Infinite Ensemble Learning with Support Vector Machines

Hsuan-Tien Lin and Ling Li

Learning Systems Group, California Institute of Technology, USA
htlin@caltech.edu, ling@caltech.edu

Abstract. Ensemble learning algorithms such as boosting can achieve better performance by averaging over the predictions of base hypotheses. However, existing algorithms are limited to combining only a finite number of hypotheses, and the generated ensemble is usually sparse. It is not clear whether we should construct an ensemble classifier with a larger or even infinite number of hypotheses. In addition, constructing an infinite ensemble itself is a challenging task. In this paper, we formulate an infinite ensemble learning framework based on SVM. The framework can output an infinite and nonsparse ensemble, and can be used to construct new kernels for SVM as well as to interpret some existing ones. We demonstrate the framework with a concrete application, the stump kernel, which embodies infinitely many decision stumps. The stump kernel is simple, yet powerful. Experimental results show that SVM with the stump kernel is usually superior than boosting, even with noisy data.

1 Introduction

Ensemble learning algorithms, such as boosting [1], are successful in practice. They construct a classifier that averages over some base hypotheses in a set $\mathcal{H}$. While the size of $\mathcal{H}$ can be infinite in theory, existing algorithms can utilize only a small finite subset of $\mathcal{H}$, and the classifier is effectively a finite ensemble of hypotheses. On the one hand, the classifier is a regularized approximation to the optimal one (see Subsection 2.2), and hence may be less vulnerable to overfitting [2]. On the other hand, it is limited in capacity [3], and may not be powerful enough. Thus, it is unclear whether an infinite ensemble would be superior for learning. In addition, it is a challenging task to construct an infinite ensemble of hypotheses [4].

The goal of this paper is to conquer the task of infinite ensemble learning in order to see if an infinite ensemble could achieve better performance. We formulate a framework for infinite ensemble learning based on the support vector machine (SVM) [4]. The key is to embed an infinite number of hypotheses into an SVM kernel. Such a framework can be applied both to construct new kernels for SVM, and to interpret some existing ones [5]. Furthermore, the framework allows a fair comparison between SVM and ensemble learning algorithms.

As a concrete application of the framework, we introduce the stump kernel, which embodies an infinite number of decision stumps. The stump kernel is novel

J. Gama et al. (Eds.): ECML 2005, LNAI 3720, pp. 242–254, 2005.

and is simpler than most existing kernels for SVM. Somehow it is powerful both in theory and in practice. Experimental results show that with the stump kernel, our framework usually achieves better performance than popular ensemble learning algorithms. Our results also bring in some important insights for both SVM and ensemble learning.

The paper is organized as follows. In Section 2, we show the connections between SVM and the ensemble learning. Next in Section 3, we propose the framework for embedding an infinite number of hypotheses into the kernel. We then present the stump kernel in Section 4. Finally, we show the experimental results in Section 5, and conclude in Section 6.

2 SVM and Ensemble Learning

2.1 Support Vector Machine

Given a training set $\{(x_i, y_i)\}_{i=1}^N$, which contains input vectors $x_i \in \mathcal{X} \subseteq \mathbb{R}^D$ and their corresponding labels $y_i \in \{-1, +1\}$, the soft-margin SVM [4] constructs a classifier $g(x) = \operatorname{sign}(\langle w, \phi_x \rangle + b)$ from the optimal solution to the following problem:[1]

$$(P_1) \quad \min_{w \in \mathcal{F}, b \in \mathbb{R}, \xi \in \mathbb{R}^N} \ \frac{1}{2} \langle w, w \rangle + C \sum_{i=1}^N \xi_i$$
$$\text{s.t.} \ \ y_i(\langle w, \phi_{x_i} \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

Here $C > 0$ is the regularization parameter, and $\phi_x = \Phi(x)$ is obtained from the feature mapping $\Phi\colon \mathcal{X} \to \mathcal{F}$. We assume the feature space $\mathcal{F}$ to be a Hilbert space equipped with the inner product $\langle \cdot, \cdot \rangle$ [6]. Because $\mathcal{F}$ can be of an infinite number of dimensions, SVM solvers usually work on the dual problem:

$$(P_2) \quad \min_{\lambda \in \mathbb{R}^N} \ \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j \mathcal{K}(x_i, x_j) - \sum_{i=1}^N \lambda_i$$
$$\text{s.t.} \ \sum_{i=1}^N y_i \lambda_i = 0, \quad 0 \leq \lambda_i \leq C.$$

Here $\mathcal{K}$ is the kernel function defined as $\mathcal{K}(x, x') = \langle \phi_x, \phi_{x'} \rangle$. Then, the optimal classifier becomes

$$g(x) = \operatorname{sign}\left(\sum_{i=1}^N y_i \lambda_i \mathcal{K}(x_i, x) + b \right), \tag{1}$$

where b can be computed through the primal-dual relationship [4, 6].

