Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Random Gradient Descent Tree: A Combinatorial Approach for SVM with Outliers[*]

Hu Ding and **Jinhui Xu**
Department of Computer Science and Engineering, State University of New York at Buffalo
{huding, jinhui}@buffalo.edu

Abstract

Support Vector Machine (SVM) is a fundamental technique in machine learning. A long time challenge facing SVM is how to deal with outliers (caused by mislabeling), as they could make the classes in SVM non-separable. Existing techniques, such as soft margin SVM, ν-SVM, and Core-SVM, can alleviate the problem to certain extent, but cannot completely resolve the issue. Recently, there are also techniques available for explicit outlier removal. But they suffer from high time complexity and cannot guarantee quality of solution. In this paper, we present a new combinatorial approach, called *Random Gradient Descent Tree (or RGD-tree)*, to explicitly deal with outliers; this results in a new algorithm called *RGD*-SVM. Our technique yields provably good solution and can be efficiently implemented for practical purpose. The time and space complexities of our approach only linearly depend on the input size and the dimensionality of the space, which are significantly better than existing ones. Experiments on benchmark datasets suggest that our technique considerably outperforms several popular techniques in most of the cases.

1 Introduction

Support Vector Machine (SVM) is a fundamental tool for classification (Chang and Lin 2011). For a given set of points in Euclidean space labeled with $+1$ or -1 each, SVM is to find a separating hyperplane so that points with different labels are located on different sides of the hyperplane and the margin of separation is maximized. In an ideal scenario where all labeled points are separable, SVM can be modeled as a convex quadratic program, and solved optimally by using *Lagrange multiplier*. However, in real world applications, data often contain noises or outliers and may not always be separable (by either linear or non-linear classifier (Aizerman, Braverman, and Rozonoer 1964)). To resolve this challenging issue, a great deal of efforts have been devoted to this problem and a number of techniques have been developed in the past, such as soft margin SVM (Cortes and Vapnik 1995; Platt 1999), ν-SVM (Scholkopf et al. 2000; Crisp and Burges 1999), Core-SVM (Tsang, Kwok, and Cheung 2005), and many other techniques (Har-Peled, Roth, and Zimak 2007; Krause and Singer 2004).

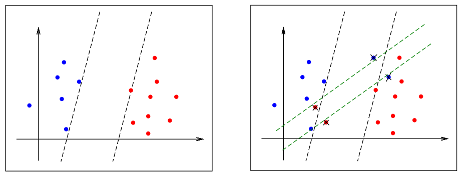

(a) (b)

Figure 1: SVM with (b) and without (a) outliers; the green margin in (b) is the output from soft margin SVM; the black margin is the output after removing outliers, which is the same as in (a).

Each of the above techniques alleviates the problem from a different perspective, and achieves good performance for certain types of data. However, since such techniques do not explicitly remove the outliers, their performance could be considerably deteriorated by outliers, especially when there exists a significant fraction of outliers in the dataset (see Fig. 1). In recent years, there are also techniques (Xu, Crammer, and Schuurmans 2006; Suzumura et al. 2014) available for explicit outliers removal. But most of them are numerical approaches (*e.g.,* by adding penalties to the objective function), and cannot guarantee the quality of solutions (*i.e.,* often trapped by local optimum).

1.1 Our Results

In this paper, we present a novel combinatorial technique called ***Random Gradient Descent (RGD) Tree*** to identify and remove outliers in SVM; this results in a new algorithm called ***RGD*-SVM**. Comparing to the existing numerical approaches, our technique yields provably good solutions (with respect to the optimal solution) and can be efficiently implemented for practical purpose. Experimental results on several benchmark data sets (from (Chang and Lin 2011)) suggest that our technique outperforms existing popular approaches.

Furthermore, our technique has significantly better time and space complexities than existing approaches. It runs in

[*]The research of this work was supported in part by NSF under grants IIS-1115220, IIS-1422591, and CCF-1422324.

linear time in the input size n, while the algorithm in (Suzumura et al. 2014) needs to solve a sequence of quadratic programs and the algorithm in (Xu, Crammer, and Schuurmans 2006) relies on a semi-definite programming formulation; thus both have a much higher time complexity than ours. Note that although linear or even sub-linear time SVM algorithms (Tsang, Kwok, and Cheung 2005; Har-Peled, Roth, and Zimak 2007; Gärtner and Jaggi 2009; Clarkson 2010; Clarkson, Hazan, and Woodruff 2012) exist, they mainly focus on hard or soft margin SVM and do not explicitly remove outliers. Thus it is not very meaningful to compare those with ours on running time. As for the space complexity, our approach takes only linear space (in n), which is significantly better than the numerical approaches (Xu, Crammer, and Schuurmans 2006; Suzumura et al. 2014).

2 Random Gradient Descent Tree

In this section, we present our main result, *Random Gradient Descent Tree* ((RGD)-tree), and show its application in SVM with outliers. Since our idea is inspired by Gilbert algorithm, we first give an overview of it in Section 2.1. Then, we present RGD-tree in Section 2.2 and its correctness in Section 2.3 for computing one-class SVM with outliers. Finally, we extend RGD-tree to handle two-class SVM with outliers in Section 2.4, and analyze its time and space complexities in Section 2.5.

