Journal of Machine Learning Research 9 (2008) 285-312 Submitted 2/07; Revised 9/07; Published 2/08

Support Vector Machinery for Infinite Ensemble Learning

Hsuan-Tien Lin HTLIN@CALTECH.EDU
Ling Li LING@CALTECH.EDU
Department of Computer Science
California Institute of Technology
Pasadena, CA 91125, USA

Editor: Peter L. Bartlett

Abstract

Ensemble learning algorithms such as boosting can achieve better performance by averaging over the predictions of some base hypotheses. Nevertheless, most existing algorithms are limited to combining only a finite number of hypotheses, and the generated ensemble is usually sparse. Thus, it is not clear whether we should construct an ensemble classifier with a larger or even an 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 the support vector machine (SVM). The framework can output an infinite and nonsparse ensemble through embedding infinitely many hypotheses into an SVM kernel. We use the framework to derive two novel kernels, the stump kernel and the perceptron kernel. The stump kernel embodies infinitely many decision stumps, and the perceptron kernel embodies infinitely many perceptrons. We also show that the Laplacian radial basis function kernel embodies infinitely many decision trees, and can thus be explained through infinite ensemble learning. Experimental results show that SVM with these kernels is superior to boosting with the same base hypothesis set. In addition, SVM with the stump kernel or the perceptron kernel performs similarly to SVM with the Gaussian radial basis function kernel, but enjoys the benefit of faster parameter selection. These properties make the novel kernels favorable choices in practice.

Keywords: ensemble learning, boosting, support vector machine, kernel

1. Introduction

Ensemble learning algorithms, such as boosting (Freund and Schapire, 1996), are successful in practice (Meir and Rätsch, 2003). They construct a classifier that averages over some base hypotheses in a set $\mathcal{H}$. While the size of $\mathcal{H}$ can be infinite, most existing algorithms use only a finite subset of $\mathcal{H}$, and the classifier is effectively a finite ensemble of hypotheses. Some theories show that the finiteness places a restriction on the capacity of the ensemble (Freund and Schapire, 1997), and some theories suggest that the performance of boosting can be linked to its asymptotic behavior when the ensemble is allowed to be of an infinite size (Rätsch et al., 2001). Thus, it is possible that an infinite ensemble is superior for learning. Nevertheless, the possibility has not been fully explored because constructing such an ensemble is a challenging task (Vapnik, 1998).

In this paper, we conquer the task of infinite ensemble learning, and demonstrate that better performance can be achieved by going from finite ensembles to infinite ones. We formulate a framework for infinite ensemble learning based on the support vector machine (SVM) (Vapnik, 1998). 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 for SVM, and to interpret some existing ones (Lin, 2005). Furthermore, the framework allows us to compare SVM and ensemble learning algorithms in a fair manner using the same base hypothesis set.

Based on the framework, we derive two novel SVM kernels, the stump kernel and the perceptron kernel, from an ensemble learning perspective (Lin and Li, 2005a). The stump kernel embodies infinitely many decision stumps, and as a consequence measures the similarity between examples by the ℓ_1-norm distance. The perceptron kernel embodies infinitely many perceptrons, and works with the ℓ_2-norm distance. While there exist similar kernels in literature, our derivation from an ensemble learning perspective is nevertheless original. Our work not only provides a feature-space view of their theoretical properties, but also broadens their use in practice. Experimental results show that SVM with these kernels is superior to successful ensemble learning algorithms with the same base hypothesis set. These results reveal some weakness in traditional ensemble learning algorithms, and help understand both SVM and ensemble learning better. In addition, SVM with these kernels shares similar performance to SVM with the popular Gaussian radial basis function (Gaussian-RBF) kernel, but enjoys the benefit of faster parameter selection. These properties make the two kernels favorable choices in practice.

We also show that the Laplacian-RBF kernel embodies infinitely many decision trees, and hence can be viewed as an instance of the framework. Experimentally, SVM with the Laplacian-RBF kernel performs better than ensemble learning algorithms with decision trees. In addition, our derivation from an ensemble learning perspective helps to explain the success of the kernel on some specific applications (Chapelle et al., 1999).

The paper is organized as follows. In Section 2, we review the connections between SVM and ensemble learning. Next in Section 3, we propose the framework for embedding an infinite number of hypotheses into a kernel. We then derive the stump kernel in Section 4, the perceptron kernel in Section 5, and the Laplacian-RBF kernel in Section 6. Finally, we show the experimental results in Section 7, and conclude in Section 8.

2. Support Vector Machine and Ensemble Learning

In this section, we first introduce the basics of SVM and ensemble learning. Then, we review some established connections between the two in literature.

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 (Vapnik, 1998) constructs a classifier

$$g(x) = \text{sign}\big(\langle w, \phi_x \rangle + b\big)$$

from the optimal solution to the following problem:[1]

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

1. When η is nonzero, $\text{sign}(\eta) \equiv \frac{\eta}{|\eta|}$. We shall let $\text{sign}(0) \equiv 0$ to make some mathematical setup cleaner.

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 $\langle \cdot, \cdot \rangle$ (Schölkopf and Smola, 2002). Because $\mathcal{F}$ can be of an infinite number of dimensions, SVM solvers usually work on the dual problem:

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

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 (Vapnik, 1998; Schölkopf and Smola, 2002).

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 (Schölkopf and Smola, 2002). Alternatively, we can begin with an arbitrary $\mathcal{K}$, and check whether there exists a space-mapping pair $(\mathcal{F},\Phi)$ 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) (Vapnik, 1998; Schölkopf and Smola, 2002).

The soft-margin SVM originates from the hard-margin SVM, which forces the margin violations ξ_i to be zero. When such a solution is feasible for (P_1), the corresponding dual solution can be obtained by setting C to ∞ in (P_2).

2.2 Adaptive Boosting and Linear Programming Boosting

The adaptive boosting (AdaBoost) algorithm (Freund and Schapire, 1996) is perhaps the most popular and successful approach 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 $w_t \ge 0$ to construct an ensemble classifier

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

The underlying algorithm for selecting $h_t \in \mathcal{H}$ is called a base learner. Under some assumptions (Rätsch et al., 2001), it is shown that when $T \to \infty$, AdaBoost asymptotically approximates an infinite ensemble classifier

$$g_\infty(x) = \operatorname{sign}\left(\sum_{t=1}^{\infty} w_t h_t(x)\right), \tag{2}$$

such that (w,h) is an optimal solution to

$$
\begin{aligned}
(P_3) \quad \min_{w_t\in\mathbb{R},h_t\in\mathcal{H}} \quad & \sum_{t=1}^{\infty} w_t \\
\text{s.t.} \quad & y_i\left(\sum_{t=1}^{\infty} w_t h_t(x_i)\right) \geq 1, && \text{for } i=1,2,\ldots,N, \\
& w_t \geq 0, && \text{for } t=1,2,\ldots,\infty.
\end{aligned}
$$

Note that there are infinitely many variables in (P_3). In order to approximate the optimal solution well with a fixed and finite T, AdaBoost resorts to two related properties of some of the optimal solutions for (P_3): finiteness and sparsity.

- **Finiteness:** When two hypotheses have the same prediction patterns on the training input vectors, they can be used interchangeably during the training time, and are thus *ambiguous*. Since there are at most 2^N prediction patterns on N training input vectors, we can partition $\mathcal{H}$ into at most 2^N groups, each of which contains mutually ambiguous hypotheses. Some optimal solutions of (P_3) only assign one or a few nonzero weights within each group (Demiriz et al., 2002). Thus, it is possible to work on a finite data-dependent subset of $\mathcal{H}$ instead of $\mathcal{H}$ itself without losing optimality.
- **Sparsity:** Minimizing the ℓ_1-norm $\|w\|_1 = \sum_{t=1}^{\infty} |w_t|$ often leads to sparse solutions (Meir and Rätsch, 2003; Rosset et al., 2007). 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. AdaBoost can be viewed as a stepwise greedy search algorithm that approximates such a finite and sparse ensemble (Rosset et al., 2004).

Another boosting approach, called the linear programming boosting (LPBoost), can solve (P_3) exactly. We will introduce the soft-margin LPBoost, which constructs an ensemble classifier like (2) with the optimal solution to

$$
\begin{aligned}
(P_4) \quad \min_{w_t\in\mathbb{R},h_t\in\mathcal{H}} \quad & \sum_{t=1}^{\infty} w_t + C\sum_{i=1}^{N} \xi_i \\
\text{s.t.} \quad & y_i\left(\sum_{t=1}^{\infty} w_t h_t(x_i)\right) \geq 1-\xi_i, && \text{for } i=1,2,\ldots,N, \\
& \xi_i \geq 0, && \text{for } i=1,2,\ldots,N, \\
& w_t \geq 0, && \text{for } t=1,2,\ldots,\infty.
\end{aligned}
$$

Demiriz et al. (2002) proposed to solve (P_4) with the column generating technique.[2] The algorithm works by adding one unambiguous h_t to the ensemble in each iteration. Because of the finiteness property, the algorithm is guaranteed to terminate within $T \leq 2^N$ iterations. The sparsity property can sometimes help speed up the convergence of the algorithm.

Rätsch et al. (2002) worked on a variant of (P_4) for regression problems, and discussed optimality conditions when $\mathcal{H}$ is of infinite size. Their results can be applied to (P_4) as well. In particular,

2. Demiriz et al. (2002) actually worked on an equivalent but slightly different formulation.

they showed that even without the finiteness property (e.g., when h_t outputs real values rather than binary values), (P_4) can still be solved using a finite subset of $\mathcal{H}$ that is associated with nonzero weights. The results justify the use of the column generating technique above, as well as a barrier, AdaBoost-like, approach that they proposed.

Recently, Rosset et al. (2007) studied the existence of a sparse solution when solving a generalized form of (P_4) with some $\mathcal{H}$ of infinite and possibly uncountable size. They showed that under some assumptions, there exists an optimal solution of (P_4) such that at most $N+1$ weights are nonzero. Thus, iterative algorithms that keep adding necessary hypotheses h_t to the ensemble, such as the proposed path-following approach (Rosset et al., 2007) or the column generating technique (Demiriz et al., 2002; Rätsch et al., 2002), could work by aiming towards such a sparse solution.

