Boosting

David Rosenberg

New York University

March 28, 2017

Boosting Introduction

Ensembles: Parallel vs Sequential

- Ensemble methods combine multiple models
- **Parallel ensembles**: each model is built independently
  - e.g. bagging and random forests
  - Main Idea: Combine many (high complexity, low bias) models to reduce variance
- **Sequential ensembles**:
  - Models are generated sequentially
  - Try to add new models that do well where previous models lack

Overview

- AdaBoost algorithm
  - weighted training sets and weighted classification error
- AdaBoost minimizes training error
- AdaBoost train/test learning curves (seems resistant to overfitting)
- (If time) AdaBoost is minimizing exponential loss function (but in a special way)
- Tomorrow
  - Forward stagewise additive modeling
  - Gradient Boosting (generalizes beyond exponential loss function)

The Boosting Question: Weak Learners

- A **weak learner** is a classifier that does slightly better than random.
- Weak learners are like "rules of thumb":
  - If an email has "Viagra" in it, more likely than not it's spam.
  - Email from a friend is probably not spam.
  - A linear decision boundary.
- Can we **combine** a set of weak classifiers to form single classifier that makes accurate predictions?
  - Posed by Kearns and Valiant (1988,1989):
- Yes! **Boosting** solves this problem. [Rob Schapire (1990).]

(We mention "weak learners" for historical context, but we'll avoid this terminology and associated assumptions...)

AdaBoost: The Algorithm

AdaBoost: Setting

- AdaBoost is for **binary classification:** $\mathcal{Y} = \{-1, 1\}$
- **Base hypothesis space** $\mathcal{H} = \{h : \mathcal{X} \to \{-1, 1\}\}$.
    - **Note**: not producing a score, but an actual class label.
    - we'll call it a **base learner**
    - (when base learner satisfies certain conditions, it's called a "weak learner")
- Typical base hypothesis spaces:
    - **Decision stumps** (tree with a single split)
    - Trees with few terminal nodes
    - Linear decision functions

Weighted Training Set

- Training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.
- Weights $(w_1, \ldots, w_n)$ associated with each example.
- **Weighted empirical risk**:

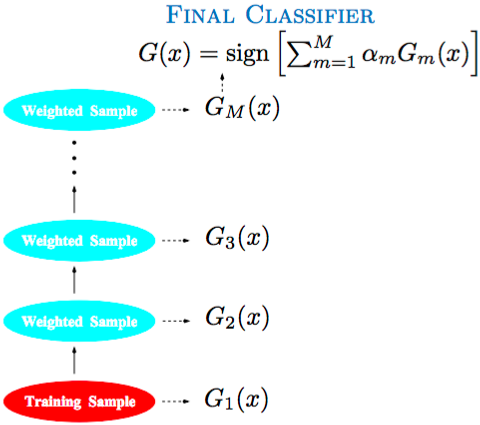
$$\hat{R}_n^w(f) = \frac{1}{W} \sum_{i=1}^{n} w_i \ell(f(x_i), y_i) \quad \text{where } W = \sum_{i=1}^{n} w_i$$

- Can train a model to minimize weighted empirical risk.
- What if model cannot conveniently be trained to reweighted data?
- Can sample a new data set from $\mathcal{D}$ with probabilities$(w_1/W, \ldots w_n/W)$.

AdaBoost - Rough Sketch

- Training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.
- Start with equal weight on all training points $w_1 = \cdots = w_n = 1$.
- Repeat for $m = 1, \ldots, M$:
  - Find base classifier $G_m(x)$ that **tries** to fit weighted training data (but may not do that well)
  - Increase weight on the points $G_m(x)$ misclassifies
- So far, we've generated M classifiers: $G_1(x), \ldots, G_m(x)$.

