Topic 4: BOOSTING

CMSC 35400/STAT 37710 Machine Learning
Risi Kondor, The University of Chicago

Ensemble methods

In supervised learning, given a collection (ensemble) of hypotheses

$$h_t \colon \mathcal{X} \to \mathcal{Y} \qquad t = 1, 2, \ldots, T$$

possibly coming from different algorithms, how do we combine them to get a "meta-hypothesis"

$$h(x) = \phi\big(\textstyle\sum_{t=1}^{T} \alpha_t \, h_t(x)\big)$$

that is better than any of them individually?

Examples:

- Bayesian model averaging
- Mixture of experts
- Bagging: different classifiers trained on different subsets of data.
- Boosting: after selecting each h_t data is reweighted to focus on hard cases.

Boosting (for classification)

- If we have a collection of really weak classifiers (e.g. decision stumps $h(x) = \mathbb{I}([x]_j \geq \theta)$) can we combine them to get a decent classifier?
- If all that we know is that at least one of the classifiers in H is better than random for any distribution generating the data, can we boost them up to a good classifier? [Kearns & Valiant, 1998]

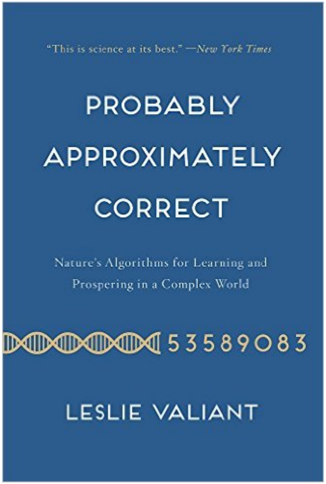
PAC learning

Definition

A deterministic concept class $\mathcal{C}$ is **strongly PAC-learnable** if for any target concept $f_{\text{true}} \in \mathcal{C}$, any distribution μ on $\mathcal{X}$, and any $\epsilon, \delta > 0$ there is a polynomial time algorithm that, given a sufficiently large training set drawn from μ, returns a hypothesis $\widehat{f}$ such that

$$\mathbb{P}[\mathcal{E}_{\text{true}}(f) > \epsilon] < \delta\,.$$

Leslie Valiant



Weak vs. strong PAC learning

Definition

A deterministic concept class $\mathcal{C}$ is **strongly PAC-learnable** if for any target concept $f_{\text{true}} \in \mathcal{C}$, any distribution μ on $\mathcal{X}$, and any $\epsilon, \delta > 0$ there is a polynomial time algorithm that, given a sufficiently large training set drawn from μ, returns a hypothesis $\widehat{f}$ such that

$$\mathbb{P}[\mathcal{E}_{\text{true}}(f) > \epsilon] < \delta\,.$$

Definition

A deterministic concept class $\mathcal{C}$ is **weakly PAC-learnable** if there is an **edge** $\tau > 0$, such that for any target concept $f_{\text{true}} \in \mathcal{C}$, any distribution μ on $\mathcal{X}$, and any $\delta > 0$ there is a poly-time algorithm that, given a sufficiently large training set drawn from μ, returns a hypothesis $\widehat{f}$ such that

$$\mathbb{P}[\mathcal{E}_{\text{true}}(f) > 1/2 - \tau] < \delta\,.$$

Does weak learnability imply strong learnability?

Yoav Freund (UCSD)

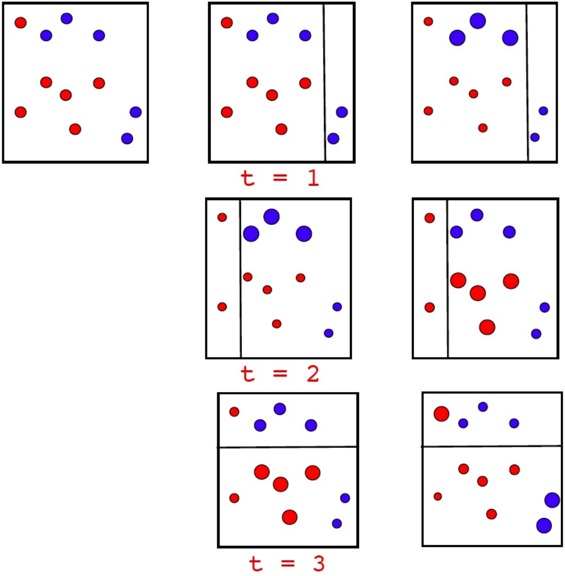
Robert Schapire (Princeton)