The use of a kernel function $\mathcal{K}$ instead of computing the inner product directly in $\mathcal{F}$ is called the kernel trick, which works when $\mathcal{K}(\cdot, \cdot)$ can be computed

[1] $\operatorname{sign}(\theta)$ is 1 when θ is nonnegative, -1 otherwise.

efficiently. Alternatively, we can begin with an arbitrary $\mathcal{K}$, and check whether there exist a space $\mathcal{F}$ and a mapping Φ such that $\mathcal{K}(\cdot,\cdot)$ is a valid inner product in $\mathcal{F}$. A key tool here is the Mercer's condition, which states that a symmetric $\mathcal{K}(\cdot,\cdot)$ is a valid inner product if and only if its Gram matrix K, defined by $K_{i,j} = \mathcal{K}(x_i, x_j)$, is always positive semi-definite (PSD) [4, 6].

The soft-margin SVM originates from the hard-margin SVM, where the margin violations ξ_i are forced to be zero. This can be achieved by setting the regularization parameter $C \to \infty$ in (P_1) and (P_2).

2.2 Adaptive Boosting

Adaptive boosting (AdaBoost) [1] is perhaps the most popular and successful algorithm for ensemble learning. For a given integer T and a hypothesis set $\mathcal{H}$, AdaBoost iteratively selects T hypotheses $h_t \in \mathcal{H}$ and weights $w_t \geq 0$ to construct an ensemble classifier

$$g(x) = \operatorname{sign}\left(\sum_{t=1}^{T} w_t h_t(x)\right).$$

Under some assumptions, it is shown that when $T \to \infty$, AdaBoost asymptotically approximates an infinite ensemble classifier $\operatorname{sign}(\sum_{t=1}^{\infty} w_t h_t(x))$ [7], such that (w, h) is an optimal solution to

$$(P_3) \quad \min_{w_t \in \mathbb{R}, h_t \in \mathcal{H}} \|w\|_1 \quad \text{s.t. } y_i \left(\sum_{t=1}^{\infty} w_t h_t(x_i)\right) \geq 1, \quad w_t \geq 0.$$

The problem (P_3) has infinitely many variables. In order to approximate the optimal solution well with a fixed T, AdaBoost has to resort to two related properties of the optimal solutions for (P_3). First, when two hypotheses have the same prediction patterns on the training vectors, they can be used interchangeably in constructing an ensemble, and are thus called "ambiguous". Since there are at most 2^N prediction patterns on N input vectors, we can partition $\mathcal{H}$ into at most 2^N groups, each containing mutually ambiguous hypotheses. Some optimal solutions of (P_3) only assign one or a few nonzero weights within each group [8]. Thus, it is possible to work on a finite subset of $\mathcal{H}$ instead of $\mathcal{H}$ itself without losing optimality.

Second, minimizing the ℓ_1-norm $\|w\|_1$ often leads to sparse solutions [2, 9]. That is, for hypotheses in the finite (but possibly still large) subset of $\mathcal{H}$, only a small number of weights needs to be nonzero. Many ensemble learning algorithms, including AdaBoost, try to find or approximate such a finite and sparse ensemble. However, it is not clear whether the performance could further be improved if either or both the finiteness and the sparsity restrictions are removed.[2]

[2] Qualitatively, sparsity is algorithm-dependent and more restricted than finiteness.

2.3 Connecting SVM to Ensemble Learning

SVM and AdaBoost are related. Consider the feature transform

$$\Phi(x) = (h_1(x), h_2(x), \dots). \tag{2}$$

We can clearly see that the problem (P_1) with this feature transform is similar to (P_3). The elements of ϕ_x in SVM and the hypotheses $h_t(x)$ in AdaBoost play similar roles. They both work on linear combinations of these elements, though SVM has an additional intercept term b. SVM minimizes the ℓ_2-norm of the weights while AdaBoost approximately minimizes the ℓ_1-norm. Note that AdaBoost requires $w_t \geq 0$ for ensemble learning.

Another difference is that for regularization, SVM introduces slack variables ξ_i, while AdaBoost relies on the choice of a finite T [2]. Note that we can also introduce proper slack variables to (P_3) and solve it by the linear programming boosting method [8]. In the scope of this paper, however, we shall focus only on AdaBoost.

The connection between SVM and AdaBoost is well known in literature [10]. Several researchers have developed interesting results based on the connection [2,7]. However, as limited as AdaBoost, previous results could utilize only a finite subset of $\mathcal{H}$ when constructing the feature mapping (2). One reason is that the infinite number of variables w_t and constraints $w_t \geq 0$ are difficult to handle. We will further illustrate these difficulties and our remedies in the next section.