2.1 Gilbert Algorithm and One-class SVM

Let o be a point and P be a set of n points in $\mathbb{R}^d$ space. The shortest distance between o and $conv(P)$ (*i.e.,* the convex hull of P) is called the *polytope distance* between o and P. Computing polytope distance is a key problem in SVM and has been extensively studied in the past (Gärtner and Jaggi 2009; Clarkson, Hazan, and Woodruff 2012; Clarkson 2010; Keerthi et al. 2000; Mavroforakis, Sdralis, and Theodoridis 2007).

Without loss of generality (WLOG), we assume that o is the origin of the coordinate system. Note that polytope distance is naturally a convex quadratic optimization problem, and can be solved optimally in polynomial time. To obtain more efficient solution, a gradient descent algorithm, called *Gilbert Algorithm*, has been developed to achieve approximate solution (Frank and Wolfe 1956; Gilbert 1966; Mitchell, Dem'yanov, and Malozemov 1974). **Algorithm 1** outlines its main steps. In each step, the algorithm finds a direction, and greedily improves the current solution along this direction until the solution becomes stable.

A theoretical analysis on the convergence of Algorithm 1 has recently been obtained. Before discussing this result, we first introduce several notations. Let ρ be the optimal solution of Algorithm 1 (*i.e.,* the polytope distance between o and P), and $D = \max_{p,q \in P} ||p - q||$ be the diameter of P. Define $E = \frac{D^2}{\rho^2}$, and denote by $p\ |_{x_i}$ the orthogonal projection of a point p on the supporting line of segment $\overline{ox_i}$.

Definition 1 (ϵ-Approximation of Polytope Distance). *Let P be a point-set in $\mathbb{R}^d$ and $x_i \in conv(P)$. x_i is an ϵ-approximation of the polytope distance of P if $||x_i|| \leq ||p\ |_{x_i}\ || + \epsilon ||x_i||$ for any $p \in P$.*

Algorithm 1 Gilbert Algorithm (Gilbert 1966)

```
Input: A point-set $P$ in $\mathbb{R}^d$.
Output: $x_i$ as an approximate solution of the polytope distance between the origin and $P$.
1. Initialize $i = 1$ and $x_1$ to be the closest point in $P$ to the origin $o$.
2. Iteratively perform the following steps until the solution becomes stable.
   (a) Find the point $p_i \in P$ whose orthogonal projection on the supporting line of segment $\overline{ox_i}$ has the closest distance to $o$ (called the projection distance of $p_i$), i.e., $p_i = \arg\min_{p \in P}\{\frac{\langle p, x_i \rangle}{||x_i||}\}$, where $\langle p, x_i \rangle$ is the inner product of $p$ and $x_i$ (see Fig. 2(a))..
   (b) Let $x_{i+1}$ be the point on segment $\overline{x_i p_i}$ closest to the origin $o$; update $i = i + 1$.
```

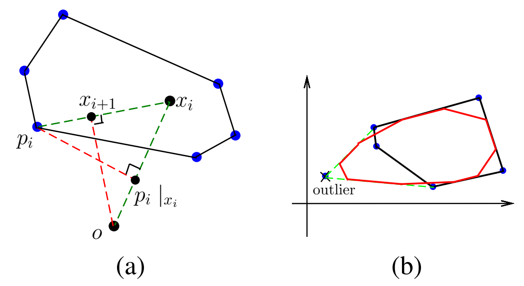


Figure 2: (a) An illustration of step 2 in Gilbert Algorithm; (b) Convex hull of the original point-set (black boundary) and the reduced convex hull after adding an outlier (red boundary).

Theorem 1 ((Gärtner and Jaggi 2009; Clarkson 2010)). *For any constant $\epsilon \in (0, 1)$, Algorithm 1 takes at most $2\lceil 2E/\epsilon \rceil$ steps to yield an ϵ-approximation.*

Theorem 1 indicates that the converging speed of Algorithm 1 depends only on E and ϵ, and is independent of the input size $|P|$ and the dimensionality d.

Now we consider one-class SVM. Let $x \neq o$ be a point in $\mathbb{R}^d$, and $\mathcal{H}_x^{\perp}$ and $\mathcal{H}_x^{\top}$ be the hyperplanes passing through o and x, respectively, and orthogonal to segment $\overline{ox}$. We define the margin $\mathcal{M}_x$ of x as the slap bounded by $\mathcal{H}_x^{\perp}$ and $\mathcal{H}_x^{\top}$. Obviously, the width of $\mathcal{M}_x$ is $||x||$. The following lemmas relate polytope distance to one-class SVM.

Lemma 1 ((Gärtner and Jaggi 2009; Clarkson 2010)). *If $x \in conv(P)$ is the optimal solution of the polytope distance from o to a point-set P, then $\mathcal{M}_x$ is the maximum margin separating the origin and P.*

Lemma 2. *Let $x \in conv(P)$ be an ϵ-approximation of the polytope distance yielded by Algorithm 1 on a point-set P for some small constant $\epsilon \in (0, 1)$. Then $\mathcal{M}_{(1-\epsilon)x}$ separates the origin and P, and its width is at least $(1 - \epsilon)$ times that of the maximum margin.*

Proof. By Definition 1, we know that for any $p \in P$,

$$||(1-\epsilon)x|| = (1-\epsilon)||x|| \le ||p\mid_x||. \quad (1)$$

Since $p\mid_x = p\mid_{(1-\epsilon)x}$, this implies that $\mathcal{M}_{(1-\epsilon)x}$ separates the origin and P.

