Efficient Data Point Pruning for One-Class SVM

Yasuhiro Fujiwara,[†‡♮] **Sekitoshi Kanai,**[†] **Junya Arai,**[†] **Yasutoshi Ida,**[†] **Naonori Ueda**[‡♯]
[†]NTT Software Innovation Center, 3-9-11 Midori-cho Musashino-shi, Tokyo, 180-8585, Japan
[‡]NTT Communication Science Laboratories, 2-4 Seika-Cho Soraku-gun, Kyoto, Japan
[♮]Osaka University, 1-5 Yamadaoka, Suita-shi, Osaka, Japan
[♯]RIKEN Center for AIP, 1-4-1 Nihonbashi, Chuo-ku, Tokyo, 103-0027, Japan
{fujiwara.yasuhiro, kanai.sekitoshi, arai.junya, ida.yasutoshi, ueda.naonori}@lab.ntt.co.jp

Abstract

One-class SVM is a popular method for one-class classification but it needs high computation cost. This paper proposes *Quix* as an efficient training algorithm for one-class SVM. It prunes unnecessary data points before applying the SVM solver by computing upper and lower bounds of a parameter that determines the hyper-plane. Since we can efficiently check optimality of the hyper-plane by using the bounds, it guarantees the identical classification results to the original approach. Experiments show that it is up to 6800 times faster than existing approaches without degrading optimality.

Introduction

Due to the rapid development of Internet and database technologies, we can improve the effectiveness of applications by using knowledge from big data (Fujiwara et al. 2017a; Mishima and Fujiwara 2015; Nakatsuji and Fujiwara 2014). Machine learning approaches play important role in extracting knowledge from big data (Fujiwara et al. 2017b; Tanaka et al. 2016; Nakatsuji et al. 2014). One-class SVM is a popular machine learning approach to achieving one-class classification (Schölkopf et al. 2001). It computes a hyper-plane that separates data points of the target class. Data points are called support vectors if they are representative of the hyper-plane, and they are called outliers if they are not labeled as the target class. One-class SVM offers the advantage of the desirable properties of traditional SVMs. For instance, it has a unique solution in computing the hyper-plane since its optimization problem is convex. However, its computation cost is excessive especially for large-scale data since it needs to solve a quadratic programming problem with n constraints where n is the number of data points.

The shrinking approach is a popular method to reduce the computation cost of SVM (Joachims 1999). Although it can compute the optimal hyper-plane, it needs high computation cost in checking the optimality. The Nyström method approximates a kernel matrix used in the solver by selecting data points as landmarks (Drineas and Mahoney 2005). Specifically, it computes a small matrix of landmarks where all data points are projected by using kernel similarities between landmarks and data points. Random Fourier features can reduce the computational cost of SVM by constructing a random mapping into a low-dimensional feature space for kernel functions (Rahimi and Recht 2007). However, these approaches need the high computation cost since they apply the solver to all the data points. Note that, since the original approaches of the Nyström method and random Fourier features use simple algorithms (Drineas and Mahoney 2005; Rahimi and Recht 2007), they are more effective in reducing computation cost comparing to the recent approaches of the Nyström method and random Fourier features (Li et al. 2016; Musco and Musco 2017; Wu et al. 2016). Noumir et al. proposed an online learning approach based on the coherence criterion (Noumir, Honeine, and Richard 2012). The approach computes the hyper-plane by updating the Gram matrix and its inverse matrix by exploiting the Woodbury matrix identity. In terms of efficiency, this approach outperforms the adaptive approach (Gómez-Verdejo et al. 2011). Gao proposed the active-set method based online approach for one-class SVM to improve the efficiency (Gao 2015). It applies the solver to data points only if their distance to the center of the hyper-plane is not less than the radius of the hyper-plane. However, they can erroneously remove data points that are actually support vectors for the optimal hyper-plane; they sacrifice the optimality of the hyper-plane to increase the efficiency. Note that there exists a gap between the performance of approximate and the optimal approaches in terms of accuracy (Li et al. 2016; Fujiwara et al. 2015; 2013; Yang et al. 2012).

We propose a novel efficient approach that guarantees the optimal hyper-plane. Given parameter ν ($0 < \nu \leq 1$) that determines the fraction of outliers and support vectors, this paper formally addresses the problem of reducing the training time of one-class SVM for parameter ν and given data points. We prune unnecessary data points in computing support vectors before applying the solver by computing upper and lower bounds of a parameter that determines the hyper-plane. Since we efficiently check the optimality of an obtained hyper-plane after applying the solver, we can obtain the same training result as the original algorithm. The proposed approach can use any solvers such as SMO (Platt 1998) and incremental approaches (Cauwenberghs and Poggio 2000; Laskov et al. 2006) since our approach is solver-independent. Note that the proposed approach itself does not have hyper-parameters to be tuned by users.

Preliminaries

In one-class SVM, vector $\mathbf{x}_i = (x_i[1], x_i[2], \ldots, x_i[m])$ corresponds to the i-th data point in m dimensional space and can be represented as the i-th row vector of data matrix $\mathbf{X}$ whose column vectors are standardized (Çeker and Upadhyaya 2016; Fisher, Camp, and Krzhizhanovskaya 2016). Let $\mathbf{w}$, ξ_i, and ρ be an m-dimensional vector, a slack variable, and an offset, respectively, it solves the following quadratic programming problem:

$$\min_{\mathbf{w} \in F, \xi \in \mathbb{R}^n, \rho \in \mathbb{R}} \tfrac{1}{2}\|\mathbf{w}\|^2 + \tfrac{1}{n\nu}\textstyle\sum_{i=1}^{n} \xi_i - \rho \quad (1)$$

$$\text{Subject to } (\mathbf{w} \cdot \Phi(\mathbf{x}_i)) \geq \rho - \xi_i, \xi_i \geq 0 \quad (2)$$

In Equation (2), Φ is a kernel map $\mathcal{X} \to F$ which transforms the training data points into a high-dimensional feature space. We have the following dual problem by introducing the Lagrange function:

$$\min_{\boldsymbol{\alpha}} \tfrac{1}{2}\textstyle\sum_{i,j=1}^{n} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (3)$$

$$\text{subject to } 0 \leq \alpha_i \leq \tfrac{1}{n\nu}, \textstyle\sum_{i=1}^{n} \alpha_i = 1 \quad (4)$$

Here, $\boldsymbol{\alpha}$ is a vector whose i-th element α_i is a Lagrange multiplier of data point $\mathbf{x}_i$, and $K(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel function such as the Gaussian kernel given as $e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2}$. In Equation (3), the data points that have $\alpha_i > 0$ and $\alpha_i = \frac{1}{n\nu}$ are called support vectors and outliers, respectively, and ν is a lower bound of the fraction of support vectors and an upper bound on the fraction of outliers. This indicates that training accuracy should be at least $1 - \nu$. Let $z = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) - \rho$ be a parameter for data point $\mathbf{x}$, the decision function is given as follows:

$$f(\mathbf{x}) = \mathrm{sgn}(z) = \mathrm{sgn}(\textstyle\sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) - \rho) \quad (5)$$

Since the dual problem is convex, we can compute the optimal hyper-plane by applying the solver. We can compute offset ρ as $\rho = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}_j)$ where $\mathbf{x}_j$ is the data point for which $0 < \alpha_j < \frac{1}{n\nu}$.

