

Infinite Ensemble Learning with Support Vector Machinery

Hsuan-Tien Lin and Ling Li

Learning Systems Group, California Institute of Technology

ECML/PKDD, October 4, 2005



Outline

- 1 Motivation of Infinite Ensemble Learning
- 2 Connecting SVM and Ensemble Learning
- 3 SVM-Based Framework of Infinite Ensemble Learning
- 4 Concrete Instance of the Framework: Stump Kernel
- 5 Experimental Comparison
- 6 Conclusion



Learning Problem

- notation: example $x \in \mathcal{X} \subseteq \mathbb{R}^D$ and label $y \in \{+1, -1\}$
- hypotheses (classifiers): functions from $\mathcal{X} \rightarrow \{+1, -1\}$
- binary classification problem: given training examples and labels $\{(x_i, y_i)\}_{i=1}^N$, find a classifier $g(x) : \mathcal{X} \rightarrow \{+1, -1\}$ that predicts the label of unseen x well



Ensemble Learning

$$g(x) : \mathcal{X} \rightarrow \{+1, -1\}$$

- ensemble learning: popular paradigm (bagging, boosting, etc.)
- ensemble: weighted vote of a committee of hypotheses
 $g(x) = \text{sign}(\sum w_t h_t(x))$
- h_t : base hypotheses, usually chosen from a set \mathcal{H}
- w_t : nonnegative weight for h_t
- ensemble usually better than individual $h_t(x)$ in stability/performance

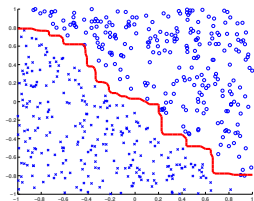


Infinite Ensemble Learning

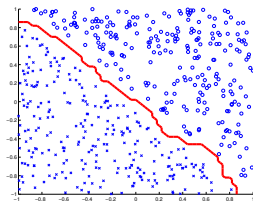
$$g(x) = \text{sign}\left(\sum w_t h_t(x)\right), h_t \in \mathcal{H}, w_t \geq 0$$

- set \mathcal{H} can be of infinite size
- traditional algorithms: assign **finite** number of nonzero w_t

- 1 is finiteness **regularization** and/or **restriction**?
- 2 how to handle **infinite** number of nonzero weights?



finite ensemble



infinite ensemble



SVM for Infinite Ensemble Learning

- Support Vector Machine (SVM): large-margin hyperplane in some feature space
- SVM: possibly **infinite** dimensional hyperplane
$$g(\mathbf{x}) = \text{sign}(\sum w_d \phi_d(\mathbf{x}) + b)$$
- an important machinery to conquer infinity: kernel trick.

how can we use Support Vector Machinery for **infinite ensemble learning**?



Properties of SVM

$$g(\mathbf{x}) = \text{sign}\left(\sum_{d=1}^{\infty} w_d \phi_d(\mathbf{x}) + b\right) = \text{sign}\left(\sum_{i=1}^N \lambda_i y_i \mathcal{K}(\mathbf{x}_i, \mathbf{x}) + b\right)$$

- a successful large-margin learning algorithm.
- goal: (infinite dimensional) large-margin hyperplane

$$\min_{w,b} \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^N \xi_i, \text{ s.t. } y_i \left(\sum_{d=1}^{\infty} w_d \phi_d(\mathbf{x}_i) + b \right) \geq 1 - \xi_i, \xi_i \geq 0$$

- optimal hyperplane: represented through duality
- key for handling infinity: computation with kernel tricks
 $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{\infty} \phi_d(\mathbf{x}) \phi_d(\mathbf{x}')$
- regularization: controlled with the trade-off parameter C



Properties of AdaBoost

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

- a successful ensemble learning algorithm
- goal: asymptotically, large-margin ensemble

$$\min_{w, h} \|w\|_1, \text{ s.t. } y_i \left(\sum_{t=1}^{\infty} w_t h_t(x_i)\right) \geq 1, w_t \geq 0$$

- optimal ensemble: approximated by finite one
- key for good approximation:
 - finiteness: some $h_{t_1}(x_i) = h_{t_2}(x_i)$ for all i
 - sparsity: optimal ensemble usually has many zero weights
- regularization: finite approximation



