AdaBoost

the original paper

Adrian Florea

14th Meetup of Papers We Love (Bucharest Chapter),
17 March 2017

Simplicity vs. fitting to data

1.2	2.8	8.0	3.3	5.0	4.5	7.4	5.6	3.8	6.6	6.1	1.7
-	-	+	-	-	-	+	+	-	+	+	-

What is your classifier?

Simplicity vs. fitting to data

1.2	2.8	8.0	3.3	5.0	4.5	7.4	5.6	3.8	6.6	6.1	1.7
-	-	+	-	-	-	+	+	-	+	+	-

What is your classifier?

$$h(x) = \begin{cases} +1 & \text{if } x \geq 5.3 \\ -1 & \text{otherwise} \end{cases}$$

It was so simple!

Simplicity vs. fitting to data

Let's change only two labels:

1.2	2.8	8.0	3.3	5.0	4.5	7.4	5.6	3.8	6.6	6.1	1.7
-	-	+	-	-	-	+	+	+	-	+	-

What is your classifier?

Simplicity vs. fitting to data

Let's change only two labels:

1.2	2.8	8.0	3.3	5.0	4.5	7.4	5.6	3.8	6.6	6.1	1.7
-	-	+	-	-	-	+	+	+	-	+	-

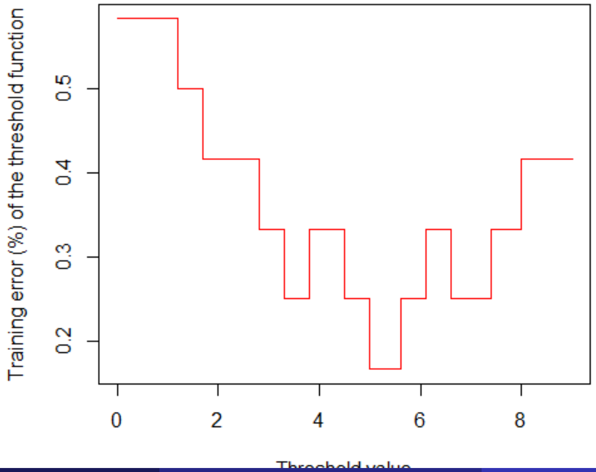
What is your classifier?

$$h(x) = \begin{cases} -1 & \text{if } x < 3.5 \\ +1 & \text{if } 3.5 \leq x < 4 \\ -1 & \text{if } 4 \leq x < 5.5 \\ +1 & \text{if } 5.5 \leq x < 6.5 \\ -1 & \text{if } 6.5 \leq x < 7 \\ +1 & \text{otherwise} \end{cases}$$

It's hard to find a single decision stump

Simplicity vs. fitting to data

Training error of a single decision stump for the last training set



PAC learning definitions I

- Let X be a set called the *instance space*.
- A *concept* c over X is a subset $c \subseteq X$ of the instance space.
- A *concept class* C over X is a set of concepts over X, typically with an associated representation.
- A *target concept* may be any concept $c^* \in C$.
- A *learning algorithm* (called also *learner*) tries to infer an unknown concept (called *hypothesis*), chosen from a known concept class.
- A set of random variables is *independent and identically distributed (i.i.d.)* if each random variable has the same *identical* probability distribution as the others and all are mutually *independent*.
- The instances the learner receives from a distribution D, are independently and identically distributed (i.i.d.).

PAC learning definitions II

- After observing a sequence S of i.i.d. *training examples* of the target concept c^*, the learner L outputs the hypothesis h (its estimate of c^*) from the set H of possible hypotheses.
- The *true error* of h with respect to c^* and D is the probability that h will misclassify an instance drawn randomly according to D:
  $err(h) \equiv \underset{x \in D}{\mathbf{Pr}}[c^*(x) \neq h(x)] = \underset{(x,y) \sim D}{\mathbf{Pr}}[h(x) \neq y]$
- The *training error* is the fraction of training examples misclassified by the hypothesis h:
  $\widehat{err}(h) \equiv \underset{x \in S}{\mathbf{Pr}}[c^*(x) \neq h(x)] =$
  $= \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}[c^*(x_i) \neq h(x_i)] = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}[h(x_i) \neq y_i]$

Weak learnability

A concept class C is *PAC-learnable* or *strongly learnable* by L if for any D over X, and for any $c \in C$, and for any positive value of ε, and for any positive value of δ, the learner L will output in a polynomial time in $\frac{1}{\varepsilon}$, $\frac{1}{\delta}$, $size(x \in X)$, $size(c)$, a hypothesis h such that $\mathbf{Pr}[err(h) > \varepsilon] \leq \delta$

An arbitrarily small true error means nearly perfect generalization and that is usually unrealistic!