Note that even though the findings above indicate that it is possible to design good algorithms to return an optimal solution when $\mathcal{H}$ is infinitely large, the resulting ensemble relies on the sparsity property, and is effectively of only finite size. Nevertheless, it is not clear whether the performance could be improved if either or both the finiteness and the sparsity restrictions are removed.

2.3 Connecting Support Vector Machine to Ensemble Learning

The connection between AdaBoost, LPBoost, and SVM is well-known in literature (Freund and Schapire, 1999; Rätsch et al., 2001; Rätsch et al., 2002; Demiriz et al., 2002). Consider the feature transform

$$\Phi(x) = \big(h_1(x), h_2(x), \ldots\big). \tag{3}$$

We can see that the problem (P_1) with this feature transform is similar to (P_4). The elements of ϕ_x in SVM are similar to the hypotheses $h_t(x)$ in AdaBoost and LPBoost. They all work on linear combinations of these elements, though SVM deals with an additional intercept term b. SVM minimizes the ℓ_2-norm of the weights while AdaBoost and LPBoost work on the ℓ_1-norm. SVM and LPBoost introduce slack variables ξ_i and use the parameter C for regularization, while AdaBoost relies on the choice of the parameter T (Rosset et al., 2004). Note that AdaBoost and LPBoost require $w_t \geq 0$ for ensemble learning.

Several researchers developed interesting results based on the connection. For example, Rätsch et al. (2001) proposed to select the hypotheses h_t by AdaBoost and to obtain the weights w_t by solving an optimization problem similar to (P_1) in order to improve the robustness of AdaBoost. Another work by Rätsch et al. (2002) introduced a new density estimation algorithm based on the connection. Rosset et al. (2004) applied the similarity to compare SVM with boosting algorithms. Nevertheless, as limited as AdaBoost and LPBoost, their results could use only a finite subset of $\mathcal{H}$ when constructing the feature mapping (3). 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.

3. SVM-Based Framework for Infinite Ensemble Learning

Vapnik (1998) proposed a challenging task of designing an algorithm that actually generates an infinite ensemble classifier, that is, an ensemble classifier with infinitely many nonzero w_t. Traditional algorithms like AdaBoost or LPBoost cannot be directly generalized to solve the task, because they select the hypotheses in an iterative manner, and only run for a finite number of iterations.

We solved the challenge via another route: the connection between SVM and ensemble learning. The connection allows us to formulate a kernel that embodies all the hypotheses in $\mathcal{H}$. Then, the classifier (1) obtained from SVM with the kernel is a linear combination over $\mathcal{H}$ (with an intercept term). Nevertheless, 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 combine several ideas to deal with these obstacles, and conquer Vapnik's task with a novel SVM-based framework for infinite ensemble learning.

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 (3) 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 $\mathcal{H} = \{h_\alpha : \alpha \in \mathcal{C}\}$, where $\mathcal{C}$ is a measure space. The kernel that embodies $\mathcal{H}$ is defined as*

$$\mathcal{K}_{\mathcal{H},r}(x,x') = \int_{\mathcal{C}} \phi_x(\alpha)\phi_{x'}(\alpha)\, d\alpha, \tag{4}$$

where $\phi_x(\alpha) = r(\alpha)h_\alpha(x)$, and $r : \mathcal{C} \to \mathbb{R}^+$ 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. The validity of the definition is formalized in the following theorem.

Theorem 2 *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 $\varphi(\cdot) : \mathcal{C} \to \mathbb{R}$ that are square integrable.*

2. *For a set of input vectors $\{x_i\}_{i=1}^N \in \mathcal{X}^N$, the Gram matrix of $\mathcal{K}_{\mathcal{H}}$ is PSD.*

Proof The first part is known in mathematical analysis (Reed and Simon, 1980), and the second part follows Mercer's condition. ■

Constructing kernels from an integral inner product is a known technique in literature (Schölkopf and Smola, 2002). The framework adopts this technique for embedding the hypotheses, and thus could handle the situation even when $\mathcal{H}$ is uncountable. Note that when $r^2(\alpha)\, d\alpha$ is a "prior" on h_α, the kernel $\mathcal{K}_{\mathcal{H},r}(x,x')$ can be interpreted as a covariance function commonly used in Gaussian process (GP) models (Williams, 1998; Rasmussen and Williams, 2006). Some Bayesian explanations can then be derived from the connection between SVM and GP, but are beyond the scope of this paper.

3.2 Negation Completeness and Constant Hypotheses

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

$$(P_5) \quad \min_{w\in\mathcal{L}_2(C),b\in\mathbb{R},\xi\in\mathbb{R}^N} \quad \frac{1}{2}\int_C w^2(\alpha)\,d\alpha + C\sum_{i=1}^{N}\xi_i$$
$$\text{s.t.} \quad y_i\left(\int_C w(\alpha)r(\alpha)h_\alpha(x_i)\,d\alpha + b\right) \geq 1-\xi_i, \qquad \text{for } i=1,2,\ldots,N,$$
$$\xi_i \geq 0, \qquad \text{for } i=1,2,\ldots,N.$$

In particular, the classifier obtained after solving (P_2) with $\mathcal{K}_{\mathcal{H}}$ is the same as the classifier obtained after solving (P_5):

$$g(x) = \text{sign}\left(\int_C w(\alpha)r(\alpha)h_\alpha(x)\,d\alpha + b\right). \tag{5}$$

When C is uncountable, it is possible that each hypothesis h_α only takes an infinitesimal weight $\big(w(\alpha)r(\alpha)\,d\alpha\big)$ in the ensemble. Thus, the classifier (5) is very different from those obtained with traditional ensemble learning, and will be discussed further in Subsection 4.2.

Note that the classifier (5) 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 such a classifier 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). Vapnik (1998) showed that even if we add a countably infinite number of constraints to (P_1), infinitely many variables and constraints would be introduced to (P_2). Then, the latter problem would still be difficult to solve.

One remedy is to assume that $\mathcal{H}$ is negation complete, that is,[3]

$$h\in\mathcal{H} \Leftrightarrow (-h)\in\mathcal{H}.$$

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 $\mathcal{H}$ (Rätsch et al., 2002). Following this assumption, the classifier (5) 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, which always predicts $c(x)=1$ for all $x\in\mathcal{X}$. We shall further add a mild assumption that $\mathcal{H}$ contains both c and $(-c)$. Then, the classifier (5) or (1) is indeed equivalent to an ensemble classifier.

We summarize our framework in Algorithm 1. The framework shall generally inherit the profound performance of SVM. Most of the steps in the framework can be done by existing SVM implementations, and the hard part is mostly in obtaining the kernel $\mathcal{K}_{\mathcal{H}}$. In the next sections, we derive some concrete instances using different base hypothesis sets.

4. Stump Kernel

In this section, we present the stump kernel, which embodies infinitely many decision stumps. The decision stump $s_{q,d,\alpha}(x) = q\cdot\text{sign}\big((x)_d-\alpha\big)$ works on the d-th element of x, and classifies x according to $q\in\{-1,+1\}$ and the threshold α (Holte, 1993). It is widely used for ensemble learning because of its simplicity (Freund and Schapire, 1996).

3. We use $(-h)$ to denote the function $(-h)(\cdot) = -(h(\cdot))$.

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 embedding function 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 $\mathcal{H}$.

Algorithm 1: SVM-based framework for infinite ensemble learning

4.1 Formulation and Properties

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

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

We also assume $\mathcal{X} \subseteq (L_1,R_1)\times(L_2,R_2)\times\cdots\times(L_D,R_D)$. Thus, the set $\mathcal{S}$ is negation complete and contains $s_{+1,1,L_1}$ as a constant hypothesis. The stump kernel $\mathcal{K}_{\mathcal{S}}$ defined below can then be used in Algorithm 1 to obtain an infinite ensemble of decision stumps.

Definition 3 *The stump kernel is $\mathcal{K}_{\mathcal{S}}$ with $r(q,d,\alpha_d) = r_{\mathcal{S}} = \frac{1}{2}$,*

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

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

Definition 3 is a concrete instance that follows Definition 1. The details of the derivation are shown in Appendix A. As we shall see further in Section 5, scaling $r_{\mathcal{S}}$ is equivalent to scaling the parameter C in SVM. Thus, without loss of generality, we use $r_{\mathcal{S}} = \frac{1}{2}$ to obtain a cosmetically cleaner kernel function.

The validity of the stump kernel follows directly from Theorem 2 of the general framework. That is, the stump kernel is an inner product in a Hilbert space of some square integrable functions $\varphi(q,d,\alpha_d)$, and it produces a PSD Gram matrix for any set of input vectors $\{x_i\}_{i=1}^N \in \mathcal{X}^N$. Given the ranges (L_d,R_d), the stump kernel is very simple to compute. Furthermore, the ranges are not even necessary in general, because dropping the constant $\Delta_{\mathcal{S}}$ does not affect the classifier obtained from SVM.

Theorem 4 *Solving (P_2) with the stump kernel $\mathcal{K}_{\mathcal{S}}$ 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, equivalent classifiers can be obtained from* (1)*.*

Proof We extend from the results of Berg et al. (1984) to show that $\tilde{\mathcal{K}}_{\mathcal{S}}(x,x')$ is conditionally PSD (CPSD). In addition, because of the constraint $\sum_{i=1}^{N} y_i\lambda_i = 0$, 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 (Schölkopf and Smola, 2002). The proof follows with $\Delta = \Delta_{\mathcal{S}}$. ■

In fact, a kernel $\hat{\mathcal{K}}(x,x') = \tilde{\mathcal{K}}(x,x') + f(x) + f(x')$ with any mapping f is equivalent to $\tilde{\mathcal{K}}(x,x')$ for (P_2) because of the constraint $\sum_{i=1}^{N} y_i\lambda_i = 0$. Now consider another kernel

$$\hat{\mathcal{K}}_{\mathcal{S}}(x,x') = \tilde{\mathcal{K}}_{\mathcal{S}}(x,x') + \sum_{d=1}^{D}(x)_d + \sum_{d=1}^{D}(x')_d = 2\sum_{d=1}^{D}\min((x)_d,(x')_d).$$

We see that $\hat{\mathcal{K}}_{\mathcal{S}}$, $\tilde{\mathcal{K}}_{\mathcal{S}}$, and $\mathcal{K}_{\mathcal{S}}$ are equivalent for (P_2). The former is called the *histogram intersection kernel* (up to a scale of 2) when the elements $(x)_d$ represent generalized histogram counts, and has been successfully used in image recognition applications (Barla et al., 2003; Boughorbel et al., 2005; Grauman and Darrell, 2005). The equivalence demonstrates the usefulness of the stump kernel on histogram-based features, which would be further discussed in Subsection 6.4. A remark here is that our proof for the PSD-ness of $\mathcal{K}_{\mathcal{S}}$ comes directly from the framework, and hence is simpler and more straightforward than the proof of Boughorbel et al. (2005) for the PSD-ness of $\hat{\mathcal{K}}_{\mathcal{S}}$.

