A PAC Bound for Approximate Support Vector Machines*

Dongwei Cao† Daniel Boley†

Abstract

We study a class of algorithms that speed up the training process of support vector machines (SVMs) by returning an approximate SVM. We focus on algorithms that reduce the size of the optimization problem by extracting from the original training dataset a small number of representatives and using these representatives to train an approximate SVM. The main contribution of this paper is a PAC-style generalization bound for the resulting approximate SVM, which provides a learning theoretic justification for using the approximate SVM in practice. The proved bound also generalizes and includes as a special case the generalization bound for the exact SVM, which denotes the SVM given by the original training dataset in this paper.

Keyword: Support Vector Machines, Approximate Solutions, Generalization Bounds, Algorithmic Stability

1 Introduction

One challenge in using support vector machines (SVM) [8, 28] for problems with a large number of training data, which are common in data mining applications, is the prohibitive computational requirement for training, which involves solving a convex optimization problem. To address this issue, many efficient training algorithms have been proposed.

The first class of algorithms attack the optimization problem directly. A commonly used strategy is to solve a series of small optimization problems, where ideas like chunking and decomposition are used [4, 17, 22], and one noteworthy example is the sequential minimal optimization (SMO) algorithm [24]. Special algorithms have also been developed for SVMs with particular kernels, such the linear kernel [18] and the Gaussian kernel [30]. We denote the SVM given by these algorithms and other algorithms that use the original training dataset directly as the *exact SVM*. The performance of exact SVMs have been theoretically justified in terms a generalization bound in many places, such as [5, 14, 16, 28].

*This work was partially supported by NSF Grant IIS-0534286.

†Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455. Email: {dcao,boley}@cs.umn.edu

A second class of algorithms resorts to an approximate solution, and the resulting SVM is denoted as an *approximate SVM*. For example, it has been proposed to construct an approximate SVM by approximating the Gram matrix with a smaller matrix using either low rank representation [12] or sampling [1, 29]. Assuming a linear kernel is used, Pavlov et al. [23] proposed to squash the original training dataset into a limited number of representatives and construct an approximate SVM using these representatives. Although these algorithms, together with many other algorithms for approximate SVMs, are well motivated and have been shown to be very effective experimentally, there is no direct theoretical justification on the generalization performance of the resulting approximate SVMs. The goal of this paper is to provide such a justification by proving a PAC-style generalization bound for the approximate SVM.

We will prove a generalization bound for the approximate SVMs given by a class of training algorithms, which is denoted as $\mathcal{A}_{\text{approx}}$ in this paper, by studying the stability of $\mathcal{A}_{\text{approx}}$ [5]. Starting with a training dataset $\mathcal{D}$ of size n and a pre-specified value for $k \leq n$, algorithm $\mathcal{A}_{\text{approx}}$ first partitions the training dataset $\mathcal{D}$ into k disjoint clusters and picks one representative for each cluster, then returns the approximate SVM trained using the k representatives, each of which is weighted by the size of the cluster it represents. Here, the representative of a cluster is defined as the feature-space center of the cluster. The algorithm $\mathcal{A}_{\text{approx}}$ subsumes a variety of specific procedures, including those in which a specific clustering algorithm is used to compute the partitions, and those in which the partitions are performed completely at random, a choice akin to random sampling. Thus, a direct consequence of the proved bound is that it provides a theoretical foundation for SVM training algorithms based on clustering, such as those studied in [3, 7, 31], thus justifying the use of approximate SVMs in practice.

In the rest of this paper, section 2 introduces the algorithm $\mathcal{A}_{\text{approx}}$ for training approximate SVMs, section 3 proves a PAC-style generalization bound for the approximate SVM given by $\mathcal{A}_{\text{approx}}$, section 4 discusses some consequences of the bound and compares it with related work, and section 5 concludes the paper.

2 Approximate Support Vector Machines

In a two-class classification problem, we are given a training dataset $\mathcal{D}$ of size n

$$\mathcal{D}=\{\mathbf{z}_j=(\mathbf{x}_j,y_j)\mid \mathbf{x}_j\in\mathcal{X}, y_j\in\{1,-1\}, j=1,\ldots,n\},$$

where the n data are assumed to be obtained by drawing independently from $\mathcal{Z}$ according to a fixed but unknown probability measure $\mathbb{P}$ defined over $\mathcal{Z}$. A hyper-plane classifier h classifies a test datum $\mathbf{x}\in\mathcal{X}$ by thresholding a real-valued function f, i.e.

