximate the Bayes point. Since SVM approximates it by a sphere center, e-SVM provides an extension to SVM for better approximation of the Bayes point. Although the idea has been mentioned before [11], no work has been done for formulating and kernelizing the method. Starting from the maximum volume ellipsoid problem, we successfully formulate and kernelize it by employing relaxations. The resulting e-SVM optimization framework has much similarity to SVM; it is naturally extendable to other loss functions and other problems. A variant of the sequential minimal optimization is provided for efficient implementation. The empirical results are shown to be consistent with the Bayes point machines, in terms of classification performance, and difference from other related methods is highlighted by using high dimensional datasets.
Keywords: Bayes point machines, Support vector machines

1 Introduction

The most common interpretation of the support vector machines (SVMs) [19, 12] is that it separates positive and negative examples by maximizing the margin that is the distance between supporting hyperplanes of both examples. Another interpretation comes from a concept called the version space. The version space is a space of consistent hypotheses, or models with no error. SVM maximizes the inscribing hypersphere to find the center that is the SVM weight vector $\boldsymbol{w}$. Given the version space, the "sphere center" completely characterizes the SVM model. The Bayes point is a point through which all hyperplanes bisect the version space by half, and is shown to have better generalization ability theoretically and empirically[6, 11].

Attempts to approximately find the Bayes point have been done since the early studies of the version space and the Bayes point. SVM can be considered as an example. The Bayes point machines (BPM) [6] uses a kernel billiard algorithm to find the center of mass in the version space. The analytic center machines (ACM) [18] approximate the Bayes point by analytic points of linear constraints.

The idea of using an ellipsoid rather than a sphere has been mentioned in [11], although it was neither formulated nor implemented because of its projected high computational cost $O(n^{3.5})$. Then a billiard algorithm including BPM has been developed to alleviate the computational challenge. However, as we have seen in the history of SVM, seemingly expensive problem can be made efficient by exploiting special structures in the problem. Sequential minimal optimization (SMO) or decomposition methods are notable examples of such algorithms[9, 3]. Furthermore, recent development of large scale linear SVMs [13, 8] impressively improves the scalability of the quadratic optimization into practially linear order. Learning from the experience, we are encouraged to develop and study the method of ellipsoidal approximation to BPM, which we refer to as the ellipsoidal SVM (e-SVM).

e-SVM is formulated just like SVMs. Advantages in formulating in such a way include possible adaptation of theoretical characterization, optimization methods developed for SVM. Furthermore, extensions to chang-

*NEC 共通基盤ソフトウェア研究所, 211-8653 川崎市中原区下沼部 1753, tel. 044-431-7656, e-mail m-momma@cd.jp.nec.com,
NEC Common Platform Software Research Laboratories, 1753 Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa 211-8666

ing loss function or application to other kind of problems should be possible. These advantages would not be obtained if we stick to BPM that has to rely on sampling techniques that scale poorly on a large scale dataset; In BPM, even the soft boundary formulation is nontrivial and the kernel regularization is used after all.

The e-SVM formulation is based on the maximum volume inscribed ellipsoid (MVIE) problem. The original MVIE problem consists of second order cone (SOC) constraints and a semidefinite constraint. Although the problem is convex and an interior point type method can be applied for polynomial time convergence, the "natural" kernelization, i.e, the inner-product based mapping, is left as a challenge. For example, the second order cone program in [16] resorts to the direct kernel method where the original data matrix is replaced by the kernel matrix. e-SVM uses relaxation of SOC constraints to make the dual kernelizable. This technique has not been used before to the best of our knowledge.

As with the MVIE problem, e-SVM has the log-determinant term in the objective. Similar problems can be seen in [14, 5, 4, 16]. e-SVM problem may be interpreted as a combination of the regular SVM and the minimum volume covering ellipsoid (MVCE) problem. Thus, when optimized separately, e-SVM becomes very similar to ellipsoidal kernel machine (EKM) [14]. If applied to one class problem, e-SVM would become similar to [5]. In other words, e-SVM can subsume other related methods and possesses bigger optimization problem. By changing the loss function to a strict convex Bregman function, the Bregman's method would be applicable to solving e-SVM [4].

One direct interpretation of e-SVM is that it learns the Mahalanobis metric in margin. That is, e-SVM maximizes margin by adjusting the metric. In terms of adjusting margin, the relative margin machines (RMM) [15] address impact of data scaling on the performance in SVM. Since SVMs do not take into consideration of spread of data, a bad scaling can hurt the performance. Although e-SVM has more degree of freedom to adjust the margin, these two methods learn quite different models as we will see in Section 4.

As the first step to solving the challenging e-SVM problem efficiently, we adopt the sequential minimal optimization (SMO). The modified SMO algorithm indeed shares many convenient features with that for SVM, such as the closed-form solution for the minimal problem, Karush-Kuhn-Tucker (KKT) condition violation check, etc. Although there should exist faster algorithm to solve depending on the type of problems, we decide to start from the simpler SMO algorithm and study how e-SVM compares against BPM, SVM and other related methods.

Section 2 formulates the e-SVM optimization problem. Section 3 describes the SMO algorithm adapted for the e-SVM problem. Section 4 compares related methods by a simple example. Section 5 gives experimental results. Section 6 concludes the paper.