The simplified stump kernel is simple to compute, yet useful in the sense of dichotomizing the training set, which comes from the following positive definite (PD) property.

Theorem 5 *(Lin, 2005) Consider training input vectors $\{x_i\}_{i=1}^{N} \in \mathcal{X}^N$. If there exists a dimension d such that $(x_i)_d \neq (x_j)_d$ 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 capacity of the SVM classifiers. Chang and Lin (2001b) showed that when the Gram matrix of the kernel is PD, a hard-margin SVM with such a kernel can always dichotomize the training set perfectly. Keerthi and Lin (2003) then applied the result to show that SVM with the popular Gaussian-RBF kernel $\mathcal{K}(x,x') = \exp\left(-\gamma\|x-x'\|_2^2\right)$ can always dichotomize the training set when $C \to \infty$. We obtain a similar theorem for the stump kernel.

Theorem 6 *Under the assumption of Theorem 5, there exists some $C^* > 0$ such that for all $C \geq C^*$, SVM with $\mathcal{K}_{\mathcal{S}}$ can always dichotomize the training set $\{(x_i,y_i)\}_{i=1}^{N}$.*

We make two remarks here. First, although the assumption of Theorem 6 is mild in practice, there are still some data sets that do not have this property. An example is the famous XOR data set (Figure 1). We can see that every possible decision stump makes 50% of errors on the training input vectors. Thus, AdaBoost and LPBoost would terminate with one bad decision stump in the ensemble. Similarly, SVM with the stump kernel cannot dichotomize this training set perfectly, regardless of the choice of C. Such a problem is inherent in any ensemble model that combines decision stumps, because the model belongs to the family of generalized additive models (Hastie and Tibshirani, 1990; Hastie et al., 2001), and hence cannot approximate non-additive target functions well.

Second, although Theorem 6 indicates how the stump kernel can be used to dichotomize the training set perfectly, the classifier obtained usually overfits to noise (Keerthi and Lin, 2003). For

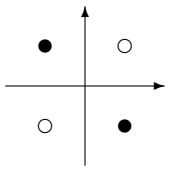


Figure 1: The XOR data set

the Gaussian-RBF kernel, it has been known that SVM with reasonable parameter selection provides suitable regularization and achieves good generalization performance even in the presence of noise (Keerthi and Lin, 2003; Hsu et al., 2003). We observe similar experimental results for the stump kernel (see Section 7).

4.2 Averaging Ambiguous Stumps

We have discussed 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 all the predictions within each group. This property is formalized in the following theorem.

Theorem 7 *Consider two ambiguous $h_\alpha, h_\beta \in \mathcal{H}$. If the kernel $\mathcal{K}_\mathcal{H}$ is used in Algorithm 1, the optimal w of (P_5) satisfies $\frac{w(\alpha)}{r(\alpha)} = \frac{w(\beta)}{r(\beta)}$.*

Proof The optimality condition between (P_1) and (P_2) leads to

$$\frac{w(\alpha)}{r(\alpha)} = \sum_{i=1}^{N} \lambda_i h_\alpha(x_i) = \sum_{i=1}^{N} \lambda_i h_\beta(x_i) = \frac{w(\beta)}{r(\beta)}. \qquad \blacksquare$$

If $w(\alpha)$ is nonzero, $w(\beta)$ would also be nonzero, which means both h_α and h_β are included in the ensemble. As a consequence, for each group of mutually ambiguous hypotheses, our framework considers the average prediction of all hypotheses as the consensus output.

The averaging process constructs a smooth representative for each group. In the following theorem, we demonstrate this behavior with the stump kernel, and show how the decision stumps group together in the final ensemble classifier.

Theorem 8 *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*

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

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

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

Proof First, for any fixed q and d, a simple integration shows that

$$\int_{(\tilde{x})_{d,a}}^{(\tilde{x})_{d,a+1}} s_{q,d,\alpha}(x)\, d\alpha \quad = \quad \Big((\tilde{x})_{d,a+1} - (\tilde{x})_{d,a}\Big)\hat{s}_{q,d,a}(x).$$

In addition, note that for all $\alpha \in \Big((\tilde{x})_{d,a}, (\tilde{x})_{d,a+1}\Big)$, $\hat{s}_{q,d,a}(x_i) = s_{q,d,\alpha}(x_i)$. Thus,

$$\begin{aligned}
& \int_{L_d}^{R_d} \Big(r(q,d,\alpha) s_{q,d,\alpha}(x_i)\Big)\Big(r(q,d,\alpha) s_{q,d,\alpha}(x)\Big)\, d\alpha \\
= & \sum_{a=0}^{A_d} \int_{(\tilde{x})_{d,a}}^{(\tilde{x})_{d,a+1}} \left(\frac{1}{2} s_{q,d,\alpha}(x_i)\right)\left(\frac{1}{2} s_{q,d,\alpha}(x)\right) d\alpha \\
= & \sum_{a=0}^{A_d} \frac{1}{4} \hat{s}_{q,d,a}(x_i) \int_{(\tilde{x})_{d,a}}^{(\tilde{x})_{d,a+1}} s_{q,d,\alpha}(x)\, d\alpha \\
= & \sum_{a=0}^{A_d} \frac{1}{4} \Big((\tilde{x})_{d,a+1} - (\tilde{x})_{d,a}\Big) \hat{s}_{q,d,a}(x_i) \hat{s}_{q,d,a}(x).
\end{aligned}$$

The theorem can be proved by summing over all q and d. ■

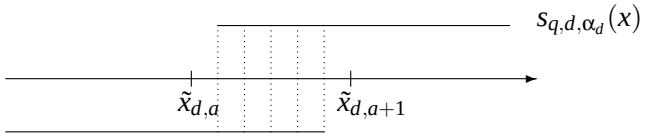
As shown in Figure 2, the function $\hat{s}_{q,d,a}$ is a smoother variant of the decision stump. Theorem 8 indicates that the infinite ensemble of decision stumps produced by our framework is equivalent to a finite ensemble of data-dependent and smoother variants. Another view of $\hat{s}_{q,d,a}$ is that they are continuous piecewise linear functions (order-2 splines) with knots defined on the training features (Hastie et al., 2001). Then, Theorem 8 indicates that an infinite ensemble of decision stumps can be obtained by fitting an additive model of finite size using these special splines as the bases. Note that although the fitting problem is of finite size, the number of possible splines can grow as large as $O(ND)$, which can sometimes be too large for iterative algorithms such as backfitting (Hastie et al., 2001). On the other hand, our SVM-based framework with the stump kernel can be thought as a route to solve this special spline fitting problem efficiently via the kernel trick.

As shown in the proof of Theorem 8, the averaged stump $\hat{s}_{q,d,a}$ represents the group of ambiguous decision stumps with $\alpha_d \in \Big((\tilde{x})_{d,a}, (\tilde{x})_{d,a+1}\Big)$. When the group is larger, $\hat{s}_{q,d,a}$ becomes smoother. Traditional ensemble learning algorithms like AdaBoost or LPBoost rely on a base learner to choose one decision stump as the only representative within each group, and the base learner usually returns the middle stump $m_{q,d,a}$. As shown in Figure 2, the threshold of the middle stump is at the mean of $(\tilde{x})_{d,a}$ and $(\tilde{x})_{d,a+1}$. 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 $\hat{s}_{q,d,a}$ has a concrete weight in the ensemble.

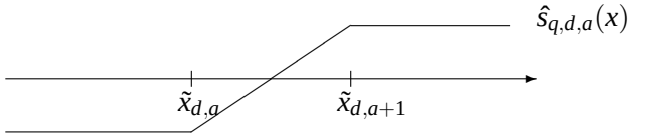
5. Perceptron Kernel

In this section, we extend the stump kernel to the perceptron kernel, which embodies infinitely many perceptrons. A perceptron is a linear threshold classifier of the form

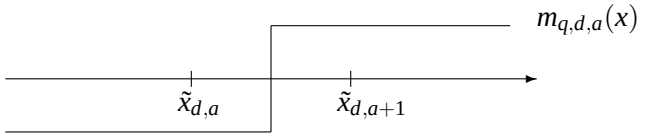
$$p_{\theta,\alpha}(x) = \mathrm{sign}\left(\theta^T x - \alpha\right).$$



(a) a group of ambiguous decision stumps s_{q,d,α_d} with $\alpha_d \in \left((\tilde{x})_{d,a}, (\tilde{x})_{d,a+1}\right)$



(b) SVM-based infinite ensemble learning uses the consensus: the averaged stump $\hat{s}_{q,d,a}$



(c) Base learners for AdaBoost and LPBoost usually only consider the middle stump $m_{q,d,a}$

Figure 2: The averaged stump and the middle stump

It is a basic theoretical model for a neuron, and is very important for building neural networks (Haykin, 1999).

To construct the perceptron kernel, 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 $\mathcal{X}$ is within the interior of $\mathcal{B}(R)$, where $\mathcal{B}(R)$ is a ball of radius R centered at the origin in $\mathbb{R}^D$. Then, the set $\mathcal{P}$ is negation complete, and contains a constant hypothesis $p_{e_1,-R}$ where $e_1 = (1,0,\ldots,0)^T$. Thus, the perceptron kernel $\mathcal{K}_{\mathcal{P}}$ defined below can be used in Algorithm 1 to obtain an infinite ensemble of perceptrons.

