Novel Distance-Based SVM Kernels for Infinite Ensemble Learning

Hsuan-Tien Lin and Ling Li
htlin@caltech.edu, ling@caltech.edu
Learning Systems Group, California Institute of Technology, USA

***Abstract*— Ensemble learning algorithms such as boosting can achieve better performance by averaging over the predictions of base hypotheses. However, most existing algorithms are limited to combining only a finite number of hypotheses, and the generated ensemble is usually sparse. It has recently been shown that the support vector machine (SVM) with a carefully crafted kernel can be used to construct a nonsparse ensemble of infinitely many hypotheses. Such infinite ensembles may surpass finite and/or sparse ensembles in learning performance and robustness. In this paper, we derive two novel kernels, the stump kernel and the perceptron kernel, for infinite ensemble learning. The stump kernel embodies an infinite number of decision stumps, and measures the similarity between examples by the ℓ_1-norm distance. The perceptron kernel embodies perceptrons, and works with the ℓ_2-norm distance. Experimental results show that SVM with these kernels is superior to boosting with the same base hypothesis set. In addition, SVM with these kernels has similar performance to SVM with the Gaussian kernel, but enjoys the benefit of faster parameter selection. These properties make the kernels favorable choices in practice.**

I. 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, most 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 II-B), 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].

Lin and Li [5] formulated an infinite ensemble learning framework based on the support vector machine (SVM) [4]. The key of the framework is to embed an infinite number of hypotheses into an SVM kernel. Such a framework can be applied both to construct new kernels, and to interpret some existing ones [6]. Furthermore, the framework allows a fair comparison between SVM and ensemble learning algorithms.

In this paper, we derive two novel SVM kernels, the stump kernel and the perceptron kernel, based on the framework. The stump kernel embodies an infinite number of decision stumps, and measures the similarity between examples by the ℓ_1-norm distance. The perceptron kernel embodies perceptrons, and works with the ℓ_2-norm distance. The two kernels are powerful both in theory and in practice. Experimental results show that SVM with these kernels is superior to famous ensemble learning algorithms with the same base hypothesis set. In addition, SVM with these kernels has similar performance to SVM with the popular Gaussian kernel, but enjoys the benefit of faster parameter selection.

The paper is organized as follows. In Section II, we show the connections between SVM and ensemble learning. Next in Section III, we introduce the framework for embedding an infinite number of hypotheses into a kernel. We then derive the stump kernel in Section IV, and the perceptron kernel in Section V. Finally, we show the experimental results in Section VI, and conclude in Section VII.

II. SVM and Ensemble Learning

A. 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:

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

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 $\langle \cdot, \cdot \rangle$ [7]. Because $\mathcal{F}$ can be of an infinite number of dimensions, SVM solvers usually work on the dual problem:

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

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], [7].

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 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], [7].

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).

B. 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 $\operatorname{sign}(\sum_{t=1}^{\infty} w_t h_t(x))$ [8], such that (w, h) is an optimal solution to

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

Problem (P_3) has infinitely many variables. In order to approximate the optimal solution well with a fixed T, AdaBoost has to resort to the sparsity of the optimal solutions for (P_3). That is, there are some optimal solutions that only need a small number of nonzero weights. The sparsity comes from the ℓ_1-norm criterion $\|w\|_1$, and allows AdaBoost to efficiently approximate the optimal solution through iterative optimization [2]. Effectively, AdaBoost only utilizes a small finite subset of $\mathcal{H}$, and approximates a sparse ensemble over $\mathcal{H}$.

C. Connecting SVM to Ensemble Learning

SVM and AdaBoost are related. Consider the feature transform

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

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 adopt proper slack variables in (P_3) and solve it by the linear programming boosting [9]. Our experimental observation shows that this does not change the conclusion of this paper, so 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], [8]. However, as limited as AdaBoost, their 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 show the remedies for these difficulties in the next section.

III. 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.

Lin and Li [5] devised another approach using the connection between SVM and ensemble learning. Their framework is based on 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). Under reasonable assumptions on $\mathcal{H}$, the framework can perform infinite ensemble learning. In this section, we shall briefly introduce the framework and the assumptions.