Notation: Throughout the paper, we assume that m data points $\boldsymbol{x}_i$ in n-dimensional space and the corresponding (target) label $y_i \in \{-1, 1\}$ are given. The bold small letters represent vectors and the capital letters represent matrices. The vector/matrix transpose is ^{T}. The kernel matrix is given by $\boldsymbol{K}$ with K_{ij} as its element. tr$A$ denotes the trace of a matrix $\boldsymbol{A}$. "s.t." in optimization problems means "subject to". I is an index set of m data points: $I \in \{1, \ldots, m\}$. $\|\boldsymbol{A}\|_2$ denotes the matrix 2-norm and $\|\boldsymbol{x}\|_2$ the L2-norm of a vector $\boldsymbol{x}$.

2 Ellipsoidal support vector machine formulations

In this section, the e-SVM optimization problem is formulated starting from that of SVM in the version space, since it is a simpler counterpart of e-SVM.

The version space is a space of error zero models. For linear models, it is the error-zero subspace of weight vectors $\boldsymbol{w}$. The data points are considered as hyperplanes and the classification constraints are the feasible region that is a polyhedron. The problem of finding a maximum hypersphere inside the polyhedron can be formulated as follows:

$$\max_{\rho, \boldsymbol{w}, b} \quad \rho \quad \text{s.t.} \quad \frac{y_i \left(\boldsymbol{x}_i^T \boldsymbol{w} + b\right)}{\|\boldsymbol{x}_i\|_2} \geq \rho, \quad \|\boldsymbol{w}\|_2 \leq 1, \; i \in I$$

which corresponds to maximization of the minimum distance between the center and the hyperplanes, in the absence of the bias b. By allowing errors in the above problem, we can get a soft-margin version of the

above problem.

$$\min_{\rho,\boldsymbol{w},b,\boldsymbol{\zeta}} \quad -m\rho + 1/\nu \sum_{i=1}^{m} \zeta_i$$
$$\text{s.t.} \quad y_i \left(\boldsymbol{x}_i^T \boldsymbol{w} + b\right) + t_i^2 \zeta_i \geq t_i^2 \rho, \ \|\boldsymbol{w}\|_2 \leq 1, i \in I \quad (1)$$

where t_i is defined to be $\|\boldsymbol{x}_i\|_2$ and $\nu > 0$ is a given constant. Note in the special case with $t_i = 1$, Problem 1 becomes identical to the ν-SVM formulation.

To better approximate the "center of models", an ellipsoid, instead of a hypersphere, will be used to inscribe the polyhedron. The MVIE problem is a well-known log-determinant optimization problem, see e.g. [2]. A representation of an ellipsoid centered at $\boldsymbol{w}$ is given by $\mathcal{E} = \{\boldsymbol{E}\boldsymbol{u} + \boldsymbol{w} \mid \|\boldsymbol{u}\|_2 \leq 1, \ \boldsymbol{E} \succeq 0\}$. Thus the constraints for SVM (1) are modified as follows:

$$y_i \left(\boldsymbol{x}_i^T \left(\boldsymbol{E}\boldsymbol{u} + \boldsymbol{w}\right) + b\right) + t_i^2 \zeta_i \geq t_i^2 \rho, \ \forall \boldsymbol{u}, \ \|\boldsymbol{u}\|_2 \leq 1 \quad (2)$$

Since Equation 2 holds for any $\boldsymbol{u}$, it suffices to use the lower bound of lhs in order to remove $\boldsymbol{u}$:

$$\begin{aligned} & y_i \left(\boldsymbol{x}_i^T \left(\boldsymbol{E}\boldsymbol{u} + \boldsymbol{w}\right) + b\right) + t_i^2 \zeta_i \\ & \geq y_i \left(\boldsymbol{x}_i^T \boldsymbol{w} + b\right) - \|\boldsymbol{E}\boldsymbol{x}_i\|_2 + t_i^2 \zeta_i \geq t_i^2 \rho \end{aligned} \quad (3)$$

where $-\frac{y_i \boldsymbol{E}\boldsymbol{x}_i}{\|\boldsymbol{E}\boldsymbol{x}_i\|_2} = \arg\min_{\boldsymbol{u}, \ \|\boldsymbol{u}\|=1} (y_i \boldsymbol{x}_i^T \boldsymbol{E}\boldsymbol{u})$ is used.

Furthermore, in order to obtain the largest ellipsoid inscribing a polyhedron, the volume of the ellipsoid should be maximized, which corresponds to maximizing the determinant of $\boldsymbol{E}$ ($|\boldsymbol{E}|$), as the volume of an ellipsoid is proportional to the determinant. In an optimization problem, log det is easier to handle and thus adopted here as well. The resulting optimization problem is given as follows:

$$\begin{aligned} \min_{\boldsymbol{E},\rho,\boldsymbol{\zeta},\boldsymbol{w},b} \quad & -\lambda \left(r \log |\boldsymbol{E}| + (1-r)\mathrm{tr}\boldsymbol{E}\right) - m\rho + \frac{1}{\nu} \sum \zeta_i \\ \text{s.t.} \quad & y_i \left(\boldsymbol{x}_i^T \boldsymbol{w} + b\right) - \|\boldsymbol{E}\boldsymbol{x}_i\|_2 \geq t_i^2 \rho - t_i^2 \zeta_i \\ & \|\boldsymbol{w}\|_2 \leq 1, \ \zeta_i \geq 0, \ i \in I, \ \boldsymbol{E} \succeq 0, \end{aligned} \quad (4)$$

where $\lambda > 0$ is a trade-off parameter and r is a constant whose value takes $0 < r \leq 1$. The additional term $\mathrm{tr}\boldsymbol{E}$ is introduced to gain numerical stability as suggested in [5].