Furthermore, since x is inside $conv(P)$ (by Algorithm 1), we know that $||x|| \ge \rho$ (*i.e.,* the polytope distance). Thus, the width of $\mathcal{M}_{(1-\epsilon)x}$ is at least $(1-\epsilon)\rho$, which is $(1-\epsilon)$ times that of the maximum margin (by Lemma 1). Thus the lemma is true. □

Lemma 2 suggests that we can find an approximate maximum margin (between the origin and P) using Gilbert Algorithm.

2.2 RGD-tree for One-class SVM with Outliers

Now we consider one-class SVM with outliers.

Definition 2 (One-class SVM with Outliers). *Let P be a set of n points in $\mathbb{R}^d$ and $\gamma \in (0,1)$ be the fraction of outliers in P. The one-class SVM with outliers problem is to find a subset $P' \subset P$ of size $(1-\gamma)n$ such that the margin separating the origin and P' is maximized.*

Despite its fast convergence, Gilbert Algorithm has been limited in applications. One of the main reasons is that it is rather sensitive to outliers. In an extreme case, Gilbert Algorithm may select an outlier (as p_i) in each step, and generates a solution far from optimal. To reduce the influence of outliers, the concept of *soft convex hull* or *reduced convex hull (RCH)* has been introduced (Crisp and Burges 1999; Mavroforakis, Sdralis, and Theodoridis 2007), which could alleviate the problem to certain extent but still cannot completely avoid it. See Fig. 2(b) for an example, where RCH is still significant different from the one without outliers. To have a better solution to this problem, we develop a new technique called *Random Gradient Descent Tree (or RGD-tree)* to explicitly remove outliers.

Main idea. Our main idea comes from the following **key observation** on Gilbert Algorithm: It is not necessary to select the point having the smallest projection distance along the direction of $\overline{ox_i}$ in each step. It is actually sufficient for maintaining the converging speed to select any point in P as long as its projection distance is one of the k smallest for some parameter k to be determined later. Based on this observation, we identify a subset of P (which may contain outliers) in each step, randomly select a sample set from this subset, instead of only a single point p_i, and try to improve the current solution using each point in the sample. Thus, if we view each current solution as a "node", and the multiple improved solutions determined by the sample points as its children, the whole procedure will form a tree, called **RGD-tree** due to its random and gradient descent nature.

To show the effectiveness of RGD-tree, we first prove that with certain probability, the sample set contains some points which are not outliers. This ensures the converging speed of the approach. We are also able to show that with certain probability, there exists one node in RGD-tree which is an approximate solution to the one-class SVM with outliers problem. Details of the RGD-tree algorithm are given in **Algorithm 2** , where the values of parameters δ, μ and h will be determined later.

Algorithm 2 RGD-Tree Algorithm

```
Input: A point-set $P$ in $\mathbb{R}^d$ with a fraction $\gamma \in (0,1)$ of it being outliers, and three parameters $0 < \mu, \delta < 1, h \in \mathbb{Z}^+$.
Output: A RGD-tree with each node associated with a candidate for an approximate solution of the one-class SVM with outliers problem.
1. Randomly select a point from $P$ as $x$. Initialize the RGD-tree as a single node tree (i.e., the root) and associate the node with $x$.
2. Starting from the root, recursively grow each node of RGD-tree as follows:
   (a) Let $\nu$ be the current node associating with a point $x_\nu$.
   (b) If the height of $\nu$ is $h$, it becomes a leaf node and stops growing. Otherwise, do the following.
       i. Find the subset $P_\nu$ of $P$ whose projection distances (see Algorithm 1) along the direction of $\overline{ox_\nu}$ are the $k$ smallest for $k = (1+\delta)\gamma|P|$.
       ii. Take a random sample $S_\nu$ from $P_\nu$ of size $(1+\frac{1}{\delta})\ln\frac{h}{\mu}$. For each point $s \in S_\nu$, create a child of $\nu$ and associate it with a point $x^s_\nu$, where $x^s_\nu$ is determined as follows (see Fig. 3(a) & (b)).
           A. If the angle $\angle osx_\nu \le \frac{\pi}{2}$, $x^s_\nu$ is the orthogonal projection of the origin $o$ on segment $\overline{sx_\nu}$.
           B. Otherwise, $x^s_\nu = s$.
```

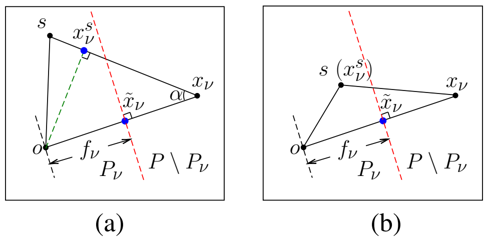


Figure 3: An illustration of Step 2b in RGD-Tree Algorithm.