3 SVM-Based Framework for Infinite Ensemble Learning

Vapnik [4] proposed a challenging task of designing an algorithm that actually generates an infinite ensemble classifier. Traditional algorithms like AdaBoost cannot be directly generalized to solve this problem, because they select the hypotheses in an iterative manner, and only run for finite number of iterations.

The connection between SVM and ensemble learning shows another possible approach. We can formulate a kernel that embodies all the hypotheses in $\mathcal{H}$. Then, the classifier (1) obtained from SVM with this kernel is a linear combination of those hypotheses (with an intercept term). However, there are still two main obstacles. One is to actually derive the kernel, and the other is to handle the constraints $w_t \geq 0$ to make (1) an ensemble classifier. In this section, we integrate several ideas to deal with these obstacles, and propose a framework of infinite ensemble learning based on SVM.

3.1 Embedding Hypotheses into the Kernel

We start by embedding the infinite number of hypotheses in $\mathcal{H}$ into an SVM kernel. We have shown in (2) that we could construct a feature mapping from $\mathcal{H}$. In Definition 1, we extend this idea to a more general form, and define a kernel based on the feature mapping.

Definition 1. *Assume that $\mathcal{H} = \{h_\alpha \colon \alpha \in \mathcal{C}\}$, where $\mathcal{C}$ is a measure space. The kernel that embodies $\mathcal{H}$ is defined as*

$$\mathcal{K}_{\mathcal{H},r}(x, x') = \int_{\mathcal{C}} \phi_x(\alpha)\phi_{x'}(\alpha)\, d\alpha, \tag{3}$$

where $\phi_x(\alpha) = r(\alpha)h_\alpha(x)$, and $r \colon \mathcal{C} \to \mathbb{R}^+$ is chosen such that the integral exists for all $x, x' \in \mathcal{X}$.

Here, α is the parameter of the hypothesis h_α. Although two hypotheses with different α values may have the same input-output relation, we would treat them as different objects in our framework. We shall denote $\mathcal{K}_{\mathcal{H},r}$ by $\mathcal{K}_{\mathcal{H}}$ when r is clear from the context.

If $\mathcal{C}$ is a closed interval $[L, R]$, the right-hand-side of (3) is obviously an inner product [6], and hence Definition 1 constructs a valid kernel. In the following theorem, the validity is formalized for a general $\mathcal{C}$.

Theorem 1. *Consider the kernel $\mathcal{K}_{\mathcal{H}} = \mathcal{K}_{\mathcal{H},r}$ in Definition 1.*

1. *The kernel is an inner product for ϕ_x and $\phi_{x'}$ in the Hilbert space $\mathcal{F} = \mathcal{L}_2(\mathcal{C})$, which contains functions $\varphi(\cdot) \colon \mathcal{C} \to \mathbb{R}$ that are square integrable.*
2. *For a set of input vectors $\{x_i\}_{i=1}^N \in \mathcal{X}^N$, the Gram matrix of $\mathcal{K}$ is PSD.*

Proof. The first part is in function analysis [11], and the second part follows Mercer's condition. □

The technique of constructing kernels from an integral inner product is known in literature [6]. Our framework utilizes this technique for embedding the hypotheses, and thus could handle the situation even when $\mathcal{H}$ is uncountable.

When we use $\mathcal{K}_{\mathcal{H}}$ in (P_2), the primal problem (P_1) becomes

$$(P_4) \quad \min_{w \in \mathcal{L}_2(\mathcal{C}), b \in \mathbb{R}, \xi \in \mathbb{R}^N} \frac{1}{2} \int_{\mathcal{C}} w^2(\alpha)\, d\alpha + C \sum_{i=1}^N \xi_i$$
$$\text{s.t. } y_i \left(\int_{\mathcal{C}} w(\alpha) r(\alpha) h_\alpha(x_i)\, d\alpha + b \right) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

In particular, the classifier obtained after solving (P_2) with $\mathcal{K}_{\mathcal{H}}$ is the same as the classifier obtained after solving (P_4):

$$g(x) = \operatorname{sign}\left(\int_{\mathcal{C}} w(\alpha) r(\alpha) h_\alpha(x)\, d\alpha + b \right). \tag{4}$$

When $\mathcal{C}$ is uncountable, it is possible that each hypothesis h_α only takes an infinitesimal weight $w(\alpha)r(\alpha)\, d\alpha$ in the ensemble. This is very different from the situation in traditional ensemble learning, and will be discussed further in Subsection 4.3.

3.2 Negation Completeness and Constant Hypotheses

Note that (4) is not an ensemble classifier yet, because we do not have the constraints $w(\alpha) \geq 0$, and we have an additional term b. Next, we would explain that (4) is equivalent to an ensemble classifier under some reasonable assumptions.