A. Embedding Hypotheses into the Kernel

The key of the framework of Lin and Li is to embed the infinite number of hypotheses in $\mathcal{H}$ into an SVM kernel [5]. We have shown with (2) that we could construct a feature mapping from $\mathcal{H}$. The idea is extended to a more general form for deriving a kernel in Definition 1.

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. We shall denote $\mathcal{K}_{\mathcal{H},r}$ by $\mathcal{K}_{\mathcal{H}}$ when r is clear from the context. The validity of the kernel for a general $\mathcal{C}$ can be formalized in the following theorem:

Theorem 1 *[5] Consider the kernel $\mathcal{K}_{\mathcal{H}}$ 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.*

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

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

which is equivalent to some ensemble classifier over $\mathcal{H}$.

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

The technique of constructing kernels from an integral inner product is known in literature [7]. The framework utilizes this technique for embedding the hypotheses, and thus could handle the situation even when $\mathcal{H}$ is uncountable. Next, we explain how the kernel $\mathcal{K}_\mathcal{H}$ can be used for infinite ensemble learning under mild assumptions.

B. Negation Completeness and Constant Hypotheses

When we use $\mathcal{K}_\mathcal{H}$ in (P_2), the classifier obtained is:

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

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. Lin and Li further assumed that $\mathcal{H}$ is negation complete, that is, $h \in \mathcal{H}$ if and only if $(-h) \in \mathcal{H}$.[1] In addition, they assumed that $\mathcal{H}$ contains a constant hypothesis.[2] Under these assumptions, the classifier g in (4) or (1) is indeed equivalent to an ensemble classifier. The framework can be summarized in Fig. 1, and shall generally inherit the profound performance of SVM. Most of the steps in the framework can be done by existing SVM algorithms, and the hard part is mostly in obtaining the kernel $\mathcal{K}_\mathcal{H}$. In the next two sections, we will show two kernels derived from the framework.

IV. STUMP KERNEL

In this section, we present the stump kernel, which embodies infinitely many decision stumps. The decision stump $s_{q,d,\alpha}(x) = q \cdot \operatorname{sign}((x)_d - \alpha)$ works on the d-th element of x, and classifies x according to $q \in \{-1, +1\}$ and the threshold α [11]. It is widely used for ensemble learning because of its simplicity [1].

[1] We use $(-h)$ to denote the function $(-h)(\cdot) = -(h(\cdot))$.

[2] A constant hypothesis $c(\cdot)$ predicts $c(x) = 1$ for all $x \in \mathcal{X}$.

A. Formulation

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

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

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 the 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 $r(q, d, \alpha_d) = \frac{1}{2}$,*

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

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

The integral in Definition 1 is easy to compute when we simply assign a constant $r_\mathcal{S}$ to all $r(q, d, \alpha_d)$. Note that scaling $r_\mathcal{S}$ is equivalent to scaling the parameter C in SVM. Thus, without loss of generality, we choose $r_\mathcal{S} = \frac{1}{2}$ 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 *[5] Solving (P_2) with $\mathcal{K}_\mathcal{S}$ is the same as solving (P_2) with the simplified stump kernel $\tilde{\mathcal{K}}_\mathcal{S}(x, x') = -\|x - x'\|_1$. That is, they obtain equivalent classifiers in* (1).

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

B. Power of the Stump Kernel

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

Theorem 3 *[6] 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.*

The PD-ness of the Gram matrix is directly connected to the classification power of the SVM classifiers. Chang and Lin [12] showed that when the Gram matrix of the kernel is PD, a hard-margin SVM can always dichotomize the training set. 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 shows that the stump kernel has theoretically almost the same power as the famous Gaussian kernel, which also provides infinite capacity to SVM [13]. 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, soft-margin SVM with suitable parameter selection can regularize the power and achieve good generalization performance even in the presence of noise [13], [14]. Soft-margin SVM with the stump kernel also has such property, which will be demonstrated experimentally in Section VI.