2.3 Correctness

Let P_{opt} be the **unknown** subset of P which yields the optimal solution in Definition 2, and ρ be the width of the corresponding maximum margin. Also denote by D the diameter of P_{opt} (*i.e.,* $D := \max_{p,q \in P_{opt}} ||p-q||$).

Before showing the correctness of Algorithm 2, we need to understand how to measure the quality of any solution to the problem of one-class SVM with outliers. Clearly, it involves two factors, the width of the margin and the number of outliers (*i.e.,* the number of misclassified points). The following definition and theorem tell us how to consider both factors using *bi-criteria* approximation.

Definition 3. *Let P and γ be the same as in Definition 2, and $0 < \epsilon, \delta < 1$ be two constants. A margin $\mathcal{M}$ is an*

(ϵ, δ)-approximation, if the width of $\mathcal{M}$ is no smaller than $(1-\epsilon)\rho$ and the number of misclassified points is no more than $(1+\delta)\gamma|P|$.

Theorem 2. *If the parameters h, μ, δ in Algorithm 2 are chosen to be $h = 3(\frac{1}{\epsilon}(\frac{D}{\rho}+1))^2 \ln(\frac{D}{\rho}+1)$ and $0 < \mu, \delta < 1$, then with probability $(1-\mu)(1-\gamma)$, there exists at least one node in the resulting RGD-tree which yields an (ϵ, δ)-approximation for the one-class SVM with outliers problem.*

To prove Theorem 2, we need the following lemmas. Lemmas 3, 4, and 5 enable us to analyze the success probability, and Lemma 6 estimates the improvement in each step of Algorithm 2.

Lemma 3 ((Ding and Xu 2014)). *Let Q be a set of elements, and Q' be a subset of Q with size $|Q'| = \beta|Q|$ for some $\beta \in (0,1)$. If one randomly samples $\frac{1}{\beta}\ln\frac{1}{\eta}$ elements from Q, then with probability at least $1-\eta$, the sample contains at least one element in Q' for any $0 < \eta < 1$.*

Lemma 4. *In Step 2b(ii) of Algorithm 2, the sample S_ν contains at least one point from P_{opt} with probability $1-\frac{\mu}{h}$.*

Proof. Since $|P_\nu| = (1+\delta)\gamma|P|$, and $|P \setminus P_{opt}| = \gamma|P|$, we have

$$\begin{aligned}\frac{|P_\nu \cap P_{opt}|}{|P_\nu|} &= 1 - \frac{|P_\nu \setminus P_{opt}|}{|P_\nu|} \\ &\geq 1 - \frac{|P \setminus P_{opt}|}{|P_\nu|} = \frac{\delta}{1+\delta}. \qquad (2)\end{aligned}$$

By Lemma 3 and the fact that $|S_\nu| = (1+\frac{1}{\delta})\ln\frac{h}{\mu}$, we know that S_ν contains at least one point from P_{opt} with probability $1-\frac{\mu}{h}$. □

Lemma 5. *With probability $(1-\gamma)(1-\mu)$, there exists a path $\mathcal{P}$ from the root to a leaf node in the RGD-tree such that the point x associated with the root belongs to P_{opt}, and for each non-leaf node ν on the path, its next node on the path is generated by a sample point $s \in P_{opt}$.*

Proof. In the beginning of Algorithm 2, it is easy to see that the point x associated with the root of the RGD-tree is in P_{opt} with probability $1-\gamma$. By Lemma 4 we know that at each time when Step 2b(ii) is executed, S_ν contains one point from P_{opt} with probability $1-\frac{\mu}{h}$. This means that with probability

$$(1-\gamma)(1-\frac{\mu}{h})^h > (1-\gamma)(1-\mu), \qquad (3)$$

the path $\mathcal{P}$ exists. □

In the following analysis, we assume that there is such a path $\mathcal{P}$ (as in Lemma 5).

Lemma 6. *In Step 2b(ii) of Algorithm 2, for each node $\nu \in \mathcal{P}$, either the point x_ν associated with node ν is an (ϵ, δ)-approximation, or the point x^s_ν associating with its next node in $\mathcal{P}$ satisfies the following inequality*

$$||x^s_\nu||^2 \leq (1 - (\frac{\epsilon\rho}{D+\rho})^2)||x_\nu||^2. \qquad (4)$$

Proof. Let f_ν be the maximum projection distance among all points in P_ν along the direction of $\overline{ox_\nu}$, *i.e.,*

$$f_\nu = \max_{p \in P_\nu}\{p\ |_{x_\nu}\}. \qquad (5)$$

Let $\tilde{x}_\nu = \frac{f_\nu}{||x_\nu||}x_\nu$ (see Fig. 3). If $f_\nu \geq (1-\epsilon)\rho$, we know that the margin determined by $\tilde{x}_\nu$, *i.e.,* $\mathcal{M}_{\tilde{x}_\nu}$, has width at least $(1-\epsilon)\rho$, and separates $|P \setminus P_\nu|$ points from the origin o. By Definition 3, we know that this margin is an (ϵ, δ)-approximation.

