Infinite Ensemble Learning with Support Vector Machines

Hsuan-Tien Lin

in collaboration with Ling Li
Learning Systems Group, Caltech

Second Symposium on Vision and Learning, 2005/09/21

Outline

Setup of our Learning Problem

- binary classification problem:

  does this image represent an apple? 
- features of the image: a vector $x \in \mathcal{X} \subseteq \mathbb{R}^D$.
  - e.g.: $(x)_1$ can describe the shape, $(x)_2$ can describe the color, etc.
  - difference to the features in vision: a **vector of properties**, not a "set of interest points."
- label (whether the image is an apple): $y \in \{+1, -1\}$.
- learning problem: give many images and their labels (training examples) $\{(x_i, y_i)\}_{i=1}^N$, find a classifier $g(x) : \mathcal{X} \to \{+1, -1\}$ that predicts **unseen** images well.
- hypotheses (classifiers): functions from $\mathcal{X} \to \{+1, -1\}$.

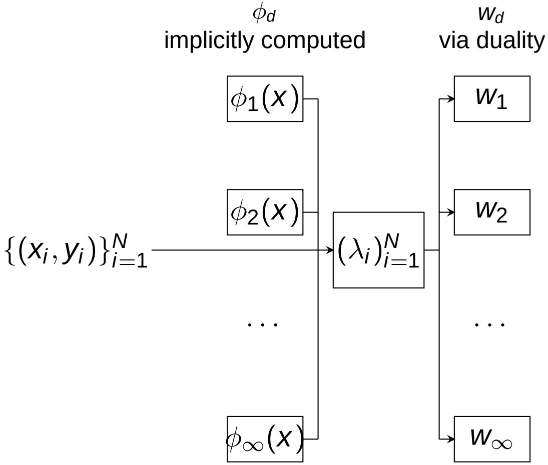
Motivation of Infinite Ensemble Learning

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

- ensemble learning: popular paradigm.
  - ensemble: weighted vote of a committee of hypotheses. $g(x) = \text{sign}(\sum w_t h_t(x)), w_t \geq 0$.
  - traditional ensemble learning: **infinite** size committee with **finite** number of nonzero weights.
  - is finiteness **restriction** and/or **regularization**?
  - how to handle **infinite** number of nonzero weights?
- SVM (large-margin hyperplane): also popular.
  - hyperplane: a weighted combination of features.
  - SVM: **infinite** dimensional hyperplane through kernels. $g(x) = \text{sign}(\sum w_d \phi_d(x) + b)$.
  - can we use SVM for **infinite ensemble learning**?

Illustration of SVM

$$g(x) = \text{sign}\left(\sum_{d=1}^{\infty} w_d \phi_d(x) + b\right)$$



SVM

- implicit computation with $\mathcal{K}(x, x') = \sum_{d=1}^{\infty} \phi_d(x)\phi_d(x')$.
- optimal solution (w, b) represented by the dual variables λ_i.

Property of SVM

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

- optimal hyperplane: represented through duality.
- key for handling infinity: kernel tricks $\mathcal{K}(x, x') = \sum_{d=1}^{\infty} \phi_d(x)\phi_d(x')$.
- quadratic programming of a margin-related criteria.
- goal: (infinite dimensional) large-margin hyperplane.

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

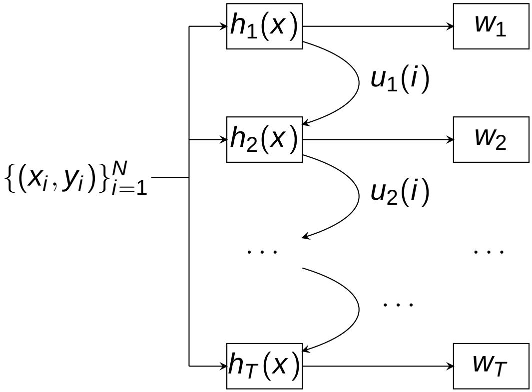
- regularization: controlled with the trade-off parameter C.

Illustration of AdaBoost

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

$h_t \in \mathcal{H}$
iteratively selected

$w_t \geq 0$
iteratively assigned



AdaBoost

- most successful ensemble learning algorithm.
- boosts up the performance of each individual h_t.
- emphasizes difficult examples by u_t and finds (h_t, w_t) iteratively.

Property of AdaBoost

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

- iterative coordinate descent of a margin-related criteria.

$$\min \sum_{i=1}^{N} \exp\left(-\rho_i\right), \text{ s.t. } \rho_i = y_i \left(\sum_{t=1}^{\infty} w_t h_t(x_i)\right), w_t \geq 0.$$

- 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: sparsity
  – some optimal ensemble has many zero weights.
- regularization: finite approximation.