However, one-class SVM has a drawback; its computation cost is high in obtaining the hyper-plane. This is because it needs to solve a quadratic programming problem with n constraints. Even though Sequential Minimal Optimization (SMO) is known to be an efficient SVM solver to obtain the optimal hyper-plane, its computation cost is quadratic to the number of data points.

Proposed method

Main ideas

The original algorithm applies the solver to all data points. Our approach exploits the property of one-class SVM that the hyper-plane is determined only by support vectors (Schölkopf et al. 2001); we select data points to apply the solver by upper and lower bounds of parameter z.

This approach has several advantages. First, the hyper-plane obtained by our approach is guaranteed to be optimal unlike the previous approaches for SVM (Drineas and Mahoney 2005; Gao 2015; Gómez-Verdejo et al. 2011; Noumir, Honeine, and Richard 2012; Rahimi and Recht 2007). This is because our approach checks whether the obtained hyper-plane is theoretically identical to the optimal hyper-plane by exploiting the upper and lower bounds. Second, our approach can efficiently check the optimality of the hyper-plane since we can efficiently compute the bounds used in the optimality check. Although the shrinking approach selects data points similar to our approach, it needs high computation cost to check the optimality of the hyper-plane by computing kernel functions (Joachims 1999). If s is the number of support vectors, the shrinking approach requires $O(snm)$ time while our approach needs at most $O(n)$ time in checking the optimality from the bounds. Finally, the proposed approach does not require any user-defined inner-parameter other than parameter ν. Note that parameter ν is also required by the original algorithm of one-class SVM. By contrast, the previous approaches (Drineas and Mahoney 2005; Noumir, Honeine, and Richard 2012; Rahimi and Recht 2007) need to set parameters that determine their performance in computing the hyper-plane. This indicates that our approach is user-friendly.

Upper and lower Bounds

Our approach iteratively computes Lagrange multipliers from a set of data points by applying the solver until we obtain the optimal hyper-plane. In computing the set of data points, we exploit the upper and lower bounds of parameter z_i. This section introduces the approach to efficiently computing the bounds. Since the RBF kernel is the most popular technique, we first assume the RBF kernel. However, we can handle other kernels as we describe in this section.

In order to efficiently compute the upper and lower bounds, we use sparse vector $\hat{\mathbf{x}}_i = (\hat{x}_i[1], \hat{x}_i[2], \ldots, \hat{x}_i[m])$ for data point $\mathbf{x}_i$. Vector $\hat{\mathbf{x}}_i$ has sparse structure since we compute each element of vector $\hat{\mathbf{x}}_i$ as $\hat{x}_i[j] = x_i[j] - \sigma_j$ where σ_j is the most frequent value of the j-th column of data matrix $\mathbf{X}$. Let $\mathbb{O}$ and $\mathbb{H}$ be a set of outliers and data points such that $\mathbb{O} = \{\mathbf{x}_i | \alpha_i = \frac{1}{n\nu}\}$ and $\mathbb{H} = \{\mathbf{x}_i | 0 < \alpha_i < \frac{1}{n\nu}\}$, respectively, we compute the bounds as follows:

Definition 1 *Let $\overline{z}_i$ and $\underline{z}_i$ be the upper and lower bounds of parameter z_i, respectively, for the given Lagrange multipliers, bound $\overline{z}_i$ and $\underline{z}_i$ are computed as follows:*

$$\overline{z}_i = \tfrac{1}{n\nu} o_i - \rho + \textstyle\sum_{\mathbf{x}_j \in \mathbb{X} \backslash \mathbb{O}} \alpha_j e^{-\gamma(\|\hat{\mathbf{x}}_i\| - \|\hat{\mathbf{x}}_j\|)^2} \quad (6)$$

$$\underline{z}_i = \tfrac{1}{n\nu} o_i - \rho + \textstyle\sum_{\mathbf{x}_j \in \mathbb{X} \backslash \mathbb{O}} \alpha_j e^{-\gamma(\|\hat{\mathbf{x}}_i\| + \|\hat{\mathbf{x}}_j\|)^2} \quad (7)$$

where $o_i = \sum_{\mathbf{x}_j \in \mathbb{O}} e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2}$.

In Definition 1, we can efficiently compute score o_i since set $\mathbb{O}$ is incrementally updated each time we compute a Lagrange multiplier, and it is not necessary to compute the kernel functions if $\alpha_i = 0$ holds in Equation (6) and (7). In addition, we can compute norm $\|\hat{\mathbf{x}}_i\|$ before applying the solver where vector $\hat{\mathbf{x}}_i$ has sparse structure by subtracting the most frequent values in data matrix $\mathbf{X}$. As a result, we can efficiently compute the bounds at $O(|\mathbb{H}|)$ time from Equation (6) and (7) where $|\mathbb{H}|$ is the number of data points included in set $\mathbb{H}$ if we have score o_i and norm $\|\hat{\mathbf{x}}_i\|$ for each data point. On the other hand, it needs $O(sm)$ time to exactly compute parameter z_i from Equation (5). This indicates that bound $\overline{z}_i$ and $\underline{z}_i$ can be more efficiently computed than parameter z_i since we have $|\mathbb{H}| \ll s$ (Schölkopf et al. 2001). Since the shrinking approach computes parameter z_i for each data points, it needs high computation cost. However, our approach can efficiently obtain the data points applied to the solver.

Definition 1 indicates that we need to obtain Lagrange multiplier α_i for data point $\mathbf{x}_i$ in computing the bounds. We describe later the approach to compute the initial Lagrange multiplier setting. . For Definition 1, we have the following lemma from the Cauchy-Schwarz inequality (Steele 2004):

Lemma 1 *If we compute parameter z_i for the obtained hyper-plane, we have $\overline{z}_i \geq z_i$ and $\underline{z}_i \leq z_i$.*

Proof Since we have $\alpha_i = \frac{1}{n\nu}$ for set $\mathbb{O}$ and $0 < \alpha_i < \frac{1}{n\nu}$ for set $\mathbb{H}$,

$$\begin{aligned} z_i &= \textstyle\sum_{j=1}^{n} \alpha_i K(\mathbf{x}_i,\mathbf{x}_j) - \rho = \sum_{\mathbf{x}_j \in \mathbb{X}} \alpha_i K(\mathbf{x}_i,\mathbf{x}_j) - \rho \\ &= \textstyle\sum_{\mathbf{x}_j \in \mathbb{O}} \alpha_i K(\mathbf{x}_i,\mathbf{x}_j) - \rho + \sum_{\mathbf{x}_j \in \mathbb{X}\setminus\mathbb{O}} \alpha_i K(\mathbf{x}_i,\mathbf{x}_j) \end{aligned} \tag{8}$$

Since we use the RBF kernel, we have

$$z_i = \tfrac{1}{n\nu} o_i - \rho + \textstyle\sum_{\mathbf{x}_j \in \mathbb{H}} \alpha_j e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2} \tag{9}$$