Definition 9 *Let*

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

where the operator angle$\langle\cdot,\cdot\rangle$ *is the angle between two vectors, and the integrals are calculated with uniform measure on the surface* $\|\theta\|_2 = 1$. *The perceptron kernel is* $\mathcal{K}_{\mathcal{P}}$ *with* $r(\theta,\alpha) = r_{\mathcal{P}}$,

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

where the constants $r_{\mathcal{P}} = (2\Xi_D)^{-\frac{1}{2}}$ *and* $\Delta_{\mathcal{P}} = \Theta_D \Xi_D^{-1} R$.

The details are shown in Appendix A. With the perceptron kernel, we can construct an infinite ensemble of perceptrons. Such an ensemble is equivalent to a neural network with one hidden layer, infinitely many hidden neurons, and the hard-threshold activation functions. Williams (1998) built an infinite neural network with either the sigmoidal or the Gaussian activation function through computing the corresponding covariance function for GP models. Analogously, our approach returns an infinite neural network with hard-threshold activation functions (ensemble of perceptrons) through

computing the perceptron kernel for SVM. Williams (1998) mentioned that "*Paradoxically, it may be easier to carry out Bayesian prediction with infinite networks rather than finite ones.*" Similar claims can be made with ensemble learning.

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 10 *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, SVM with the perceptron kernel can also dichotomize the training set perfectly, which comes from the usefulness of the simplified perceptron kernel $\tilde{\mathcal{K}}_{\mathcal{P}}$ in interpolation.

Theorem 11 *(Micchelli, 1986) Consider input vectors $\{x_i\}_{i=1}^N \in \mathcal{X}^N$, and the perceptron kernel $\mathcal{K}_{\mathcal{P}}$ in Definition 9. If $x_i \neq x_j$ for all $i \neq j$, then the Gram matrix of $\mathcal{K}_{\mathcal{P}}$ is PD.*

Then, similar to Theorem 6, we get the following result.

Theorem 12 *If $x_i \neq x_j$ for all $i \neq j$, there exists some $C^* > 0$ such that for all $C \geq C^*$, SVM with $\mathcal{K}_{\mathcal{P}}$ can always dichotomize the training set $\{(x_i,y_i)\}_{i=1}^N$.*

Another important property, called *scale-invariance*, accompanies the simplified perceptron kernel, which was also named the *triangular kernel* by Fleuret and Sahbi (2003). They proved that when the kernel is used in the hard-margin SVM, scaling all training input vectors x_i by some positive γ does not change the optimal solution.

In fact, in the soft-margin SVM, a well-known result is that scaling the Gram matrix K by some $\gamma > 0$ is equivalent to scaling C by γ in (P_2). Because the simplified perceptron kernel $\tilde{\mathcal{K}}_{\mathcal{P}}$ satisfies $\gamma\tilde{\mathcal{K}}_{\mathcal{P}}(x,x') = \tilde{\mathcal{K}}_{\mathcal{P}}(\gamma x,\gamma x')$, the effect of scaling training examples can be equivalently performed with the parameter selection step on C. That is, when C is selected reasonably, there is no need to explicitly have a scaling parameter γ.

Recall that we construct the perceptron kernel (and the stump kernel) with an embedding constant $r_{\mathcal{P}}$ (and $r_{\mathcal{S}}$), and from Definition 1, multiplying the constant by $\sqrt{\gamma} > 0$ is equivalent to scaling the Gram matrix K by γ. Thus, when C is selected reasonably, there is also no need to explicitly try different $r_{\mathcal{P}}$ or $r_{\mathcal{S}}$ for these two kernels. We will further discuss the benefits of this property in Subsection 6.4.

6. Laplacian-RBF Kernel

In the previous sections, we applied Definition 1 on some simple base hypothesis sets. Next, we show how complex hypothesis sets can also be embedded in a kernel by suitably combining the kernels that embody simpler sets. We will introduce two useful tools: summation and multiplication. The tools would eventually allow us to embed infinitely many decision trees in a kernel. Interestingly, the kernel obtained is equivalent to the well-known Laplacian-RBF kernel in some parameters.

6.1 Summation: Embedding Multiple Sets of Hypotheses

Summation can be used to embed multiple sets of hypotheses altogether. For example, given kernels $\mathcal{K}_{\mathcal{H}_1}$ and $\mathcal{K}_{\mathcal{H}_2}$, their summation

$$\mathcal{K}(x,x') = \mathcal{K}_{\mathcal{H}_1}(x,x') + \mathcal{K}_{\mathcal{H}_2}(x,x')$$

embodies both $\mathcal{H}_1$ and $\mathcal{H}_2$. In other words, if we use $\mathcal{K}(x,x')$ in Algorithm 1, we could obtain an ensemble classifier over $\mathcal{H}_1 \cup \mathcal{H}_2$ when the union is negation complete and contains a constant hypothesis.

In traditional ensemble learning, when multiple sets of hypotheses are considered altogether, it is usually necessary to call a base learner for each set. On the other hand, our framework only requires a simple summation on the kernel evaluations. In fact, as shown in the next theorem, our framework can be applied to work with any countable sets of hypotheses, which may not be an easy task for traditional ensemble learning algorithms.

Theorem 13 *Assume that the kernels $\mathcal{K}_{\mathcal{H}_1}, \ldots, \mathcal{K}_{\mathcal{H}_J}$ are defined for some $J \in \mathbb{N} \bigcup \{\infty\}$ with sets of hypotheses $\mathcal{H}_1, \ldots, \mathcal{H}_J$, respectively. Then, let*

$$\mathcal{K}(x,x') = \sum_{j=1}^{J} \mathcal{K}_{\mathcal{H}_j}(x,x').$$

If $\mathcal{K}(x,x')$ exists for all $x,x' \in \mathcal{X}$, and $\mathcal{H} = \bigcup_{j=1}^{J} \mathcal{H}_j$ is negation complete and contains a constant hypothesis, Algorithm 1 using $\mathcal{K}(x,x')$ outputs an ensemble classifier over $\mathcal{H}$.

Proof The theorem comes from the following result in mathematical analysis: any countable direct sum over Hilbert spaces is a Hilbert space (Reed and Simon, 1980, Example 5). Lin (2005, Theorem 6) showed the details of the proof. ■

A remark on Theorem 13 is that we do not intend to define a kernel with $\mathcal{H}$ directly. Otherwise we need to choose suitable $\mathcal{C}$ and r first, which may not be an easy task for such a complex hypothesis set. Using the summation of the kernels, on the other hand, allow us to obtain an ensemble classifier over the full union with less efforts.

6.2 Multiplication: Performing Logical Combination of Hypotheses

It is known that we can combine two kernels by point-wise multiplication to form a new kernel (Schölkopf and Smola, 2002). When the two kernels are associated with base hypothesis sets, a natural question is: what hypothesis set is embedded in the new kernel?

Next, let output $+1$ represent logic TRUE and -1 represent logic FALSE. We show that multiplication can be used to perform common logical combinations on the hypotheses.

Theorem 14 *For two sets of hypotheses $\mathcal{H}_1 = \{h_\alpha : \alpha \in \mathcal{C}_1\}$ and $\mathcal{H}_2 = \{h_\beta : \beta \in \mathcal{C}_2\}$, define*

$$\mathcal{H} = \left\{h_{\alpha,\beta} : h_{\alpha,\beta}(x) = -h_\alpha(x) \cdot h_\beta(x), \alpha \in \mathcal{C}_1, \beta \in \mathcal{C}_2\right\}.$$

In addition, let $r(\alpha,\beta) = r_1(\alpha) r_2(\beta)$. Then,

$$\mathcal{K}_{\mathcal{H},r}(x,x') = \mathcal{K}_{\mathcal{H}_1,r_1}(x,x') \cdot \mathcal{K}_{\mathcal{H}_2,r_2}(x,x')$$

for all $x,x' \in \mathcal{X}$.

The proof simply follows from Definition 1. Note that when representing logic, the combined hypothesis $h_{\alpha,\beta}$ is the XOR operation on h_α and h_β. More complicated results about other operations can be introduced under a mild assumption called *neutrality*.

Definition 15 *A set of hypothesis $\mathcal{H} = \{h_\alpha : \alpha \in \mathcal{C}\}$ is neutral to $\mathcal{X}$ with a given r if and only if for all $x \in \mathcal{X}$, $\int_{\alpha\in\mathcal{C}} h_\alpha(x) r^2(\alpha)\, d\alpha = 0$.*

Note that for a negation complete set $\mathcal{H}$, neutrality is usually a mild assumption (e.g., by assigning the same r for h_α and $-h_\alpha$). We can easily verify that the set of decision stumps in Definition 3 and the set of perceptrons in Definition 9 are both neutral.

Theorem 16 *For two sets of hypotheses $\mathcal{H}_1 = \{h_\alpha : \alpha \in \mathcal{C}_1\}$ and $\mathcal{H}_2 = \{h_\beta : \beta \in \mathcal{C}_2\}$, define*

$$\mathcal{H} \quad = \quad \left\{h_{q,\alpha,\beta} : h_{q,\alpha,\beta}(x) = q \cdot \min\left(h_\alpha(x), h_\beta(x)\right), \alpha \in \mathcal{C}_1, \beta \in \mathcal{C}_2, q \in \{-1,+1\}\right\}.$$

Assume that $\mathcal{H}_1$ and $\mathcal{H}_2$ are neutral with r_1 and r_2, respectively, and both integrals

$$\Delta_1 = \int_{\alpha\in\mathcal{C}_1} r_1^2(\alpha)\, d\alpha,\ \Delta_2 = \int_{\beta\in\mathcal{C}_2} r_2^2(\beta)\, d\beta$$

are finite. In addition, let $r(q,\alpha,\beta) = \sqrt{2} r_1(\alpha) r_2(\beta)$. Then,

$$\mathcal{K}_{\mathcal{H},r}(x,x') = \left(\mathcal{K}_{\mathcal{H}_1,r_1}(x,x') + \Delta_1\right) \cdot \left(\mathcal{K}_{\mathcal{H}_2,r_2}(x,x') + \Delta_2\right)$$

for all $x, x' \in \mathcal{X}$. Furthermore, $\mathcal{H}$ is neutral to $\mathcal{X}$ with r.