Note the role of ρ and $|\boldsymbol{E}|$ as maximizing margin is similar and redundant; the determinant maximization term can subsume the linear maximization of ρ [1]. Hence, ρ is dropped from the problem hereafter, allowing us to remove λ:

$$\begin{aligned} \min_{\boldsymbol{E},\boldsymbol{\zeta},\boldsymbol{w},b} \quad & -r \log |\boldsymbol{E}| + (1-r)\mathrm{tr}\boldsymbol{E} + \frac{1}{\nu} \sum \zeta_i \\ \text{s.t.} \quad & y_i \left(\boldsymbol{x}_i^T \boldsymbol{w} + b\right) + t_i^2 \zeta_i \geq \|\boldsymbol{E}\boldsymbol{x}_i\|_2 \\ & \|\boldsymbol{w}\|_2 \leq 1, \ \zeta_i \geq 0, \ i \in I, \ \boldsymbol{E} \succeq 0. \end{aligned} \quad (5)$$

This MVIE problem can be solved by using existing techniques, including interior point methods or cutting plane based approaches. Here we relax the SOC constraint in Problem 5 in order to ease the high computational complexity. This change, as we shall see, plays a significant role in making the kernelized formulation possible. As the first step, assume the matrix $\boldsymbol{E}$ is written as $\boldsymbol{E} = \boldsymbol{E}_0 + \boldsymbol{B}$, where $\boldsymbol{E}_0$ is the current solution and $\boldsymbol{B}$ is a deviation from it. By the Taylor expansion, the SOC constraint is written as $\|\boldsymbol{E}\boldsymbol{x}_i\|_2 = \kappa_i + \frac{1}{\kappa_i} \boldsymbol{x}_i^T \boldsymbol{E}_0 \boldsymbol{B}\boldsymbol{x}_i + O(\|\boldsymbol{B}\|_2^2)$ where κ_i is given by $\kappa_i = \|\boldsymbol{E}_0 \boldsymbol{x}_i\|_2$. Using the convexity of SOC, we get the following inequality.

$$\|\boldsymbol{E}\boldsymbol{x}_i\|_2 \geq \kappa_i + (1/\kappa_i) \boldsymbol{x}_i^T \boldsymbol{E}_0 \boldsymbol{B}\boldsymbol{x}_i. \quad (6)$$

Now the SOC constraints are replaced by linear constraints that are much easier to handle. In the special case with $\boldsymbol{E}_0 = c\boldsymbol{I}$, $c \to +0$, the problem becomes simple and may be used as the initial problem.

$$\begin{aligned} \min_{\boldsymbol{B},\boldsymbol{\xi},\boldsymbol{w},b} \quad & -r \log |\boldsymbol{B}| + (1-r)\mathrm{tr}\boldsymbol{B} + \sum_i C_i \xi_i \\ \text{s.t.} \quad & y_i \left(\boldsymbol{x}_i^T \boldsymbol{w} + b\right) + \xi_i \geq \boldsymbol{x}_i^T \boldsymbol{B}\boldsymbol{x}_i \\ & \|\boldsymbol{w}\|_2 \leq 1, \ \boldsymbol{\xi} \geq 0, \ i \in I, \ \boldsymbol{B} \succeq 0 \end{aligned} \quad (7)$$

where we define $\xi_i = \kappa_i^2 \zeta_i$ and $C_i = \frac{1}{t_i^2 \nu}$. This formulates the ellipsoidal support vector machines primal problem. Note the Taylor approximation gets less accurate when $\|\boldsymbol{B}\|_2$ becomes larger, which is the cost for making the formulation feasible for kernelization done in Section 2.2.

Problem 7 has some interesting similarity with other methods. By putting $\boldsymbol{B} = \Sigma^{-1}$, it can be seen as a variant of MVCE problem in which the radius in the original problem is modified to a prediction dependent constraint. Hence it can be viewed as a supervised version of [14, 5]; unlike EKM, e-SVM solves the classification problem at the same time. Shivaswamy et al.'s formulation for handling missing and uncertain data [16] looks similar to Problem 4, where the metric in margin is given by the uncertainty in the data point.

[1] Our preliminary study confirmed that ρ becomes zero in most cases

In e-SVM, margin is given by the $\boldsymbol{B}$-norm, which is optimized simultaneously.

2.1 Dual formulation

It can be readily shown that Problem 7 is a convex optimization problem with no duality gap. Hence the complementarity can be used to solve the primal and the dual problems, just like SVMs. The Lagrangian is given as follows:

$$\begin{aligned}\mathcal{L} &= -r\log|\boldsymbol{B}| + (1-r)\,\mathrm{tr}\boldsymbol{B} + \sum_i C_i\xi_i \\ &\quad - \sum_i \alpha_i\left(y_i\left(\boldsymbol{x}_i^T\boldsymbol{w}+b\right) - \boldsymbol{x}_i^T\boldsymbol{B}\boldsymbol{x}_i - \xi_i\right) \\ &\quad + \gamma\left(\|\boldsymbol{w}\|_2^2 - 1\right) - \boldsymbol{\pi}^T\boldsymbol{\xi} - \mathrm{tr}(\boldsymbol{B}\boldsymbol{D}),\end{aligned}$$

where $\boldsymbol{\alpha}$, γ, $\boldsymbol{\pi}$ and $\boldsymbol{D}$ are the Lagrange multipliers for the classification constraints, norm constraint on $\boldsymbol{w}$, nonnegativity on $\boldsymbol{\xi}$ and positive semidefiniteness on $\boldsymbol{B}$, respectively. The optimality condition gives the following relations[2]:

$$\boldsymbol{B}^{-1} = \frac{1}{r}\left((1-r)\boldsymbol{I} + \sum_i \alpha_i\boldsymbol{x}_i\boldsymbol{x}_i^T\right),\ \boldsymbol{D}=0$$
$$\boldsymbol{w} = \frac{1}{2\gamma}\sum_i \alpha_i y_i\boldsymbol{x}_i,\ \sum_i y_i\alpha_i = 0,\ C_i - \alpha_i - \pi_i = 0.$$