A concept class C is *weakly learnable* by L if for any D over X, and for any $c \in C$, and for any positive value of δ, and for some fixed positive value of $\varepsilon = \frac{1}{2} - \gamma$ with $\gamma > 0$, the learner L will output in a polynomial time in $\frac{1}{\varepsilon}$, $\frac{1}{\delta}$, $size(x \in X)$, $size(c)$, a hypothesis h such that $\mathbf{Pr}[err(h) > \varepsilon] \leq \delta$

Weak $\equiv$ Strong

Hypothesis Boosting Problem - Michael Kearns, December 1988

> *"is it the case that any target class that is weakly learnable is in fact strongly learnable? Note that we pose the question in a representation-independent setting in that the hypothesis class used by the strong learning algorithm need not be the same as that used by the weak learning algorithm."*

Proof - Robert Schapire, June 1990

> *"Theorem: A concept class C is weakly learnable if and only if it is strongly learnable"*

Boosting introduction

Let's consider a simple classifier, the decision stump:
$h(\mathbf{x}; \theta) = \text{sign}(w_1 x_k - w_0), \mathbf{x} \in \mathbb{R}^d, \theta = \{k, w_1, w_0\}$
Note it acts on only one component of $\mathbf{x}$

Voted combination of m classifiers, built iteratively in m rounds:
$h_m(\mathbf{x}) = \alpha_1 h(\mathbf{x}; \theta_1) + \ldots + \alpha_m h(\mathbf{x}; \theta_m)$
where the votes α_i are non-negative (a higher vote means a better classifier)

Let's consider this loss function:
$L(y, f(\mathbf{x})) = \exp(-yf(\mathbf{x}))$
with the value $1/e$ (small) when classifies correctly,
or the value e (large) when classifies incorrectly

$$\min \sum_{i=1}^{n} L(y_i, h_m(\mathbf{x}_i)) = \min \sum_{i=1}^{n} \exp(-y_i h_m(\mathbf{x}_i)) =$$
$$\min \sum_{i=1}^{n} \exp(-y_i(h_{m-1}(\mathbf{x}_i) + \alpha_m h(\mathbf{x}_i;\theta_m))) =$$
$$\min \sum_{i=1}^{n} \exp(-y_i h_{m-1}(\mathbf{x}_i))\exp(-y_i\alpha_m h(\mathbf{x}_i;\theta_m)))=$$

At round m, the expression $\exp(-y_i h_{m-1}(\mathbf{x}_i))$ under min is fixed (it's already constructed), let's denote it by W_i^{m-1} so:

$$= \min \sum_{i=1}^{n} W_i^{m-1}\exp(-y_i\alpha_m h(\mathbf{x}_i;\theta_m))) =$$
$$= \min\{\sum_{y_i=h(\mathbf{x}_i;\theta_m)} W_i^{m-1}\exp(-\alpha_m) + \sum_{y_i\neq h(\mathbf{x}_i;\theta_m)} W_i^{m-1}\exp(\alpha_m)\} =$$
$$= \min\{\exp(-\alpha_m)\sum_{y_i=h(\mathbf{x}_i;\theta_m)} W_i^{m-1} + \exp(\alpha_m)\sum_{y_i\neq h(\mathbf{x}_i;\theta_m)} W_i^{m-1}\} =$$
$$= \min\{\exp(-\alpha_m)\sum_{i=1}^{n} W_i^{m-1}\mathbf{1}(y_i = h(\mathbf{x}_i;\theta_m))+$$
$$+\exp(\alpha_m)\sum_{i=1}^{n} W_i^{m-1}\mathbf{1}(y_i \neq h(\mathbf{x}_i;\theta_m))\} =$$
$$= \min\{\exp(-\alpha_m)(\sum_{i=1}^{n} W_i^{m-1} - \sum_{i=1}^{n} W_i^{m-1}\mathbf{1}(y_i \neq h(\mathbf{x}_i;\theta_m))+$$
$$+\exp(\alpha_m)\sum_{i=1}^{n} W_i^{m-1}\mathbf{1}(y_i \neq h(\mathbf{x}_i;\theta_m))\} =$$
$$= \min\{\exp(-\alpha_m)\sum_{i=1}^{n} W_i^{m-1}-$$
$$-\exp(-\alpha_m)\sum_{i=1}^{n} W_i^{m-1}\mathbf{1}(y_i \neq h(\mathbf{x}_i;\theta_m))+$$
$$+\exp(\alpha_m)\sum_{i=1}^{n} W_i^{m-1}\mathbf{1}(y_i \neq h(\mathbf{x}_i;\theta_m))\}$$