Since we have $\hat{x}_i[j] = x_i[j] - \sigma_j$ for vector $\hat{\mathbf{x}}_i$,

$$\begin{aligned} \|\mathbf{x}_i - \mathbf{x}_j\|^2 &= \textstyle\sum_{k=1}^{m} (x_i[k] - x_j[k])^2 \\ &= \textstyle\sum_{k=1}^{m} \{(x_i[k] - \sigma_j) - (x_j[k] - \sigma_j)\}^2 = \|\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j\|^2 \end{aligned} \tag{10}$$

If $\langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle$ is the inner product of $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$, from the Cauchy-Schwarz inequality, we have

$$\begin{aligned} \|\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j\|^2 &= \textstyle\sum_{k=1}^{m} (\hat{x}_i[k] - \hat{x}_j[k])^2 \\ &= \textstyle\sum_{k=1}^{m} \{(\hat{x}_i[k])^2 + (\hat{x}_j[k])^2\} - 2\sum_{k=1}^{m} \hat{x}_i[k]\hat{x}_j[k] \\ &= \|\hat{\mathbf{x}}_i\|^2 + \|\hat{\mathbf{x}}_j\|^2 - 2\langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle \geq (\|\hat{\mathbf{x}}_i\| - \|\hat{\mathbf{x}}_j\|)^2 \end{aligned} \tag{11}$$

Therefore, $z_i \leq \frac{1}{n\nu} o_i - \rho + \sum_{\mathbf{x}_j \in \mathbb{H}} \alpha_j e^{-\gamma(\|\hat{\mathbf{x}}_i\| - \|\hat{\mathbf{x}}_j\|)^2} = \overline{z}_i$ holds from Equation (9). Similarly, since $\|\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j\|^2 \leq (\|\hat{\mathbf{x}}_i\| + \|\hat{\mathbf{x}}_j\|)^2$ holds from the Cauchy-Schwarz inequality, we have $z_i \geq \frac{1}{n\nu} o_i - \rho + \sum_{\mathbf{x}_j \in \mathbb{H}} \alpha_j e^{-\gamma(\|\hat{\mathbf{x}}_i\| + \|\hat{\mathbf{x}}_j\|)^2} = \underline{z}_i$. □

We can compute the bounds for other kernel functions such as linear ($K(\mathbf{x}_i,\mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$), polynomial ($K(\mathbf{x}_i,\mathbf{x}_j) = (\gamma\langle \mathbf{x}_i, \mathbf{x}_j \rangle + r)^p$), and sigmoid ($K(\mathbf{x}_i,\mathbf{x}_j) = \tanh(\gamma\langle \mathbf{x}_i, \mathbf{x}_j \rangle + r)$) based on the following property:

Lemma 2 *For the inner product of vector $\mathbf{x}_i$ and $\mathbf{x}_j$, we have $\langle \mathbf{x}_i, \mathbf{x}_j \rangle \leq \frac{1}{2}\{\|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2 - (\|\hat{\mathbf{x}}_i\| - \|\hat{\mathbf{x}}_j\|)^2\}$ and $\langle \mathbf{x}_i, \mathbf{x}_j \rangle \geq \frac{1}{2}\{\|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2 - (\|\hat{\mathbf{x}}_i\| + \|\hat{\mathbf{x}}_j\|)^2\}$.*

Proof Since $\|\mathbf{x}_i - \mathbf{x}_j\|^2 = \|\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j\|^2$ from Equation (10), $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \frac{1}{2}(\|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2 - \|\hat{\mathbf{x}}_i\|^2 - \|\hat{\mathbf{x}}_j\|^2 + 2\langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle)$ holds. Since $\langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle \leq \|\hat{\mathbf{x}}_i\|\|\hat{\mathbf{x}}_j\|$ holds from the Cauchy-Schwarz inequality, we have $-\|\hat{\mathbf{x}}_i\|^2 - \|\hat{\mathbf{x}}_j\|^2 + 2\langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle \leq -(\|\hat{\mathbf{x}}_i\| - \|\hat{\mathbf{x}}_j\|)^2$. Similarly, since $\langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle \geq -\|\hat{\mathbf{x}}_i\|\|\hat{\mathbf{x}}_j\|$ holds, we have $-\|\hat{\mathbf{x}}_i\|^2 - \|\hat{\mathbf{x}}_j\|^2 + 2\langle \hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j \rangle \geq -(\|\hat{\mathbf{x}}_i\| + \|\hat{\mathbf{x}}_j\|)^2$, which completes the proof. □

Note that we can compute norm $\|\mathbf{x}_i\|$ and $\|\mathbf{x}_j\|$ before applying the solver. As a result, we can exploit Lemma 2 to efficiently compute the bounds for linear, polynomial, and sigmoid kernel functions since these kernel functions can be computed from inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$.

Selective computation

In order to reduce the computation cost of one-class SVM, we effectively use set of selected data points $\mathbb{S}$ and set of pruned data points $\mathbb{P}$. More specifically, we apply the solver for data points in set $\mathbb{S}$ and check the optimality of the hyper-plane from data points in set $\mathbb{P}$. If the hyper-plane is not confirmed to be optimal, we update set $\mathbb{S}$ and $\mathbb{P}$. Let $\mathbb{X}$ be a set of the given data points, i.e., $\mathbb{X} = \{\mathbf{x}_i | 1 \leq i \leq n\}$, we determine set $\mathbb{S}$ and $\mathbb{P}$ so that they meet the three conditions of (1) $\mathbb{S} \cup \mathbb{P} = \mathbb{X}$, (2) $\mathbb{S} \cap \mathbb{P} = \emptyset$, and (3) $\forall \mathbf{x}_i \in \mathbb{P}$, $\alpha_i = 0$ or $\alpha_i = \frac{1}{n\nu}$. We later described our approach to determine set $\mathbb{S}$ and $\mathbb{P}$. Theoretically, our approach is based on the following property for set $\mathbb{S}$ and $\mathbb{P}$.