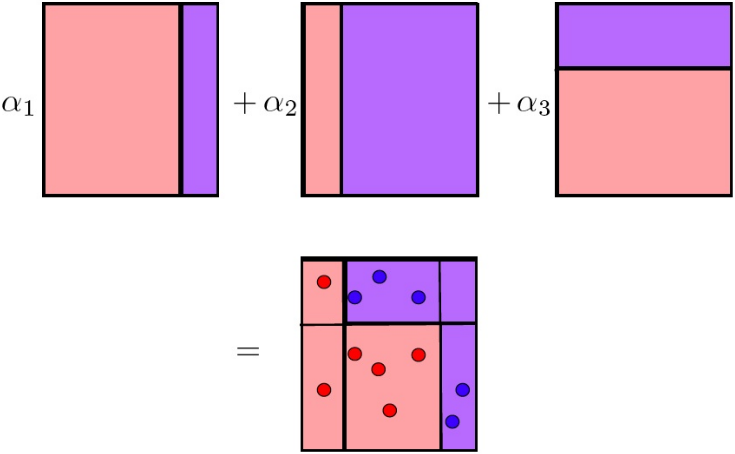
Boosting is a meta-classifier

- In classification tasks it is often relatively easy to come up with a set of N weak learners $H = \{h^1, h^2, \ldots, h^N\}$ where each h^i is a **base classifier** $h^i \colon \mathcal{X} \to \{-1, +1\}$.
- In the first round of boosting pick out the **base classifier** $h_1 \in H$ that does best on the training set.
- *Reweight* the training set so as to emphasize the misclassified examples and in the second round pick the base classifier h_2 that does the best on this reweighted training set (can simulate reweighting with filtering).
- Iterate T times.
- Return the final classifier

$$h(x) = \operatorname{sgn} \sum_{t=1}^{T} \alpha_t h_t(x).$$

- By VC-type arguments, usually sufficient to prove that this drives down the training error.
- As we will see, Boosting is similar to gradient descent to reduce $\mathcal{E}_{\text{train}}[h]$.


t = 1
t = 2
t = 3


α_1
$+\alpha_2$
$+\alpha_3$
$=$

Notation

- Set of base classifiers: $H = \{h^1, h^2, \ldots, h^N\}$.
- Base classifier (weak learner) selected at round t: h_t
- Distribution over examples at time t: D_t
- Weighted error of h_t (assumed $< 1/2$):

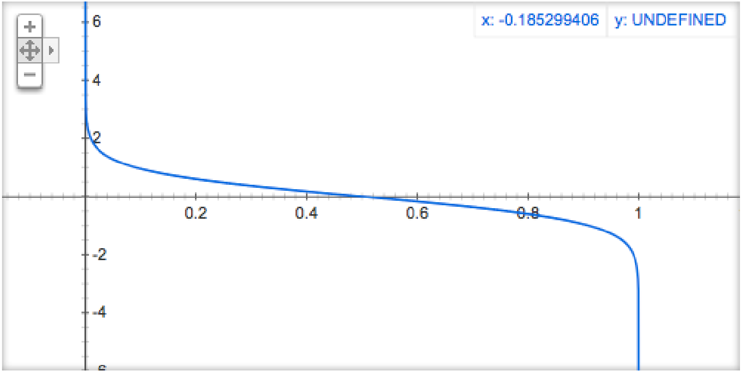
$$\epsilon_t := \mathsf{Pr}_{D_t}[h_t(x) \neq y] = \sum_{i=1}^{m} D_t(i)\, \ell_{0/1}(h_t(x_i), y_i).$$

- In the most famous boosting algorithm, **Adaboost**, the weights in the final hypothesis h are

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t} \in (0, \infty],$$

which is a measure of how good h_t is on the training set reweighted by D_t. Weight updates are also related: $D_{t+1}(i) \sim D_t(i)\, e^{-\alpha_t y_i h(x_i)}$.