V. PERCEPTRON KERNEL

In this section, we extend the stump kernel to the perceptron kernel, which embodies an infinite number of perceptrons. A perceptron is a linear threshold classifier of the form $p_{\theta,\alpha}(x) = \operatorname{sign}\big(\theta^T x - \alpha\big)$. It is a basic theoretical model for a neuron, and is very important for building neural networks [15].

A. Formulation

We consider the following set of perceptrons

$$\mathcal{P} = \left\{p_{\theta,\alpha} \colon \theta \in \mathbb{R}^D, \|\theta\|_2 = 1, \alpha \in [-R, R]\right\}.$$

We assume that $\mathcal{X} \subseteq \mathcal{B}(R)$, where $\mathcal{B}(R)$ is a ball of radius R centered at the origin in $\mathbb{R}^D$. Then, $\mathcal{P}$ is negation complete, and contains a constant classifier $p_{e_1,-R}(\cdot)$ where $e_1 = (1, 0, \cdots, 0)^T$. Thus, we can apply Definition 1 to $\mathcal{P}$ and obtain the perceptron kernel $\mathcal{K}_\mathcal{P}$.

Definition 3 *The perceptron kernel is $\mathcal{K}_\mathcal{P}$ with $r(\theta, \alpha) = r_\mathcal{P}$,*

$$\mathcal{K}_\mathcal{P}(x, x') = \Delta_\mathcal{P} - \|x - x'\|_2\,,$$

where $r_\mathcal{P}$ and $\Delta_\mathcal{P}$ are constants to be defined below. The integral in Definition 1 is done with uniform measure in all possible parameters of $\mathcal{P}$.

Proof: Define two constants

$$\Theta_D = \int_{\|\theta\|_2=1} d\theta, \quad \Xi_D = \int_{\|\theta\|_2=1} \left|\cos\left(\operatorname{angle}\langle\theta, e_1\rangle\right)\right| \, d\theta.$$

Here the operator $\operatorname{angle}\langle\cdot,\cdot\rangle$ is the angle between two vectors. Noticing that $p_{\theta,\alpha}(x) = s_{+1,1,\alpha}(\theta^T x)$, we have

$$\begin{aligned}
&\mathcal{K}_\mathcal{P}(x, x') \\
&= r_\mathcal{P}^2 \int_{\|\theta\|_2=1} \left[\int_{-R}^{R} s_{+1,1,\alpha}(\theta^T x) s_{+1,1,\alpha}(\theta^T x') \, d\alpha\right] d\theta \\
&= 2r_\mathcal{P}^2 \int_{\|\theta\|_2=1} \left(R - \left|\theta^T x - \theta^T x'\right|\right) d\theta \\
&= 2r_\mathcal{P}^2 \int_{\|\theta\|_2=1} \left(R - \|x - x'\|_2 \left|\cos\left(\operatorname{angle}\langle\theta, x - x'\rangle\right)\right|\right) d\theta \\
&= 2r_\mathcal{P}^2 \Theta_D R - 2r_\mathcal{P}^2 \Xi_D \|x - x'\|_2\,.
\end{aligned}$$

Because the integral is over every possible direction of θ, the symmetry leads to the last equality. Then, we can set $r_\mathcal{P} = (2\Xi_D)^{-\frac{1}{2}}$ and $\Delta_\mathcal{P} = \Theta_D \Xi_D^{-1} R$ to obtain the definition. ■

With the perceptron kernel, we can construct an infinite ensemble classifier over perceptrons. Such a classifier is equivalent to a neural network with one hidden layer, infinitely many hidden neurons, and the hard-threshold activation functions. Even without the infinity, it is difficult to optimize with the hard-threshold activation functions or to obtain a good and efficient learning algorithm for perceptrons [16]. Hence, traditional neural network or ensemble learning algorithms can never build such a classifier. Using the perceptron kernel, however, the infinite neural network (ensemble of perceptrons) can be easily obtained through SVM.

The perceptron kernel shares many similar properties to the stump kernel. First, the constant $\Delta_\mathcal{P}$ can also be dropped, as formalized below.