$$h(\mathbf{x})=\operatorname{sign}(f(\mathbf{x})),\ \text{with}\ f(\mathbf{x})=\langle\mathbf{w},\phi(\mathbf{x})\rangle, \tag{2.1}$$

where $\mathbf{w}$ is the weight vector of the hyper-plane, ϕ is a mapping from the data space $\mathcal{X}$ to a reproducing kernel Hilbert "feature" space $\mathbb{H}$ equipped with a dot product $\langle\cdot,\cdot\rangle$. The space $\mathbb{H}$ is induced by a Mercer kernel K, which satisfies $K(\mathbf{x}',\mathbf{x}'')=\langle\phi(\mathbf{x}'),\phi(\mathbf{x}'')\rangle$ for all $\mathbf{x}',\mathbf{x}''\in\mathcal{X}$. Given a function f, we define a regularized functional $R(f;\mathcal{D},\lambda)$ parameterized by the training set $\mathcal{D}$ and real regularization parameter $\lambda>0$ in a manner similar to that in [5]

$$R(f;\mathcal{D},\lambda)=R_{\text{emp}}(f;\mathcal{D})+\lambda\Omega(f) \tag{2.2a}$$

$$R_{\text{emp}}(f;\mathcal{D})=\frac{1}{n}\sum_{\mathbf{z}\in\mathcal{D}}\ell_{\text{h}}(f,\mathbf{z});\ \Omega(f)=\|\mathbf{w}\|^2, \tag{2.2b}$$

where $\|\cdot\|$ is the norm induced by the inner product in $\mathbb{H}$, and ℓ_{h} is the hinge loss

$$\ell_{\text{h}}(f,\mathbf{z})=\ell_{\text{h}}(f,(\mathbf{x},y))=\max(0,1-yf(\mathbf{x})), \tag{2.3}$$

measuring by how much a data sample lies on the wrong side of the classification boundary. The function f^* corresponding to a SVM classifier h^* is the minimizer of $R(f;\mathcal{D},\lambda)$

$$f^*=\operatorname*{argmin}_f R(f;\mathcal{D},\lambda). \tag{2.4}$$

Compared with the familiar SVM formulation [6], there is no bias term in the real-valued function f here. This choice simplifies the presentation later on while preserves the generality of the results developed.

We call the SVM h^* defined in equation (2.4) the *exact SVM*, since the original training dataset $\mathcal{D}$ is used directly in finding f^*. Usually, instead of minimizing $R(f;\mathcal{D},\lambda)$, the corresponding dual problem is solved and the time complexity is often $\mathcal{O}(n^p)$, where $1.7\le p\le 3$ [6,24].

Algorithm 1 describes the algorithm $\mathcal{A}_{\text{approx}}$, which speeds up the training process by returning an approximate SVM trained using a small number (k) of representatives of $\mathcal{D}$. The choice of k in algorithm $\mathcal{A}_{\text{approx}}$ depends on the available computational resources, such as the amount of memory. The main result of this paper is a PAC-style generalization bound for the approximate SVM $\widetilde{h}$ returned by $\mathcal{A}_{\text{approx}}$, which is proved in section 3. The effectiveness of algorithm $\mathcal{A}_{\text{approx}}$ has been demonstrated experimentally over several real datasets in [7], and will not be repeated here due to space limit.

Algorithm 1 Algorithm $\mathcal{A}_{\text{approx}}$

Require: A training dataset $\mathcal{D}$ of size n, number $k\le n$ of the representatives, kernel K, regularization coefficient λ.

1: Extract k representatives from $\mathcal{D}$ satisfying the following three assumptions:

- **Assumption I:** Each datum in $\mathcal{D}$ has one and only one representative, i.e., the k representatives correspond to a pairwise disjoint partition $\mathcal{D}_1,\ldots,\mathcal{D}_k$ of $\mathcal{D}$;
- **Assumption II:** For $j=1,\ldots,k$, all data in $\mathcal{D}_j$ have the same label, which is either 1 or -1;
- **Assumption III:** For $j=1,\ldots,k$, the representative $\widetilde{\mathbf{z}}_j^c=(\widetilde{\mathbf{x}}_j^c,\widetilde{y}_j^c)$ of $\mathcal{D}_j$ satisfies

$$\phi(\widetilde{\mathbf{x}}_j^c)=\frac{1}{n_j}\sum_{(\mathbf{x},y)\in\mathcal{D}_j}\phi(\mathbf{x})$$

$$\widetilde{y}_j^c=\begin{cases}1:\text{Data in }\mathcal{D}_j\text{ belong to class 1}\\ -1:\text{Data in }\mathcal{D}_j\text{ belong to class -1}\end{cases},$$

where n_j is the number of data in $\mathcal{D}_j$ and ϕ is the mapping induced by the kernel K;

2: Construct a reduced training dataset $\widetilde{\mathcal{D}}$ from $\mathcal{D}$ by replacing every datum in $\mathcal{D}$ with its representative;

$$\widetilde{\mathcal{D}}=\{\widetilde{\mathbf{z}}_j=(\widetilde{\mathbf{x}}_j,\widetilde{y}_j)\mid j=1,\ldots,n\},$$

3: **return** The approximate SVM $\widetilde{h}$ and corresponding real-valued function $\widetilde{f}$ (related by (2.1)), where

$$\widetilde{f}=\operatorname*{argmin}_f R\left(f;\widetilde{\mathcal{D}},\lambda\right).$$