Connection between SVM and AdaBoost

$$\phi_d(x) \Leftrightarrow h_t(x)$$

<p>SVM</p> $G(x) = \sum_k w_k \phi_k(x) + b$	<p>AdaBoost</p> $G(x) = \sum_k w_k h_k(x)$ $w_k \geq 0$
---	--

hard-goal

$\min \ w\ _p, \text{ s.t. } y_i G(x_i) \geq 1$ $p = 2$	$p = 1$
---	---------

key for infinity

kernel trick	finiteness and sparsity
--------------	-------------------------

regularization

soft-margin trade-off	finite approximation
-----------------------	----------------------



Challenge

- challenge: how to design a good infinite ensemble learning algorithm?
- traditional ensemble learning: iterative and cannot be directly generalized
- our main contribution: **novel and powerful infinite ensemble learning algorithm with Support Vector Machinery**
- our approach: embedding infinite number of hypotheses in SVM kernel, i.e., $\mathcal{K}(x, x') = \sum_{t=1}^{\infty} h_t(x)h_t(x')$
 – then, SVM classifier: $g(x) = \text{sign}(\sum_{t=1}^{\infty} w_t h_t(x) + b)$

- 1 does the kernel exist?
- 2 how to ensure $w_t \geq 0$?



Embedding Hypotheses into the Kernel

Definition

The kernel that embodies $\mathcal{H} = \{h_\alpha : \alpha \in \mathcal{C}\}$ is defined as

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

where \mathcal{C} is a measure space, $\phi_x(\alpha) = r(\alpha)h_\alpha(x)$, and $r: \mathcal{C} \rightarrow \mathbb{R}^+$ is chosen such that the integral always exists

- integral instead of sum: works even for uncountable \mathcal{H}
- existence problem handled with a suitable $r(\cdot)$
- $\mathcal{K}_{\mathcal{H},r}(x, x')$: an inner product for ϕ_x and $\phi_{x'}$ in $\mathcal{F} = \mathcal{L}_2(\mathcal{C})$
- the classifier: $g(x) = \text{sign}(\int_{\mathcal{C}} w(\alpha)r(\alpha)h_\alpha(x) d\alpha + b)$



Negation Completeness and Constant Hypotheses

$$g(x) = \text{sign} \left(\int_{\mathcal{C}} w(\alpha) r(\alpha) h_{\alpha}(x) d\alpha + b \right)$$

- not an ensemble classifier yet
- $w(\alpha) \geq 0$?
 - hard to handle: possibly uncountable constraints
 - simple with negation completeness assumption on \mathcal{H} ($h \in \mathcal{H}$ if and only if $(-h) \in \mathcal{H}$)
 - e.g. neural networks, perceptrons, decision trees, etc.
 - for any w , exists nonnegative \tilde{w} that produces same g
- What is b ?
 - equivalently, the weight on a constant hypothesis
 - another assumption: \mathcal{H} contains a constant hypothesis
- with mild assumptions, $g(x)$ is equivalent to an ensemble classifier



Framework of Infinite Ensemble Learning

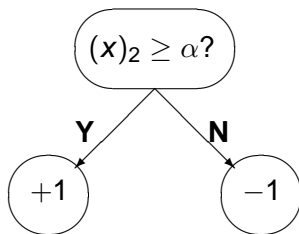
Algorithm

- 1 Consider a hypothesis set \mathcal{H} (negation complete and contains a constant hypothesis)
 - 2 Construct a kernel $\mathcal{K}_{\mathcal{H},r}$ with proper $r(\cdot)$
 - 3 Properly choose other SVM parameters
 - 4 Train SVM with $\mathcal{K}_{\mathcal{H},r}$ and $\{(x_i, y_i)\}_{i=1}^N$ to obtain λ_i and b
 - 5 Output $g(x) = \text{sign}\left(\sum_{i=1}^N y_i \lambda_i \mathcal{K}_{\mathcal{H}}(x_i, x) + b\right)$
- hard: kernel construction
 - SVM as an optimization machinery: training routines are widely available
 - SVM as a well-studied learning model: inherit the profound regularization properties