Thus using the above equations the dual problem is written as follows:

$$\begin{aligned}\max_{\boldsymbol{\alpha},\gamma}\quad & r\log\left|\boldsymbol{B}^{-1}\right| - \frac{1}{4\gamma}\sum_{i,j}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j - \gamma \\ \text{s.t.}\quad & \boldsymbol{B}^{-1} = \frac{1}{r}\left((1-r)\boldsymbol{I} + \sum_i \alpha_i\boldsymbol{x}_i\boldsymbol{x}_i^T\right) \\ & \sum_i y_i\alpha_i = 0,\ 0\le\alpha_i\le C_i,\ \gamma>0 \qquad (8)\end{aligned}$$

A pleasant surprise is that $\boldsymbol{B}^{-1}$ is **always positive definite** since $\alpha_i \ge 0$, which is a great advantage, allowing us to remove the constraint $\boldsymbol{B}^{-1}\succeq 0$ in (8).

2.2 Kernel formulation

In this subsection, we show how Problem 8 is kernelized. For notational convenience, we use the matrix notation as well as the vector notation wherever appropriate. Note Problem 8 is very similar to the SVM problems, with the only difference being the additional $r\log\left|\boldsymbol{B}^{-1}\right|$ in the objective. By the matrix determinant lemma, the following equality can be shown to hold; $\left|\boldsymbol{B}^{-1}\right| = \left|\boldsymbol{I} + \frac{\boldsymbol{A}^{1/2}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{A}^{1/2}}{(1-r)}\right|\left|\frac{1-r}{r}\boldsymbol{I}\right|$, where $\boldsymbol{X}$ is the data matrix $\boldsymbol{X} = [\boldsymbol{x}_1\ldots\boldsymbol{x}_m]^T$ and $\boldsymbol{A}$ is a diagonal matrix whose elements are give by $\boldsymbol{A}_{i,i} = \alpha_i$. Note that the last factor is a constant so it can be ignored.

By employing the kernel defined feature mapping $\boldsymbol{x}\mapsto\phi(\boldsymbol{x})$, or $\boldsymbol{X}\boldsymbol{X}^T\mapsto\boldsymbol{K}$, we have

$$\begin{aligned}\left|\boldsymbol{I} + \tfrac{1}{(1-r)}\boldsymbol{A}^{1/2}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{A}^{1/2}\right| &\mapsto \left|\boldsymbol{I} + \frac{1}{(1-r)}\boldsymbol{A}^{1/2}\boldsymbol{K}\boldsymbol{A}^{1/2}\right| \\ &= \left|\boldsymbol{I} + \frac{1}{(1-r)}\boldsymbol{A}^{1/2}\boldsymbol{Z}\boldsymbol{Z}^T\boldsymbol{A}^{1/2}\right| \\ &= \left|\boldsymbol{I} + \frac{1}{(1-r)}\boldsymbol{Z}^T\boldsymbol{A}\boldsymbol{Z}\right|\end{aligned}$$

where $\boldsymbol{K}$ is decomposed as $\boldsymbol{K} = \boldsymbol{Z}\boldsymbol{Z}^T$ From the 2nd line to the 3rd line the Sylvester's determinant theorem, a generalization of the Matrix determinant lemma, is used. Note also the dimensionality of $\boldsymbol{I}$ changes before and after the theorem is applied. In general, $\boldsymbol{Z}$ can be any matrix such that $\boldsymbol{K} = \boldsymbol{Z}\boldsymbol{Z}^T$ including $\boldsymbol{Z} = \boldsymbol{K}^{1/2}$ so that a rank reduction method or sparsification method such as the incomplete Cholesky factorization may be used. After removing the constant terms, the kernel e-SVM optimization problem is given by

$$\begin{aligned}\max\quad & r\log\left|\boldsymbol{I} + \frac{1}{(1-r)}\sum_i \alpha_i\boldsymbol{z}_i{\boldsymbol{z}_i}^T\right| \\ & -\frac{1}{4\gamma}\sum_{i,j} y_i y_j\alpha_i\alpha_j K_{ij} - \gamma \\ \text{s.t.}\quad & \sum_i y_i\alpha_i = 0,\ 0\le\alpha_i\le C_i,\ \gamma\ge 0, \qquad (9)\end{aligned}$$

with $\boldsymbol{z}_i$ being the transpose of the i-th **row** of the matrix $\boldsymbol{Z}$: $\boldsymbol{Z} = [\boldsymbol{z}_1\ldots\boldsymbol{z}_m]^T$.

3 Sequential minimal optimization

Although Problem 9 can be solved by an optimization package, a customized solver should be developed to take advantage of its similarity to the familiar SVM formulation; ideally an SVM solver can be modified to handle e-SVM. For this purpose, we develop a variant of SMO for e-SVM.