Remark 2.1. When using a non-linear kernel, the representative $\phi(\widetilde{\mathbf{x}}_j^c)$ in "feature space" might not correspond to any particular point in $\mathcal{X}$. To train an approximate SVM, we need to compute the kernel value $K(\widetilde{\mathbf{x}}_i^c,\widetilde{\mathbf{x}}_j^c)$ between two representatives $\widetilde{\mathbf{x}}_i^c$ and $\widetilde{\mathbf{x}}_j^c$. To use the trained approximate SVM, we need to compute the kernel value $K(\widetilde{\mathbf{x}}_j^c,\widehat{\mathbf{x}})$ between a representative $\widetilde{\mathbf{x}}_j^c$ and an arbitrary datum $\widehat{\mathbf{x}}\in\mathcal{X}$. Fortunately, these two kernel values remain well-defined regardless of whether $\widetilde{\mathbf{x}}_j^c$ is a real or "virtual" point,

$$K(\widetilde{\mathbf{x}}_i^c, \widetilde{\mathbf{x}}_j^c) = \langle \phi(\widetilde{\mathbf{x}}_i^c), \phi(\widetilde{\mathbf{x}}_j^c) \rangle = \frac{1}{n_i n_j} \sum_{\substack{(\mathbf{x}', y') \in \mathcal{D}_i \\ (\mathbf{x}'', y'') \in \mathcal{D}_j}} K(\mathbf{x}', \mathbf{x}'')$$

$$K(\widetilde{\mathbf{x}}_j^c, \widehat{\mathbf{x}}) = \langle \phi(\widetilde{\mathbf{x}}_j^c), \phi(\widehat{\mathbf{x}}) \rangle = \frac{1}{n_j} \sum_{(\mathbf{x}, y) \in \mathcal{D}_j} K(\mathbf{x}, \widehat{\mathbf{x}}).$$

In other words, there is no need to construct the feature space explicitly. ■

3 Generalization Performance of Approximate SVM

3.1 Algorithmic Stability The approach of algorithmic stability has been exploited in many places, including [5,9,11,19–21,25,26]. For example, Bousquet et al. [5] proved generalization bounds for the hypotheses given by learning algorithms that are based on either Hilbert space regularization or Kullback-Leibler regularization, Poggio et al. [21,25] proved that certain type of stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. The basic idea of this approach is to study the generalization performance of a hypothesis by exploiting the robustness of the algorithm returning the hypothesis under small perturbations of the training set.

Given a training dataset $\mathcal{D}$, one possible perturbation is to replace a single datum in $\mathcal{D}$ with another datum drawn from the same distribution as the data in $\mathcal{D}$. Without losing generality, we assume that the i-th datum $\mathbf{z}_i$ in $\mathcal{D}$ is replaced by $\mathbf{z}'$ and denote the perturbed training dataset as $\mathcal{D}^i$ [5]

$$\mathcal{D} = \{\mathbf{z}_1, \ldots, \mathbf{z}_{i-1}, \mathbf{z}_i, \mathbf{z}_{i+1}, \ldots, \mathbf{z}_n\} \tag{3.5a}$$

$$\mathcal{D}^i = \{\mathbf{z}_1, \ldots, \mathbf{z}_{i-1}, \mathbf{z}', \mathbf{z}_{i+1}, \ldots, \mathbf{z}_n\}, \tag{3.5b}$$

where $\mathbf{z}', \mathbf{z}_1, \ldots, \mathbf{z}_n$ are independent and identically distributed according to the probability distribution $\mathbb{P}$ defined over $\mathcal{Z}$. We use $\mathcal{A}$ to denote a learning algorithm returning a binary classifier that performs classification by thresholding a real-valued discriminant function, i.e., for all $\mathbf{x} \in \mathcal{X}$,

$$h(\mathbf{x}) = \text{sign}(f(\mathbf{x})) \text{ and } h^i(\mathbf{x}) = \text{sign}(f^i(\mathbf{x})), \tag{3.6}$$

where h and h^i are classifiers given by $\mathcal{A}$ with $\mathcal{D}$ and $\mathcal{D}^i$ as training datasets, respectively, and f and f^i are corresponding real-valued functions. The stability of the algorithm $\mathcal{A}$ is characterized by the following *uniform replacement classification stability*[1] η_n

[1]There have been two definitions of "uniform stability" [5,26]. The definition here follows that in [26]. The definition in [5] assumes that the perturbed training dataset is obtained by deleting one datum from $\mathcal{D}$. To differentiate these two cases, we use the notion "uniform replacement classification stability" to denote the uniform stability defined in [26], and use the notion "uniform deletion classification stability" to denote the uniform stability defined in [5].

DEFINITION 3.1. *A learning algorithm $\mathcal{A}$ yielding real-valued classifiers* (3.6) *has uniform replacement classification stability η_n if*

$$\left| f(\mathbf{x}) - f^i(\mathbf{x}) \right| \leq \eta_n$$

holds for all training datasets $\mathcal{D}$ of size n, for all $i = 1, \ldots, n$, for all $\mathbf{z}' \in \mathcal{Z}$, and for all $(\mathbf{x}, y) \in \mathcal{Z}$.