Theorem 5 *Solving (P_2) with the simplified perceptron kernel $\tilde{\mathcal{K}}_\mathcal{P}(x, x') = -\|x - x'\|_2$ is the same as solving (P_2) with $\mathcal{K}_\mathcal{P}(x, x')$.*

Second, the perceptron kernel also provides infinite capacity to SVM, which is shown below.

B. Power of the Perceptron Kernel

The power of the perceptron kernel comes from the following PD-ness theorem known in interpolation literature:

Theorem 6 *[6], [17] Consider input vectors $\{x_i\}_{i=1}^N \in \mathcal{X}^N$, and the perceptron kernel $\mathcal{K}_\mathcal{P}$ in Definition 3. If $\mathcal{X} \subset \mathcal{B}(R)$ but $\mathcal{X} \neq \mathcal{B}(R)$, and $x_i \neq x_j$ for all $i \neq j$, then the Gram matrix of $\mathcal{K}_\mathcal{P}$ is PD.*

Then, similar to Theorem 4, we get:

Theorem 7 *The class of SVM classifiers with $\mathcal{K}_\mathcal{P}$, or equivalently, the class of infinite ensemble classifiers over $\mathcal{P}$, has an infinite V-C dimension.*

The stump kernel, the perceptron kernel, and the Gaussian kernel all evaluates the similarity between examples by distance. They all provide infinite power to SVM, while the first two are simpler and have explanations from an ensemble point-of-view. We shall further compare them experimentally in Subsection VI-B.

VI. EXPERIMENTS

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

SVM with the simplified stump kernel is denoted as SVM-Stump. It is compared with AdaBoost-Stump, AdaBoost with decision stumps as base hypotheses.

SVM with the simplified perceptron kernel is denoted SVM-Perc. We compare it to AdaBoost-Perc, AdaBoost with perceptrons as base hypotheses. Note that AdaBoost requires a base learner to choose the perceptrons. Unlike the decision stumps for which a deterministic and efficient learning algorithm is available, perceptron learning is usually probabilistic and difficult, especially when the dataset is not linearly separable. We use the random coordinate descent algorithm [16] which is shown to work well with AdaBoost as the base learner.

TABLE I

TEST ERROR (%) OF SEVERAL ENSEMBLE LEARNING ALGORITHMS

dataset	SVM-Stump	AB-Stump $T = 100$	AB-Stump $T = 1000$	SVM-Perc	AB-Perc $T = 100$	AB-Perc $T = 1000$
twonorm	$\mathbf{2.86 \pm 0.04}$	5.06 ± 0.06	4.97 ± 0.06	$\mathbf{2.55 \pm 0.03}$	3.08 ± 0.05	3.00 ± 0.04
twonorm-n	$\mathbf{3.08 \pm 0.06}$	12.6 ± 0.14	15.5 ± 0.17	$\mathbf{2.76 \pm 0.05}$	5.93 ± 0.08	4.31 ± 0.07
threenorm	$\mathbf{17.7 \pm 0.10}$	21.8 ± 0.09	22.9 ± 0.12	$\mathbf{14.6 \pm 0.08}$	18.2 ± 0.12	16.7 ± 0.09
threenorm-n	$\mathbf{19.0 \pm 0.14}$	25.9 ± 0.13	28.2 ± 0.14	$\mathbf{16.3 \pm 0.10}$	21.9 ± 0.14	19.2 ± 0.11
ringnorm	$\mathbf{3.97 \pm 0.07}$	12.2 ± 0.13	9.95 ± 0.14	$\mathbf{2.46 \pm 0.04}$	24.0 ± 0.19	20.0 ± 0.24
ringnorm-n	$\mathbf{5.56 \pm 0.11}$	19.4 ± 0.20	20.3 ± 0.19	$\mathbf{3.50 \pm 0.09}$	30.5 ± 0.22	26.2 ± 0.24
australian	$\mathbf{14.5 \pm 0.21}$	$\mathbf{14.7 \pm 0.18}$	16.9 ± 0.18	$\mathbf{14.5 \pm 0.17}$	15.5 ± 0.16	15.6 ± 0.14
breast	$\mathbf{3.11 \pm 0.08}$	4.27 ± 0.11	4.51 ± 0.11	$\mathbf{3.23 \pm 0.08}$	3.50 ± 0.09	$\mathbf{3.38 \pm 0.09}$
german	$\mathbf{24.7 \pm 0.18}$	$\mathbf{25.0 \pm 0.18}$	26.9 ± 0.18	$\mathbf{24.6 \pm 0.20}$	26.2 ± 0.20	$\mathbf{24.9 \pm 0.19}$
heart	$\mathbf{16.4 \pm 0.27}$	19.9 ± 0.36	22.6 ± 0.39	$\mathbf{17.6 \pm 0.31}$	18.6 ± 0.29	$\mathbf{17.8 \pm 0.30}$
ionosphere	$\mathbf{8.13 \pm 0.17}$	11.0 ± 0.23	11.0 ± 0.25	$\mathbf{6.40 \pm 0.20}$	11.8 ± 0.28	11.3 ± 0.26
pima	$\mathbf{24.2 \pm 0.23}$	24.8 ± 0.22	27.0 ± 0.25	$\mathbf{23.5 \pm 0.21}$	24.9 ± 0.22	24.2 ± 0.20
sonar	$\mathbf{16.6 \pm 0.42}$	19.0 ± 0.37	19.0 ± 0.35	$\mathbf{15.6 \pm 0.40}$	21.4 ± 0.41	19.2 ± 0.42
votes84	4.76 ± 0.14	$\mathbf{4.07 \pm 0.14}$	5.29 ± 0.15	$\mathbf{4.43 \pm 0.14}$	$\mathbf{4.43 \pm 0.16}$	$\mathbf{4.49 \pm 0.14}$