Thus we only need to consider the other case, and can assume that

$$f_\nu < (1-\epsilon)\rho. \qquad (6)$$

By Lemma 1, we know that ρ is the polytope distance between the origin o and P_{opt}. This implies that $||x_\nu||$ is no smaller than ρ. Combining this with (6), we have

$$||x_\nu - \tilde{x}_\nu|| \geq \epsilon\rho. \qquad (7)$$

Furthermore, since we only consider the nodes in the path $\mathcal{P}$, we know that the sample point $s \in P_{opt}$. This implies that

$$||s - x_\nu|| \leq D. \qquad (8)$$

If $\angle osx_\nu \leq \frac{\pi}{2}$, then

$$||x^s_\nu|| = ||x_\nu||\sin\alpha, \qquad (9)$$

where α is $\angle ox_\nu s$ (see Fig. 3(a)). Combining (7) and (8), we have the following

$$\cos\alpha \geq \frac{\epsilon\rho}{D} \Longrightarrow \sin\alpha \leq \sqrt{1-(\frac{\epsilon\rho}{D})^2}. \qquad (10)$$

From (9) and (10), we immediately have (4) in the case of $\angle osx_\nu \leq \frac{\pi}{2}$.

If $\angle osx_\nu > \frac{\pi}{2}$ (see Fig. 3(b)), we have

$$||x^s_\nu||^2 + ||x^s_\nu - x_\nu||^2 \leq ||x_\nu||^2. \qquad (11)$$

By (7), it is easy to see that $||x^s_\nu - x_\nu|| \geq ||x_\nu - \tilde{x}_\nu|| \geq \epsilon\rho$. Thus, we have

$$\begin{aligned}||x^s_\nu||^2 &\leq ||x_\nu||^2 - (\epsilon\rho)^2 \\ &= (1 - \frac{(\epsilon\rho)^2}{||x_\nu||^2})||x_\nu||^2 \\ &\leq (1 - \frac{(\epsilon\rho)^2}{(D+\rho)^2})||x_\nu||^2, \qquad (12)\end{aligned}$$

where the last inequality follows from the fact that $||x_\nu|| \leq D + \rho$ (by triangle inequality). Thus, (4) also holds in the case of $\angle osx_\nu > \frac{\pi}{2}$.

This completes the proof of the lemma. □

Below, we prove Theorem 2.

Proof. (**of Theorem 2**) We prove the theorem following the flow of Algorithm 2. We kown that path $\mathcal{P}$ exists with probability $(1-\gamma)(1-\mu)$ (by Lemma 5), and claim that there exists one node ν on the path whose x_ν is an (ϵ, δ)-approximation. We prove this claim by contradiction. Suppose that no such a node ν exists. Let $x_\mathcal{L}$ be the point associated with the leaf of the path. Then by Lemma 6, we have

$$||x_\mathcal{L}||^2 \leq ((1 - (\frac{\epsilon\rho}{D+\rho})^2)^h||x||^2. \qquad (13)$$

By Lemma 1 we know that ρ is the polytope distance between the origin and P_{opt}. Thus, $||x_{\mathcal{L}}||$ is no smaller than ρ. Also by triangle inequality, we know that $||x|| \leq D + \rho$. Hence, (13) implies that

$$\rho^2 \leq ((1 - (\frac{\epsilon\rho}{D+\rho})^2)^h (D+\rho)^2. \tag{14}$$

Let λ denote $\frac{D+\rho}{\rho}$. Then (14) implies that

$$\begin{aligned} h &\leq \log_{1/(1-(\epsilon/\lambda)^2)} \lambda^2 \\ &= \frac{2\ln\lambda}{\ln(1+\frac{(\epsilon/\lambda)^2}{1-(\epsilon/\lambda)^2})} \\ &< \frac{2\ln\lambda}{\ln(1+(\epsilon/\lambda)^2)} \\ &< 3(\frac{\lambda}{\epsilon})^2 \ln\lambda = 3(\frac{1}{\epsilon}(\frac{D}{\rho}+1))^2 \ln(\frac{D}{\rho}+1), \end{aligned} \tag{15}$$

where the final inequality follows from the fact that $(\epsilon/\lambda)^2 < 1$. Since $h = 3(\frac{1}{\epsilon}(\frac{D}{\rho}+1))^2 \ln(\frac{D}{\rho}+1)$ in the theorem, we have a contradiction. This completes the proof. □

Remark 1. *The above theorem ensures the correctness. Several other issues, such as the time and space complexities, will be addressed in Section 2.5.*

2.4 Two-class SVM with Outliers

Now, we consider two-class SVM with outliers.

Definition 4 (Two-class SVM with Outliers). *Let P_1 and P_2 be two sets of points in $\mathbb{R}^d$, and $\gamma \in (0,1)$ (or $\gamma_1, \gamma_2 \in (0,1)$) be the fraction of outliers in $P_1 \cup P_2$ (or in P_1 and P_2, respectively). The problem of two-class SVM with outliers is to find two subsets $P_1' \subset P_1$ and $P_2' \subset P_2$, such that $|P_1' \cup P_2'| \geq (1-\gamma)|P_1 \cup P_2|$ (or $|P_1'| \geq (1-\gamma_1)|P_1|$ and $|P_2'| \geq (1-\gamma_2)|P_2|$), and the margin separating P_1' and P_2' is maximized.*

Corresponding to the two slightly different definitions, we have two ways to define the bi-criteria approximation.