But this min is on α_m, so:

$\frac{\partial \sum_{i=1}^{n} L(y_i, h_m(\mathbf{x}_i))}{\partial \alpha_m} = 0$, i.e.

$$-\exp(-\alpha_m) \sum_{i=1}^{n} W_i^{m-1} +$$
$$+\exp(-\alpha_m) \sum_{i=1}^{n} W_i^{m-1} \mathbf{1}(y_i \neq h(\mathbf{x}_i; \theta_m)) +$$
$$+\exp(\alpha_m) \sum_{i=1}^{n} W_i^{m-1} \mathbf{1}(y_i \neq h(\mathbf{x}_i; \theta_m)) = 0$$

Results:

$$\exp(-\alpha_m) \frac{\sum_{i=1}^{n} W_i^{m-1} \mathbf{1}(y_i \neq h(\mathbf{x}_i; \theta_m))}{\sum_{i=1}^{n} W_i^{m-1}} +$$
$$+\exp(\alpha_m) \frac{\sum_{i=1}^{n} W_i^{m-1} \mathbf{1}(y_i \neq h(\mathbf{x}_i; \theta_m))}{\sum_{i=1}^{n} W_i^{m-1}} = \exp(-\alpha_m)$$

$$\frac{\sum_{i=1}^{n} W_i^{m-1} \mathbf{1}(y_i \neq h(\mathbf{x}_i; \theta_m))}{\sum_{i=1}^{n} W_i^{m-1}} = \frac{\exp(-\alpha_m)}{\exp(-\alpha_m) + \exp(\alpha_m)}$$

The left term is the *weighted* training error at round m, so:

$$\varepsilon_m = \frac{\exp(-\alpha_m)}{\exp(-\alpha_m)+\exp(\alpha_m)} = \frac{1}{1+\exp(2\alpha_m)}$$

We obtain the formula of the vote at round m:

$$\alpha_m = \frac{1}{2}\ln(\frac{1-\varepsilon_m}{\varepsilon_m})$$

We have denoted above the weights by:

$$W_i^m = \exp(-y_i h_m(\mathbf{x}_i)) = exp(-y_i(h_{m-1}(\mathbf{x}_i) + \alpha_m h(\mathbf{x}_i;\theta_m))) =$$
$$= W_i^{m-1}\exp(-y_i\alpha_m h(\mathbf{x}_i;\theta_m))$$

Results a recursive update rule of the weights at round m based on the value at the previous round:

$$W_i^m = W_i^{m-1}\exp(-y_i\alpha_m h(\mathbf{x}_i;\theta_m))$$

At every round, these weights can be normalized such that: $\sum_{i=1}^n W_i^m = 1$

AdaBoost Algorithm - Freund & Schapire, 1995

Input:

a sequence of n labeled examples $\{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$,
a number T of iterations,
and a weak learner *WeakLearn* that at round m will produce a hypothesis $h(\mathbf{x}; \theta_m)$

Initialize:

$W_i^1 = \frac{1}{n}, i = 1, \ldots, n$, because we don't have any information about weights at the first round.

Iterate from m = 1 to T:

Call *WeakLearn* to produce the hypothesis $h(\mathbf{x}; \theta_m)$ (*AdaBoost has no direct control on the hypothesis*)

Calculate the weighted training error: $\varepsilon_m = \frac{\sum_{i=1}^n W_i^m \mathbf{1}(y_i \neq h(\mathbf{x}_i; \theta_m))}{\sum_{i=1}^n W_i^m}$

Calculate the vote: $\alpha_m = \frac{1}{2} \ln(\frac{1-\varepsilon_m}{\varepsilon_m})$

Update the weights: $W_i^m \leftarrow W_i^m \exp(-y_i \alpha_m h(\mathbf{x}_i; \theta_m))/Z_m$ and normalize their values (division by Z_m).