Lemma 3 *Let $\mathbb{S}$ and $\mathbb{P}$ be sets of data points such that (1) $\mathbb{S} \cup \mathbb{P} = \mathbb{X}$, (2) $\mathbb{S} \cap \mathbb{P} = \emptyset$, and (3) $\forall \mathbf{x}_i \in \mathbb{P}$, $\alpha_i = 0$ or $\alpha_i = \frac{1}{n\nu}$. In addition, let α'_i be the Lagrange multiplier of data point $\mathbf{x}_i$ before applying the solver. The hyper-plane is optimal if the following two conditions hold where the bounds are computed from Lagrange multipliers after applying the solver to the data points in set $\mathbb{S}$:*

$$(1) \forall \mathbf{x}_i \in \mathbb{P} \text{ s.t. } \alpha'_i = \tfrac{1}{n\nu},\ \overline{z}_i < 0 \tag{12}$$

$$(2) \forall \mathbf{x}_i \in \mathbb{P} \text{ s.t. } \alpha'_i = 0,\ \underline{z}_i > 0 \tag{13}$$

Proof Each α_i of data point $\mathbf{x}_i$ included in set $\mathbb{S}$ reaches convergence after applying the solver since the optimization problem of one-class SVM is convex (Schölkopf et al. 2001). As a result, since we have $\mathbb{S} \cup \mathbb{P} = \mathbb{X}$ and $\mathbb{S} \cap \mathbb{P} = \emptyset$, we need to show that (1) $\alpha_i = \frac{1}{n\nu}$ holds for data point $\mathbf{x}_i \in \mathbb{P}$ such that $\alpha'_i = \frac{1}{n\nu}$ if we have $\overline{z}_i < 0$ and (2) $\alpha_i = 0$ holds for data point $\mathbf{x}_i \in \mathbb{P}$ such that $\alpha'_i = 0$ if we have $\underline{z}_i > 0$ even if we apply the solver to all the data points in set $\mathbb{X}$ in order to prove Lemma 3. Note that set $\mathbb{P}$ does not include data point $\mathbf{x}_i$ such that $0 < \alpha_i < \frac{1}{n\nu}$ from the condition for set $\mathbb{P}$.

If $\overline{z}_i < 0$ holds, we have $z_i < 0$ since $z_i \leq \overline{z}_i$ holds, and we have $\xi_i > 0$ for such data points (Cristianini and Shawe-Taylor 2000). In addition, if β_i is a Lagrange multiplier, we have (1) $\beta_i \xi_i = 0$ and (2) $\alpha_i(z_i + \xi_i) = 0$ from the Kuhn-Tucker Theorem (Cristianini and Shawe-Taylor 2000). If we assume $\alpha_i = 0$ for such data points as $\overline{z}_i < 0$, we have $\beta_i = \frac{1}{n\nu}$ since $\beta_i = \frac{1}{n\nu} - \alpha_i$ holds by differentiating the Lagrange function with respect to ξ_i. As a result, we have $\xi_i = 0$ since $\beta_i \xi_i = 0$ holds. However, this contradicts the result of $\overline{z}_i < 0$. Therefore, we have $\alpha_i \neq 0$. If we assume $0 < \alpha_i < \frac{1}{n\nu}$ for such data points of $\overline{z}_i < 0$, we similarly have $\xi_i = 0$ since we have $\beta_i \xi_i = 0$ where $\beta_i \neq 0$. However, this contradicts the result of $\overline{z}_i < 0$. As a result, we have $\alpha_i = \frac{1}{n\nu}$ for such data points of $\overline{z}_i < 0$. If $\underline{z}_i > 0$ holds, we have $z_i > 0$ and $\xi_i = 0$ (Cristianini and Shawe-Taylor 2000). As a result, we have $z_i + \xi_i \neq 0$. Therefore, $\alpha_i = 0$ holds from $\alpha_i(z_i + \xi_i) = 0$, which completes the proof. □

As shown in Lemma 3, we can check the optimality of the hyper-plane from data points in set $\mathbb{P}$; we do not need set $\mathbb{S}$. In addition, this lemma indicates that we can check the optimality by using bound $\overline{z}_i$ and $\underline{z}_i$; we do not compute parameter z_i. As a result, we can check the optimality at most $O(n)$ time if we have the bounds from Lemma 3. On the other hand, the shrinking approach checks the optimality of the hyper-plane by exactly computing parameter z_i for all the data points. Therefore, our approach can more efficiently check the optimality than the shrinking approach.

In exploiting Lemma 3, we have the assumption for set $\mathbb{S}$ and $\mathbb{P}$ that they meet the three conditions of (1) $\mathbb{S} \cup \mathbb{P} = \mathbb{X}$, (2) $\mathbb{S} \cap \mathbb{P} = \emptyset$, and (3) $\forall \mathbf{x}_i \in \mathbb{P}$, $\alpha_i = 0$ or $\alpha_i = \frac{1}{n\nu}$. In our approach, we initialize and update set $\mathbb{S}$ and $\mathbb{P}$ as follows in the iterative process to compute the optimal hyper-plane:

Definition 2 *We initialize set $\mathbb{S}$ by including data points that meet one of the following conditions:*

$$(1)\ \alpha'_i = \tfrac{1}{n\nu} \text{ and } \overline{z}_i \geq 0 \tag{14}$$

$$(2)\ 0 < \alpha'_i < \tfrac{1}{n\nu} \tag{15}$$

$$(3)\ \alpha'_i = 0 \text{ and } \underline{z}_i \leq 0 \tag{16}$$

In addition, we initialize set $\mathbb{P}$ by including data points that meet either of the following two conditions:

$$(1)\ \alpha'_i = \tfrac{1}{n\nu} \text{ and } \overline{z}_i < 0 \tag{17}$$

$$(2)\ \alpha'_i = 0 \text{ and } \underline{z}_i > 0 \tag{18}$$

Definition 3 *If $\mathbb{S}'$ and $\mathbb{P}'$ are the set of selected and pruned data points before the update, respectively, we update $\mathbb{S}$ and $\mathbb{P}$ as follows after applying the solver if the optimality is not confirmed:*

$$\mathbb{S} = \mathbb{S}' \cup \{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{P}' \text{ and } \underline{z}_i \leq 0 \leq \overline{z}_i\} \tag{19}$$

$$\mathbb{P} = \mathbb{P}' \setminus \{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{P}' \text{ and } \underline{z}_i \leq 0 \leq \overline{z}_i\} \tag{20}$$

As shown in Definition 2, we use the initial settings of the Lagrange multipliers in computing set $\mathbb{S}$ and $\mathbb{P}$. We describe the approach to initializing the Lagrange multipliers in the next section. In Definition 3, we compute the bounds after applying the solver. Definition 3 indicates that we move data points to set $\mathbb{S}$ from set $\mathbb{P}$ if the hyper-plane is not confirmed to be optimal; set $\mathbb{S}$ and $\mathbb{P}$ are monotonically increased and decreased, respectively. Definition 2 and 3 indicate that we can obtain set $\mathbb{S}$ and $\mathbb{P}$ within $O(n)$ time if we have the bounds. For set $\mathbb{S}$ and $\mathbb{P}$, we have the following property:

Lemma 4 *If set $\mathbb{S}$ and $\mathbb{P}$ are given by Definition 2 and 3, we have (1) $\mathbb{S} \cup \mathbb{P} = \mathbb{X}$, (2) $\mathbb{S} \cap \mathbb{P} = \emptyset$, and (3) $\forall \mathbf{x}_i \in \mathbb{P}$, $\alpha_i = 0$ or $\alpha_i = \frac{1}{n\nu}$.*