For an algorithm $\mathcal{A}$ with uniform replacement classification stability η_n, the generalization performance of the resulting classifier $h = \mathcal{A}(\mathcal{D})$ is characterized by the following Lemma 3.1, whose proof is similar to that of Theorem 17 in [5] and is omitted here.

LEMMA 3.1. *Let $\mathcal{A}$ be a learning algorithm which outputs a real-valued classifier and has uniform replacement classification stability η_n. Then, for any $n \geq 1$ and $\delta \in (0, 1)$, we have, with confidence at least $1 - \delta$ over a random draw of a training dataset $\mathcal{D}$ of size n, (where the elements of $\mathcal{D}$ are drawn i.i.d. from $\mathcal{Z}$)*

$$\mathbb{E}\left[\mathbb{I}_{h(\mathbf{x}) \neq y}\right] \leq \frac{1}{n} \sum_{i=1}^{n} \ell_{\text{h}}(h, \mathbf{z}_i) + \eta_n + (2n\eta_n + 1) \sqrt{\frac{\ln 1/\delta}{2n}},$$

where $h = sign(f)$ is the classifier returned by $\mathcal{A}$, $\mathbb{I}$ is the indicator function, $\mathbb{E}\left[\mathbb{I}_{h(\mathbf{x}) \neq y}\right]$ is the expected (with respect to $\mathbb{P}$) error rate of h, and $\frac{1}{n} \sum_{i=1}^{n} \ell_{\text{h}}(h, \mathbf{z}_i)$ is the empirical hinge loss of h over $\mathcal{D}$.

3.2 Analysis of Algorithm $\mathcal{A}_{\text{approx}}$ Using algorithm $\mathcal{A}_{\text{approx}}$, for a pre-specified number k of representatives and a regularization coefficient λ, we denote the approximate SVM corresponding to the training dataset $\mathcal{D}$ as $\widetilde{h}$, which corresponds to the real-valued function $\widetilde{f}$, and denote the approximate SVM corresponding to the perturbed training dataset $\mathcal{D}^i$ as $\widetilde{h}^i$, which corresponds to the real-valued function $\widetilde{f}^i$. As shown in Algorithm 1, the real-valued functions $\widetilde{f}$ and $\widetilde{f}^i$ are

$$\widetilde{f} = \underset{f}{\text{argmin}}\, R\left(f; \widetilde{\mathcal{D}}, \lambda\right), \quad \widetilde{f}^i = \underset{f}{\text{argmin}}\, R\left(f; \widetilde{\mathcal{D}}^i, \lambda\right),$$

where $\widetilde{\mathcal{D}}$ and $\widetilde{\mathcal{D}}^i$ are the reduced training datasets corresponding to $\mathcal{D}$ and $\mathcal{D}^i$, respectively,

$$\widetilde{\mathcal{D}} = \{\widetilde{\mathbf{z}}_j = (\widetilde{\mathbf{x}}_j, \widetilde{y}_j) \mid j = 1, \ldots, n\} \tag{3.7a}$$

$$\widetilde{\mathcal{D}}^i = \{\widetilde{\mathbf{z}}_j^i = (\widetilde{\mathbf{x}}_j^i, \widetilde{y}_j^i) \mid j = 1, \ldots, n\}. \tag{3.7b}$$

We also use $\widetilde{\mathbf{w}}$ and $\widetilde{\mathbf{w}}^i$ to denote the weight vectors of $\widetilde{f}$ and $\widetilde{f}^i$, respectively.

The following Lemma 3.2 shows the uniform replacement classification stability of algorithm $\mathcal{A}_{\text{approx}}$.

LEMMA 3.2. *The algorithm $\mathcal{A}_{approx}$ has uniform replacement classification stability $\eta_n = \frac{\chi^2}{\lambda n}$, where $\chi = \sup_{\mathbf{x}\in\mathcal{X}} \sqrt{K(\mathbf{x},\mathbf{x})}$.*

Proof. The following inequality holds for all $\mathbf{z} = (\mathbf{x}, y) \in \mathcal{Z}$,

$$\left|\widetilde{f}(\mathbf{x}) - \widetilde{f}^i(\mathbf{x})\right| = \left|\langle\widetilde{\mathbf{w}}, \phi(\mathbf{x})\rangle - \langle\widetilde{\mathbf{w}}^i, \phi(\mathbf{x})\rangle\right| \le \chi \left\|\widetilde{\mathbf{w}} - \widetilde{\mathbf{w}}^i\right\|,$$

where $\chi = \sup_{\mathbf{x}\in\mathcal{X}} \sqrt{K(\mathbf{x},\mathbf{x})}$. According to Definition 3.1, it suffices to show that $\left\|\widetilde{\mathbf{w}} - \widetilde{\mathbf{w}}^i\right\| \le \frac{\chi}{\lambda n}$.

We begin with the observation that the functionals R, Ω, and R_{emp} defined in equation (2.2) are convex in f. Hence they induce Bregman divergences [27] whose properties, which are summarized in the Appendix, will be used below in some steps of the proof.