Connection between SVM and AdaBoost

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

SVM	AdaBoost
$G(x) = \sum_k w_k \phi_k(x) + b$	$G(x) = \sum_k w_k h_k(x)$ $w_k \geq 0$
hard-goal	
$\min \|w\|_p, \text{ s.t. } y_i G(x_i) \geq 1$	
$p = 2$	$p = 1$
optimization	
quadratic programming	iterative coordinate descent
key for infinity	
kernel trick	sparsity
regularization	
soft-margin trade-off	finite approximation

Challenge

designing an infinite ensemble learning algorithm

- traditional ensemble learning: iterative and cannot directly be generalized.
- another 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: $g(x) = \text{sign}(\sum_{t=1}^{\infty} w_t h_t(x) + b)$.
- **does the kernel exist?**
- **how to ensure** $w_t \geq 0$**?**
- our main contribution: **a framework that conquers the challenge**.

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 $r\colon \mathcal{C} \to \mathbb{R}^+$ is chosen such that the integral always exists.

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

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 $\mathcal{H}$.
  - negation completeness: $h \in \mathcal{H}$ if and only if $(-h) \in \mathcal{H}$.
  - 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.
- both assumptions: mild in practice.
- $g(x)$ is equivalent to an ensemble classifier.

Framework of Infinite Ensemble Learning

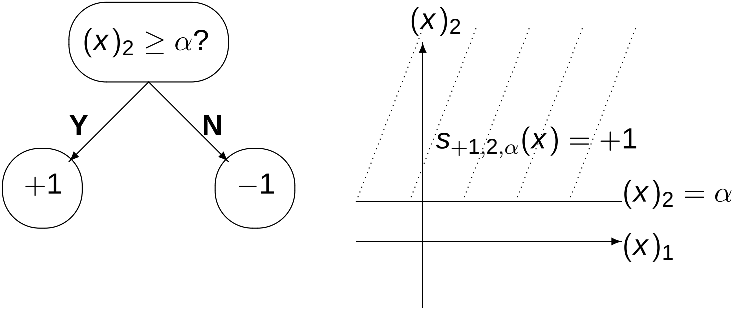
Algorithm

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

- easy: SVM routines.
- hard: kernel construction.
- shall inherit the profound properties of SVM.

Decision Stump

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



(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 $\mathcal{S} = \left\{ s_{q,d,\alpha_d} : q \in \{+1, -1\}, d \in \{1, \ldots, D\}, \alpha_d \in [L_d, R_d] \right\}$.
- when $\mathcal{X} \subseteq [L_1, R_1] \times [L_2, R_2] \times \cdots \times [L_D, R_D]$, $\mathcal{S}$ is negation complete, and contains a constant hypothesis.

Definition

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

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

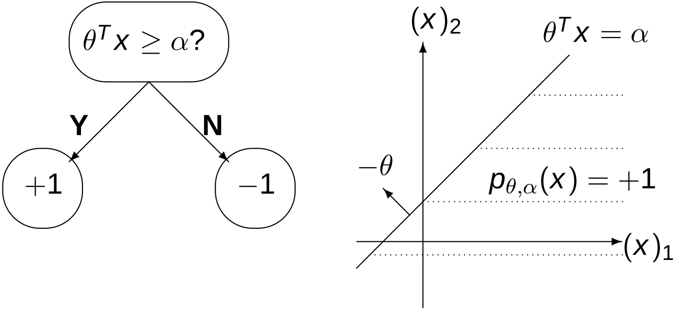
where $\Delta_{\mathcal{S}} = \frac{1}{2} \sum_{d=1}^{D} (R_d - L_d)$ is a constant.

Property of Stump Kernel

- simple to compute: the constant $\Delta_{\mathcal{S}}$ can even be dropped $\tilde{K}(x, x') = -\|x - x'\|_1$.
- infinite power: under mild assumptions, SVM with $C = \infty$ can perfectly classify training examples with stump kernel.
  - the popular Gaussian kernel $\exp(-\gamma\|x - x'\|_2^2)$ also.
- fast parameter selection: scaling the stump kernel is equivalent to scaling soft-margin parameter C.
  - Gaussian kernel depends on a good (γ, C) pair.
  - stump kernel only needs good C: roughly ten times faster.
- feature space explanation for ℓ_1-norm similarity.
- well suited in some specific applications: cancer prediction with gene expressions.

Perceptron

- perceptron: $p_{\theta,\alpha}(x) = \text{sign}(\theta^T x - \alpha)$.
- not easy for ensemble learning: hard to design good algorithm.



(a) Decision Process

(b) Decision Boundary

Figure: Illustration of the perceptron $p_{\theta,\alpha}(x)$

Perceptron Kernel

- consider the set of perceptrons
  $\mathcal{P} = \left\{ p_{\theta,\alpha} \colon \theta \in \mathbb{R}^D, \|\theta\|_2 = 1, \alpha \in [-R, R] \right\}$.
- when $\mathcal{X}$ is within a ball of radius R centered at the origin, $\mathcal{P}$ is negation complete, and contains a constant hypothesis.