AdaBoost: Schematic



From ESL Figure 10.1

AdaBoost - Rough Sketch

- Training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.
- Start with equal weight on all training points $w_1 = \cdots = w_n = 1$.
- Repeat for $m = 1, \ldots, M$:
  - Base learner fits weighted training data and returns $G_m(x)$
  - Increase weight on the points $G_m(x)$ misclassifies
- Final prediction $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$. (recall $G_m(x) \in \{-1, 1\}$)
- The α_m's are nonnegative,
  - larger when G_m fits its weighted $\mathcal{D}$ well
  - smaller when G_m fits weighted $\mathcal{D}$ less well

Adaboost: Weighted Classification Error

- In round m, base learner gets a weighted training set.
    - Returns a base classifier $G_m(x)$ that roughly minimizes weighted $0-1$ error.
- The **weighted 0-1 error** of $G_m(x)$ is

$$\text{err}_m = \frac{1}{W}\sum_{i=1}^{n} w_i 1(y_i \neq G_m(x_i)) \quad \text{where } W = \sum_{i=1}^{n} w_i.$$

- Notice: $\text{err}_m \in [0,1]$.

AdaBoost: Classifier Weights

- The weight of classifier $G_m(x)$ is $\alpha_m = \ln\left(\frac{1-\text{err}_m}{\text{err}_m}\right)$.

- Note that weight $\alpha_m \to 0$ as weighted error $\text{err}_m \to 0.5$ (random guessing).

AdaBoost: Example Reweighting

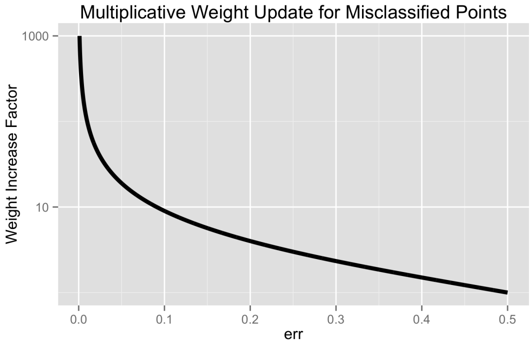
- We train G_m to minimize weighted error, and it achieves err_m.
- Then $\alpha_m = \ln\left(\frac{1-\text{err}_m}{\text{err}_m}\right)$ is the weight of G_m in final ensemble.
- Suppose w_i is weight of example i before training:
  - If G_m classfies x_i correctly, then w_i is unchanged.
  - Otherwise, w_i is increased as

$$
\begin{aligned}
w_i &\leftarrow w_i e^{\alpha_m} \\
&= w_i \left(\frac{1-\text{err}_m}{\text{err}_m}\right)
\end{aligned}
$$

  - For $\text{err}_m < 0.5$, this always increases the weight.

Adaboost: Example Reweighting

- Any misclassified point has weight adjusted as $w_i \leftarrow w_i \left(\frac{1-\text{err}_m}{\text{err}_m}\right)$.



- The smaller err_m, the more we increase weight of misclassified points.

AdaBoost: Algorithm

Given training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

1. Initialize observation weights $w_i = 1$, $i = 1, 2, \ldots, n$.
2. For $m = 1$ to M:
   1. Base learner fits weighted training data and returns $G_m(x)$
   2. Compute **weighted empirical 0-1 risk**:

   $$\text{err}_m = \frac{1}{W} \sum_{i=1}^{n} w_i 1(y_i \neq G_m(x_i)) \quad \text{where } W = \sum_{i=1}^{n} w_i.$$

   3. Compute $\alpha_m = \ln\left(\frac{1-\text{err}_m}{\text{err}_m}\right)$ **[classifier weight]**
   4. Set $w_i \leftarrow w_i \cdot \exp\left[\alpha_m 1(y_i \neq G_m(x_i))\right], \quad i = 1, 2, \ldots, n$ **[example weight adjustment]**
3. Ouptut $G(x) = \operatorname{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

AdaBoost with Decision Stumps

- After 1 round:

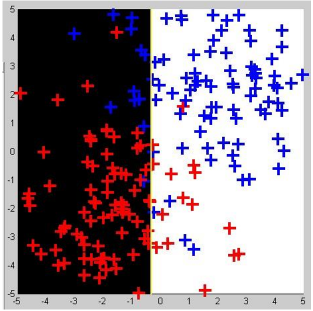


Figure: Plus size represents weight. Blackness represents score for red class.