The differences from the standard implementation of SMO include $\|\boldsymbol{w}\|_2$ being normalized to one, step

[2] The $\log|\boldsymbol{B}|$ term forces $\boldsymbol{B}$ to be full-rank. Thus $\boldsymbol{D}=0$ holds by complementarity.

size optimization formula, and KKT conditions. The weight normalization concerns optimization with respect to γ and can be done via the iterative projection. Step size optimization and active set selection using the KKT condition are done very similar to those for SVM. This section focuses on describing essential differences as a guide to implementation.

3.1 Optimality conditions

SMO chooses an active set, a pair of data points, to optimize at any iteration. The selection of a pair critically affects the convergence speed. We adopt the selection heuristic described in [9]: choose ones that violate the KKT condition most. This subsection derives the KKT condition and thus gives the criterion for choosing the active set.

First, consider the dual of (9). The Lagrangian is given by

$$\begin{aligned}\mathcal{L} &= -r\log\left|\frac{1-r}{r}\boldsymbol{I}+\frac{1}{r}\sum_i \alpha_i \boldsymbol{z}_i {\boldsymbol{z}_i}^T\right| \\ &\quad +\frac{1}{4\gamma}\sum_{ij} y_i y_j \alpha_i \alpha_j K_{ij} + \gamma - \eta\sum_i y_i\alpha_i \\ &\quad -\sum_i \delta_i\alpha_i + \sum \mu_i\left(\alpha_i - C_i\right).\end{aligned}$$

Solving the optimality conditions, we have

$$\left(F_i-\gamma\right)y_i - \delta_i + \mu_i - {\boldsymbol{z}_i}^T\widetilde{\boldsymbol{B}}\boldsymbol{z}_i = 0 \quad (10)$$

$$\gamma = \sqrt{\sum_{ij} y_i y_j \alpha_i \alpha_j K_{ij}}/2. \quad (11)$$

with $F_i = \frac{1}{2\gamma}\sum K_{ij}y_j\alpha_j$ and $\widetilde{\boldsymbol{B}} = \left(\frac{1-r}{r}\boldsymbol{I} + \sum_i \alpha_i \boldsymbol{z}_i {\boldsymbol{z}_i}^T\right)^{-1}$. Hence, by the complementarity, we have the following KKT conditions:

- For $\alpha_i = 0,\ \delta_i > 0,\ \mu_i = 0 \Rightarrow \left(H_i - \beta\right)y_i \geq 0$
- For $0 < \alpha_i < C_i,\ \delta_i,\ \mu_i = 0, \Rightarrow \left(H_i-\beta\right)y_i = 0$
- For $\alpha_i = C_i,\ \delta_i = 0,\ \mu_i > 0 \Rightarrow \left(H_i-\beta\right)y_i \leq 0$

with $H_i = F_i - y_i {\boldsymbol{z}_i}^T\widetilde{\boldsymbol{B}}\boldsymbol{z}_i$. Note the first term F_i corresponds to that in [9] and the second term is newly introduced for the e-SVM problem. This means that replacing F_i by H_i suffices to establish a version of the SMO algorithm for e-SVM and can be easily integrated into an existing SVM solver.

3.2 Step size computation

As explained, the KKT condition for e-SVM is easily adopted to the existing SMO algorithm. Another important piece in SMO algorithm is to find the optimal step size. The incremental step for α_i can be expressed as

$$\boldsymbol{\alpha}^{new} = \boldsymbol{\alpha}^{old} + s\left(\boldsymbol{e}_i - y_i y_j \boldsymbol{e}_j\right), \quad (13)$$

which satisfies the constraint $\sum_i y_i \alpha_i^{new} = 0$ given $\boldsymbol{\alpha}^{old}$ is a feasible solution. $\boldsymbol{e}_i$ is a vector of zeros except for the i-th element being unity. Consider the following objective function, $U(s)$, after removing any constant terms with respect to s:

$$U(s) = r\log\det\widetilde{\boldsymbol{B}}^{-1} - \sum_{i,j}\frac{1}{4\gamma}\alpha_i^{new}\alpha_j^{new}y_i y_j K_{ij} - \gamma. \quad (14)$$

The first term is modified using the update formula:

$$\begin{aligned}\left|\widetilde{\boldsymbol{B}}^{-1}\right| &= \left|\widetilde{\boldsymbol{B}}^{old\,-1} + \frac{s}{r}\left(\boldsymbol{z}_i{\boldsymbol{z}_i}^T - y_i y_j \boldsymbol{z}_j {\boldsymbol{z}_j}^T\right)\right| \\ &= \left|\boldsymbol{I} + \frac{s}{r}\begin{bmatrix}{\boldsymbol{z}_i}^T \\ -y_i{\boldsymbol{z}_j}^T\end{bmatrix}\widetilde{\boldsymbol{B}}^{old}\left[\boldsymbol{z}_i\ y_j\boldsymbol{z}_j\right]\right|\left|\widetilde{\boldsymbol{B}}^{old\,-1}\right| \\ &= \begin{vmatrix}1+\frac{s}{r}\omega_{ii} & \frac{s}{r}y_j\omega_{ij} \\ -\frac{s}{r}y_i\omega_{ij} & 1-\frac{s}{r}y_i y_j\omega_{jj}\end{vmatrix} \times const \quad (15)\end{aligned}$$

where ω_{ij} is defined to be $\omega_{ij} = {\boldsymbol{z}_i}^T\widetilde{\boldsymbol{B}}^{old}\boldsymbol{z}_i$ and the matrix determinant lemma is used for deriving the 2nd line. The resulting matrix is merely a 2×2 matrix determinant and easily expandable.