Graph for log((1-x)/x)



AdaBoost (Freund & Schapire, 1997)

For binary classification $y \in \{-1, +1\}$:

$\textsc{AdaBoost}(S = ((x_1, y_1), \ldots, (x_m, y_m)))$

for $i \leftarrow 1$ **to** m **do**
 $D_1(i) \leftarrow \frac{1}{m}$
for $t \leftarrow 1$ **to** T **do**
 $h_t \leftarrow$ base classifier in H with small error $\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$
 $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
 $Z_t \leftarrow 2[\epsilon_t(1-\epsilon_t)]^{\frac{1}{2}}$ $\triangleright$ normalization factor
 for $i \leftarrow 1$ **to** m **do**
  $D_{t+1}(i) \leftarrow \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
$f \leftarrow \sum_{t=1}^{T} \alpha_t h_t$
return $h = \operatorname{sgn}(f)$

Trivial observations

- Z_t does indeed normalize the distribution, because

$$\epsilon \exp\left(\tfrac{1}{2}\ln\tfrac{1-\epsilon}{\epsilon}\right) + (1-\epsilon)\exp\left(-\tfrac{1}{2}\ln\tfrac{1-\epsilon}{\epsilon}\right) =$$
$$\epsilon\sqrt{\frac{1-\epsilon}{\epsilon}} + (1-\epsilon)\sqrt{\frac{\epsilon}{1-\epsilon}} = 2\sqrt{\epsilon(1-\epsilon)}$$

- The example weight $D_t(i)$ reflects how badly we are doing so far on i'th example:

$$D_{t+1}(i) = \frac{e^{-\alpha_t h_t(x_i) y_i} D_t(i)}{Z_t} = \frac{e^{-\alpha_t h_t(x_i) y_i} e^{-\alpha_{t-1} h_{t-1}(x_i) y_i} D_{t-1}(i)}{Z_t Z_{t-1}}$$

$$= \ldots = \frac{1}{m} \frac{e^{-y_i \sum_{s=1}^{t} \alpha_s h_s(x_i)}}{\prod_{s=1}^{t} Z_s}.$$

Boosting reduces $\mathcal{E}_{\text{train}}$ exponentially

Theorem

The empirical error of the final hypothesis $\widehat{h}$ obeys

$$\mathcal{E}_{\textit{train}}(\widehat{h}) = \frac{1}{m}\sum_{i=1}^{m} \ell_{0/1}(\widehat{h}(x_i), y_i)$$

$$\leq \exp\Big(-2\sum_{t=1}^{T}(1/2-\epsilon_t)^2\Big) \leq \exp(-2\gamma^2 T),$$

where $\gamma = \min_t \gamma_t$ and $\gamma_t = 1/2 - \epsilon_t$ is the ***edge*** *of h_t.*

The usual assumption is that no matter what D_t is, there is some weak learner h_t that has edge $\gamma_t > \tau$. $\rightarrow$ Error decreases with $e^{-\tau^2 T}$.

Proof

$$\ell_{0/1}(z,1) \le e^{-z} \quad \Rightarrow$$

$$\ell_{0/1}(\widehat{h}(x_i), y_i) \le e^{-y_i \sum_t \alpha_t h_t} = m\big(\prod_t Z_t\big) D_{T+1}(i) \quad \Rightarrow$$

$$\mathcal{E}_{\text{train}}(\widehat{h}) = \frac{1}{m}\sum_{i=1}^{m} \ell_{0/1}(\widehat{h}(x_i), y_i) \le \prod_t Z_t = \prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)} =$$

$$\prod_t \sqrt{1-4\gamma_t^2} \le \exp\big(-2\sum_t \gamma_t^2\big) \le \exp\big(-2\gamma^2 T\big)$$

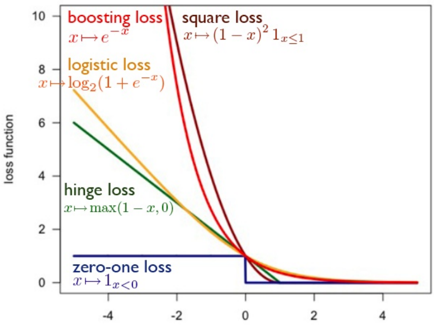
But why is $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$?

Simply minimize $\mathcal{E}_{\text{train}}(\widehat{h}) = \prod_t Z_t$ with respect to $\alpha_1, \alpha_2, \ldots, \alpha_T$: We don't need to know γ for any of this, that's why it is called adaptive boosting.