Since $\Omega(f) = \|\mathbf{w}\|^2$, we observe that $\nabla\Omega = 2\mathbf{w}$, and the corresponding induced Bregman divergence is $d_{\Omega(\cdot)}(\widetilde{f}^i, \widetilde{f}) = \|\widetilde{\mathbf{w}}^i\|^2 - \|\widetilde{\mathbf{w}}\|^2 - 2(\widetilde{\mathbf{w}}^i - \widetilde{\mathbf{w}}) \cdot \widetilde{\mathbf{w}}$. Hence a simple calculation shows

$$d_{\Omega(\cdot)}(\widetilde{f}^i, \widetilde{f}) + d_{\Omega(\cdot)}(\widetilde{f}, \widetilde{f}^i) = 2\|\widetilde{\mathbf{w}} - \widetilde{\mathbf{w}}^i\|^2. \tag{3.8}$$

In the remainder of the proof, we will show

$$\lambda\left(d_{\Omega(\cdot)}(\widetilde{f}^i, \widetilde{f}) + d_{\Omega(\cdot)}(\widetilde{f}, \widetilde{f}^i)\right) \le \frac{2\chi}{n}\left\|\widetilde{\mathbf{w}} - \widetilde{\mathbf{w}}^i\right\|, \tag{3.9}$$

which will imply

$$\left\|\widetilde{\mathbf{w}} - \widetilde{\mathbf{w}}^i\right\| \le \frac{\chi}{\lambda n},$$

thus completing the proof.

We now prove (3.9). Let $d_{R(\cdot;\widetilde{\mathcal{D}},\lambda)}(\widetilde{f}^i, \widetilde{f})$ and $d_{R(\cdot;\widetilde{\mathcal{D}}^i,\lambda)}(\widetilde{f}, \widetilde{f}^i)$ be the Bregman divergences induced by the convex functions $R(f;\widetilde{\mathcal{D}},\lambda)$ and $R(f;\widetilde{\mathcal{D}}^i,\lambda)$, respectively. Since $\widetilde{f}$ and $\widetilde{f}^i$ minimize $R(f;\widetilde{\mathcal{D}},\lambda)$ and $R(f;\widetilde{\mathcal{D}}^i,\lambda)$ respectively, using Property II of the Bregman divergence (in the Appendix), we have

$$d_{R(\cdot;\widetilde{\mathcal{D}},\lambda)}(\widetilde{f}^i, \widetilde{f}) = R(\widetilde{f}^i;\widetilde{\mathcal{D}},\lambda) - R(\widetilde{f};\widetilde{\mathcal{D}},\lambda) \tag{3.10a}$$

$$d_{R(\cdot;\widetilde{\mathcal{D}}^i,\lambda)}(\widetilde{f}, \widetilde{f}^i) = R(\widetilde{f};\widetilde{\mathcal{D}}^i,\lambda) - R(\widetilde{f}^i;\widetilde{\mathcal{D}}^i,\lambda). \tag{3.10b}$$

By the definition of R in equation (2.2a) and Property III of the Bregman divergence, we can relate d_R and $d_{R_{\text{emp}}}$

$$d_{R(\cdot;\widetilde{\mathcal{D}},\lambda)}(\widetilde{f}^i, \widetilde{f}) = d_{R_{\text{emp}}(\cdot;\widetilde{\mathcal{D}})}(\widetilde{f}^i, \widetilde{f}) + \lambda d_{\Omega(\cdot)}(\widetilde{f}^i, \widetilde{f}) \tag{3.11a}$$

$$d_{R(\cdot;\widetilde{\mathcal{D}}^i,\lambda)}(\widetilde{f}, \widetilde{f}^i) = d_{R_{\text{emp}}(\cdot;\widetilde{\mathcal{D}}^i)}(\widetilde{f}, \widetilde{f}^i) + \lambda d_{\Omega(\cdot)}(\widetilde{f}, \widetilde{f}^i). \tag{3.11b}$$

Combining equations (3.10) and (3.11) and using the non-negativity of $d_{R_{\text{emp}}}$ (Property I of the Bregman divergence), we have

$$\begin{aligned}
&\lambda\left(d_{\Omega(\cdot)}(\widetilde{f}^i, \widetilde{f}) + d_{\Omega(\cdot)}(\widetilde{f}, \widetilde{f}^i)\right) \\
&\le R(\widetilde{f}^i;\widetilde{\mathcal{D}},\lambda) - R(\widetilde{f};\widetilde{\mathcal{D}},\lambda) + R(\widetilde{f};\widetilde{\mathcal{D}}^i,\lambda) - R(\widetilde{f}^i;\widetilde{\mathcal{D}}^i,\lambda) \\
&= \frac{1}{n}\sum_{j=1}^n \ell_{\text{h}}\left(\widetilde{f}^i, \widetilde{\mathbf{z}}_j\right) - \frac{1}{n}\sum_{j=1}^n \ell_{\text{h}}\left(\widetilde{f}, \widetilde{\mathbf{z}}_j\right) \\
&\qquad + \frac{1}{n}\sum_{j=1}^n \ell_{\text{h}}\left(\widetilde{f}, \widetilde{\mathbf{z}}_j^i\right) - \frac{1}{n}\sum_{j=1}^n \ell_{\text{h}}\left(\widetilde{f}^i, \widetilde{\mathbf{z}}_j^i\right) \\
&\overset{\text{def}}{=} \boxed{\texttt{A}},
\end{aligned}$$

where the third line follows from equations (2.2).