Likewise, we can rewrite the second term in (14) as follows:

$$\begin{aligned}\sum_{i,j}\alpha_i^{new}\alpha_j^{new}y_i y_j K_{ij} &= \sum_{i,j}\alpha_i^{old}\alpha_j^{old}y_i y_j K_{ij} \\ &\quad -4\beta s y_i\left(F_i - F_j\right) \\ &\quad -s^2\left(K_{ii}-2K_{ij}+K_{jj}\right).\end{aligned}$$

Hence by putting all the pieces together, we have the following optimality condition on s.

$$\begin{aligned}\frac{\partial U(s)}{\partial s} &= r\frac{\partial\log\det\widetilde{\boldsymbol{B}}^{-1}}{\partial s} - \frac{\partial}{\partial s}\left(\frac{1}{4\gamma}\sum_{ij}\alpha_i\alpha_j y_i y_j K_{ij}\right) = 0 \\ &\Rightarrow\ ra_1 - a_3 + (2ra_2 - a_1a_3 - a_4)s \\ &\quad -(a_2a_3 + a_1a_4)s^2 - a_2a_4s^3 = 0\end{aligned}$$

with $a_1 = r(\omega_{ii} - y_i y_j\omega_{jj})$, $a_2 = y_i y_j(\omega_{ij}^2 - \omega_{ii}\omega_{jj})$, $a_3 = y_i\left(G_i - G_j\right)$, $a_4 = \frac{K_{ii}+K_{ii}-2K_{ij}}{2\gamma}$. This is merely a cubic equation and can be solved **analytically**.

3.3 Computing $\widetilde{\boldsymbol{B}}$

At each iteration, access to $\widetilde{\boldsymbol{B}}$ is needed to calculate ω's. Specifically, the diagonal elements ω_{ii} are required

for the KKT violaiton check and ω_{ij} as well as ω_{ii} and ω_{jj} for the step size computation concerning an update of α_i and α_j. Since we solve the dual $\boldsymbol{\alpha}$, as well as γ in SMO, $\widetilde{\boldsymbol{B}}^{-1}$ is easily obtained, but getting $\widetilde{\boldsymbol{B}}$, in a naive way, would require an inverse matrix operation that is never done in practice.

A way to efficiently compute $\widetilde{\boldsymbol{B}}$ is to employ the rank-one update of matrix inversion and factorize the matrix in the following way: $\widetilde{\boldsymbol{B}} = \boldsymbol{B}_0 + \sum_i \sigma_i \boldsymbol{v}_i \boldsymbol{v}_i^T$. By using the Woodbury formula, $\widetilde{\boldsymbol{B}}$ is updated at each SMO step involving update of α_i and α_j: $\widetilde{\boldsymbol{B}}^{new} = \widetilde{\boldsymbol{B}}^{old} + \sigma_i \boldsymbol{v}_i \boldsymbol{v}_i^T + \sigma_j \boldsymbol{v}_j \boldsymbol{v}_j^T$, where $\boldsymbol{v}_i = \widetilde{\boldsymbol{B}}^{old} \boldsymbol{z}_i$, $\omega_{ii} = {\boldsymbol{z}_i}^T \boldsymbol{v}_i$, $\sigma_i = -\frac{s}{r+s\omega_{ii}}$. $\boldsymbol{v}_j = \left(\widetilde{\boldsymbol{B}}^{old} + \sigma_i \boldsymbol{v}_i \boldsymbol{v}_i^T\right) \boldsymbol{z}_j$, $\sigma_j = \frac{s y_i y_j}{r - s y_i y_j {\boldsymbol{z}_j}^T \boldsymbol{v}_j}$. Note this decomposition formula on $\widetilde{\boldsymbol{B}}$ enables us to do an incremental update of ω:

$$\omega_{kl}^{new} = \omega_{kl}^{old} + \sigma_i {\boldsymbol{z}_i}^T \boldsymbol{z}_k {\boldsymbol{z}_i}^T \boldsymbol{z}_l + \sigma_j {\boldsymbol{z}_j}^T \boldsymbol{z}_k {\boldsymbol{z}_j}^T \boldsymbol{z}_l$$

where $\omega_{kl}^{old} = {\boldsymbol{z}_k}^T \widetilde{\boldsymbol{B}}^{old} \boldsymbol{z}_l$. This update formula is particularly useful when exploiting efficiency; it is evident that computing ω_{kl} with $\widetilde{\boldsymbol{B}}$ close to full-rank or dense is overkill as SMO requires many updates. ω_{ij}'s that are cached will be updated via the vector-vector multiplications. For ω_{ij}'s that are not in the cache may require the full matrix-vector multiplication, if $\widetilde{\boldsymbol{B}}$ is computed and kept in memory. Otherwise, the conjugate gradient would need to be employed to calculate ω_{ij} from $\widetilde{\boldsymbol{B}}^{-1}$, see [7], for example. A recommendation is that the diagonal elememts ω_{ii} are kept in memory as all of them are used in any case for the KKT condition violation check.

4 Related methods

Recently, several methods are proposed for improving generalization ability of SVMs. Ellipsoidal kernel machines (EKM) are ellipsoidal gap-tolerant classifiers that have lower bound of the VC dimension than that of SVMs. The EKM algorithm first finds the minimum volume ellipsoid enclosing data points. Then the normal SVM is applied to the transformed space where data points are placed in a hypersphere.