Proof Just after the initialization by Definition 2, each data point included in set $\mathbb{S} \cup \mathbb{P}$ meets one of the following conditions: (1) $\overline{z}_i \geq 0$ or $\overline{z}_i < 0$ if $\alpha'_i = \frac{1}{n\nu}$, (2) $0 < \alpha'_i < \frac{1}{n\nu}$, and (3) $\underline{z}_i \leq 0$ or $\underline{z}_i > 0$ if $\alpha'_i = 0$. Since $\overline{z}_i$ and $\underline{z}_i$ are the bounds of parameter z_i, we have $-\infty \leq \underline{z}_i \leq \overline{z}_i \leq \infty$. Therefore, it is clear that all the data points are included in set $\mathbb{S} \cup \mathbb{P}$, i.e., $\mathbb{S} \cup \mathbb{P} = \mathbb{X}$. Similarly, we have the following conditions for data points in set $\mathbb{S} \cap \mathbb{P}$: (1) $\overline{z}_i \geq 0$ and $\overline{z}_i < 0$ if $\alpha'_i = \frac{1}{n\nu}$ and (2) $\underline{z}_i \leq 0$ and $\underline{z}_i > 0$ if $\alpha'_i = 0$. Therefore, it is clear that $\mathbb{S} \cap \mathbb{P} = \emptyset$. In addition, as shown in Equation (17) and (18), a data point can be included in set $\mathbb{P}$ only if $\alpha'_i = \frac{1}{n\nu}$ or $\alpha'_i = 0$. Therefore, $\mathbb{P}$ does not include data point $\mathbf{x}_i$ such that $0 < \alpha_i < \frac{1}{n\nu}$. After updating sets $\mathbb{S}$ and $\mathbb{P}$, since the set of data points $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{P}'$ and $\underline{z}_i \leq 0 \leq \overline{z}_i\}$ is moved from set $\mathbb{P}$ to set $\mathbb{S}$ as shown in Equation (19) and (20), we have $\mathbb{S} \cup \mathbb{P} = \mathbb{S}' \cup \mathbb{P}' = \mathbb{X}$ and $\mathbb{S} \cap \mathbb{P} = \mathbb{S}' \cap \mathbb{P}' = \emptyset$. In addition, from Equation (20), any data points are not added to set $\mathbb{P}$ in the update computation. As a result, since set $\mathbb{P}$ does not initially have a data point such that $0 < \alpha_i < \frac{1}{n\nu}$, all the data points in set $\mathbb{P}$ meet the condition of $\alpha_i = 0$ or $\alpha_i = \frac{1}{n\nu}$. □

Algorithm 1 Initialization of $\boldsymbol{\alpha}$

```
Input: Matrix $\mathbf{X}$, parameter $\nu$, number of data point $n$
Output: Lagrange multiplier $\boldsymbol{\alpha}$
 1: $\boldsymbol{\alpha} = \mathbf{0}$, $\alpha_{sum} = 0$, $\mathbb{A} = \emptyset$;
 2: repeat
 3:     if $\mathbb{A} \neq \emptyset$ then
 4:         $\mathbf{x}_i = \arg\max\{p_i | \mathbf{x}_i \in \mathbb{X} \backslash \mathbb{A}\}$;
 5:     else
 6:         $\mathbf{x}_i = \arg\max\{\|\hat{\mathbf{x}}_i\| | \mathbf{x}_i \in \mathbb{X}\}$;
 7:     add $\mathbf{x}_i$ to $\mathbb{A}$ and randomly set $\alpha_i$;
 8:     if $\alpha_{sum} + \alpha_i > 1$ then
 9:         $\alpha_i = 1 - \alpha_{sum}$;
10:     $\alpha_{sum} = \alpha_{sum} + \alpha_i$;
11:     for $i = 1$ to $n$ do
12:         incrementally compute priority $p_i$;
13: until $\alpha_{sum} = 1$
```

Lemma 4 indicates that set $\mathbb{S}$ and $\mathbb{P}$ given by Definition 2 and 3 meet the conditions assumed in Lemma 3. Therefore, we can check the optimality from Definition 2 and 3.

Lagrange multiplier initialization

As described in the previous sections, we need to compute the initial setting of Lagrange multiplier α_i in computing set $\mathbb{S}$ and $\mathbb{P}$ from the bounds. This section introduces our approach that determines the initial setting.

In the proposed approach, we first set $\alpha_i = 0$ for each data point. We then add data points one by one to the set of support vectors from the data points that have the highest priorities until we have $\sum_{i=1}^{n} \alpha_i = 1$. We randomly set the Lagrange multiplier of the added data point to $0 < \alpha_i \leq \frac{1}{n\nu}$. Note that Lagrange multipliers have constraints such that $0 \leq \alpha_i \leq \frac{1}{n\nu}$ and $\sum_{i=1}^{n} \alpha_i = 1$ as shown in Equation (4). We iteratively compute priority p_i of data point $\mathbf{x}_i$ as $p_i = -\sum_{j=1}^{n} \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$ by adding data points as support vectors. Note that priority p_i is computed from the Lagrange multipliers and the kernel function of data point $\mathbf{x}_i$ similar to the definition of parameter z_i in Equation (5) although offset ρ is not used in computing priority p_i. This indicates that priority p_i is expected to have high value if data point $\mathbf{x}_i$ is determined to be an outlier such that $\alpha_i > 0$. As a result, we can effectively compute the initial setting of Lagrange multiplier α_i from priority p_i. Note that we can incrementally compute priority p_i of each data point within $O(m)$ time in the process of adding data points one by one; we can efficiently update priority of each data points.

Algorithm 1 shows the approach to compute the initial setting of Lagrange multipliers. If $\mathbb{A}$ is a set of data points that are added as support vectors, it selects the data point that has maximum priority if $\mathbb{A} \neq \emptyset$ holds; otherwise, it selects the data point offering the maximum norm since such data points are expected to be outliers (lines 3-6). For the selected data point, it randomly sets the Lagrange multiplier as $0 < \alpha_i \leq \frac{1}{n\nu}$, and, if the sum of the Lagrange multiplier

Algorithm 2 Quix

```
Input: Matrix $\mathbf{X}$, parameter $\nu$, Lagrange multiplier $\boldsymbol{\alpha}$
Output: Lagrange multiplier $\boldsymbol{\alpha}$, offset $\rho$
 1: $\mathbb{S} = \emptyset$, $\mathbb{P} = \emptyset$;
 2: for $i = 1$ to $m$ do
 3:     sort elements in the $i$-th column vector of $\mathbf{X}$;
 4:     find the most frequent value from the sorted elements;
 5: for $i = 1$ to $n$ do
 6:     compute vector $\hat{\mathbf{x}}_i$ from the most frequent values;
 7: compute the initial Lagrange multiplier by Algorithm 1;
 8: for $i = 1$ to $n$ do
 9:     compute bound $\overline{z}_i$ and $\underline{z}_i$ from Definition 1;
10: compute set $\mathbb{S}$ and $\mathbb{P}$ from Definition 2;
11: repeat
12:     apply the solver to the data points included in set $\mathbb{S}$;
13:     for $i = 1$ to $n$ do
14:         incrementally compute $\overline{z}_i$ and $\underline{z}_i$ by updating set $\mathbb{O}$;
15:     check the optimality of the hyper-plane by using Lemma 3;
16:     update set $\mathbb{S}$ and $\mathbb{P}$ from Definition 3;
17: until the hyper-plane is optimal
18: compute $\rho$ from $\boldsymbol{\alpha}$;
```

exceeds 1, it resets α_i by following the constraint for one-class SVM (lines 7-10). It then incrementally computes the priority of each data point (lines 11-12).