We start from the constraints $w(\alpha) \geq 0$, which cannot be directly considered in (P_1). It has been shown that even if we add a countably infinite number of constraints to (P_1), we introduce infinitely many variables and constraints in (P_2), which makes the later problem difficult to solve [4].

One remedy is to assume that $\mathcal{H}$ is negation complete, that is, $h \in \mathcal{H}$ if and only if $(-h) \in \mathcal{H}$.[3] Then, every linear combination over $\mathcal{H}$ has an equivalent linear combination with only nonnegative weights. Negation completeness is usually a mild assumption for a reasonable $\mathcal{H}$. Following this assumption, the classifier (4) can be interpreted as an ensemble classifier over $\mathcal{H}$ with an intercept term b. Somehow b can be viewed as the weight on a constant hypothesis c.[4] We shall further add a mild assumption that $\mathcal{H}$ contains both c and $(-c)$, which makes $g(\cdot)$ in (4) or (1) indeed equivalent to an ensemble classifier.

We summarize our framework in Fig. 1. The framework shall generally inherit the profound performance of SVM. Most of the steps in the framework could be done by existing SVM algorithms, and the hard part is mostly in obtaining the kernel $\mathcal{K}_{\mathcal{H}}$. We have derived several useful kernels with the framework [5]. In the next section, we demonstrate one concrete instance of those kernels.

---

1. Consider a training set $\{(x_i, y_i)\}_{i=1}^N$ and the hypothesis set $\mathcal{H}$, which is assumed to be negation complete and to contain a constant hypothesis.
2. Construct a kernel $\mathcal{K}_{\mathcal{H}}$ according to Definition 1 with a proper r.
3. Choose proper parameters, such as the soft-margin parameter C.
4. Solve (P_2) with $\mathcal{K}_{\mathcal{H}}$ and obtain Lagrange multipliers λ_i and the intercept term b.
5. Output the classifier $g(x) = \text{sign}\left(\sum_{i=1}^N y_i \lambda_i \mathcal{K}_{\mathcal{H}}(x_i, x) + b\right)$, which is equivalent to some ensemble classifier over $\mathcal{H}$.

Fig. 1. Steps of the SVM-based framework for infinite ensemble learning.

4 Stump Kernel

In this section, we present the stump kernel, which embodies infinitely many decision stumps, as a concrete application of our framework. The decision stump $s_{q,d,\alpha}(x) = q \cdot \text{sign}\big((x)_d - \alpha\big)$ works on the d-th element of x, and classifies x according to $q \in \{-1, +1\}$ and the threshold α [12]. It is widely used for ensemble learning because of its simplicity [1].

[3] We use $(-h)$ to denote the function $(-h)(\cdot) = -(h(\cdot))$.
[4] A constant hypothesis $c(\cdot)$ predicts $c(x) = 1$ for all $x \in \mathcal{X}$.

4.1 Formulation

To construct the stump kernel, we consider the following set of decision stumps

$$\mathcal{S} = \left\{ s_{q,d,\alpha_d} \colon q \in \{-1,+1\}, d \in \{1,\ldots,D\}, \alpha_d \in [L_d, R_d] \right\}.$$

In addition, we assume that $\mathcal{X} \subseteq [L_1, R_1] \times [L_2, R_2] \times \cdots \times [L_D, R_D]$. Then, $\mathcal{S}$ is negation complete, and contains $s_{+1,1,L_1}(\cdot)$ as a constant hypothesis. Thus, the stump kernel $\mathcal{K}_{\mathcal{S}}$ defined below can be used in our framework (Fig. 1) to obtain an infinite ensemble of decision stumps.

Definition 2. *The stump kernel $\mathcal{K}_{\mathcal{S}}$ is defined as in Definition 1 for the set $\mathcal{S}$ with $r(q,d,\alpha_d) = \frac{1}{2}$,*

$$\mathcal{K}_{\mathcal{S}}(x,x') = \Delta_{\mathcal{S}} - \sum_{d=1}^{D} \left| (x)_d - (x')_d \right| = \Delta_{\mathcal{S}} - \left\| x - x' \right\|_1,$$

where $\Delta_{\mathcal{S}} = \frac{1}{2} \sum_{d=1}^{D} (R_d - L_d)$ is a constant.

To obtain the stump kernel in Definition 2, we separate the integral (3) into two parts: stumps having the same outputs on x and x', and stumps having different outputs on x and x'. Both parts exist and are easy to compute when we simply assign a constant r to all $r(q,d,\alpha_d)$. Note that scaling r is equivalent to scaling the parameter C in SVM. Thus, without loss of generality, we choose $r = \frac{1}{2}$ to obtain a cosmetically cleaner kernel function.