(results that are as significant as the best ones with the same base hypothesis set are marked in bold)

We also compare SVM-Stump and SVM-Perc with SVM-Gauss, which is SVM with the Gaussian kernel. For AdaBoost-Stump and AdaBoost-Perc, 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 [14], which selects a suitable parameter with cross validation before actual training.

The three artificial datasets from Breiman [19] (twonorm, threenorm, and ringnorm) are used. 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. The settings are the same as the ones used by Lin and Li [5]. All the results are averaged over 100 runs, presented with standard error bar.

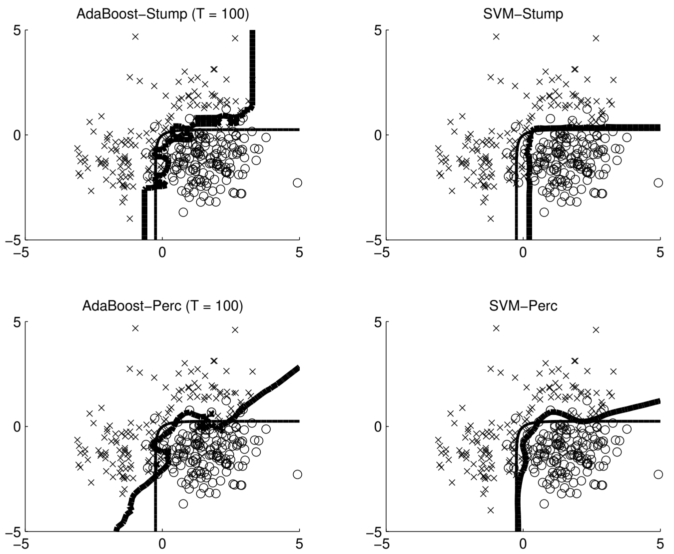

Fig. 2. Decision boundaries from four ensemble learning algorithms on a 2-D threenorm dataset. (thin curves: Bayes optimal boundary; thick curves: boundaries from the algorithms)

A. Comparison of Ensemble Learning Algorithms

Table I shows the test performance of several ensemble learning algorithms. We can see that SVM-Stump and SVM-Perc are usually better than AdaBoost with the same base hypothesis set, and especially have superior performance in the presence of noise. These results demonstrate that it is beneficial to go from a finite ensemble to an infinite one with suitable regularization.