To bound $\boxed{\texttt{A}}$, we use the following property of the hinge loss that holds for all $\mathbf{z} = (\mathbf{x}, y) \in \mathcal{Z}$

$$\begin{cases} \ell_{\text{h}}(f_1, \mathbf{z}) - \ell_{\text{h}}(f_2, \mathbf{z}) \le f_2(\mathbf{x}) - f_1(\mathbf{x}) & \text{if } y = 1, \\ \ell_{\text{h}}(f_1, \mathbf{z}) - \ell_{\text{h}}(f_2, \mathbf{z}) \le f_1(\mathbf{x}) - f_2(\mathbf{x}) & \text{if } y = -1. \end{cases}$$

We separate the terms in $\boxed{\texttt{A}}$ by the labels $\widetilde{y} = 1$ and $\widetilde{y} = -1$ to get:

$$\boxed{\texttt{A}} \le \boxed{\texttt{B}},$$

where

$$\begin{aligned}
\boxed{\texttt{B}} &\overset{\text{def}}{=} \frac{1}{n}\sum_{j=1,\widetilde{y}_j=1}^n \left(\widetilde{f}(\widetilde{\mathbf{x}}_j) - \widetilde{f}^i(\widetilde{\mathbf{x}}_j) + \widetilde{f}^i(\widetilde{\mathbf{x}}_j^i) - \widetilde{f}(\widetilde{\mathbf{x}}_j^i)\right) \\
&\quad + \frac{1}{n}\sum_{j=1,\widetilde{y}_j=-1}^n \left(\widetilde{f}^i(\widetilde{\mathbf{x}}_j) - \widetilde{f}(\widetilde{\mathbf{x}}_j) + \widetilde{f}(\widetilde{\mathbf{x}}_j^i) - \widetilde{f}^i(\widetilde{\mathbf{x}}_j^i)\right) \\
&= \frac{1}{n}\left\langle \widetilde{\mathbf{w}} - \widetilde{\mathbf{w}}^i, \underbrace{\sum_{j=1,\widetilde{y}_j=1}^n (\phi(\widetilde{\mathbf{x}}_j) - \phi(\widetilde{\mathbf{x}}_j^i))}_{\vartheta_5} \right. \\
&\qquad \left. - \underbrace{\sum_{j=1,\widetilde{y}_j=-1}^n (\phi(\widetilde{\mathbf{x}}_j) - \phi(\widetilde{\mathbf{x}}_j^i))}_{\vartheta_6} \right\rangle.
\end{aligned}$$

$$\begin{aligned}
&\le \frac{1}{n}\left\|\widetilde{\mathbf{w}} - \widetilde{\mathbf{w}}^i\right\| \|\vartheta_5 - \vartheta_6\|
\end{aligned} \tag{3.12}$$

We now show that $\|\vartheta_5 - \vartheta_6\| \le 2\chi$. Recall that the perturbed training dataset $\mathcal{D}^i$ is obtained by replacing $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ in $\mathcal{D}$ with $\mathbf{z}' = (\mathbf{x}', y')$. Both the outgoing label y_i and the incoming label y' can take on either value ± 1, giving rise to 4 possible cases. The inequality $\|\vartheta_5 - \vartheta_6\| \le 2\chi$ must be verified for each case. We show the calculation for the case $y_i = 1$ and $y' = -1$. We have

$$\begin{aligned}
\vartheta_5 &= \sum_{j=1,\widetilde{y}_j=1}^n \phi(\widetilde{\mathbf{x}}_j) - \sum_{j=1,\widetilde{y}_j=1}^n \phi(\widetilde{\mathbf{x}}_j^i) \\
&= \sum_{j=1,y_j=1}^n \phi(\mathbf{x}_j) - \left(\sum_{j=1,y_j=1}^n \phi(\mathbf{x}_j) - \phi(\mathbf{x}_i)\right) = \phi(\mathbf{x}_i)
\end{aligned}$$

$$\vartheta_6 = \sum_{\substack{j=1 \\ y_j=-1}}^{n} \phi(\mathbf{x}_j) - \left(\sum_{\substack{j=1 \\ y_j=-1}}^{n} \phi(\mathbf{x}_j) + \phi(\mathbf{x}') \right) = -\phi(\mathbf{x}'),$$

which imply that

$$\begin{aligned} &\|\vartheta_5 - \vartheta_6\| \\ &= \|\phi(\mathbf{x}_i) + \phi(\mathbf{x}')\| \le \|\phi(\mathbf{x}_i)\| + \|\phi(\mathbf{x}')\| \le 2\chi \end{aligned} \tag{3.13}$$

The other three combinations can be verified similarly.