Definition

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

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

where $r_{\mathcal{P}}$ and $\Delta_{\mathcal{P}}$ are constants.

Property of Perceptron Kernel

- similar properties to the stump kernel.
- also simple to compute.
- infinite power: equivalent to a D-∞-1 neural network.
- fast parameter selection: also shown in (Fleuret and Sahbi, ICCV 2003 workshop, called triangular kernel) without feature space explanation.

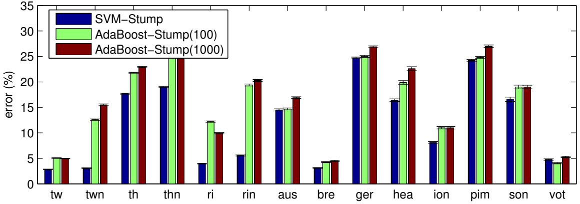
Histogram Intersection Kernel

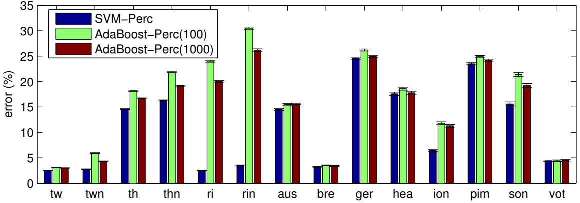
- introduced for scene recognition (Odone et al., IEEE TIP, 2005).
- assume $(x)_d$: counts in the histogram (how many pixels are red?) – an integer between $[0, \text{size of image}]$.
- histogram intersection kernel: $\mathcal{K}(x, x') = \sum_{d=1}^{D} \min((x)_d, (x')_d)$.
- generalized with difficult math when $(x)_d$ is not an integer (Boughorbel et al., ICIP, 2005), similar tasks.
- let $\hat{s}(x) = (s(x) + 1)/2$: HIK can be constructed easily from the framework.
- furthermore, HIK equivalent to stump kernel.
- insights on why HI (stump) kernel works well for the task?

Other Kernels

- Laplacian kernel: $\mathcal{K}(x, x') = \exp\left(-\gamma \|x - x'\|_1\right)$.
  - provably embodies infinite number of decision trees.
- generalized Laplacian: $\mathcal{K}(x, x') = \exp\left(-\gamma \sum |(x)_d^a - (x')_d^a|\right)$.
  - can be similarly constructed with a slightly different r function.
  - standard kernel for histogram-based image classification with SVM (Chappelle et al., IEEE TNN, 1999).
  - insights on why it should work well?
- exponential kernel: $\mathcal{K}(x, x') = \exp\left(-\gamma \|x - x'\|_2\right)$.
  - provably embodies infinite number of decision trees of perceptrons.

Comparison between SVM and AdaBoost

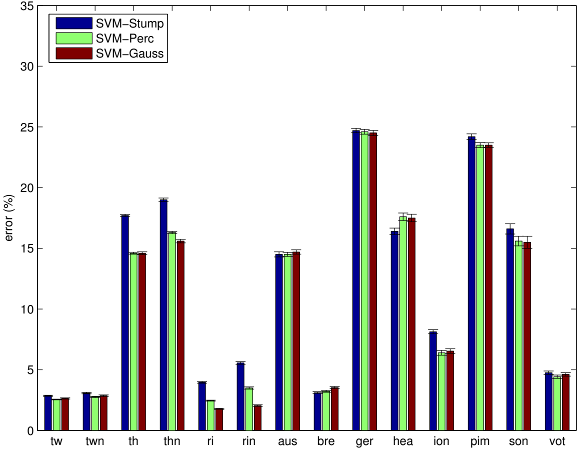




Results

- fair comparison between AdaBoost and SVM.
- SVM is usually best – benefits to go to infinity.
- sparsity (finiteness) is a **restriction**.

Comparison of SVM Kernels



Results

- SVM-Perc very similar to SVM-Gauss.
- SVM-Stump comparable to, but sometimes a bit worse than others.

Conclusion and Discussion

- constructed: general framework for infinite ensemble learning.
- infinite ensemble learning could be better – existing AdaBoost-Stump applications may switch.
- derived new and meaningful kernels.
  - stump kernel: succeeded in specific applications.
  - perceptron kernel: similar to Gaussian, faster in parameter selection.
- gave novel interpretation to existing kernels.
  - histogram intersection kernel: equivalent to stump kernel.
  - Laplacian kernel: ensemble of decision trees.
- possible thoughts for vision
  - would fast parameter selection be important for some problems?
  - any vision applications in which those kernel models are reasonable?
  - do the novel interpretations give any insights?
  - any domain knowledge that can be brought into kernel construction?