KPM Figure 16.10

AdaBoost with Decision Stumps

- After 3 rounds:

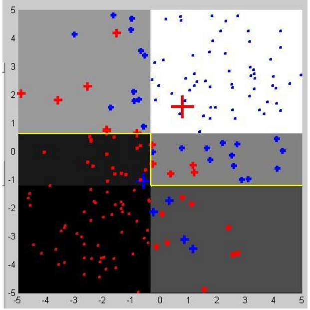


Figure: Plus size represents weight. Blackness represents score for red class.

KPM Figure 16.10

AdaBoost with Decision Stumps

- After 120 rounds:

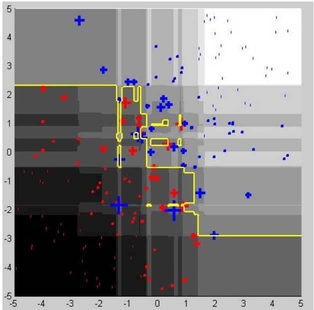


Figure: Plus size represents weight. Blackness represents score for red class.

KPM Figure 16.10

Does AdaBoost Minimize Training Error?

AdaBoost: Does it actually minimize training error?

- Methods we've seen so far come in two categories:
  - Regularized empirical risk minimization (L1/L2 regression, SVM, kernelized versions)
  - Trees
- GD and SGD converge to minimizers of objective function on training data
- Trees achieve 0 training error unless same input occurs with different outputs
  - without any limit on tree complexity
- So far, AdaBoost is just an algorithm.
- Does an AdaBoost classifier $G(x)$ even minimize training error?
- Yes, if our weak classifiers have an "**edge**" over random.

AdaBoost: Does it actually minimize training error?

- Assume base classifier, $G_m(x)$ has $\text{err}_m \leqslant \frac{1}{2}$.
    - (Otherwise, let $G_m(x) \leftarrow -G_m(x)$.)
- Define the **edge** of classifier $G_m(x)$ at round m to be

$$\gamma_m = \frac{1}{2} - \text{err}_m.$$

- Measures how much better than random G_m performs.

AdaBoost: Does it actually minimize training error?

Theorem

The empirical 0-1 risk of the AdaBoost classifier $G(x)$ is bounded as

$$\frac{1}{n}\sum_{i=1}^{n} 1(y_i \neq G(x)) \leqslant \prod_{m=1}^{M} \sqrt{1-4\gamma_m^2}.$$

- What's are the possible values for $\sqrt{1-4\gamma_m^2}$?.
- Proof is an optional homework problem on Homework 6.

AdaBoost: Does it actually minimize training error?

Suppose $\text{err}_m \leqslant 0.4$ for all m.

- Then the "edge" is $\gamma_m = .5 - .4 = .1$, and training error is bounded as follows:

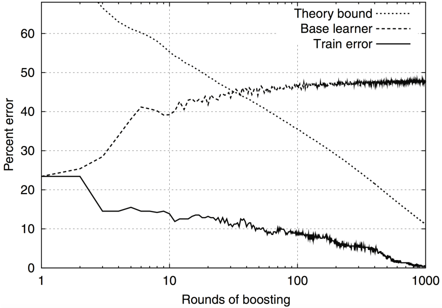
$$\frac{1}{n}\sum_{i=1}^{n} 1(y_i \neq G(x)) \leqslant \prod_{m=1}^{M}\sqrt{1-4(.1)^2} \approx (.98)^M$$

- Bound decreases exponentially:

$$\begin{aligned} .98^{100} &\approx .133 \\ .98^{200} &\approx .018 \\ .98^{300} &\approx .002 \end{aligned}$$

- With a consistent edge, training error decreases very quickly to 0.