Proof Because $h_\alpha(x), h_\beta(x) \in \{-1,+1\}$,

$$h_{+1,\alpha,\beta}(x) = \frac{1}{2}\left(h_\alpha(x)h_\beta(x) + h_\alpha(x) + h_\beta(x) - 1\right).$$

Then,

$$\begin{aligned}
& \mathcal{K}_{\mathcal{H},r}(x,x') \\
= \ & 2\int h_{+1,\alpha,\beta}(x) h_{+1,\alpha,\beta}(x') r^2(\alpha,\beta)\, d\beta\, d\alpha \\
= \ & \frac{1}{2}\int \left(h_\alpha(x)h_\beta(x) + h_\alpha(x) + h_\beta(x) - 1\right)\left(h_\alpha(x')h_\beta(x') + h_\alpha(x') + h_\beta(x') - 1\right) r^2(\alpha,\beta)\, d\beta\, d\alpha \\
= \ & \int \left(h_\alpha(x)h_\beta(x)h_\alpha(x')h_\beta(x') + h_\alpha(x)h_\alpha(x') + h_\beta(x)h_\beta(x') + 1\right) r_1^2(\alpha) r_2^2(\beta)\, d\beta\, d\alpha \qquad (6) \\
= \ & \left(\mathcal{K}_{\mathcal{H}_1,r_1}(x,x') + \Delta_1\right) \cdot \left(\mathcal{K}_{\mathcal{H}_2,r_2}(x,x') + \Delta_2\right).
\end{aligned}$$

Note that (6) comes from the neutrality assumption, which implies that during integration, the cross-terms like

$$\int h_\alpha(x) h_\beta(x') r_1^2(\alpha) r_2^2(\beta)\, d\alpha\, d\beta$$

are all 0. Neutrality of $\mathcal{H}$ follows from the symmetry in q. ■

The arithmetic operation $(+1 \cdot \min)$ is equivalent to the AND operation when the outputs represent logic, and hence $(-1 \cdot \min)$ represents the NAND operation. If $\mathcal{H}_1$ and $\mathcal{H}_2$ are negation complete, the NOT operation is implicit in the original sets, and hence OR can be equivalently performed through $\mathrm{OR}(a,b) = \mathrm{NAND}(\mathrm{NOT}(a), \mathrm{NOT}(b))$.

6.3 Stump Region Kernel, Decision Tree Kernel, and Laplacian-RBF Kernel

Next, we use the stump kernel to demonstrate the usefulness of summation and multiplication. When $\mathcal{H}_1 = \mathcal{H}_2 = \mathcal{S}$, the resulting $\mathcal{K}_{\mathcal{H}}$ from Theorem 16 embodies AND/OR combinations of two decision stumps in $\mathcal{S}$. Extending this concept, we get the following new kernels.

Definition 17 *The L-level stump region kernel $\mathcal{K}_{\mathcal{I}_L}$ is recursively defined by*

$$\begin{aligned} \mathcal{K}_{\mathcal{I}_1}(x,x') &= \mathcal{K}_{\mathcal{S}}(x,x') + \Delta_{\mathcal{S}},\ \Delta_1 = 2\Delta_{\mathcal{S}}, \\ \mathcal{K}_{\mathcal{I}_{L+1}}(x,x') &= \left(\mathcal{K}_{\mathcal{I}_L}(x,x') + \Delta_L\right)\left(\mathcal{K}_{\mathcal{S}}(x,x') + \Delta_{\mathcal{S}}\right),\ \Delta_{L+1} = 2\Delta_L\Delta_{\mathcal{S}} \text{ for } L \in \mathbb{N}. \end{aligned}$$

If we construct a kernel from $\{c,-c\}$ with $r = \sqrt{\frac{1}{2}\Delta_{\mathcal{S}}}$ on each hypothesis, we can see that the constant $\Delta_{\mathcal{S}}$ is also a neutral kernel. Since neutrality is preserved by summation, the kernel $\mathcal{K}_{\mathcal{I}_1}$ is neutral as well. By repeatedly applying Theorem 16 and maintaining Δ_L as the constant associated with $\mathcal{I}_L$, we see that $\mathcal{K}_{\mathcal{I}_L}$ embodies all possible AND/OR combinations of L decision stumps in $\mathcal{S}$. We call these hypotheses the *L-level stump regions*.

Note that we can solve the recurrence and get

$$\mathcal{K}_{\mathcal{I}_L}(x,x') = 2^L\Delta_{\mathcal{S}}^L \sum_{\ell=1}^{L} \left(\frac{\mathcal{K}_{\mathcal{S}}(x,x') + \Delta_{\mathcal{S}}}{2\Delta_{\mathcal{S}}}\right)^{\ell}, \text{ for } L \in \mathbb{N}.$$

Then, by applying Theorem 13, we obtain an ensemble classifier over stump regions of any level.

Theorem 18 *For $0 < \gamma < \frac{1}{\Delta_{\mathcal{S}}}$, the infinite stump region (decision tree) kernel*

$$\mathcal{K}_{\mathcal{I}}(x,x') = \exp\left(\gamma \cdot \left(\mathcal{K}_{\mathcal{S}}(x,x') + \Delta_{\mathcal{S}}\right)\right) - 1$$

can be applied to Algorithm 1 to obtain an ensemble classifier over $\mathcal{I} = \bigcup_{L=1}^{\infty} \mathcal{I}_L$.

Proof By Taylor's series expansion of $\exp(\varepsilon)$ near $\varepsilon = 0$, we get

$$\begin{aligned} \mathcal{K}_{\mathcal{I}}(x,x') &= \sum_{L=1}^{\infty} \frac{\gamma^L}{L!}\left(\mathcal{K}_{\mathcal{S}}(x,x') + \Delta_{\mathcal{S}}\right)^L \\ &= \gamma\mathcal{K}_{\mathcal{I}_1}(x,x') + \sum_{L=2}^{\infty} \frac{\gamma^L}{L!}\left(\mathcal{K}_{\mathcal{I}_L}(x,x') - 2\Delta_{\mathcal{S}}\,\mathcal{K}_{\mathcal{I}_{L-1}}(x,x')\right) \\ &= \sum_{L=1}^{\infty} \frac{\gamma^L}{L!}\mathcal{K}_{\mathcal{I}_L}(x,x') - \sum_{L=1}^{\infty} \frac{\gamma^{L+1}}{(L+1)!} 2\Delta_{\mathcal{S}}\,\mathcal{K}_{\mathcal{I}_L}(x,x') \\ &= \sum_{L=1}^{\infty} \left(\frac{\gamma^L}{L!} - \frac{\gamma^{L+1}2\Delta_{\mathcal{S}}}{(L+1)!}\right)\mathcal{K}_{\mathcal{I}_L}(x,x'). \end{aligned}$$

Note that $\tau_L = \frac{\gamma^L}{L!} - \frac{\gamma^{L+1}2\Delta_{\mathcal{S}}}{(L+1)!} > 0$ for all $L \geq 1$ if and only if $0 < \gamma < \frac{1}{\Delta_{\mathcal{S}}}$. The desired result simply follows Theorem 13 by scaling the r functions of each $\mathcal{K}_{\mathcal{I}_L}$ by $\sqrt{\tau_L}$. ■

The set of stump regions of any level contains all AND/OR combinations of decision stumps. It is not hard to see that every stump region can be represented by recursive axis-parallel partitions

that output $\{-1,+1\}$, that is, a decision tree (Quinlan, 1986; Hastie et al., 2001). In addition, we can view the nodes of a decision tree as logic operations:

$$\begin{aligned} & \mathsf{tree} \\ = \; & \mathrm{OR}(\mathrm{AND}(\text{root node condition}, \mathsf{left}), \mathrm{AND}(\mathrm{NOT}(\text{root node condition}), \mathsf{right})). \end{aligned}$$

By recursively replacing each root node condition with a decision stump, we see that every decision tree can be represented as a stump region hypothesis. Thus, the set $\mathcal{T}$ that contains stump regions of any level is the same as the set of all possible decision trees, which leads to the name *decision tree kernel*.[4]

Decision trees are popular for ensemble learning, but traditional algorithms can only deal with trees of finite levels (Breiman, 1999; Dietterich, 2000). On the other hand, when the decision tree kernel $\mathcal{K}_{\mathcal{T}}$ is plugged into our framework, it allows us to actually build an infinite ensemble over decision trees of arbitrary levels.

Note that the decision tree kernel $\mathcal{K}_{\mathcal{T}}(x,x')$ is of the form

$$\kappa_1 \exp\left(-\kappa_2 \left\|x-x'\right\|_1\right) + \kappa_3$$

where $\kappa_1, \kappa_2, \kappa_3$ are constants and κ_1, κ_2 are positive. We mentioned in Section 4 that scaling the kernel with κ_1 is equivalent to scaling the soft-margin parameter C in SVM, and in Theorem 4 that dropping κ_3 does not affect the solution obtained from SVM. Then, the kernel $\mathcal{K}_{\mathcal{T}}(x,x')$ is similar to the Laplacian-RBF kernel $\mathcal{K}_{\mathcal{L}}(x,x') = \exp(-\gamma\|x-x'\|_1)$. This result is a novel interpretation of the Laplacian-RBF kernel: under suitable parameters, SVM with the Laplacian-RBF kernel allows us to obtain an infinite ensemble classifier over decision trees of any level.[5]

Not surprisingly, when all training input vectors x_i are distinct (Micchelli, 1986; Baxter, 1991), the Gram matrix of $\mathcal{K}_{\mathcal{L}}$ (and hence $\mathcal{K}_{\mathcal{T}}$) is PD. Then, the Laplacian-RBF kernel and the decision tree kernel could be used to dichotomize the training set perfectly.

6.4 Discussion on Radial Basis Function Kernels

Note that the stump kernel, the perceptron kernel, the Laplacian-RBF kernel, and the Gaussian-RBF kernel are all radial basis functions. They can all be used to dichotomize the training set perfectly under mild conditions, while the first three connect to explanations from an ensemble perspective. Next, we compare two properties of these kernels, and discuss their use in SVM applications.