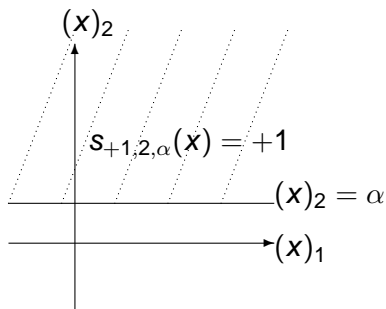


Decision Stump

- decision stump: $s_{q,d,\alpha}(x) = q \cdot \text{sign}((x)_d - \alpha)$
- simplicity: popular for ensemble learning



(a) Decision Process



(b) Decision Boundary

Figure: Illustration of the decision stump $s_{+1,2,\alpha}(x)$



Stump Kernel

- consider the set of decision stumps
 $\mathcal{S} = \{s_{q,d,\alpha_d} : q \in \{+1, -1\}, d \in \{1, \dots, D\}, \alpha_d \in [L_d, R_d]\}$
- when $\mathcal{X} \subseteq [L_1, R_1] \times [L_2, R_2] \times \dots \times [L_D, R_D]$, \mathcal{S} is negation complete, and contains a constant hypothesis

Definition

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

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

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



Properties of Stump Kernel

- simple to compute: can even use a simpler one
 $\tilde{\mathcal{K}}_{\mathcal{S}}(\mathbf{x}, \mathbf{x}') = -\|\mathbf{x} - \mathbf{x}'\|_1$ while getting the same solution
 - under the dual constraint $\sum_i y_i \lambda_i = 0$, using $\mathcal{K}_{\mathcal{S}}$ or $\tilde{\mathcal{K}}_{\mathcal{S}}$ is the same
 - feature space explanation for ℓ_1 -norm distance
- infinite power: under mild assumptions, SVM-Stump with $C = \infty$ can perfectly classify all training examples
 - if there is a dimension for which all feature values are different, the kernel matrix K with $K_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$ is strictly positive definite
 - similar power to the popular Gaussian kernel
 $\exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|_2^2)$
 - suitable control on the power leads to good performance



Properties of Stump Kernel (Cont'd)

- fast automatic parameter selection: only needs to search for a good soft-margin parameter C
 - scaling the stump kernel is equivalent to scaling soft-margin parameter C
 - Gaussian kernel depends on a good (γ, C) pair (10 times slower)
- well suited in some specific applications:
cancer prediction with gene expressions
(Lin and Li, ECML/PKDD Discovery Challenge, 2005)



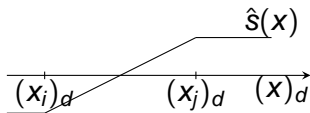
Infinite Decision Stump Ensemble

$$g(\mathbf{x}) = \text{sign} \left(\sum_{q \in \{+1, -1\}} \sum_{d=1}^D \int_{L_d}^{R_d} w_{q,d}(\alpha) s_{q,d,\alpha}(\mathbf{x}) d\alpha + b \right)$$

- each $s_{q,d,\alpha}$: infinitesimal influence $w_{q,d}(\alpha) d\alpha$
- equivalently,

$$g(\mathbf{x}) = \text{sign} \left(\sum_{q \in \{+1, -1\}} \sum_{d=1}^D \sum_{a=0}^{A_d} \hat{w}_{q,d,a} \hat{s}_{q,d,a}(\mathbf{x}) + b \right)$$

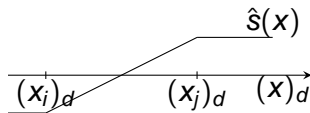
- \hat{s} : a smoother variant of decision stump



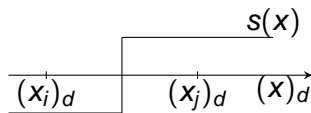
Infinitesimal Influence

$$g(x) = \text{sign} \left(\sum_{q \in \{+1, -1\}} \sum_{d=1}^D \sum_{a=0}^{A_d} \hat{w}_{q,d,a} \hat{s}_{q,d,a}(x) + b \right)$$

- infinity \rightarrow dense combination of finite number of smooth stumps
- infinitesimal influence \rightarrow concrete weight of the smooth stumps