We now use (3.13) to complete our sequence of inequalities, picking up from (3.12):

$$\begin{aligned} &\lambda \left(d_{\Omega(\cdot)}(\widetilde{f}^i, \widetilde{f}) + d_{\Omega(\cdot)}(\widetilde{f}, \widetilde{f}^i) \right) \\ &\le \boxed{\mathtt{A}} \le \boxed{\mathtt{B}} \le \frac{1}{n} \left\| \widetilde{\mathbf{w}} - \widetilde{\mathbf{w}}^i \right\| \|\vartheta_5 - \vartheta_6\| \le \frac{2\chi}{n} \left\| \widetilde{\mathbf{w}} - \widetilde{\mathbf{w}}^i \right\|, \end{aligned}$$

proving (3.9) and completing the proof. ■

Putting together Lemma 3.1 and Lemma 3.2, we arrive at our main result, namely the following Theorem 3.1 for the generalization performance of the approximate SVM $\widetilde{h}$ given by algorithm $\mathcal{A}_{\text{approx}}$.

THEOREM 3.1. *For any $n \ge 1$ and $\delta \in (0,1)$, we have, with confidence at least $1-\delta$ over a random draw of a training dataset $\mathcal{D}$ of size n,*

$$\mathbb{E}\left[\mathbb{I}_{\widetilde{h}(\mathbf{x}) \ne y}\right] \le \frac{1}{n} \sum_{\mathbf{z} \in \mathcal{D}} \ell_{\text{h}}(\widetilde{h}, \mathbf{z}) + \frac{\chi^2}{\lambda n} + \left(\frac{2\chi^2}{\lambda} + 1 \right) \sqrt{\frac{\ln 1/\delta}{2n}},$$

where $\widetilde{h}$ is the approximate SVM returned by algorithm $\mathcal{A}_{approx}$ and, as shown step 3 of Algorithm 1, it minimizes the functional $R(f; \widetilde{\mathcal{D}}, \lambda)$ (c.f. equation (2.2)), and $\widetilde{\mathcal{D}}$ is the reduced training dataset given by step 2 of Algorithm 1. Furthermore, $\mathbb{E}\left[\mathbb{I}_{\widetilde{h}(\mathbf{x}) \ne y}\right]$ is the expected error rate of $\widetilde{h}$, and $\frac{1}{n}\sum_{\mathbf{z} \in \mathcal{D}} \ell_{\text{h}}(\widetilde{h}, \mathbf{z})$ is the empirical hinge loss of $\widetilde{h}$ over the original training dataset $\mathcal{D}$. All other quantities are defined as in Lemma 3.2.

4 Discussions

Except requiring the data of the same cluster belong to the same class, the assumptions in Algorithm 1 put no restrictions on how the training dataset $\mathcal{D}$ is partitioned. Thus, Theorem 3.1 applies regardless of the way $\mathcal{D}$ is partitioned, as long as the representatives satisfy the three assumptions in Algorithm 1. In other words, one can use any clustering algorithm preferred, such as kernel k-means or even random partitioning, to partition $\mathcal{D}$ in Algorithm 1 and, like other generalization bounds, Theorem 3.1 guarantees that the resulting approximate SVM will very likely to have a small expected error rate if its empirical performance over $\mathcal{D}$ is good.

Theorem 3.1 thus provides a theoretical justification for using approximate SVMs in practice. Cao et al. [7] showed that, in order for the approximate SVM to agree fairly closely with the exact SVM, kernel k-means [10, 13] should be used to partition $\mathcal{D}$. However, kernel k-means often suffers from the problem of multiple local minima, and may yield different partitions for different initializations. Theorem 3.1 suggests that all these adverse phenomena can be safely ignored and the resulting approximate SVM $\widetilde{h}$ will very likely have a small expected error rate if it has a small empirical loss over the dataset $\mathcal{D}$. Furthermore, since kernel k-means is an expensive procedure, to speed up the training process further, Cao et al. [7] proposed to partition $\mathcal{D}$ in the space $\mathcal{X}$ using a more efficient clustering algorithm called Principal Direction Divisive Partition [2], which was found experimentally to be effective. Theorem 3.1 provides a theoretical justification for this strategy.

Theorem 3.1 includes as a special case the generalization bound for the exact SVM proved in [5], which was also proved by using the approach of algorithmic stability. The reason is that the approximate SVM $\widetilde{h}$ given by algorithm $\mathcal{A}_{\text{approx}}$ will be the same as the exact SVM if $k = n$ and each cluster contains exactly one training datum. However, the proof in [5] can not be directly used to prove Theorem 3.1 because $\widetilde{\mathcal{D}}$ and $\widetilde{\mathcal{D}}^i$ corresponding to approximate SVMs could be completely different, while $\mathcal{D}$ and $\mathcal{D}^i$ corresponding to exact SVMs only differs by a single datum according to [5].

Finally, the bound in Theorem 3.1 is different from and generally tighter than the bound that would be obtained based on VC dimension (see e.g. [15]).