First, we can group these kernels by the distance metrics they use. The stump kernel and the Laplacian-RBF kernel deal with the ℓ_1-norm distance between input vectors, while the others work on the ℓ_2-norm distance. An interesting property of using the ℓ_2-norm distance is the invariance to rotations. From the construction of the perceptron kernel, we can see how the rotation invariance is obtained from an ensemble point-of-view. The transformation vectors θ in perceptrons represent the rotation, and rotation invariance comes from embedding all possible θ uniformly in the kernel.

4. We use the name decision tree kernel for $\mathcal{K}_{\mathcal{T}}$ in Theorem 18 because the kernel embodies an infinite number of decision tree "hypotheses" and can be used in our framework to construct an infinite ensemble of decision trees. As pointed out by a reviewer, however, the kernel is derived in a particular way, which makes the metric of the underlying feature space different from the metrics associated with common decision tree "algorithms."

5. Note that the techniques in Theorem 18 can be coupled with Theorem 14 to show that Laplacian-RBF kernel with any $\gamma > 0$ embodies XOR stump regions (a special type of decision tree) of any level. We emphasize on the AND-OR stump regions here to connect better to general decision trees.

Some applications, however, may not desire rotation invariance. For example, when representing an image with color histograms, rotation could mix up the information in each color component. Chapelle et al. (1999) showed some successful results with the Laplacian-RBF kernel on this application. In Subsection 4.1, we have also discussed some image recognition applications using the histogram intersection kernel, which is equivalent to the stump kernel, on histogram-based features. Gene expression analysis, as demonstrated by Lin and Li (2005b), is another area that the stump kernel could be helpful.

Second, we can group kernels by whether they are scale-invariant (see also Section 5). The simplified stump kernel and the simplified perceptron kernel are scale-invariant, which means that C is the only parameter that needs to be determined. On the other hand, different combinations of (γ, C) need to be considered for the Gaussian-RBF kernel or the Laplacian-RBF kernel during parameter selection (Keerthi and Lin, 2003). Thus, SVM with the simplified stump kernel or the simplified perceptron kernel enjoys an advantage on speed during parameter selection. As we will see in Section 7.2, experimentally they perform similarly to the Gaussian-RBF kernel on many data sets. Thus, SVM applications that consider speed as an important factor may benefit from using the simplified stump kernel or the simplified perceptron kernel.

7. Experiments

We first compare our SVM-based infinite ensemble learning framework with AdaBoost and LPBoost using decision stumps, perceptrons, or decision trees as the base hypothesis set. The simplified stump kernel (SVM-Stump), the simplified perceptron kernel (SVM-Perc), and the Laplacian-RBF kernel (SVM-Dec) are plugged into Algorithm 1 respectively. We also compare SVM-Stump, SVM-Perc, and SVM-Dec with SVM-Gauss, which is SVM with the Gaussian-RBF kernel.

The deterministic decision stump algorithm (Holte, 1993), the random coordinate descent perceptron algorithm (Li and Lin, 2007), and the C4.5 decision tree algorithm (Quinlan, 1986) are taken as base learners in AdaBoost and LPBoost for the corresponding base hypothesis set. For perceptrons, we use the RCD-bias setting with 200 epochs of training; for decision trees, we take the pruned tree with the default settings of C4.5. All base learners above have been shown to work reasonably well with boosting in literature (Freund and Schapire, 1996; Li and Lin, 2007).

We discussed in Subsection 4.2 that a common implementation of AdaBoost-Stump and LPBoost-Stump only chooses the middle stumps. For further comparison, we include all the middle stumps in a set $\mathcal{M}$, 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 (4) becomes a summation when computed with the counting measure. We test our framework with this kernel, and call it SVM-Mid.

LIBSVM 2.8 (Chang and Lin, 2001a) is adopted as the soft-margin SVM solver, with a suggested procedure that selects a suitable parameter with a five-fold cross validation on the training set (Hsu et al., 2003). For SVM-Stump, SVM-Mid, and SVM-Perc, the parameter $\log_2 C$ is searched within $\{-17, -15, \ldots, 3\}$, and for SVM-Dec and SVM-Gauss, the parameters $(\log_2 \gamma, \log_2 C)$ are searched within $\{-15, -13, \ldots, 3\} \times \{-5, -3, \ldots, 15\}$. We use different search ranges for $\log_2 C$ because the numerical ranges of the kernels could be quite different. After the parameter selection procedure, a new model is trained using the whole training set, and the generalization ability is evaluated on an unseen test set.

For boosting algorithms, we conduct the parameter selection procedure similarly. The parameter $\log_2 C$ of LPBoost is also searched within $\{-17, -15, \ldots, 3\}$. For AdaBoost, the parameter T

data set	number of training examples	number of test examples	number of features
twonorm	300	3000	20
twonorm-n	300	3000	20
threenorm	300	3000	20
threenorm-n	300	3000	20
ringnnorm	300	3000	20
ringnorm-n	300	3000	20
australian	414	276	14
breast	409	274	10
german	600	400	24
heart	162	108	13
ionosphere	210	141	34
pima	460	308	8
sonar	124	84	60
votes84	261	174	16
a1a	1605	30956	123
splice	1000	2175	60
svmguide1	3089	4000	4
w1a	2477	47272	300

Table 1: Summarized information of the data sets used

is searched within $\{10, 20, \ldots, 1500\}$. Note that because LPBoost can be slow when the ensemble size is too large (Demiriz et al., 2002), we set a stopping criterion to generate at most 1000 columns (hypotheses) in order to obtain an ensemble within a reasonable amount of time.

The three artificial data sets from Breiman (1999) (twonorm, threenorm, and ringnorm) are generated with training set size 300 and test set size 3000. We create three more data sets (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 data sets from the UCI repository (Hettich et al., 1998): australian, breast, german, heart, ionosphere, pima, sonar, and votes84. Their feature elements are scaled to $[-1, 1]$. We randomly pick 60% of the examples for training, and the rest for testing. For the data sets above, we compute the means and the standard errors of the results over 100 runs. In addition, four larger real-world data sets are used to test the validity of the framework for large-scale learning. They are a1a (Hettich et al., 1998; Platt, 1999), splice (Hettich et al., 1998), svmguide1 (Hsu et al., 2003), and w1a (Platt, 1999).[6] Each of them comes with a benchmark test set, on which we report the results. Some information of the data sets used is summarized in Table 1.

6. These data sets are downloadable on tools page of LIBSVM (Chang and Lin, 2001a).

data set	SVM-Stump	SVM-Mid	AdaBoost-Stump	LPBoost-Stump
twonorm	$\mathbf{2.86 \pm 0.04}$	3.10 ± 0.04	5.02 ± 0.06	5.58 ± 0.07
twonorm-n	$\mathbf{3.08 \pm 0.06}$	3.29 ± 0.05	12.7 ± 0.17	17.9 ± 0.19
threenorm	$\mathbf{17.7 \pm 0.10}$	18.6 ± 0.12	22.1 ± 0.12	24.1 ± 0.15
threenorm-n	$\mathbf{19.0 \pm 0.14}$	19.6 ± 0.13	26.1 ± 0.17	30.3 ± 0.16
ringnorm	$\mathbf{3.97 \pm 0.07}$	5.30 ± 0.07	10.1 ± 0.14	10.3 ± 0.14
ringnorm-n	$\mathbf{5.56 \pm 0.11}$	7.03 ± 0.14	19.6 ± 0.20	22.4 ± 0.21
australian	$\mathbf{14.4 \pm 0.21}$	15.9 ± 0.18	$\mathbf{14.2 \pm 0.18}$	19.8 ± 0.24
breast	3.11 ± 0.08	$\mathbf{2.77 \pm 0.08}$	4.41 ± 0.10	4.79 ± 0.12
german	$\mathbf{24.7 \pm 0.18}$	$\mathbf{24.9 \pm 0.17}$	25.4 ± 0.19	31.6 ± 0.20
heart	$\mathbf{16.4 \pm 0.27}$	19.1 ± 0.35	19.2 ± 0.35	24.4 ± 0.39
ionosphere	$\mathbf{8.13 \pm 0.17}$	$\mathbf{8.37 \pm 0.20}$	11.3 ± 0.25	11.5 ± 0.24
pima	$\mathbf{24.1 \pm 0.23}$	$\mathbf{24.4 \pm 0.23}$	24.8 ± 0.23	31.0 ± 0.24
sonar	$\mathbf{16.6 \pm 0.42}$	18.0 ± 0.37	19.4 ± 0.38	19.8 ± 0.37
votes84	4.76 ± 0.14	4.76 ± 0.14	$\mathbf{4.27 \pm 0.15}$	5.87 ± 0.16
a1a	16.2	16.3	**16.0**	16.3
splice	6.21	6.71	**5.75**	8.78
svmguide1	**2.92**	3.20	3.35	4.50
w1a	**2.09**	2.26	2.18	2.79

Table 2: Test error (%) of several ensemble learning algorithms using decision stumps

data set	SVM-Perc	AdaBoost-Perc	LPBoost-Perc
twonorm	$\mathbf{2.55 \pm 0.03}$	3.11 ± 0.04	3.52 ± 0.05
twonorm-n	$\mathbf{2.75 \pm 0.05}$	4.53 ± 0.10	6.89 ± 0.11
threenorm	$\mathbf{14.6 \pm 0.08}$	17.3 ± 0.11	18.2 ± 0.11
threenorm-n	$\mathbf{16.3 \pm 0.10}$	20.0 ± 0.18	22.1 ± 0.13
ringnorm	$\mathbf{2.46 \pm 0.04}$	36.3 ± 0.14	37.4 ± 0.13
ringnorm-n	$\mathbf{3.50 \pm 0.09}$	37.8 ± 0.20	39.1 ± 0.15
australian	$\mathbf{14.5 \pm 0.17}$	15.7 ± 0.16	16.4 ± 0.17
breast	$\mathbf{3.23 \pm 0.08}$	3.49 ± 0.10	3.80 ± 0.10
german	$\mathbf{24.6 \pm 0.20}$	25.0 ± 0.18	26.4 ± 0.21
heart	$\mathbf{17.6 \pm 0.31}$	$\mathbf{18.2 \pm 0.32}$	19.8 ± 0.32
ionosphere	$\mathbf{6.40 \pm 0.20}$	11.4 ± 0.23	12.1 ± 0.25
pima	$\mathbf{23.5 \pm 0.21}$	24.8 ± 0.20	26.4 ± 0.19
sonar	$\mathbf{15.6 \pm 0.40}$	19.8 ± 0.43	22.5 ± 0.47
votes84	$\mathbf{4.43 \pm 0.14}$	$\mathbf{4.37 \pm 0.16}$	4.92 ± 0.16
a1a	**15.7**	20.0	18.6
splice	**10.4**	13.7	14.7
svmguide1	**3.10**	3.28	3.62
w1a	**1.91**	2.35	2.13