AdaBoost is like coordinate descent

Error with respect to the **surrogate loss** $\ell_{\text{exp}}(\widehat{f}(x), y) = e^{-yf(x)}$ is

$$F(\boldsymbol{\alpha}) = \sum_{i=1}^{m} e^{-y_i \sum_t \alpha_t h_t(x_i)}$$



$\rightarrow$ Coordinate-wise descent on surrogate loss leads exactly to the AdaBoost!

Use in practice

- Base learners can be e.g., decision trees
- Often they are just *decision stumps*, e.g., "$x_j = \texttt{TRUE}$" or "$x_5 \geq 3.48$" or "blood pressure $>$ 140".
- Stumps are very fast to evaluate, total complexity something like $O((m \log m)N + mNT)$.
- Stumps are not necessarily weak learners (XOR).
- But why doesn't it overfit???

Margin argument

Assuming that zero training error has been achieved, the ℓ_1 –**margin**

$$\rho = \min_{i \in \{1,\ldots,m\}} y_i \frac{\sum_t \alpha_t h_t(x_i)}{\| \boldsymbol{\alpha} \|_1}$$

is a measure of confidence in classifying all the points.

Corresponding bound:

$$\mathcal{E}_{\text{true}}(\widehat{h}) \leq \mathcal{E}_{\text{train}}(\widehat{h}) + \frac{2}{\rho}\mathcal{R} + \sqrt{\frac{\log(1/\delta)}{2m}}.$$

AdaBoost quasi maximizes the margin.

AdaBoost summary

Pros:

- Very simple to code.
- Efficient, $O(mnT)$ for stumps.
- There is some theory.

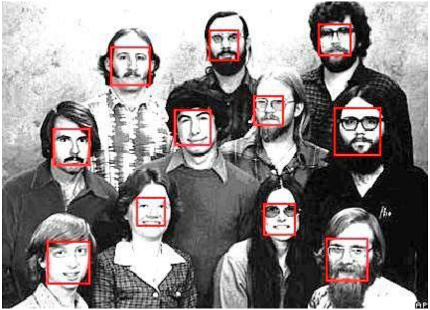
Cons:

- Hard to come up with stopping criterion (overfitting).
- NOISE!!! All the analysis was in the deterministic setting. Even small amounts of label noise can hurt AdaBoost.

Application: Face detection

The Viola–Jones detector

Face detection



To detect where the faces are, need to slide a window over entire image, so

- detector must be very fast,
- must have low false positive rate (typically only a few faces in any image),
- it's okay if training is expensive.

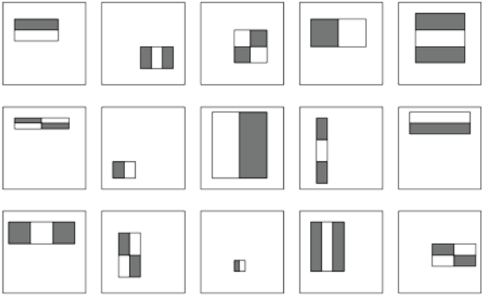
The Viola–Jones approach

In their seminal paper “Rapid object detection using a cascade of simple features” (CVPR 2001) Viola and Jones combine three ideas:

- The “integral image” representation to efficiently compute Haar-like filters
- Boosting on decision stumps to find a very small number of relevant features
- Classifier cascade to drive down false positive rate.

This framework is now standard for detecting faces, cars, pedestrians, etc..

Haar–like image features



The VJ paper uses just three types of Haar–like filters as features:

$x_j =$ average pixel intensity in black area$-$average pixel intensity in white area.

Since the offset and size of the rectangles can be anything, this still gives a lot of features: for 24×24 patches 160,000 features!

The integral image

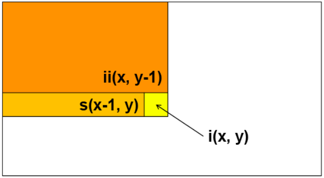
Define $ii(x, y)$ to be the sum of all pixels above and to the left:

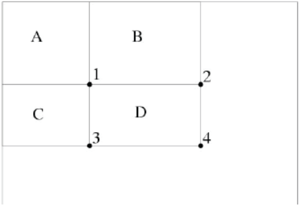
$$ii(x, y) = \sum_{u=1}^{x} \sum_{j=1}^{y} i(x, y).$$