Definition 5. *Let $P_1, P_2, \gamma, \gamma_1$ and γ_2 be the same as in Definition 4, and ρ be the width of the maximum margin in the problem of two-class SVM with outliers. Then a margin $\mathcal{M}$ is an (ϵ, δ)-approximation, if the width of $\mathcal{M}$ is no smaller than $(1-\epsilon)\rho$ and the number of misclassified points is no more than $(1+\delta)\gamma|P_1 \cup P_2|$ (or $(1+\delta)\gamma_1|P_1|$ and $(1+\delta)\gamma_2|P_2|$), where $0 < \epsilon, \delta < 1$ are two constants.*

In the above definitions, the case of one-outlier-parameter (γ) can actually be reduced to the case of two-outlier-parameters (γ_1 and γ_2). The main idea is to use ***discretization*** and "guess" the fractions of outliers in P_1 and P_2, respectively. Due to space limit, we leave the details in the full version of our paper. Thereafter, we always assume that two outlier parameters γ_1 and γ_2 are given. The following definition and theorem reveal the connection between polytope distance and two-class SVM.

Definition 6 ((Gärtner and Jaggi 2009)). *Let P_1 and P_2 be two point-sets in $\mathbb{R}^d$. The Minkowski difference $MD(P_1, P_2)$ of polytopes $conv(P_1)$ and $conv(P_2)$ is the set (which is also a polytope (Ziegler 1995)) of all difference vectors,* i.e., $MD(P_1, P_2) = \{u - v \mid u \in conv(P_1), v \in conv(P_2)\}$.

Theorem 3. *[(Gärtner and Jaggi 2009)] Finding the shortest distance between two polytopes $conv(P_1)$ and $conv(P_2)$ is equivalent to finding the polytope distance from the origin to $MD(P_1, P_2)$.*

Theorem 3 tells us that to find the maximum margin separating two point-sets P_1 and P_2, we only need to find the maximum margin separating the origin and $MD(P_1, P_2)$. This indicates the possibility of using RGD-tree to solve the problem of two-class SVM with outliers. A careful analysis shows that we actually do not need to explicitly compute $MD(P_1, P_2)$ (which would take quadratic time); a slight modification to Algorithm 2 will be sufficient.

Revisited Algorithm 2. The revisited algorithm will have the following main differences from the original one. (1) The input has two point-sets P_1 and P_2, and two outlier parameters γ_1 and γ_2. (2) In Step 2b(i), take two subsets $P_\nu^1 \subset P_1$ and $P_\nu^2 \subset P_2$, which respectively consist of points having the $(1+\delta)\gamma_1|P_1|$ smallest projection distances among all points in P_1 and the $(1+\delta)\gamma_2|P_2|$ largest projection distances among all points in P_2. (3) In Step 2b(ii), take a set of vectors S_v of the same size as in the original algorithm, where each vector $s \in S_\nu$ is the difference of two random sampled points from P_ν^1 and P_ν^2, respectively.

Using a similar argument given in the proof of Theorem 2, we have the following result.

Theorem 4. *If the parameters in the revisited Algorithm 2 are chosen as follows, $h = 3(\frac{1}{\epsilon}(\frac{D}{\rho}+1))^2 \ln(\frac{D}{\rho}+1)$ and $0 < \mu, \delta < 1$, then with probability $(1-\mu)(1-\gamma)$, there exists at least one node in the resulting RGD-tree which yields an (ϵ, δ)-approximation for the two-class SVM with outliers problem.*

2.5 Time and Space Complexity

In this section, we analyze the time and space complexities of the RGD-tree algorithm.

Time complexity. Let n be the size of P (or $P_1 \cup P_2$ in the two-class case), and d be the dimensionality of the space. For each node of the RGD-tree, we create $|S_\nu| = (1+\frac{1}{\delta}) \ln\frac{h}{\mu}$ children. Since the height h of the tree is $3(\frac{1}{\epsilon}\frac{D+\rho}{\rho})^2 \ln\frac{D+\rho}{\rho}$, the total size of the tree is thus $O(|S_\nu|^h) = O((\ln\frac{D+\rho}{\epsilon\rho})^h)$ (after omitting the constant factor depending on δ and μ). This means that the size of the tree depends only on the ratio of $\frac{D}{\rho}$ and ϵ, and is independent of n and d. Now, we consider the computation on each node ν. Firstly, we need to compute the projections of P (or P_1 and P_2 for the two-class case) on $\overline{ox_\nu}$, which takes $O(nd)$ time in total. Secondly, we need to find subset P_ν (or P_ν^1 and P_ν^2 in the two-class case). This can be done in $O(nd)$ time by using the linear-time selection algorithm[1]. Thirdly, creating all the $|S_\nu|$ children costs $O(|S_\nu|d)$ time in Step 2b(ii). Thus the total time for each node ν is

[1]Given a set of n numbers Y and an integer $k \leq n$, we can first find the k-th smallest number y by the selection algorithm in linear time (Blum et al. 1973), and use y as the pivot to select the k smallest numbers in linear time.