Table 3: Test error (%) of several ensemble learning algorithms using perceptrons

data set	SVM-Dec	AdaBoost-Dec	LPBoost-Dec
twonorm	$\mathbf{2.87 \pm 0.04}$	3.74 ± 0.05	4.80 ± 0.06
twonorm-n	$\mathbf{3.10 \pm 0.05}$	6.46 ± 0.09	7.89 ± 0.10
threenorm	$\mathbf{15.0 \pm 0.11}$	16.8 ± 0.09	18.3 ± 0.10
threenorm-n	$\mathbf{16.8 \pm 0.15}$	20.2 ± 0.16	22.0 ± 0.12
ringnorm	$\mathbf{2.25 \pm 0.05}$	4.33 ± 0.06	6.00 ± 0.10
ringnorm-n	$\mathbf{2.67 \pm 0.06}$	7.32 ± 0.12	8.76 ± 0.12
australian	14.3 ± 0.18	$\mathbf{13.7 \pm 0.16}$	$\mathbf{13.8 \pm 0.17}$
breast	3.18 ± 0.08	$\mathbf{2.92 \pm 0.09}$	3.94 ± 0.16
german	24.9 ± 0.20	$\mathbf{24.5 \pm 0.17}$	24.9 ± 0.19
heart	$\mathbf{16.8 \pm 0.31}$	19.5 ± 0.33	20.4 ± 0.35
ionosphere	$\mathbf{6.48 \pm 0.19}$	$\mathbf{6.59 \pm 0.19}$	$\mathbf{6.81 \pm 0.22}$
pima	$\mathbf{24.0 \pm 0.24}$	26.1 ± 0.21	26.4 ± 0.20
sonar	$\mathbf{14.7 \pm 0.42}$	19.6 ± 0.45	21.3 ± 0.42
votes84	$\mathbf{4.59 \pm 0.15}$	5.04 ± 0.14	5.95 ± 0.17
a1a	**15.7**	18.8	20.3
splice	3.77	**2.94**	3.77
svmguide1	3.28	3.22	**3.17**
w1a	**2.37**	2.53	3.01

Table 4: Test error (%) of several ensemble learning algorithms using decision trees

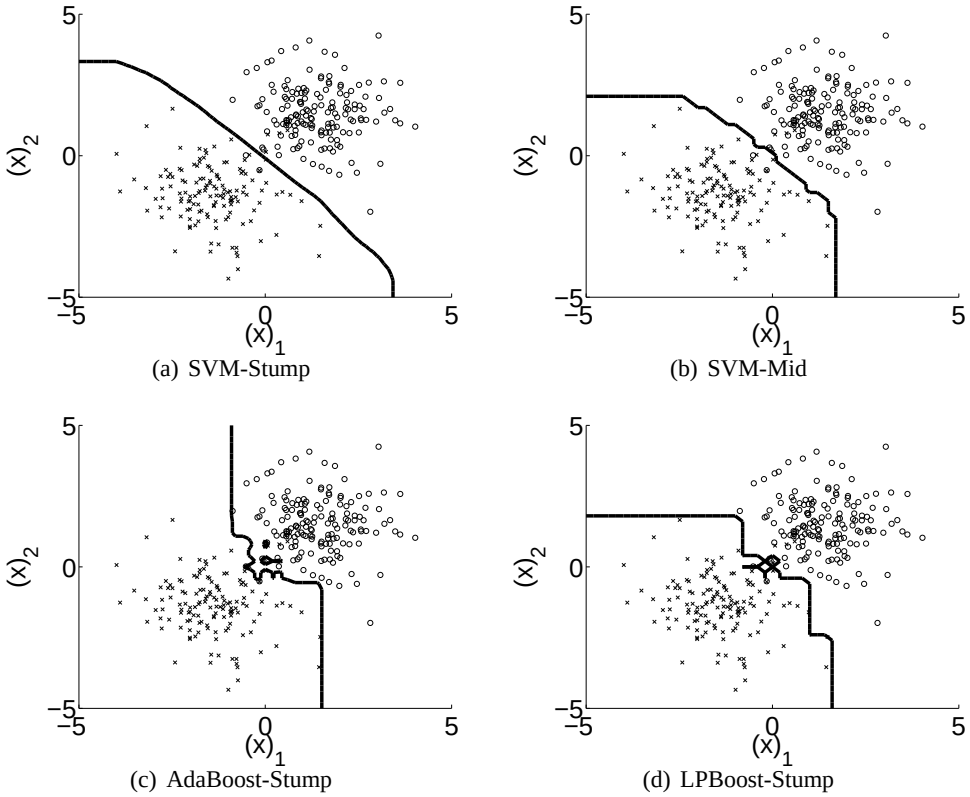
7.1 Comparison of Ensemble Learning Algorithms

Tables 2, 3, and 4 show the test performance of several ensemble learning algorithms on different base hypothesis sets.[7] We can see that SVM-Stump, SVM-Perc, and SVM-Dec are usually better than AdaBoost and LPBoost with the same base hypothesis set, especially for the cases of decision stumps and perceptrons. In noisy data sets, SVM-based infinite ensemble learning always significantly outperforms AdaBoost and LPBoost. These results demonstrate that it is beneficial to go from a finite ensemble to an infinite one with suitable regularization. When comparing the two boosting approaches, LPBoost is at best comparable to AdaBoost on a small number of the data sets, which suggests that the success of AdaBoost may not be fully attributed to its connection to (P_3) or (P_4).

Note that SVM-Stump, SVM-Mid, AdaBoost-Stump, and LPBoost-Stump usually generate different kinds of ensembles: SVM-Stump produces infinite and nonsparse ones; SVM-Mid produces finite and nonsparse ones; AdaBoost-Stump produces finite and sparse ones; LPBoost-Stump produces finite and even sparser ones (since some of the selected h_t may end up having $w_t = 0$). In Table 2, we see that SVM-Stump often outperforms SVM-Mid, which is another evidence that an infinite ensemble could help. Interestingly, SVM-Mid often performs better than AdaBoost-Stump, which means that a nonsparse ensemble introduced by minimizing the ℓ_2-norm of w is better than a sparse one.

In Figure 3, we further illustrate the difference between the finite and infinite ensemble learning algorithms by a simplified experiment. We show the decision boundaries generated by the four algorithms on 300 training examples from the 2-D version of the twonorm data set. The Bayes-

7. For the first 14 rows of Tables 2, 3, 4, and 5, results that are as significant as the best ones are marked in bold; for the last 4 rows, the best results are marked in bold.



(a) SVM-Stump (b) SVM-Mid

(c) AdaBoost-Stump (d) LPBoost-Stump

Figure 3: Decision boundaries of ensemble learning algorithms on a 2-D twonorm data set

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, while AdaBoost-Stump and LPBoost-Stump fail to generate a decent boundary. SVM-Stump obtains the smooth boundary by averaging over infinitely many decision stumps; SVM-Mid can also generate a smooth boundary by constructing a nonsparse ensemble over a finite number of decision stumps. Nevertheless, both LPBoost-Stump and AdaBoost-Stump, for which sparsity could be observed from the axis-parallel decision boundaries, do not have the ability to approximate the Bayes optimal boundary well. In addition, as can be seen near the origin point of Figure 3(c), AdaBoost-Stump could suffer from overfitting the noise.

Note that traditional ensemble learning and our SVM-based framework differ in the concept of sparsity. As illustrated in Subsection 2.2, traditional ensemble learning prefers sparse ensemble classifiers, that is, ensembles that include a small number of hypotheses. Our framework works with an infinite number of hypotheses, but results in a sparse classifier in the support vector domain. Both concepts can be justified with various generalization bounds (Freund and Schapire, 1997; Graepel et al., 2005). Nevertheless, our experimental results indicate that sparse ensemble classifiers are sometimes not sophisticated enough in practice, especially when the base hypothesis set is simple. For example, when using the decision stumps, a general data set may require many of them to describe a suitable decision boundary. Thus, AdaBoost-Stump and LPBoost-Stump could be limited by the finiteness and sparsity restrictions. The comparison between AdaBoost-Stump and SVM-Mid

data set	SVM-Stump	SVM-Perc	SVM-Dec	SVM-Gauss
twonorm	2.86 ± 0.04	$\mathbf{2.55 \pm 0.03}$	2.87 ± 0.04	2.64 ± 0.05
twonorm-n	3.08 ± 0.06	$\mathbf{2.75 \pm 0.05}$	3.10 ± 0.05	$\mathbf{2.86 \pm 0.07}$
threenorm	17.7 ± 0.10	$\mathbf{14.6 \pm 0.08}$	15.0 ± 0.11	$\mathbf{14.6 \pm 0.11}$
threenorm-n	19.0 ± 0.14	16.3 ± 0.10	16.8 ± 0.15	$\mathbf{15.6 \pm 0.15}$
ringnorm	3.97 ± 0.07	2.46 ± 0.04	2.25 ± 0.05	$\mathbf{1.77 \pm 0.04}$
ringnorm-n	5.56 ± 0.11	3.50 ± 0.09	2.67 ± 0.06	$\mathbf{2.05 \pm 0.07}$
australian	$\mathbf{14.4 \pm 0.21}$	$\mathbf{14.5 \pm 0.17}$	$\mathbf{14.3 \pm 0.18}$	14.7 ± 0.18
breast	$\mathbf{3.11 \pm 0.08}$	$\mathbf{3.23 \pm 0.08}$	$\mathbf{3.18 \pm 0.08}$	3.53 ± 0.10
german	$\mathbf{24.7 \pm 0.18}$	$\mathbf{24.6 \pm 0.20}$	$\mathbf{24.9 \pm 0.20}$	$\mathbf{24.5 \pm 0.21}$
heart	$\mathbf{16.4 \pm 0.27}$	17.6 ± 0.31	$\mathbf{16.8 \pm 0.31}$	17.5 ± 0.31
ionosphere	8.13 ± 0.17	$\mathbf{6.40 \pm 0.20}$	$\mathbf{6.48 \pm 0.19}$	$\mathbf{6.54 \pm 0.19}$
pima	24.1 ± 0.23	$\mathbf{23.5 \pm 0.21}$	24.0 ± 0.24	$\mathbf{23.5 \pm 0.20}$
sonar	16.6 ± 0.42	15.6 ± 0.40	$\mathbf{14.7 \pm 0.42}$	$\mathbf{15.5 \pm 0.50}$
votes84	4.76 ± 0.14	$\mathbf{4.43 \pm 0.14}$	$\mathbf{4.59 \pm 0.15}$	$\mathbf{4.62 \pm 0.14}$
a1a	16.2	**15.7**	**15.7**	16.2
splice	6.21	10.4	**3.77**	9.56
svmguide1	**2.92**	3.10	3.28	3.12
w1a	2.09	**1.92**	2.37	2.03

Table 5: Test error (%) of SVM with radial basis function kernels

indicates that the second restriction could be crucial. On the other hand, our framework (SVM-Stump), which suffers from neither restrictions, can perform better by averaging over an infinite number of hypotheses.