5 Conclusion

We proved in this paper a PAC-style generalization bound for approximate SVMs, which have been proposed to speed up the training process of SVM. The bound provides a theoretical justification for using approximate SVMs in practice, and generalizes the generalization bound for exact SVMs. One future direction is to develop a generalization bound that explicitly takes into the account the fact that some specific algorithm, such as kernel k-means, is used to partition $\mathcal{D}$. The resulting bound is expected to be tighter than Theorem 3.1, although its applicability may be less general. The intuition here is that, taking kernel k-means as an example, the set of representatives would change a rather small amount when the training dataset is perturbed in the way described in equation (3.5), i.e., the corresponding stability η_n is expected to be less than $\chi^2/(\lambda n)$ (c.f. Lemma 3.2).

Appendix: Bregman Divergence

DEFINITION 5.1. (BREGMAN DIVERGENCE [27]) *Let $J(f)$ be a strictly convex, differentiable real-valued function of f. The Bregman divergence $d_J(f_1, f_2)$ induced by J is*

$$d_J(f_1, f_2) = J(f_1) - J(f_2) - \langle f_1 - f_2, \nabla J(f_2) \rangle .$$

The Bregman divergence has following properties.

- **Property I**: $d_J(f_1, f_2) \geq 0$ for all f_1, f_2;
- **Property II**: If f^* is the minimizer of $J(f)$, then $d_J(f', f^*) = J(f') - J(f^*)$ for all f';
- **Property III**: If $J_\mathrm{a}, J_\mathrm{b}$ are two strictly convex, differentiable real-valued functions, then so is $J_\mathrm{a}+J_\mathrm{b}$, and its Bregman divergence is $d_{J_\mathrm{a}+J_\mathrm{b}}(f_1, f_2) = d_{J_\mathrm{a}}(f_1, f_2) + d_{J_\mathrm{b}}(f_1, f_2)$.

References

[1] D. Achlioptas, F. McSherry, and B. Schölkopf. Sampling techniques for kernel methods. In S. B. Thomas, G. Dietterich, and Z. Ghahramani, editors, *NIPS 14*, pages 335–342, 2002.

[2] D. L. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[3] D. L. Boley and D. Cao. Training support vector machine using adaptive clustering. In *SIAMDM*, pages 126–137, 2004.

[4] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, pages 144–152, 1992.

[5] O. Bousquet and A. Elisseeff. Stability and generalization. *JMLR*, 2:499–526, 2002.

[6] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[7] D. Cao and D. L. Boley. On approximate solutions to support vector machines. In *SIAMDM*, 2006.

[8] C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

[9] L. Devroye and T. J. Wagner. Distribution-free performance bounds for potential function rules. *IEEE Trans. on Information Theory*, 25(5):601–604, 1979.

[10] I. Dhillon, Y. Guan, and B. Kulis. Kernel k-means, spectral clustering and normalized cuts. In *KDD*, pages 551–556, 2004.

[11] A. Elisseeff, T. Evgeniou, and M. Pontil. Stability of randomized learning algorithms. *JMLR*, 6:55–79, 2005.

[12] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *JMLR*, 2:243–264, 2001.

[13] M. Girolami. Mercer kernel-based clustering in feature space. *IEEE Trans. on Neural Networks*, 13(3):780–784, May 2002.

[14] T. Graepel, R. Herbrich, and J. Shawe-Taylor. Generalisation error bounds for sparse linear classifiers. In *COLT*, pages 298–303, 2000.

[15] R. Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press, 2002.

[16] R. Herbrich and T. Graepel. A PAC-Bayesian margin bound for linear classifiers: Why SVMs work. In *NIPS*, pages 224–230, 2001.

[17] T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 169–184. MIT Press, 1999.

[18] T. Joachims. Training linear SVMs in linear time. In *KDD*, pages 217–226, 2006.

[19] M. Kearns and D. Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross validation. *Neural Computation*, 11(6):1427–1453, August 1999.

[20] S. Kutin and P. Niyogi. Almost-everywhere algorithmic stability and generalization error. In *UAI*, pages 275–282, 2002.

[21] S. Mukherjee, P. Niyogi, T. Poggio, and R. Rifkin. Statistical learning: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. Technical Report AI Memo 2002-024, MIT, 2002. Revised 2003.

[22] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In A. Island, editor, *IEEE Neural Networks for Signal Processing VII Workshop*, pages 276–285, 1997.

[23] D. Pavlov, D. Chudova, and P. Smyth. Towards scalable support vector machines using squashing. In *KDD*, pages 295–299, 2000.

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

[25] T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi. General conditions for predictivity in learning theory. *Nature*, 428:419–422, March 2004.

[26] R. M. Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning*. PhD thesis, MIT, September 2002.

[27] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970. Reprint in 1997.

[28] V. N. Vapnik. *Statistical Learning Theory*. Wiley, NY, 1998.

[29] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *NIPS 13*, pages 682–688, 2001.

[30] C. Yang, R. Duraiswami, and L. Davis. Efficient kernel machines using fast Gauss transform. In *NIPS 17*, 2005.

[31] H. Yu, J. Yang, and J. Han. Classifying large data sets using SVM with hierarchical clusters. In *KDD*, pages 306–315, 2003.