Output the final hypothesis: $\text{sign}(h_T(\mathbf{x}))$

From the normalized weights update formula, we have:

$$W_i^T = W_i^1 \frac{\exp(-\sum_{m=1}^T y_i \alpha_m h(\mathbf{x}_i;\theta_m))}{\prod_{m=1}^T Z_m} = W_i^1 \frac{\exp(-y_i \sum_{m=1}^T \alpha_m h(\mathbf{x}_i;\theta_m))}{\prod_{m=1}^T Z_m}$$

But $W_i^1 = \frac{1}{n}$ and $\sum_{m=1}^T \alpha_m h(\mathbf{x}_i;\theta_m) = h_T(\mathbf{x}_i)$, so:
$W_i^T = \frac{1}{n} \frac{\exp(-y_i h_T(\mathbf{x}_i))}{\prod_{m=1}^T Z_m}$

The training error of the final classifier is:

$\widehat{err}(h_T) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}[h_T(\mathbf{x_i}) \neq y_i] = \frac{1}{n} \sum_{i=1}^n \begin{cases} 1, & \text{if } y_i \neq h_T(\mathbf{x_i}) \\ 0, & \text{otherwise} \end{cases} =$

$= \frac{1}{n} \sum_{i=1}^n \begin{cases} 1, & \text{if } y_i h_T(\mathbf{x_i}) < 0. \\ 0, & \text{otherwise} \end{cases}$

But $\exp(-y_i h_T(\mathbf{x_i})) > 1$ if $y_i h_T(\mathbf{x_i}) < 0$, so:
$\widehat{err}(h_T) < \frac{1}{n} \sum_{i=1}^n \exp(-y_i h_T(\mathbf{x}_i)) = \frac{1}{n} n \prod_{m=1}^T Z_m \sum_{i=1}^n W_i^T$
From $\sum_{i=1}^n W_i^T = 1$ we obtain the following upper bound for the training error of the final classifier:

$\widehat{err}(h_T) < \prod_{m=1}^{T} Z_m$

$$W_i^{m+1} = W_i^m \exp(-y_i \alpha_m h(\mathbf{x}_i; \theta_m))/Z_m =$$
$$\frac{W_i^m}{Z_m} \begin{cases} \sqrt{\frac{1-\varepsilon_m}{\varepsilon_m}}, & \text{if } y_i h_T(\mathbf{x_i}) < 0. \\ \sqrt{\frac{\varepsilon_m}{1-\varepsilon_m}}, & \text{otherwise} \end{cases} \text{ because } \alpha_m = \frac{1}{2} \ln(\frac{1-\varepsilon_m}{\varepsilon_m})$$
$$1 = \sum_{i=i}^{n} W_i^{m+1} = \sqrt{\frac{\varepsilon_m}{1-\varepsilon_m}} \sum_{y_i = h_m(\mathbf{x}_i)} \frac{W_i^m}{Z_m} + \sqrt{\frac{1-\varepsilon_m}{\varepsilon_m}} \sum_{y_i \neq h_m(\mathbf{x}_i)} \frac{W_i^m}{Z_m} =$$
$$= \frac{1}{Z_m}(\sqrt{\frac{\varepsilon_m}{1-\varepsilon_m}}(1-\varepsilon_m) + \sqrt{\frac{1-\varepsilon_m}{\varepsilon_m}}\varepsilon_m) = \frac{2}{Z_m}\sqrt{\varepsilon_m(1-\varepsilon_m)}, \text{ so:}$$
$$Z_m = 2\sqrt{\varepsilon_m(1-\varepsilon_m)}$$

We denote by *edge* $\gamma_m = \frac{1}{2} - \varepsilon_m$, i.e. $\varepsilon_m = \frac{1-2\gamma_m}{2}$, so:

$$Z_m = \sqrt{1 - 4\gamma_m^2}$$

From the inequality $\sqrt{1-x} \leq \exp(-x/2)$, we obtain:
$$Z_m = \sqrt{1-4\gamma_m^2} \leq exp(-2\gamma_m^2)$$

That means:
$\widehat{err}(h_T) < \prod_{m=1}^{T} Z_m \leq \exp(-2\sum_{m=1}^{T} \gamma^2)$