Relative margin machines (RMM) addresses the issue of scale invariance in SVMs. Being motivated by a simple example where enlarging one feature dimension can drastically change the solution of an SVM, RMM regularizes the amplitude of prediction value, or projection of data point onto the weight vector $\boldsymbol{w}$ so as to reduce the effect of influence from badly scaled features.

5 Experimental study

5.1 Comparison against BPM

It is important to examine the quality of e-SVM solutions to understand how the approximation and relaxation used in e-SVM affect the performance. In this paper, we directly compare the classification performance using the standard real dataset as a delegate to an assessment from an optimization perspective, which is left for a longer version of the paper. Also, BPM and e-SVM are compared against SVM to understand how much performance lift can be realized.

In this regard, a wide range of datasets in the UCI machine learning repository [1] are used. Both SVM and e-SVM are implemented in pure MATLAB, using the SMO. Note ν-SVM formulation is adopted in this study, since e-SVM is based on ν-SVM. BPM's implementation follow [6] and is implemented in C with an R interface. For the experimental setting, 100 randomizations are done and the average error rates in percent is reported in Table 1 and 2. In order to evaluate significance of statistics, the paired t-tests are conducted for comparing BPM with SVM, and e-SVM with SVM. SVM. Bold numbers denote the test results being significant.

Both hard and soft margin/boundary cases are examined. The training and testing splits are identical to [10] except for *sonar*, *iono*, *wisconsin-breastcancer* (WISC-BC). The splits for *sonar* are set identical to [6] and those for *iono* are created randomly in advance and identical splits are applied for all methods. For e-SVM and SVM, the tolerance of KKT violation is set to 10^{-3}. For BPM the tolerance parameter for convergence check is set to 10^{-3}. Model parameters are selected following [10]: using five-fold cross validation on the first five set and take the median of the best parameters. The best model parameters are fixed to conduct the 100 repetitions. Parameters including C and r for e-SVM, ν for SVM and λ in BPM are selected by the cross validation, in which 10-20 parameter values are used to cover a wide range of the search space. For hard margin e-SVM, r is set to $1 - 10^{-6}$. The radial basis function kernel is used for this experiment.

Table 1: Error rates for hard boundary/margin classifiers

Data set	SVM	BPM	e-SVM
THYROID	4.96 (.24)	**4.24 (.22)**	**4.42 (.25)**
HEART	25.86 (.40)	**22.58 (.33)**	**20.87 (.32)**
DIABETES	33.87 (.21)	**31.06 (.22)**	**29.68 (.24)**
WAVE	13.19 (.12)	**12.02 (.08)**	**11.59 (.07)**
BANANA	16.24 (.14)	**13.70 (.10)**	**12.76 (.08)**
WISC-BC	4.22 (.13)	**2.56 (.10)**	**3.28 (.12)**
BUPA	37.04 (.39)	**34.5 (.38)**	**32.71 (.35)**
GERMAN	30.07 (.22)	**27.16 (.24)**	**26.34 (.27)**
BREST	35.17 (.51)	**33.04 (.48)**	**31.96 (.51)**
SONAR	**14.90 (.38)**	16.26 (.36)	16.87 (.38)
IONO	7.94 (.25)	11.45 (.25)	**5.92 (.21)**

The kernel width parameters are set to identical to those reported in [6], except for *iono*, for the rest of the datasets, those reported for SVMs in [10] are used.

The overall performance for BPM and e-SVM is very similar. This suggests that the approximations made to formulate e-SVM do not affect the classification performance for the datasets examined. In comparison with SVM, the hard boundary/margin classifiers significantly outperform those of SVM. For soft boundary/margin cases, however, the advantage is reduced. Although performance of BPM and e-SVM is slightly better than that of SVM, the difference is small, which is a consistent observation with [6].

Scalability

The computational cost is illustrated to see how e-SVM scales in comparison with BPM, using the *adult* dataset [1]. The entire dataset is split into training, testing and validation. Model selection is done using the validation set of size 1000 and fixed for the rest of the experiments. The size of the training set is increased from 100 up to 4000. Test performance is observed to check if the there is no significant difference between the methods. Obviously, e-SVM runs much faster than BPM as the data size grows. Using the log-log fit, e-SVM scales as $m^{2.1}$ whereas BPM takes as much as $m^{2.7}$. Note SVM takes only 30 secs for the problem of size 4,000 (not shown); although e-SVM significantly beats BPM, there is much room to improve the efficiency when compared with SVM.

5.2 High dimensional datasets

The datasets used in Subsection 5.1 are in relatively low dimensions. With the ability to adjust the margin metric, it is more interesting to see the performance

Table 2: Error rates for soft boundary/margin classifiers

Data set	SVM	BPM	e-SVM
THYROID	5.05 (.22)	**4.32 (.20)**	**4.44 (.22)**
HEART	**15.58 (.30)**	16.19 (.29)	16.17 (.32)
DIABETES	**23.36 (.17)**	23.12 (.18)	23.79 (.19)
WAVE	**9.88 (.04)**	11.82 (.05)	10.03 (.04)
BANANA	10.90 (.07)	**10.40 (.04)**	**10.59 (.05)**
WISC-BC	2.70 (.10)	**2.32 (.10)**	2.77 (.09)
BUPA	28.80 (.30)	**28.37 (.30)**	**28.38 (.32)**
GERMAN	23.61 (.20)	**23.27 (.22)**	**23.21 (.22)**
BREAST	29.01 (.43)	**26.21 (.46)**	**26.08 (.48)**
SONAR	**14.93 (.38)**	16.49 (.36)	16.12 (.36)
IONO	5.94 (.18)	10.16 (.23)	**5.16 (.18)**