7.2 Comparison to Gaussian Kernel

In Table 5, we compare all the radial basis function kernels. We can see that SVM-Gauss, being the state-of-the-art setting (Hsu et al., 2003), usually performs well. Its superior performance on the artificial data sets is because they are generated from certain Gaussian distributions. Nevertheless, all SVM-Stump, SVM-Perc, and SVM-Dec outperform SVM-Gauss on some data sets, which demonstrates that they could achieve decent test performances as well.

Furthermore, SVM-Perc and SVM-Gauss share almost indistinguishable performance on the real-world data sets, which is possibly because they both use the ℓ_2-norm distance for measuring similarity. In addition, as discussed in Subsection 6.4, SVM-Perc enjoys the benefit of faster parameter selection. For example, in our experiments, SVM-Gauss involves solving 550 optimization problems, but SVM-Perc deals with only 55 problems. Table 6 shows the speed comparison.[8] We can qualitatively see that SVM-Stump and SVM-Perc are much faster than SVM-Dec and SVM-Gauss.

8. Solving each SVM optimization problem heavily depends on the condition number of the Gram matrix (which depends on the kernel) and the soft-margin parameter C. Thus, it is not easy to compare the training time between different kernels and the numbers here are meant to be interpreted qualitatively. Similar results are observed when using real-world data sets as well (Lin, 2005).

data set	SVM-Stump	SVM-Perc	SVM-Dec	SVM-Gauss
twonorm	1.34	1.44	19.5	23.1
threenorm	1.69	1.69	23.1	31.1
ringnorm	1.50	1.60	23.7	27.9

Table 6: Parameter selection time (sec.) on a dual 1.7 GHz Intel Xeon CPU machine

Since SVM-Perc and SVM-Gauss perform similarly on real-world data sets, the benefit of faster parameter selection makes SVM-Perc a favorable choice in practice. Furthermore, SVM-Stump, albeit slightly worse than SVM-Perc or SVM-Gauss, could still be a useful alternative in some applications where decision stump ensemble models are preferred, such as those described in Subsection 6.4.

8. Conclusion

We derived two novel kernels based on the infinite ensemble learning framework. The stump kernel embodies infinitely many decision stumps, and the perceptron kernel embodies infinitely many perceptrons. These kernels can be simply evaluated by the ℓ_1- or ℓ_2-norm distance between examples. We also explained that the Laplacian-RBF kernel embodies infinitely many decision trees. SVM equipped with the kernels can generate infinite and nonsparse ensembles, which are usually more robust than finite and sparse ones.

Experimental comparisons with AdaBoost and LPBoost showed that SVM with the kernels usually performs much better than boosting approaches with the same base hypothesis set, especially in the cases of decision stumps or perceptrons. Therefore, existing applications that use boosting with decision stumps, perceptrons, or decision trees may be improved by switching to SVM with the corresponding kernel.

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

Acknowledgments

We thank Yaser Abu-Mostafa, Amrit Pratap, Kai-Min Chung, and the anonymous reviewers for valuable suggestions. Most of the work was done in 2005, in which Hsuan-Tien Lin was supported by the Caltech Center for Neuromorphic Systems Engineering under the US NSF Cooperative Agreement EEC-9402726, and Ling Li was sponsored by the Caltech SISL Graduate Fellowship.

Appendix A. Derivation of Kernels

$$\begin{aligned}
\mathcal{K}_{\mathcal{S}}(x,x') &= \sum_{q\in\{-1,+1\}}\sum_{d=1}^{D}\int_{L_d}^{R_d} r_{\mathcal{S}}^2 s_{q,d,\alpha}(x)s_{q,d,\alpha}(x')\,d\alpha \\
&= 2r_{\mathcal{S}}^2\sum_{d=1}^{D}\int_{L_d}^{R_d}\operatorname{sign}\big((x)_d-\alpha\big)\operatorname{sign}\big((x')_d-\alpha\big)\,d\alpha \\
&= 2r_{\mathcal{S}}^2\sum_{d=1}^{D}\left((R_d-L_d)-2\int_{\min((x)_d,(x')_d)}^{\max((x)_d,(x')_d)} d\alpha\right) \\
&= 2r_{\mathcal{S}}^2\sum_{d=1}^{D}(R_d-L_d)-4r_{\mathcal{S}}^2\sum_{d=1}^{D}\left|(x)_d-(x')_d\right| \\
&= 2r_{\mathcal{S}}^2\sum_{d=1}^{D}(R_d-L_d)-4r_{\mathcal{S}}^2\left\|x-x'\right\|_1.
\end{aligned}$$

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

The last equality comes from the symmetry when integrating over every possible θ.

References

Annalisa Barla, Francesca Odone, and Alessandro Verri. Histogram intersection kernel for image classification. In *Proceedings of the 2003 Conference on Image Processing*, volume 3, pages 513–516, 2003.

Brad J. C. Baxter. Conditionally positive functions and *p*-norm distance matrices. *Constructive Approximation*, 7(1):427–440, 1991.

Christian Berg, Jens P. R. Christensen, and Paul Ressel. *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*. Springer-Verlag, 1984.

Sabri Boughorbel, Jean-Philippe Tarel, and Nozha Boujemaa. Generalized histogram intersection kernel for image recognition. In *Proceedings of the 2005 Conference on Image Processing*, volume 3, pages 161–164, 2005.

Leo Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1517, 1999.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A library for support vector machines*, 2001a. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chih-Chung Chang and Chih-Jen Lin. Training ν-support vector classifiers: Theory and algorithms. *Neural Computation*, 13(9):2119–2147, 2001b.

Olivier Chapelle, Patrick Haffner, and Vladimir N. Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, 1999.

Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1–3):225–254, 2002.

Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

François Fleuret and Hichem Sahbi. Scale-invariance of support vector machines based on the triangular kernel. In *Proceedings of the Third International Workshop on Statistical and Computational Theories of Vision*, 2003.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Yoav Freund and Robert E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

Thore Graepel, Ralf Herbrich, and John Shawe-Taylor. PAC-Bayesian compression bounds on the prediction error of learning algorithms for classification. *Machine Learning*, 59(1–2):55–76, 2005.

Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1458–1465, 2005.

Trevor Hastie and Robert Tibshirani. *Generalized Additive Models*. Chapman and Hall, 1990.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, second edition, 1999.

Seth Hettich, Catherine L. Blake, and Christopher J. Merz. UCI repository of machine learning databases, 1998. Downloadable at http://www.ics.uci.edu/~mlearn/MLRepository.html.

Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–91, 1993.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, National Taiwan University, 2003.

S. Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.

Ling Li and Hsuan-Tien Lin. Optimizing 0/1 loss for perceptrons by random coordinate descent. In *Proceedings of the 2007 International Joint Conference on Neural Networks*, pages 749–754, 2007.

Hsuan-Tien Lin. Infinite ensemble learning with support vector machines. Master's thesis, California Institute of Technology, 2005.

Hsuan-Tien Lin and Ling Li. Novel distance-based SVM kernels for infinite ensemble learning. In *Proceedings of the 12th International Conference on Neural Information Processing*, pages 761–766, 2005a.

Hsuan-Tien Lin and Ling Li. Analysis of SAGE results with combined learning techniques. In P. Berka and B. Crémilleux, editors, *Proceedings of the ECML/PKDD 2005 Discovery Challenge*, pages 102–113, 2005b.

Ron Meir and Gunnar Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. J. Smola, editors, *Advanced Lectures on Machine Learning*, pages 118–183. Springer-Verlag, 2003.

Charles A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2(1):11–22, 1986.

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

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Gunnar Rätsch, Takashi Onoda, and Klaus-Robert Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

Gunnar Rätsch, Ayhan Demiriz, and Kristin P. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning*, 48(1–3):189–218, 2002.

Gunnar Rätsch, Sebastian Mika, Bernhard Schölkopf, and Klaus-Robert Müller. Constructing boosting algorithms from SVMs: an application to one-class classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1184–1199, 2002.

Michael Reed and Barry Simon. *Functional Analysis*. Academic Press, revised and enlarged edition, 1980.

Saharon Rosset, Ji Zhu, and Trevor Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, 2004.

Saharon Rosset, Grzegorz Swirszcz, Nathan Srebro, and Ji Zhu. ℓ_1 regularization in infinite dimensional feature spaces. In N. H. Bshouty and C. Gentile, editors, *Learning Theory: 20th Annual Conference on Learning Theory*, volume 4539 of *Lecture Notes in Computer Science*, pages 544–558. Springer-Verlag, 2007.

Bernhard Schölkopf and Alex J. Smola. *Learning with Kernels*. MIT Press, 2002.

Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

Christopher K. I. Williams. Computation with infinite neural networks. *Neural Computation*, 10 (5):1203–1216, 1998.