Training Error Rate Curves



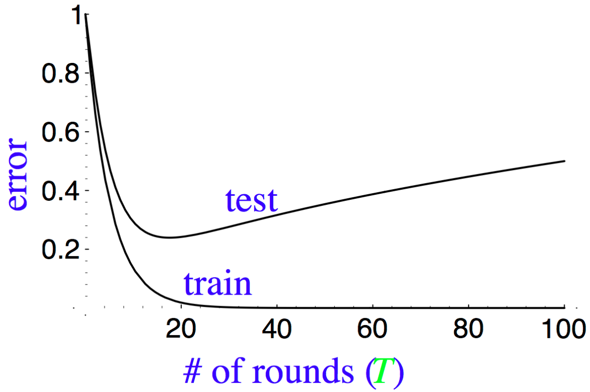
- "Base learner" plots error rates err_M on weighted training sets after M rounds of boosting
- "Train error" is the training error of the combined classifier
- "Theory bound" plots the training error bound given by the theorem

Figure 3.1 from *Boosting: Foundations and Algorithms* by Schapire and Freund.

Test Performance of Boosting

Typical Train / Test Learning Curves

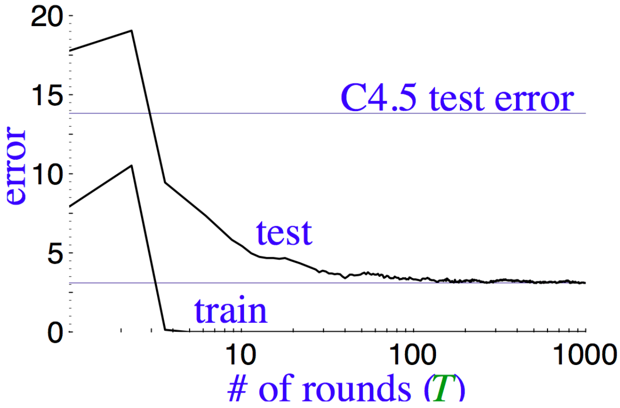
- Might expect too many rounds of boosting to overfit:



From Rob Schapire's NIPS 2007 Boosting tutorial.

Learning Curves for AdaBoost

- In typical performance, AdaBoost is surprisingly resistant to overfitting.
- Test continues to improve even after training error is zero!



From Rob Schapire's NIPS 2007 Boosting tutorial.

Boosting Fits an Additive Model

Adaptive Basis Function Model

- AdaBoost produces a classification score function of the form

$$\sum_{m=1}^{M} \alpha_m G_m(x)$$

  - each G_m is a **base classifier**
- The G_m's are like basis functions, but they are learned from the data.
- Let's move beyond classification models...

Adaptive Basis Function Model

- **Base hypothesis space** $\mathcal{H}$
- An **adaptive basis function expansion** over $\mathcal{H}$ is

$$f(x) = \sum_{m=1}^{M} \nu_m h_m(x),$$

  - $h_m \in \mathcal{H}$ chosen in a learning process ("adaptive")
  - $\nu_m \in \mathbf{R}$ are **expansion coefficients.**
- **Note:** We are taking linear combination of outputs of $h_m(x)$.
  - Functions in $h_m \in \mathcal{H}$ must produce values in $\mathbf{R}$ (or a vector space)

How to fit an adaptive basis function model?

- **Loss function**: $\ell(y, \hat{y})$
- **Base hypothesis space**: $\mathcal{H}$ of **real-valued** functions
- Want to find

$$f(x) = \sum_{m=1}^{M} \nu_m h_m(x)$$

that **minimizes empirical risk**

$$\frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, f(x_i)\right).$$

- We'll proceed in stages, adding a new h_m in every stage.

Forward Stagewise Additive Modeling (FSAM)

- Start with $f_0 \equiv 0$.
- After $m-1$ stages, we have

$$f_{m-1} = \sum_{i=1}^{m-1} \nu_i h_i,$$

  where $h_1, \ldots, h_{m-1} \in \mathcal{H}$.

- Want to find
  - **step direction** $h_m \in \mathcal{H}$ and
  - **step size** $\nu_i > 0$
- So that

$$f_m = f_{m-1} + \nu_i h_m$$

  minimizes empirical risk.