Since we can obtain sub-optimal hyper-plane by using the initialization approach, we can effectively reduce the training time. Even if we randomly initialize the dual variables, we can obtain the optimal hyper-plane since our approach checks the optimality of the obtained hyper-plane by Lemma 3, however, training time would be a little bit long.

Algorithm

Algorithm 2 gives a full description of our approach, Quix. It starts by computing the most frequent value σ_i from the i-th column to obtain sparse vectors (lines 2-6). It then obtains the initial Lagrange multiplier settings by using Algorithm 1 (line 7). It computes the upper and lower bounds of each data point and the sets of selected and pruned data points from the definitions (lines 8-10). In order to compute the hyper-plane, it applies the solver to the selected data points and updates the bounds (lines 12-14). It then checks the optimality of the obtained hyper-plane by exploiting Lemma 3 and incrementally computes the set of selected and pruned data points from Definition 3 (lines 15-16). It performs these procedures by using Lagrange multipliers of the previous iteration as the warm-start until the optimal hyper-plane is assured (line 17). Note that we can use various solvers since our approach is solver-independent as shown in Algorithm 2. In addition, our approach can be easily extended to online fashion. When we have a new data point, we obtain set $\mathbb{A}$ by adding data points of previous support vectors and the new data points. After that, we randomly set the initial setting of Lagrange multiplier so that the condition of Equation (4) should meet.

Our approach has the following properties:

Theorem 1 *The proposed approach is guaranteed to yield the optimal hyper-plane.*

Proof Since we have $\mathbb{S} \cup \mathbb{P} = \mathbb{X}$ and $\mathbb{S} \cap \mathbb{P} = \emptyset$ from Lemma 4, Algorithm 2 assigns all the data points to either set $\mathbb{S}$ or $\mathbb{P}$ after the initialization by Definition 2 (line 10). After initialization, we apply the solver to each data point in set $\mathbb{S}$ (line 12). It is clear that the hyper-plane is optimal if it passes the optimality check according to Lemma 3. From the obtained hyper-plane, we update set $\mathbb{S}$ and $\mathbb{P}$ by using Definition 3 (line 16). In this process, set $\mathbb{S}$ and $\mathbb{P}$ are monotonically increased and decreased, respectively. This is because, as shown in Definition 3, set of data points $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{P}' \text{ and } \underline{z}_i \leq 0 \leq \overline{z}_i\}$ is added to set $\mathbb{S}$ by subtracting it from set $\mathbb{P}$. This indicates that this process can update set $\mathbb{S}$ and $\mathbb{P}$ until $\mathbb{S} = \mathbb{X}$ and $\mathbb{P} = \emptyset$. In this case, it is clear that our approach can obtain the optimal hyper-plane since it applies the solver to all data points. □

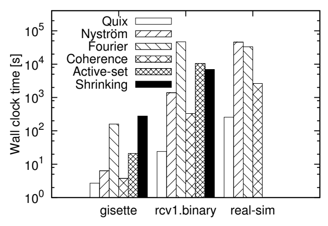
Theorem 2 *Let $C(\mathbb{S})$ and t be the computation cost of the solver applied to set $\mathbb{S}$ and the number of applications of the solver, respectively. In addition, let $|\mathbb{A}|$ and $|\mathbb{O}|$ be the number of data points in $\mathbb{A}$ and $\mathbb{O}$, respectively. The computation cost of the proposed approach is $O(nm(\log n + |\mathbb{A}| + |\mathbb{O}|) + (C(\mathbb{S}) + n|\mathbb{H}|)t)$.*

Proof As shown in Algorithm 1, in computing the initial setting of Lagrange multipliers, it identifies the data points that have the largest priorities if $\mathbb{A} \neq \emptyset$ holds (lines 2-4); otherwise, it obtains the data point with maximum norm (lines 5-6). Since we can update priorities within $O(nm)$ time after adding a data point to set $\mathbb{A}$, the computation cost for these processes is $O(nm|\mathbb{A}|)$. As shown in Algorithm 2, in computing the optimal hyper-plane, we first compute the most frequent value of each column of data matrix $\mathbf{X}$ to obtain vector $\hat{\mathbf{x}}_i$ (lines 2-6). This process needs $O(nm \log n)$ time by using Quicksort. It computes the bounds from set $\mathbb{A}$ at $O(nm|\mathbb{A}|)$ time (lines 8-9) and initializes sets $\mathbb{S}$ and $\mathbb{P}$ within $O(n)$ time (line 10). In computing the hyper-plane, it applies the solver to the selected data points within $O(C(\mathbb{S})t)$ time. It needs $O(nm|\mathbb{O}| + n|\mathbb{H}|t)$ time to update the bounds (lines 13-14). In addition, it requires $O(nt)$ time to check the optimality of the hyper-plane (line 15). To compute offset ρ, it needs $O(n)$ time (line 18). As a result, the computation cost of the proposed approach is $O(nm(\log n + |\mathbb{A}| + |\mathbb{O}|) + (C(\mathbb{S}) + n|\mathbb{H}|)t)$. □

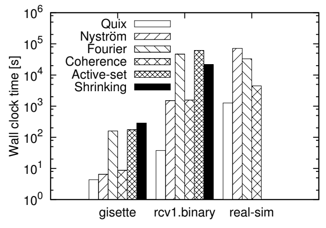
Since we randomly set Lagrange multiplier from 0 to $1/(n\nu)$, average of Lagrange multiplier initially included in set $\mathbb{S}$ is $2n\nu$. In addition, at least one data point is added to set $\mathbb{S}$ in each iteration. Note that if no data point is added, we terminate the iterations. Therefore, The upper bound of number of iterations is $O(|\mathbb{S}| - 2n\nu)$.

Experimental evaluation

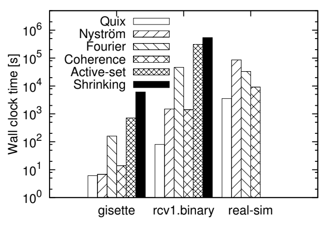
In this section, we evaluated the efficiency and effectiveness of the proposed approach. We performed the experiments on three datasets of *gisette*, *rcv1.binary*, and *real-sim* downloaded from the website of LIBSVM. They have 6000, 20242, and 72309 data points, respectively; these datasets have dimensions of 5000, 47236, and 20958, respectively. In the datasets, each column vector of data matrix $\mathbf{X}$ is standardized to have mean zero and variance one by following the previous papers (Çeker and Upadhyaya 2016; Fisher, Camp, and Krzhizhanovskaya 2016).



(1) $\nu = 0.02$



(2) $\nu = 0.04$



(3) $\nu = 0.06$

Figure 1: Training time of each approach.