Recursive computation:

$$s(x, y) = s(x, y-1) + i(x, y) \quad \text{(row sum)}$$

$$ii(x, y) = ii(x-1, y) + s(x, y)$$





To compute intensity in D:

$$ii(D) = ii(x_4, y_4) + ii(x_1, y_1) - ii(x_2, y_2) - ii(x_3, y_3).$$

Boosting

Run boosting on the set of weak learners $\{h_{i,p,\theta}\}$, where i selects the feature, $p \in \{+1, -1\}$ is the polarity, and $\theta \in \mathbb{R}$ is the threshold:

$$h_{i,p,\theta}(x) = \mathsf{sgn}(p(f_i(x) - \theta)),$$

where $f_i(x)$ is the value of the i'th feature in the image x.

- Here boosting is just used as a method to select a very sparse set of features.
- Modify AdaBoost to set the final threshold so that there are *no* false negatives.

Quickly finding p and θ

Let $f_i(x_j)$ be the value of feature i on example j. For the corresponding weak learner h_i, the optimal polarity p and threshold θ can be quickly found:

- Find the permutation σ that sorts the examples according to $f_i(x_j)$:

$$f_i(x_{\sigma(1)}) \leq f_i(x_{\sigma(2)}) \leq \ldots \leq f_i(x_{\sigma(N)})$$

- For each $j = 0, 1, 2, \ldots, N$ compute

$$\mathcal{E}_j = \min\{S^+ + (T^- - S^-), S^- + (T^+ - S^+)\},$$

$S^+ = \sum_{k=1}^{j} \mathbb{I}(y_{\sigma(j)} = +) D_t(\sigma(j))$ total weight of + examples to the left

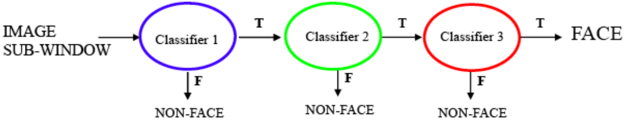
$S^- = \sum_{k=1}^{j} \mathbb{I}(y_{\sigma(j)} = -) D_t(\sigma(j))$ total weight of - examples to the left

$T^+ = \sum_{k=1}^{N} \mathbb{I}(y_{\sigma(j)} = +) D_t(\sigma(j))$ total weight of + examples

$T^- = \sum_{k=1}^{N} \mathbb{I}(y_{\sigma(j)} = -) D_t(\sigma(j))$ total weight of - examples

- Set θ by $f_i(x_{\sigma(j)}) \leq \theta \leq f_i(x_{\sigma(j+1)})$ and p based on which side of the min is smaller.

Classifier cascade



If we have a cascade of classifiers $f_1, \ldots, f_k$, overall

$$\text{FPR}(f) = \prod_{i=1}^{k} FPR(f_i) \qquad \text{DR}(f) = \prod_{i=1}^{k} DR(f_i).$$

Example: If $k = 10$ and $\text{DR}(f_i) \geq 0.99$ and $\text{FPR}(f_i) < 0.3$ for each i, then $\text{FPR}(f) < 6 \cdot 10^{-6}$ while $\text{DR}(f) > 0.9$.

Results

Table 3. Detection rates for various numbers of false positives on the MIT + CMU test set containing 130 images and 507 faces.

	False detections							
Detector	10	31	50	65	78	95	167	422
Viola-Jones	76.1%	88.4%	91.4%	92.0%	92.1%	92.9%	93.9%	94.1%
Viola-Jones (voting)	81.1%	89.7%	92.1%	93.1%	93.1%	93.2%	93.7%	–
Rowley-Baluja-Kanade	83.2%	86.0%	–	–	–	89.2%	90.1%	89.9%
Schneiderman-Kanade	–	–	–	94.4%	–	–	–	–
Roth-Yang-Ahuja	–	–	–	–	(94.8%)	–	–	–

FURTHER READING

- R. Schapire: The boosting approach to machine learning – an overview
- Y. Freund and R. Schapire: A decision-theoretic generalization of on-line learning and an application to boosting (1997)
- R. Schapire and Y. Freund: Boosting: foundations and algorithms (book)
- P. Long and R. Servedio: Random classification noise defeats all convex potential boosters (2008)
- P. Viola and M. Jones: Rapid object detection using a cascade of simple features (CVPR 2001)