$O((n + |S_\nu|)d)$. Consequently, the time complexity of the algorithm is $O((\ln \frac{D+\rho}{\epsilon\rho})^{h+1} nd)$, which is linear in n and d.

We note that the above time complexity is only for the worst case analysis. In practice, many nodes are actually not needed for generating the final solution, and thus do not need to be kept in the tree. Thus in implementation, we can set an upper bound L on the number of nodes in each level of the RGD-tree. In this way, we can bring to the total time down to $O(Lhnd)$, which linearly, rather than exponentially, depends on h. To achieve this, we prune a number nodes which produce bad results (*i.e.,* too narrow margin) in each level while generating the RGD-tree. Note that this will not increase the time complexity, since we just need to check the already computed projections of P (or P_1 and P_2) to obtain the width of the margin.

Space complexity. The space complexity is obviously bounded by the total size of the input and the RGD-tree, which is $O(((\ln \frac{D+\rho}{\epsilon\rho})^h + n)d)$ in the worst case and linear in n and d. With a careful analysis, we can actually significantly reduce it as we do not need to store the whole tree. Instead, we only need to store two levels of the tree, the current level and its next level. Additionally, we need to save the best solutions from the root to the current level, which takes only a constant number of extra space. Thus, if we perform the above pruning strategy, the space complexity becomes $O((L + n)d)$, which is independent of the height h.

3 Boosting

To further improve the performance of our RGD-tree algorithm, we propose the following boosting procedure. The key idea is to build a sequence of RGD-trees, instead of a single one.

1. Initially, build the RGD-tree for the input.
2. Iteratively perform the following steps until the result becomes stable:
   (a) Select the node corresponding to the best solution in the newly built RGD-tree.
   (b) Use the selected node as the root, build a new RGD-tree.

In Section 4, we will show that this boosting procedure can not only improve the result, but also reduce the running time significantly.

4 Experiments

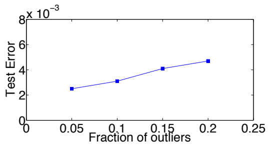
We test our RGD-SVM on both synthetic and benchmark data sets. All results are obtained by using a MacBook Pro (2.4 GHz CPU and 4 GB memory).

Synthetic data. We design three experiments on synthetic data for the two-class SVM with outlier problem. In each experiment, the sizes of the two classes vary randomly in the ranges of $[10^3, 3 \times 10^3]$ for training and $[10^4, 3 \times 10^4]$ for testing, and the dimensionality is set to be 100. Each class is generated following a *multivariate Gaussian distribution*; in order to make the distribution as arbitrary as possible, the covariance matrix Σ is also generated randomly and the ratio between the maximum and the minimum diagonal elements is set to be 10. Each of the three experiments has repeatedly run 10 times, and the average results are reported. Experiment **(1)** tests how our algorithm performs under different level of outliers. We vary the fraction of outliers between 5% to 20%. Experimental results in Fig. 4(a) show that the error is a slowly growing function and is always around 0.5%, which is quite stable. Experiment **(2)** tests the performance of our algorithm under different values of ratio D/ρ. We fix the fraction of outliers to be 10%, and vary the ratio D/ρ from 5 to 30. The height of RGD-tree is restricted to be no more than 10 so that the running time is always within 5 minutes. Results in Fig. 4(b) suggest that error grows (in the range of $(0.5\%, 2.5\%)$) roughly linearly with the ratio. Experiment **(3)** tests the improvement of boosting procedure (in Section 3). In this experiment, we fix the fraction of outliers to be 10% and D/ρ to be 30, and perform the boosting procedure $1 - 4$ rounds; in each round, the height of the RGD-tree is no more than 5 (instead of 10 as in experiment (2)). Results in Fig. 4(c) show that the solution improves rapidly in the first couple of rounds and becomes stable after 3 rounds. The error becomes comparable to that in Fig. 4(b) after 3 rounds, but the running time is reduced by at least half. This is because the RGD-tree becomes much smaller and thus the total running time improves significantly.

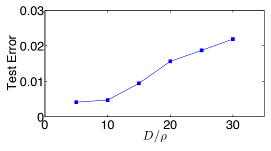
Benchmark data. We select 10 benchmark data sets from (Chang and Lin 2011), and compare our algorithm with three popular methods, soft margin SVM (Cortes and Vapnik 1995; Platt 1999), robust SVM based on CCCP (Yuille and Rangarajan 2001; Krause and Singer 2004), and homotopy algorithm (Suzumura et al. 2014). For each data set, we use the best result from the three methods as the baseline. We set the data the same way as that in (Suzumura et al. 2014). That is, randomly divide each data set into training (40%), validation (30%), and testing (30%) sets, and flip 15% of the labels in the training and validation sets as outliers. From Fig. 4(d), we can see that our RGD-SVM outperforms the baseline on 8 data sets, and particularly has more than 20% improvement on #1 and #2, and more than 5% improvement on #3, #5, #8, and #9.