Now, turning back to our weak learning assumption, we have: $\gamma_m \geq \gamma$ for $m = 1, \ldots, T$, so have completely proved that:

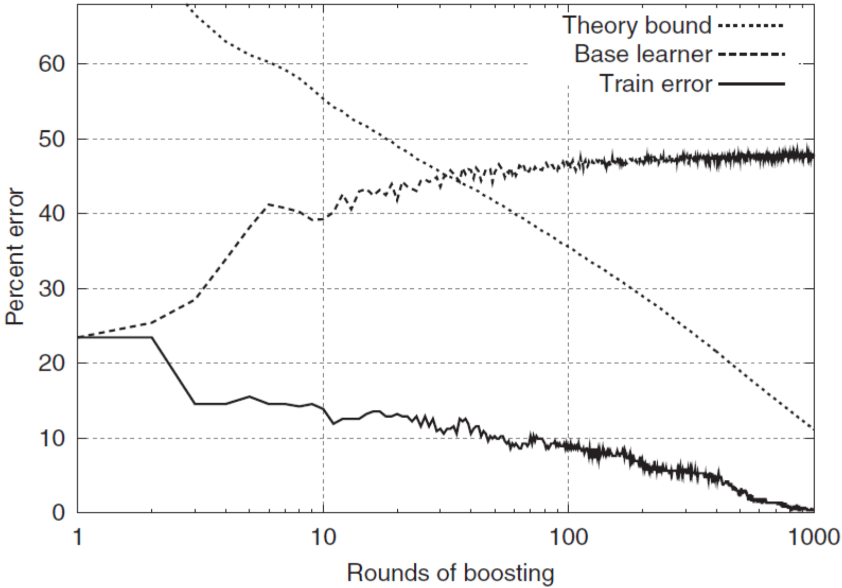
Theorem

$\widehat{err}(h_T) < \exp(-2T\gamma^2)$

The training error of the final classifier constructed with AdaBoost is exponentially converging to 0 for any γ with the number of rounds.

In reality, the convergence is even better, as it results from the following figure:

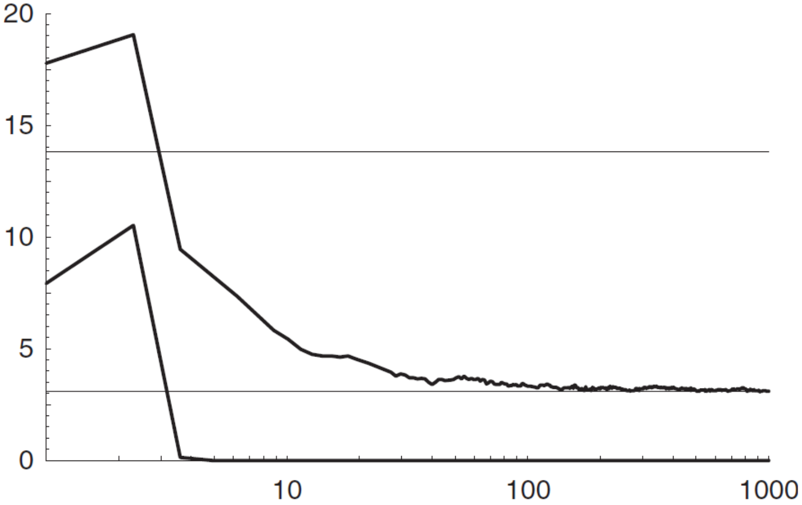
[Schapire & Freund's book, p. 57]



No overfitting

Test error and training error

[Schapire & Freund's book, p. 16]



Bibliography I

R.E. Schapire, Y. Freund
Boosting. Foundations and Algorithms
MIT Press, 2012

Y. Freund, R.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

T. Jaakkola
6.867 Machine Learning
MIT CSAIL, 2004

M. Kearns
Thoughts on Hypothesis Boosting
Project for Ron Rivest's machine learning course at MIT, Unpublished manuscript, December 1988

Bibliography II

J. Kun
Weak Learning, Boosting, and the AdaBoost algorithm
Jeremy Kun's Blog, May 2015

R.E. Schapire
The Strength of Weak Learnability
Machine Learning, 5(2):197-227 (1990)

R.E. Schapire
A Boosting Tutorial
May 2005