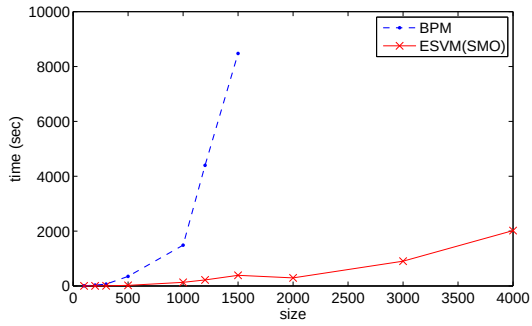

Figure 1: Scalability for BPM and e-SVM

with a high dimensional dataset. We compare the e-SVM with other methods mentioned in Section 4. The 20-newsgroup ($20NG$) and *mnist* [1] are used for this purpose. EKM is implemented in MATLAB Optimization Toolbox. RMM is implemented in SDPT3 [17]. For $20NG$, the number of words is limited to the top 5,000 frequent words and each data point is normalized to length one after the tf-idf weighting. There are 11,250 data points in the training and 7,486 in the testing set. *mnist* contains 60,000 observations in the training and 10,000 in the testing set in 784 dimensions.

The one-versus-one strategy is applied to solving the multiclass problem. We set the number of training data points to 50, 100 and 200 for each pair of classes in training (100, 200 & 400 in training samples for both classes). The rest of the training data is reserved as the validation set that is used for model selection. The linear kernel is used for all the methods. The results are shown in Table 3. For $20NG$, e-SVM outperforms other methods. Interestingly, both EKM and RMM show slightly worse performance than SVM. This indicates the transformation in EKM and regularization for scaling resistance in RMM seem to have adversary effect. In turn, for *mnist*, RMM outperforms other

Table 3: Error rates for SVM, EKM, RMM and e-SVM

	SVM	EKM	RMM	e-SVM
NG100	35.0	36.4	37.7	**34.0**
NG200	30.2	32.8	31.2	**29.1**
NG400	27.0	31.5	27.8	**26.4**
MNIST100	10.8	13.0	**10.5**	10.7
MNIST200	9.31	10.9	**8.92**	9.40
MNIST400	8.11	9.44	**8.10**	8.28

methods, which confirms RMM works well when scale invariance is desired.

6 Conclusion

In this paper, the ellipsoidal support vector machine was proposed. The formulation is based on that of the familiar SVM and the sequential minimal optimization was successfully adapted to solve the e-SVM optimization problem. The framework is flexible for possible modification of loss functions or application to other problems. Also, by the minimum volume ellipsoid interpretation, it can be used to learn the metric guided through the maximum margin framework. None of these advantages is available in BPM and thus novel in e-SVM. Furthermore, e-SVM showed comparable performance with BPM, indicating the approximations in e-SVM do not affect the performance over wide variety of datasets.

The SMO algorithm was shown to provide acceptable scalability and was much more efficient than BPM. e-SVM thus can be applicable to real world applications up to several thousands of data points. Future work includes developing a more efficient algorithm that handles caching better and an algorithm for out-of-memory computation.

References

[1] C. L. Blake and C. J. Merz. UCI Repository of machine learning databases, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[3] Pai-Hsuen Chen, Chih-Jen Lin, and Bernhard Schölkopf. A tutorial on ν-support vector machines: Research articles. *Appl. Stoch. Model. Bus. Ind.*, 21(2):111–136, 2005.

[4] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 209–216, New York, NY, USA, 2007. ACM.

[5] A.N. Dolia, T. De Bie, C.J. Harris, J. Shawe-Taylor, and D.M. Titterington. The minimum volume covering ellipsoid estimation in kernel-defined feature spaces. In *Proceedings of the 17th European Conference on Machine Learning (ECML 2006), Berlin, September 2006*. 2006.

[6] Ralf Herbrich, Thore Graepel, and Colin Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279, 2001.

[7] Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. Learning nonparametric kernel matrices from pairwise constraints. In *ICML*, pages 361–368, 2007.

[8] C. Hsieh, K. Chang, C. Lin, S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. pages 408–415, 2008.

[9] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Comput.*, 13(3):637–649, 2001.

[10] Gunnar Rätsch. Benchmark repository, 1998. `http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm`.

[11] Pál Ruján. Playing billiards in version space. *Neural Comput.*, 9(1):99–122, 1997.

[12] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[13] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 807–814, New York, NY, USA, 2007. ACM.

[14] P. Shivaswamy and T. Jebara. Ellipsoidal kernel machines. *Artificial Intelligence and Statistics*, 2007.

[15] P. Shivaswamy and T. Jebara. Relative margin machines. In *Neural Information Processing Systems*, 2008.

[16] Pannagadatta K. Shivaswamy, Chiranjib Bhattacharyya, and Alexander J. Smola. Second order cone programming approaches for handling missing and uncertain data. *J. Mach. Learn. Res.*, 7:1283–1314, 2006.

[17] K. C. Toh, M. J. Todd, and R. Tutuncu. SDPT3 — a Matlab software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.

[18] Theodore B. Trafalis and Alexander M. Malyscheff. An analytic center machine. *Mach. Learn.*, 46(1-3):203–223, 2002.

[19] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1996.