Following Theorem 1, the stump kernel produces a PSD Gram matrix for $x_i \in \mathcal{X}$. Given the ranges $[L_d, R_d]$, the stump kernel is very simple to compute. In fact, the ranges are even not necessary in general, because dropping the constant $\Delta_{\mathcal{S}}$ does not affect the classifier obtained from SVM:

Theorem 2. *Solving (P_2) with $\mathcal{K}_{\mathcal{S}}$ is the same as solving (P_2) with the simplified stump kernel $\tilde{\mathcal{K}}_{\mathcal{S}}(x,x') = -\left\| x - x' \right\|_1$. That is, they obtain equivalent classifiers in (1).*

Proof. We extend from [13] to show that $\tilde{\mathcal{K}}_{\mathcal{S}}(x,x')$ is conditionally PSD (CPSD). In addition, a CPSD kernel $\tilde{\mathcal{K}}(x,x')$ works exactly the same for (P_2) as any PSD kernel of the form $\tilde{\mathcal{K}}(x,x') + \Delta$, where Δ is a constant, because of the linear constraint $\sum_{i=1}^{N} y_i \lambda_i = 0$ [6,14]. The proof follows with $\Delta = \Delta_{\mathcal{S}}$. □

Although the simplified stump kernel is simple to compute, it provides comparable classification ability for SVM, as shown below.

4.2 Power of the Stump Kernel

The classification ability of the stump kernel comes from the following positive definite (PD) property under some mild assumptions:

Theorem 3. *Consider input vectors* $\{x_i\}_{i=1}^N \in \mathcal{X}^N$*. If there exists a dimension* d *such that* $(x_i)_d \in (L_d, R_d)$ *and* $(x_i)_d \neq (x_j)_d$ *for all* $i \neq j$*, the Gram matrix of* $\mathcal{K}_\mathcal{S}$ *is PD.*

Proof. See [5] for details. □

The PD-ness of the Gram matrix is directly connected to the classification power of the SVM classifiers. Chang and Lin [15] show that when the Gram matrix of the kernel is PD, a hard-margin SVM with such kernel can always dichotomize the training vectors. Thus, Theorem 3 implies:

Theorem 4. *The class of SVM classifiers with* $\mathcal{K}_\mathcal{S}$*, or equivalently, the class of infinite ensemble classifiers over* $\mathcal{S}$*, has an infinite V-C dimension.*

Theorem 4 indicates the power of the stump kernel. A famous kernel that also provides infinite power to SVM is the Gaussian kernel [16]. The theorem shows that the stump kernel has theoretically almost the same power as the Gaussian kernel. Note that such power needs to be controlled with care because the power of fitting any data can also be abused to fit noise. For the Gaussian kernel, it has been observed that soft-margin SVM with suitable parameter selection can regularize the power and achieve good generalization performance even in the presence of noise [16, 17]. The stump kernel, which is similar to the Gaussian kernel, also has such property when used in soft-margin SVM. We shall further demonstrate this property experimentally in Section 5.

4.3 Averaging Ambiguous Stumps

We have shown in Subsection 2.2 that the set of hypotheses can be partitioned into groups and traditional ensemble learning algorithms can only pick a few representatives within each group. Our framework acts in a different way: the ℓ_2-norm objective function of SVM leads to an optimal solution that combines the average predictions of each group. In other words, the consensus output of each group is the average prediction of all hypotheses in the group rather than the predictions of a few selected ones. The averaging process constructs a smooth representative for each group. In the following theorem, we shall demonstrate this with our stump kernel, and show how the decision stumps group together in the final ensemble classifier.

Theorem 5. *Define* $(\tilde{x})_{d,a}$ *as the a-th smallest value in* $\{(x_i)_d\}_{i=1}^N$*, and* A_d *as the number of different* $(\tilde{x})_{d,a}$*. Let* $(\tilde{x})_{d,0} = L_d$*,* $(\tilde{x})_{d,(A_d+1)} = R_d$*, and*

$$\hat{s}_{q,d,a}(x) = q \cdot \begin{cases} 1, & \text{when } (x)_d \geq (\tilde{x})_{d,t+1}; \\ -1, & \text{when } (x)_d \leq (\tilde{x})_{d,t}; \\ \frac{2(x)_d - (\tilde{x})_{d,a} - (\tilde{x})_{d,a+1}}{(\tilde{x})_{d,a+1} - (\tilde{x})_{d,a}}, & \text{otherwise.} \end{cases}$$

Then, for $r(q,d,a) = \frac{1}{2}\sqrt{(\tilde{x})_{d,a+1} - (\tilde{x})_{d,a}}$*,*

$$K_\mathcal{S}(x, x') = \sum_{q \in \{-1,+1\}} \sum_{d=1}^{D} \sum_{a=0}^{A_d} r^2(q,d,a)\hat{s}_{q,d,a}(x)\hat{s}_{q,d,a}(x').$$