To further demonstrate the difference between the finite and infinite ensemble learning algorithms, in Fig. 2 we show the decision boundaries generated by the four algorithms on 300 training examples from a 2-D version of the threenorm dataset. We can see that both SVM-Stump and SVM-Perc produce a decision boundary close to the optimal, while AdaBoost-Stump and AdaBoost-Perc fail to generate a decent boundary. One reason is that a sparse and finite ensemble can be easily influenced by a few hypotheses. For example, in Fig. 2, the boundary of AdaBoost-Stump is influenced by the vertical line at the right, and the boundary of AdaBoost-Perc is affected by the inclined line. The risk is that those hypotheses may only represent an unstable approximation of the underlying model. In contrast, the infinite ensemble produced by SVM averages over the predictions of many hypotheses, and hence can produce a smoother and stabler boundary that approximates the optimal one well.

Another reason for AdaBoost-like ensemble learning algorithms to perform worse is the overfitting in the center areas of the figures. Although AdaBoost performs inherit regularization for a suitable choice of T [2], the goal of the algorithm is to fit the difficult examples well. Hence, for any T, many of the finite T hypotheses are used to create a sophisticated boundary in the center rather than to globally approximate the optimal boundary. Thus, in the case of noisy or difficult datasets (e.g., ringnorm), AdaBoost-like ensemble learning algorithms overfit the noise easily. On the other hand, SVM-based ensemble learning algorithms can give regularization with a suitable choice of C, and hence achieve good performance even in the presence of noise.

TABLE II

TEST ERROR (%) OF SVM WITH DIFFERENT KERNELS

dataset	SVM-Stump	SVM-Perc	SVM-Gauss
twonorm	2.86 ± 0.04	2.55 ± 0.03	$\mathbf{2.64 \pm 0.05}$
twonorm-n	3.08 ± 0.06	$\mathbf{2.76 \pm 0.05}$	$\mathbf{2.86 \pm 0.07}$
threenorm	17.7 ± 0.10	$\mathbf{14.6 \pm 0.08}$	$\mathbf{14.6 \pm 0.11}$
threenorm-n	19.0 ± 0.14	16.3 ± 0.10	$\mathbf{15.6 \pm 0.15}$
ringnorm	3.97 ± 0.07	2.46 ± 0.04	$\mathbf{1.78 \pm 0.04}$
ringnorm-n	5.56 ± 0.11	3.50 ± 0.09	$\mathbf{2.05 \pm 0.07}$
australian	$\mathbf{14.5 \pm 0.21}$	$\mathbf{14.5 \pm 0.17}$	$\mathbf{14.7 \pm 0.18}$
breast	$\mathbf{3.11 \pm 0.08}$	$\mathbf{3.23 \pm 0.08}$	3.53 ± 0.09
german	$\mathbf{24.7 \pm 0.18}$	$\mathbf{24.6 \pm 0.20}$	$\mathbf{24.5 \pm 0.21}$
heart	$\mathbf{16.4 \pm 0.27}$	17.6 ± 0.31	17.5 ± 0.31
ionosphere	8.13 ± 0.17	$\mathbf{6.40 \pm 0.20}$	$\mathbf{6.54 \pm 0.19}$
pima	24.2 ± 0.23	$\mathbf{23.5 \pm 0.21}$	$\mathbf{23.5 \pm 0.19}$
sonar	16.6 ± 0.42	$\mathbf{15.6 \pm 0.40}$	$\mathbf{15.5 \pm 0.50}$
votes84	4.76 ± 0.14	$\mathbf{4.43 \pm 0.14}$	$\mathbf{4.62 \pm 0.14}$

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

B. Comparison to Gaussian Kernel

To further test the performance of the two novel kernels in practice, we compare SVM-Stump and SVM-Perc with a popular and powerful setting, SVM-Gauss. Table II shows their test errors. We can see that SVM-Perc and SVM-Gauss have almost indistinguishable performance in the real-world datasets, which is possibly because they both use the ℓ_2-norm distance for measuring similarity. Note that SVM-Gauss has an advantage in the artificial datasets because they are generated from certain Gaussian distributions. Thus, the indistinguishable performance on real-world datasets makes SVM-Perc a useful alternative to SVM-Gauss in practice.