SVM: dense
combination of
smoothed stumps



AdaBoost: sparse
combination of middle
stumps

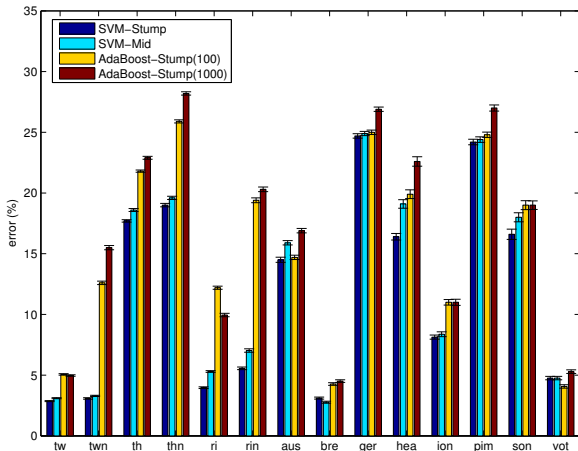


Experiment Setting

- ensemble learning algorithms:
 - SVM-Stump: infinite ensemble of decision stumps (dense ensemble of smooth stumps)
 - SVM-Mid: dense ensemble of middle stumps
 - AdaBoost-Stump: sparse ensemble of middle stumps
- SVM algorithms: SVM-Stump versus SVM-Gauss
- artificial, noisy, and realworld datasets
- cross-validation for automatic parameter selection of SVM
- evaluate on hold-out test set and averaged over 100 different splits



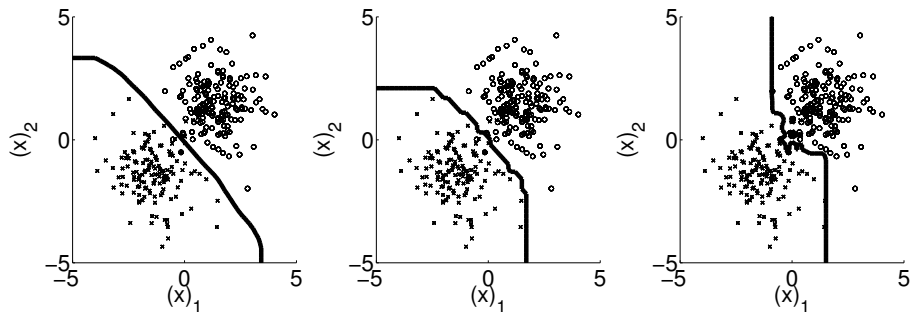
Comparison between SVM and AdaBoost



Results

- fair comparison between AdaBoost and SVM
- SVM-Stump is usually best – benefits to go to infinity
- SVM-Mid is also good – benefits to have dense ensemble
- sparsity and finiteness are **restrictions**

Comparison between SVM and AdaBoost (Cont'd)

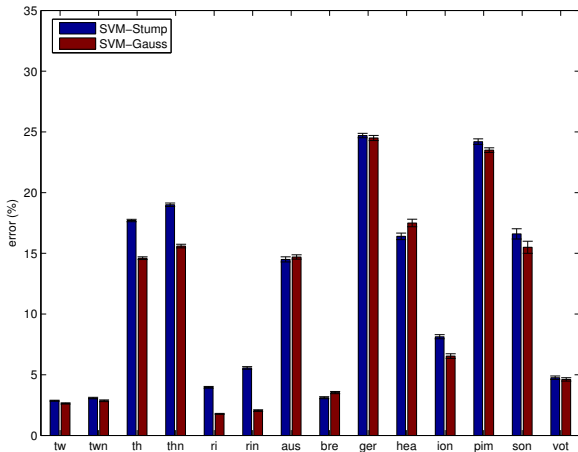


left to right: SVM-Stump, SVM-Mid, AdaBoost-Stump

- smoother boundary with infinite ensemble (SVM-Stump)
- still fits well with dense ensemble (SVM-Mid)
- cannot fit well when sparse and finite (AdaBoost-Stump)



Comparison of SVM Kernels



Results

- SVM-Stump is only a bit worse than SVM-Gauss
- still benefit with faster parameter selection in some applications



Conclusion

- novel and powerful framework for infinite ensemble learning
- derived a new and meaningful kernel
 - stump kernel: succeeded in specific applications
- infinite ensemble learning could be better – existing AdaBoost-Stump applications may switch
- not the only kernel:
 - perceptron kernel → infinite ensemble of perceptrons
 - Laplacian kernel → infinite ensemble of decision trees
- SVM: our machinery for conquering infinity
 - possible to apply similar machinery to areas that need infinite or lots of aggregation