Table 1: Results on 10 benchmark data sets

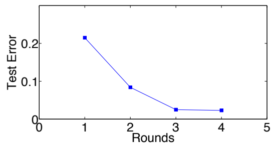
#	Name	n	d	Test error
1	BreastCancer	596	30	2.2%
2	AustralianCredit	690	14	10.2%
3	GermanNumber	1000	24	24.0%
4	SVMGuide1	3089	4	4.0%
5	Spambase	4601	57	7.6%
6	Musk	6598	166	7.0%
7	Gisette	6000	5000	18.2%
8	w5a	9888	300	1.9%
9	a6a	11220	122	15.2%
10	a7a	16100	122	16.4%



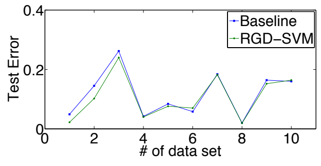
(a)



(b)



(c)



(d)

Figure 4: (a) and (b) Classification errors under different values of γ and ratio D/ρ; (c) Improvement on classification error by using boosting; (d) Comparison between our algorithm and the baseline on benchmark data sets.

5 Conclusions

In this paper, we present a new combinatorial approach for dealing with outliers in SVM. Most existing techniques for this problem are numerical approaches, and cannot guarantee the quality of solution despite a high time complexity. On the contrary, our combinatorial approach yields provably good solution, and has time and space complexities linearly depending on the input size and the dimensionality. Furthermore, experimental results suggest that our approach has better performance (in terms of accuracy) than several popular existing methods.

References

Aizerman, M. A.; Braverman, E. A.; and Rozonoer, L. 1964. Theoretical foundations of the potential function method in pattern recognition learning. In *Automation and Remote Control,*, number 25 in Automation and Remote Control,, 821–837.

Blum, M.; Floyd, R. W.; Pratt, V. R.; Rivest, R. L.; and Tarjan, R. E. 1973. Time bounds for selection. *J. Comput. Syst. Sci.* 7(4):448–461.

Chang, C.-C., and Lin, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM TIST* 2(3):27.

Clarkson, K. L.; Hazan, E.; and Woodruff, D. P. 2012. Sublinear optimization for machine learning. *J. ACM* 59(5):23.

Clarkson, K. L. 2010. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms* 6(4).

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Machine Learning* 20:273.

Crisp, D. J., and Burges, C. J. C. 1999. A geometric interpretation of v-SVM classifiers. In Solla, S. A.; Leen, T. K.; and Müller, K.-R., eds., *NIPS*, 244–250. The MIT Press.

Ding, H., and Xu, J. 2014. Sub-linear time hybrid approximations for least trimmed squares estimator and related problems. In Cheng, S., and Devillers, O., eds., *SOCG'14, Kyoto, Japan, June 08 - 11, 2014*, 110. ACM.

Frank, M., and Wolfe, P. 1956. An algorithm for quadratic programming. *Naval Research Logistics Quarterly* 3:149–154.

Gärtner, B., and Jaggi, M. 2009. Coresets for polytope distance. In Hershberger, J., and Fogel, E., eds., *Proceedings of the 25th ACM Symposium on Computational Geometry, Aarhus, Denmark, June 8-10, 2009*, 33–42. ACM.

Gilbert, E. G. 1966. An iterative procedure for computing the minimum of a quadratic form on a convex set. *SIAM Journal on Control* 4(1):61–80.

Har-Peled, S.; Roth, D.; and Zimak, D. 2007. Maximum margin coresets for active and noise tolerant learning. In Veloso, M. M., ed., *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, 836–841.

Keerthi, S. S.; Shevade, S. K.; Bhattacharyya, C.; and Murthy, K. R. K. 2000. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE Trans. Neural Netw. Learning Syst* 11(1):124–136.

Krause, N., and Singer, Y. 2004. Leveraging the margin more carefully. In Brodley, C. E., ed., *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM.

Mavroforakis, M. E.; Sdralis, M.; and Theodoridis, S. 2007. A geometric nearest point algorithm for the efficient solution of the svm classification task. *IEEE Transactions on Neural Networks* 18(5):1545–1549.

Mitchell, B. F.; Dem'yanov, V. F.; and Malozemov, V. N. 1974. Finding the point of a polyhedron closest to the origin. *SIAM Journal on Control* 12(1):19–26.

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

Scholkopf, B.; Smola, A. J.; Muller, K. R.; and Bartlett, P. L. 2000. New support vector algorithms. *Neural Computation* 12:1207–1245.

Suzumura, S.; Ogawa, K.; Sugiyama, M.; and Takeuchi, I. 2014. Outlier path: A homotopy algorithm for robust svm. In Jebara, T., and Xing, E. P., eds., *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 1098–1106. JMLR Workshop and Conference Proceedings.

Tsang, I. W.; Kwok, J. T.; and Cheung, P.-M. 2005. Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research* 6:363–392.

Xu, L.; Crammer, K.; and Schuurmans, D. 2006. Robust support vector machine training via convex outlier ablation. In *AAAI*, 536–542. AAAI Press.

Yuille, A. L., and Rangarajan, A. 2001. The concave-convex procedure (CCCP). In Dietterich, T. G.; Becker, S.; and Ghahramani, Z., eds., *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, 1033–1040. MIT Press.

Ziegler, G. M. 1995. *Lectures on polytopes*, volume 152 of *Graduate Texts in Mathematics*. Berlin-Heidelberg-New York-London-Paris-Tokyo-Hong Kong-Barcelona-Budapest: Springer-Verlag.