In addition, SVM-Perc enjoys the benefit of faster parameter selection because scaling the 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, but we only need to deal with 55 problems for SVM-Perc. None of the commonly-used nonlinear SVM kernel can do fast parameter selection like this. With the indistinguishable performance, SVM-Perc should be a more favorable choice.

SVM-Stump also enjoys the benefit of fast parameter selection. From Table II, SVM-Stump is only slightly worse than SVM-Perc. With the comparable performance, SVM-Stump could still be useful when we have the prior knowledge or preference to model the dataset by an ensemble of decision stumps.

VII. CONCLUSION

We derived two novel kernels based on the infinite ensemble learning framework. The stump kernel embodies an infinite number of decision stumps, and the perceptron kernel embodies an infinite number of perceptrons. These kernels can be simply evaluated by the ℓ_1- or ℓ_2-norm distance between examples. SVM equipped with such kernels can generate infinite and nonsparse ensembles, which are usually more robust than sparse ones.

Experimental comparisons with AdaBoost showed that SVM with the novel kernels usually performs much better than AdaBoost with the same base hypothesis set. Therefore, existing applications that use AdaBoost with stumps or perceptrons may be improved by switching to SVM with the stump kernel or the perceptron kernel.

In addition, we showed that the perceptron kernel has similar performance to the Gaussian kernel, while it benefits from faster parameter selection. This property makes the perceptron kernel favorable to the Gaussian kernel in practice.

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] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Machine Learning: Proceedings of the Thirteenth International Conference*, 1996, pp. 148–156.

[2] S. Rosset, J. Zhu, and T. Hastie, "Boosting as a regularized path to a maximum margin classifier," *Journal of Machine Learning Research*, vol. 5, pp. 941–973, 2004.

[3] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, pp. 119–139, 1997.

[4] V. N. Vapnik, *Statistical Learning Theory*. New York: John Wiley & Sons, 1998.

[5] H.-T. Lin and L. Li, "Infinite ensemble learning with support vector machines," in *Machine Learning: ECML 2005*, 2005.

[6] H.-T. Lin, "Infinite ensemble learning with support vector machines," Master's thesis, California Institute of Technology, 2005.

[7] B. Schölkopf and A. Smola, *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.

[8] G. Rätsch, T. Onoda, and K. Müller, "Soft margins for AdaBoost," *Machine Learning*, vol. 42, pp. 287–320, 2001.

[9] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," *Machine Learning*, vol. 46, pp. 225–254, 2002.

[10] Y. Freund and R. E. Schapire, "A short introduction to boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, pp. 771–780, 1999.

[11] R. C. Holte, "Very simple classification rules perform well on most commonly used datasets," *Machine Learning*, vol. 11, pp. 63–91, Apr. 1993.

[12] C.-C. Chang and C.-J. Lin, "Training ν-support vector classifiers: Theory and algorithms," *Neural Computation*, vol. 13, pp. 2119–2147, 2001.

[13] S. S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with Gaussian kernel," *Neural Computation*, vol. 15, pp. 1667–1689, 2003.

[14] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," National Taiwan University, Tech. Rep., July 2003.

[15] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Prentice Hall, 1999.

[16] L. Li, "Perceptron learning with random coordinate descent," California Institute of Technology, Tech. Rep. CaltechCSTR:2005.006, Aug. 2005.

[17] C. A. Micchelli, "Interpolation of scattered data: Distance matrices and conditionally positive definite functions," *Constructive Approximation*, vol. 2, pp. 11–22, 1986.

[18] C.-C. Chang and C.-J. Lin, *LIBSVM: A library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[19] L. Breiman, "Prediction games and arcing algorithms," *Neural Computation*, vol. 11, pp. 1493–1517, 1999.

[20] S. Hettich, C. L. Blake, and C. J. Merz, "UCI repository of machine learning databases," 1998, downloadable at http://www.ics.uci.edu/~mlearn/MLRepository.html.