We can prove Theorem 5 by carefully writing down the equations. Note that the function $\hat{s}_{q,d,t}(\cdot)$ is a smoother variant of the decision stump. Each $\hat{s}_{q,d,t}(\cdot)$ represents the group of ambiguous decision stumps with $\alpha_d \in ((\tilde{x})_{d,t}, (\tilde{x})_{d,t+1})$. When the group is larger, $\hat{s}_{q,d,t}(\cdot)$ is smoother because it is the average over more decision stumps. Traditional ensemble learning algorithms like AdaBoost usually consider the middle stump $m_{q,d,t}(\cdot)$, which has threshold at the mean of $(\tilde{x})_{d,t}$ and $(\tilde{x})_{d,t+1}$, as the only representative of the group. Our framework, on the other hand, enjoys a smoother decision by averaging over more decision stumps. Even though each decision stump only has an infinitesimal hypothesis weight, the averaged stump $\hat{s}_{q,d,t}(\cdot)$ could have a concrete weight in the ensemble, which explains how the infinitesimal weights work.

5 Experiments

We test and compare several ensemble learning algorithms, including our framework with the stump kernel, on various datasets.

The first algorithm we test is our framework with the simplified stump kernel, denoted as SVM-Stump. It is compared with AdaBoost-Stump, AdaBoost with decision stumps as base hypotheses. A common implementation of AdaBoost-Stump only chooses the middle stumps (see Subsection 4.3). For further comparison, we take the set of middle stumps $\mathcal{M}$, and construct a kernel $\mathcal{K}_{\mathcal{M}}$ with $r = \frac{1}{2}$ according to Definition 1. Because $\mathcal{M}$ is a finite set, the integral in (3) becomes a summation when computed with the counting measure. We test our framework with this kernel, and call it SVM-Mid. We also compare SVM-Stump with SVM-Gauss, which is SVM with the Gaussian kernel. For AdaBoost-Stump, we demonstrate the results using $T = 100$ and $T = 1000$. For SVM algorithms, we use LIBSVM [18] with the general procedure of soft-margin SVM [17], which selects a suitable parameter with cross validation before actual training.

The three artificial datasets from Breiman [19] (twonorm, threenorm, and ringnorm) are used with training set size 300 and test set size 3000. We create three more datasets (twonorm-n, threenorm-n, ringnorm-n), which contain mislabeling noise on 10% of the training examples, to test the performance of the algorithms on noisy data. We also use eight real-world datasets from the UCI repository [20]: australian, breast, german, heart, ionosphere, pima, sonar, and votes84. Their feature elements are normalized to $[-1, 1]$. We randomly pick 60% of the examples for training, and the rest for testing. All the results are averaged over 100 runs, presented with standard error bar.

5.1 Comparison of Ensemble Learning Algorithms

Table 1 shows the test performance of our framework and traditional ensemble learning algorithms. We can see that SVM-Stump is usually the best of the four algorithms, and also has superior performance even in the presence of noise. That is, SVM-Stump performs significantly better than AdaBoost-Stump. Not surprisingly, SVM-Stump also performs better than SVM-Mid. These results demonstrate that it is beneficial to go from a finite ensemble to an infinite one.

Table 1. Test error (%) of several ensemble learning algorithms

dataset	SVM-Stump	SVM-Mid	AdaBoost-Stump $T = 100$	AdaBoost-Stump $T = 1000$
twonorm	$\mathbf{2.86 \pm 0.04}$	3.10 ± 0.04	5.06 ± 0.06	4.97 ± 0.06
twonorm-n	$\mathbf{3.08 \pm 0.06}$	3.29 ± 0.05	12.6 ± 0.14	15.5 ± 0.17
threenorm	$\mathbf{17.7 \pm 0.10}$	18.6 ± 0.12	21.8 ± 0.09	22.9 ± 0.12
threenorm-n	$\mathbf{19.0 \pm 0.14}$	19.6 ± 0.13	25.9 ± 0.13	28.2 ± 0.14
ringnorm	$\mathbf{3.97 \pm 0.07}$	5.30 ± 0.07	12.2 ± 0.13	9.95 ± 0.14
ringnorm-n	$\mathbf{5.56 \pm 0.11}$	7.03 ± 0.14	19.4 ± 0.20	20.3 ± 0.19
australian	$\mathbf{14.5 \pm 0.21}$	15.9 ± 0.18	$\mathbf{14.7 \pm 0.18}$	16.9 ± 0.18
breast	3.11 ± 0.08	$\mathbf{2.77 \pm 0.08}$	4.27 ± 0.11	4.51 ± 0.11
german	$\mathbf{24.7 \pm 0.18}$	$\mathbf{24.9 \pm 0.17}$	$\mathbf{25.0 \pm 0.18}$	26.9 ± 0.18
heart	$\mathbf{16.4 \pm 0.27}$	19.1 ± 0.35	19.9 ± 0.36	22.6 ± 0.39
ionosphere	$\mathbf{8.13 \pm 0.17}$	$\mathbf{8.37 \pm 0.20}$	11.0 ± 0.23	11.0 ± 0.25
pima	$\mathbf{24.2 \pm 0.23}$	$\mathbf{24.4 \pm 0.23}$	24.8 ± 0.22	27.0 ± 0.25
sonar	$\mathbf{16.6 \pm 0.42}$	18.0 ± 0.37	19.0 ± 0.37	19.0 ± 0.35
votes84	4.76 ± 0.14	4.76 ± 0.14	$\mathbf{4.07 \pm 0.14}$	5.29 ± 0.15