We compared our approach to the Nyström method (Drineas and Mahoney 2005), random Fourier features (Rahimi and Recht 2007), the coherence criterion-based approach (Noumir, Honeine, and Richard 2012), the active-set method-based approach (Gao 2015), and the shrinking approach (Joachims 1999). In the experiments, we set the number of landmark to $0.01 \cdot n$ for the Nyström method. We set the number of features to $0.01 \cdot m$ for Random Fourier features. The coherence criterion-based approach incrementally adds data points to a set of support vectors by exploiting threshold μ_0 if it is not well approximated by the support vectors. The active-set method-based approach updates the hyper-plane by adding a data point to a set of support vectors if its distance to the center of hyper-plane is not less than the radius of the hyper-plane. In the experiments of the coherence criterion-based approach and the active-set method-based approach, we show the results in training all the data points. Note that these four approaches inherently can discard data points that can be support vectors of the optimal hyper-plane unlike our approach. On the other hand, the shrinking approach obtains the optimal hyper-plane by checking the optimality of the hyper-plane on the basis of Kuhn-Tucker Theorem. In this section, "Quix", "Nyström", "Fourier", "Coherence", "Active-set", and "Shrinking" represent the results of the proposed approach, the Nyström method, random Fourier features, the coherence criterion-based approach, the active-set method-based approach, and the shrinking approach, respectively.

Our approach can use SMO variants since it prunes data points before applying the solver as shown in Algorithm 2. Similarly, SMO variants can be used in the Nyström method (Drineas and Mahoney 2005), random Fourier features (Rahimi and Recht 2007), the active-set method-based approach (Gao 2015), and the shrinking approach (Joachims 1999). However, since the coherence criterion-based approach (Noumir, Honeine, and Richard 2012) uses the incremental approach, it cannot use SMO variants. In addition, the solver of LIBSVM exploits several heuristics to improve the efficiency. Therefore, in order to ensure fair comparisons, we used the original SMO (Platt 1998) as the solver. We used the RBF kernel function and set kernel parameter $\gamma = \frac{1}{m}$. We conducted all experiments on a Linux server with 2.70 GHz Intel Xeon.

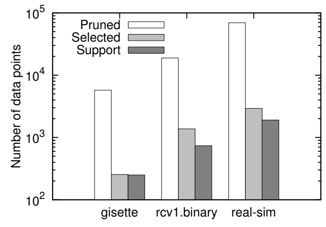
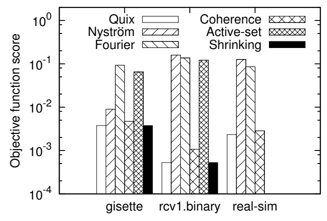


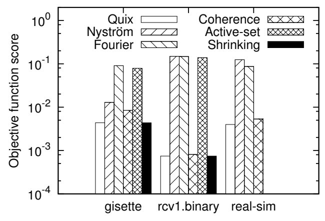
Figure 2: Effectiveness of the pruning

Efficiency

We evaluated the training time of each approach. Figure 1 shows the results with parameter ν values of 0.02, 0.04, and 0.06. For the coherence criterion-based approach, we set threshold μ_0 so that the number of support vectors is $n\nu$ for parameter ν to ensure fair comparisons. Note that ν is the lower bound on the fraction of support vectors. Therefore, we changed ν to evaluate our approach for various numbers of support vectors. For real-sim dataset, we omit the results of the active-set method-based approach and the shrinking approach since training could not be completed within a week. Figure 2 shows the number of data points included in set $\mathbb{S}$ and $\mathbb{P}$ of our approach and the number of support vectors after we have the optimal hyper-plane. In this figure, "Pruned", "Selected", and "Support" represent the number of selected data points in set $\mathbb{S}$, pruned data points in set $\mathbb{P}$, and support vectors, respectively, where we set $\nu = 0.02$.

As shown in Figure 1, our approach is much faster than the previous approaches. Specifically, it is up to 170, 190, 40, 3800, and 6800 times faster than the Nyström method, random Fourier features, the coherence criterion-based approach, active-set method-based approach, and the shrinking approach, respectively. The Nyström method and random Fourier features are not so effective in reducing the computation time. This is because, although these approaches can efficiently compute kernel functions used in the solver, they need to apply the solver for all the data points. In addition, the proposed approach is more efficient than the coherence criterion-based and active-set method-based approaches. The active-set method-based approach incurs high computation cost since it must compute the distances of data points from the center of the hyper-plane to discard data


(1) $\nu = 0.02$


(2) $\nu = 0.04$

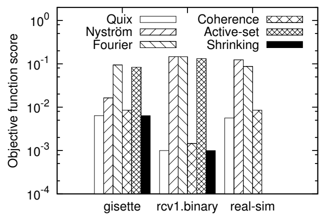

(3) $\nu = 0.06$

Figure 3: Objective function score of each approach.

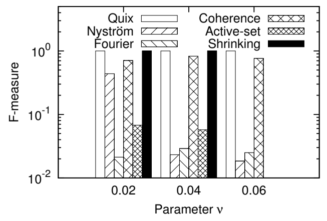


Figure 4: F-measures of output support vectors.

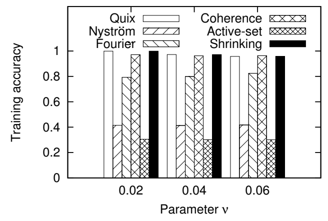


Figure 5: Training accuracy of each approach

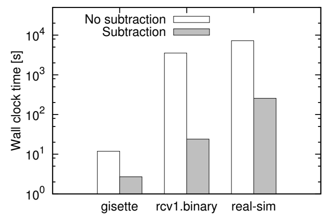


Figure 6: Training time of the subtraction approach

points. Since the coherence criterion-based approach can efficiently check the data points by using the criterion, it is faster than the active-set method-based approach. However, the effectiveness of the coherence criterion-based approach is moderate. This is because, although it incrementally updates the Lagrange multipliers by using the Woodbury matrix identity, it needs to update the Lagrange multipliers every time data points are added to the set of support vectors. Although the shrinking approach can reduce data points before applying the solver, it needs to compute parameter z_i of each data point from the kernel functions in checking the optimality of the hyper-plane. Therefore, the shrinking approach does not effectively reduce the computation cost. On the other hand, our approach efficiently computes the upper and lower bounds of each data point before applying the solver. In addition, we can effectively reduce sets of selected data points applied to the solver as shown in Figure 2. Note that the number of applications of the solver in our approach, t, was at most two in the experiments. Even though we use the set of pruned data points in checking the optimality, we can efficiently check the optimality by using upper and lower bounds. Note that our approach is more efficient than the incremental SVM approach (Laskov et al. 2006). Incremental SVM needs to iteratively add data point one by one. Therefore, it needs $O(n|\mathbb{S}|^2)$ time to perform batch learning ($n \gg |\mathbb{S}|$) if SMO is used as the solver. On the other hand, we compute hyper-plane for s data points and thus needs $O(|\mathbb{S}|^2)$ time. Therefore, our approach is more efficient than the previous approaches.

Optimality

One major advantage of our approach is that it yields the optimal hyper-plane that minimizes the objective function given by Equation (1). In Figure 3, we show the scores of the objective function yielded by each approach for parameter ν values of 0.02, 0.04, and 0.06. In addition, Figure 4 shows the F-measure against the original approach of one-class SVM in identifying support vectors, and Figure 5 show the training accuracy of each approach where we used gisette dataset. F-measure of an approach is 1 if the obtained support vectors by the approach exactly match those of the original approach.