Forward Stagewise Additive Modeling

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
   1. Compute:
   $$(\nu_m, h_m) = \operatorname*{arg\,min}_{\nu \in \mathbf{R}, h \in \mathcal{H}} \sum_{i=1}^{n} \ell\left(y_i, f_{m-1}(x_i) \underbrace{+\nu h(x_i)}_{\text{new piece}}\right).$$
   2. Set $f_m = f_{m-1} + \nu_m h$.
3. Return: f_M.

Exponential Loss and AdaBoost

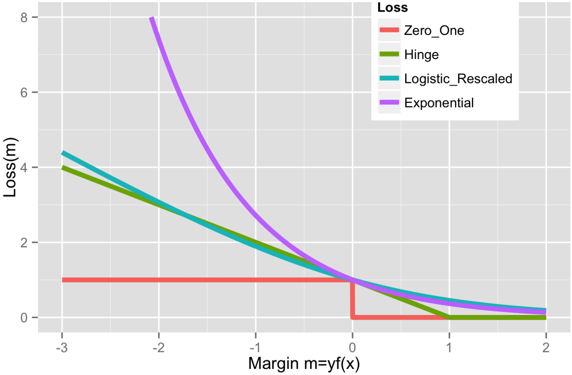
- Take loss function to be
$$\ell(y, f(x)) = \exp\left(-yf(x)\right).$$
- Let $\mathcal{H}$ be our base hypothesis space of classifiers $h : \mathcal{X} \to \{-1, 1\}$.
- Then Forward Stagewise Additive Modeling (FSAM) reduces to AdaBoost!
  - Proof on Homework #6 (and see HTF Section 10.4).
- Only difference:
  - AdaBoost gets whichever G_m the base learner returns from $\mathcal{H}$ – no guarantees it's best in $\mathcal{H}$.
  - FSAM explicitly requires getting the best in $\mathcal{H}$

$$G_m = \underset{G \in \mathcal{H}}{\arg\min} \sum_{i=1}^{N} w_i^{(m)} 1(y_i \neq G(x_i))$$

Robustness and AdaBoost

Exponential Loss

- Note that exponential loss puts a very large weight on bad misclassifications.



AdaBoost / Exponential Loss: Robustness Issues

- When Bayes error rate is high (e.g. $\mathbb{P}(f^*(X) \neq Y) = 0.25$)
  - e.g. there's some intrinsic randomness in the label
  - e.g. training examples with same input, but different classifications.
- Best we can do is predict the most likely class for each X.
- Some training predictions **should be wrong** (because example doesn't have majority class)
  - AdaBoost / exponential loss puts a lot of focus on geting those right
- Empirically, AdaBoost has degraded performance in situations with
  - high Bayes error rate, or when there's
  - high "**label noise**"
- Logistic loss performs better in settings with high Bayes error

Population Minimizer

Population Minimizers

- In traditional statistics, the **population** refers to
  - the full population of a group, rather than a sample.
- In machine learning, the **population case** is the hypothetical case of
  - an infinite training sample from $P_{\mathcal{X}\times\mathcal{Y}}$.
- A **population minimizer** for a loss function is another name for the risk minimizer.
- For the exponential loss $\ell(m) = e^{-m}$, the population minimizer is given by

$$f^*(x) = \frac{1}{2}\ln\frac{\mathbb{P}(Y=1 \mid X=x)}{\mathbb{P}(Y=-1 \mid X=x)}$$

- (Short proof in KPM 16.4.1)
- By solving for $\mathbb{P}(Y=1 \mid X=x)$, we can give probabilistic predictions from AdaBoost as well.

Population Minimizers

- AdaBoost has the robustness issue because of the exponential loss.
- Logistic loss $\ell(m) = \ln(1+e^{-m})$ has the same population minimizer.
  - But works better with high label noise or high Bayes error rate
- Population minimizer of SVM hinge loss is

$$f^*(x) = \operatorname{sign}\left[\mathbb{P}(Y=1 \mid X=x) - \frac{1}{2}\right].$$

- Because of the sign, we cannot solve for the probabilities.