(results that are as significant as the best ones are marked in bold)

The three algorithms, AdaBoost-Stump, SVM-Mid, and SVM-Stump, generate three different kinds of ensembles. AdaBoost-Stump produces finite and sparse ensembles, SVM-Mid produces finite but nonsparse ensembles, and SVM-Stump produces infinite and nonsparse ensembles. Interestingly, SVM-Mid often performs better than AdaBoost-Stump, too. This indicates that a nonsparse ensemble, introduced by the ℓ_2-norm objective function, may be better than a sparse one. We further illustrate this by a simplified experiment. In Fig. 2 we show the decision boundaries generated by the three algorithms on 300 training examples from the 2-D version of the twonorm dataset. AdaBoost-Stump performs similarly with $T = 100$ or $T = 1000$. Hence only the former is shown. The Bayes optimal decision boundary is the line $(x)_1 + (x)_2 = 0$. We can see that SVM-Stump produces a decision boundary close to the optimal, SVM-Mid is slightly worse, and AdaBoost-Stump fails to generate a decent boundary. SVM-Stump obtains the smooth boundary by averaging over infinitely many decision stumps. SVM-Mid, although using finite number of decision stumps, can still have a smooth boundary in the center area by constructing a nonsparse ensemble. However, AdaBoost-Stump, which produces a finite and sparse ensemble, does not have the ability to approximate the Bayes optimal boundary well.

Although sparsity is often considered beneficial in learning paradigms like Occam's razor, a sparse classifier is not always good. In our case, because the decision stumps are very simple, a general dataset would require many of them to describe a suitable decision boundary. Thus, AdaBoost would suffer from the finite choice of middle stumps, the sparsity introduced by the ℓ_1-norm, and the approximation by T iterations. The comparison between AdaBoost-Stump and SVM-Mid indicates that the second restriction could be crucial. On the other hand, our framework (SVM-Stump), which does not have all those restrictions, has an advantage by averaging over an infinite number of hypotheses.

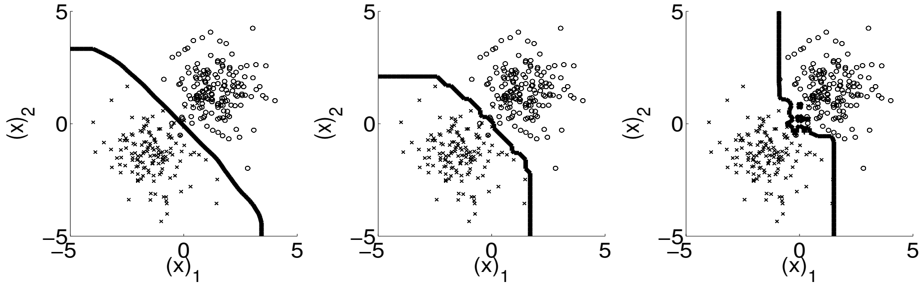

Fig. 2. Decision boundaries of SVM-Stump (left), SVM-Mid (middle), and AdaBoost-Stump with $T = 100$ (right) on a 2-D twonorm dataset.

Table 2. Test error (%) of SVM with different kernels

dataset	SVM-Stump	SVM-Gauss	dataset	SVM-Stump	SVM-Gauss
twonorm	2.86 ± 0.04	$\mathbf{2.64 \pm 0.05}$	twonorm-n	3.08 ± 0.06	$\mathbf{2.86 \pm 0.07}$
threenorm	17.7 ± 0.10	$\mathbf{14.6 \pm 0.11}$	threenorm-n	19.0 ± 0.14	$\mathbf{15.6 \pm 0.15}$
ringnorm	3.97 ± 0.07	$\mathbf{1.78 \pm 0.04}$	ringnorm-n	5.56 ± 0.11	$\mathbf{2.05 \pm 0.07}$
australian	$\mathbf{14.5 \pm 0.21}$	$\mathbf{14.7 \pm 0.18}$	breast	$\mathbf{3.11 \pm 0.08}$	3.53 ± 0.09
german	$\mathbf{24.7 \pm 0.18}$	$\mathbf{24.5 \pm 0.21}$	heart	$\mathbf{16.4 \pm 0.27}$	17.5 ± 0.31
ionosphere	8.13 ± 0.17	$\mathbf{6.54 \pm 0.19}$	pima	24.2 ± 0.23	$\mathbf{23.5 \pm 0.19}$
sonar	16.6 ± 0.42	$\mathbf{15.5 \pm 0.50}$	votes84	$\mathbf{4.76 \pm 0.14}$	$\mathbf{4.62 \pm 0.14}$