Figure 3 shows that our approach has smaller objective function scores than the previous approximate approaches. The Nyström method randomly selects data points as landmarks and random Fourier features randomly selects features in order to compute low-rank approximation. However, they erroneously discard data points that can be support vectors of the optimal hyper-plane as shown in Figure 4; they cannot exactly obtain the support vectors. As a result, they fail to minimize the score of the objective function given by Equation (1) as shown in Figure 3. In addition, the online approaches exploit heuristics in checking data points to improve the efficiency. However, these approaches cannot obtain the optimal hyper-plane since the heuristics have no theoretical foundation. Note that approximate approaches result in reducing accuracy of SVM as demonstrated in the previous papers (Li et al. 2016; Yang et al. 2012). On the other hand, although our approach also discards data points it is carefully designed so that support vectors of the optimal hyper-plane can be obtained. Therefore, as shown in Figure 4, our approach can accurately identify the support vectors for the given data points. As a result, the training accuracy of the proposed approach is high as shown in 5. Note that, as described in the preliminaries section, the training accuracy of one-class SVM cannot smaller than $1-\nu$. Although the shrinking approach can obtained the optimal hyper-plane, it needs high computation cost as shown in Figure 1.

Subtraction approach

The proposed approach uses sparse vector $\hat{\mathbf{x}}_i$ to efficiently compute the upper and lower bounds. So as to evaluate the effectiveness of this approach, we perform experiments. In Figure 6, we show the training times where we set $\nu = 0.02$. In this figure, "No subtraction" and "Subtraction" represent the results of approaches without and with the subtraction approaches, respectively. Note that gisette dataset obtained from the website of LIBSVM has numerical features.

Figure 6 shows that we can increase the training speed by reducing the number of non-zero elements. In order to increase the number of non-zero elements, the proposed approach subtracts the most frequent value in each vector. As a result, our approach can use the sparse vectors in computing the bounds. Since we can effectively reduce non-zero elements, we can increase the training efficiency of one-class SVM; the training time is up to 147 times faster by exploiting the subtraction approach.

Conclusions

In this paper, we proposed Quix, an efficient algorithm for one-class SVM that is guaranteed to yield the optimal hyper-plane. The proposed approach computes upper and lower bounds of a parameter that determines the hyper-plane to improve calculation efficiency. Experiments indicated that the proposed approach outperforms previous approaches in terms of efficiency and optimality. The proposed approach is an attractive option in the application of one-class SVM.

References

Cauwenberghs, G., and Poggio, T. A. 2000. Incremental and Decremental Support Vector Machine Learning. In *NIPS*, 409–415.

Çeker, H., and Upadhyaya, S. J. 2016. User Authentication with Keystroke Dynamics in Long-text Data. In *BTAS*, 1–6.

Cristianini, N., and Shawe-Taylor, J. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

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

Fisher, W.; Camp, T.; and Krzhizhanovskaya, V. V. 2016. Crack Detection in Earth Dam and Levee Passive Seismic Data Using Support Vector Machines. In *ICCS*, 577–586.

Fujiwara, Y.; Nakatsuji, M.; Shiokawa, H.; Mishima, T.; and Onizuka, M. 2013. Fast and Exact Top-k Algorithm for PageRank. In *AAAI*.

Fujiwara, Y.; Nakatsuji, M.; Shiokawa, H.; Ida, Y.; and Toyoda, M. 2015. Adaptive Message Update for Fast Aaffinity Propagation. In *SIGKDD*, 309–318.

Fujiwara, Y.; Marumo, N.; Blondel, M.; Takeuchi, K.; Kim, H.; Iwata, T.; and Ueda, N. 2017a. Scaling Locally Linear Embedding. In *SIGMOD*, 1479–1492.

Fujiwara, Y.; Marumo, N.; Blondel, M.; Takeuchi, K.; Kim, H.; Iwata, T.; and Ueda, N. 2017b. SVD-based Screening for the Graphical Lasso. In *IJCAI*, 1682–1688.

Gao, K. 2015. Online One-class SVMs with Active-set Optimization for Data Streams. In *ICMLA*, 116–121.

Gómez-Verdejo, V.; Arenas-García, J.; Lázaro-Gredilla, M.; and Navia-Vázquez, Á. 2011. Adaptive One-class Support Vector Machine. *IEEE Trans. Signal Processing* 59(6):2975–2981.

Joachims, T. 1999. Advances in kernel methods. Cambridge, MA, USA: MIT Press. chapter Making Large-scale Support Vector Machine Learning Practical, 169–184.

Laskov, P.; Gehl, C.; Krüger, S.; and Müller, K. 2006. Incremental Support Vector Learning: Analysis, Implementation and Applications. *Journal of Machine Learning Research* 7:1909–1936.

Li, Z.; Yang, T.; Zhang, L.; and Jin, R. 2016. Fast and Accurate Refined Nyström-based Kernel SVM. In *AAAI*, 1830–1836.

Mishima, T., and Fujiwara, Y. 2015. Madeus: Database Live Migration Middleware under Heavy Workloads for Cloud Environment. In *SIGMOD*, 315–329.

Musco, C., and Musco, C. 2017. Recursive Sampling for the Nystrom Method. In *NIPS*, 3836–3848.

Nakatsuji, M., and Fujiwara, Y. 2014. Linked Taxonomies to Capture Users' Subjective Assessments of Items to Facilitate Accurate Collaborative Filtering. *Artif. Intell.* 207:52–68.

Nakatsuji, M.; Fujiwara, Y.; Toda, H.; Sawada, H.; Zheng, J.; and Hendler, J. A. 2014. Semantic Data Representation for Improving Tensor Factorization. In *AAAI*, 2004–2012.

Noumir, Z.; Honeine, P.; and Richard, C. 2012. Online One-class Machines Based on the Coherence Criterion. In *EUSIPCO*, 664–668.

Platt, J. 1998. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Technical report.

Rahimi, A., and Recht, B. 2007. Random Reatures for Large-scale Kernel Machines. In *NIPS*, 1177–1184.

Schölkopf, B.; Platt, J. C.; Shawe-Taylor, J.; Smola, A. J.; and Williamson, R. C. 2001. Estimating the Support of a High-dimensional Distribution. *Neural Computation* 13(7):1443–1471.

Steele, J. M. 2004. *The Cauchy-Schwarz Master Class: An Introduction to the Art of Mathematical Inequalities*. Cambridge University Press.

Tanaka, Y.; Kurashima, T.; Fujiwara, Y.; Iwata, T.; and Sawada, H. 2016. Inferring Latent Triggers of Purchases with Consideration of Social Effects and Media Advertisements. In *WSDM*, 543–552.

Wu, L.; Yen, I. E.; Chen, J.; and Yan, R. 2016. Revisiting Random Binning Features: Fast Convergence and Strong parallelizability.P In *KDD*, 1265–1274.

Yang, T.; Li, Y.; Mahdavi, M.; Jin, R.; and Zhou, Z. 2012. Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison. In *NIPS*, 485–493.