(results that are as significant as the best one are marked in bold)

5.2 Comparison to Gaussian Kernel

To further test the performance of the stump kernel in practice, we compare SVM-Stump with a popular and powerful setting, SVM-Gauss. Table 2 shows the test errors of them. From the table, SVM-Stump could have comparable yet slightly worse performance. However, the stump kernel has the advantage of faster parameter selection because scaling the stump kernel is equivalent to scaling the soft-margin parameter C. Thus, only a simple parameter search on C is necessary. For example, in our experiments, SVM-Gauss involves solving 550 optimization problems using different parameters, but we only need to deal with 55 problems for SVM-Stump. None of the commonly-used nonlinear SVM kernel can do fast parameter selection like the stump kernel. With the comparable performance, when time is a big concern, SVM-Stump could be a first-hand choice.

6 Conclusion

We proposed a framework to construct ensemble classifiers that average over an infinite number of base hypotheses. This is achieved with SVM by embedding infinitely many hypotheses in an SVM kernel. In contrast to ensemble learning algorithms like AdaBoost, our framework inherits the profound generalization performance from the soft-margin SVM, and would generate infinite and non-sparse ensembles, which are usually more robust than sparse ones.

We demonstrated our framework with decision stumps and obtained the stump kernel, which is novel and useful. Experimental comparisons with AdaBoost showed that SVM with the stump kernel usually performs much better than AdaBoost with stumps. Therefore, existing applications that use AdaBoost with stumps may be improved by switching to SVM with the stump kernel. In addition, we can benefit from the property of fast parameter selection when using the stump kernel. The property makes the kernel favorable to the Gaussian kernel in the case of large datasets.

Acknowledgment

We thank Yaser Abu-Mostafa, Amrit Pratap, Kai-Min Chung, and the anonymous reviewers for valuable suggestions. This work has been mainly supported by the Caltech Center for Neuromorphic Systems Engineering under the US NSF Cooperative Agreement EEC-9402726. Ling Li is currently sponsored by the Caltech SISL Graduate Fellowship.

References

1. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Machine Learning: Proceedings of the Thirteenth International Conference. (1996) 148–156
2. Rosset, S., Zhu, J., Hastie, T.: Boosting as a regularized path to a maximum margin classifier. Journal of Machine Learning Research **5** (2004) 941–973
3. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences **55** (1997) 119–139
4. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, New York (1998)
5. Lin, H.T.: Infinite ensemble learning with support vector machines. Master's thesis, California Institute of Technology (2005)
6. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge, MA (2002)
7. Rätsch, G., Onoda, T., Müller, K.: Soft margins for AdaBoost. Machine Learning **42** (2001) 287–320
8. Demiriz, A., Bennett, K.P., Shawe-Taylor, J.: Linear programming boosting via column generation. Machine Learning **46** (2002) 225–254
9. Meir, R., Rätsch, G.: An introduction to boosting and leveraging. In Mendelson, S., Smola, A.J., eds.: Advanced Lectures on Machine Learning. Springer-Verlag, Berlin (2003) 118–183
10. Freund, Y., Schapire, R.E.: A short introduction to boosting. Journal of Japanese Society for Artificial Intelligence **14** (1999) 771–780
11. Reed, M., Simon, B.: Functional Analysis. Revised and enlarged edn. Methods of Modern Mathematical Physics. Academic Press (1980)
12. Holte, R.C.: Very simple classification rules perform well on most commonly used datasets. Machine Learning **11** (1993) 63–91
13. Berg, C., Christensen, J.P.R., Ressel, P.: Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions. Springer-Verlag, New York (1984)

14. Lin, H.T., Lin, C.J.: A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. Technical report, National Taiwan University (2003)
15. Chang, C.C., Lin, C.J.: Training ν-support vector classifiers: Theory and algorithms. Neural Computation **13** (2001) 2119–2147
16. Keerthi, S.S., Lin, C.J.: Asymptotic behaviors of support vector machines with Gaussian kernel. Neural Computation **15** (2003) 1667–1689
17. Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classification. Technical report, National Taiwan University (2003)
18. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. (2001) Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
19. Breiman, L.: Prediction games and arcing algorithms. Neural Computation **11** (1999) 1493–1517
20. Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998) Downloadable at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.