STAT 154
Modern Statistical Prediction and Machine Learning
Lecture 24: Boosting

Raaz Dwivedi

UC Berkeley

April 30, 2019

Logistics

- Last class, May 2 Thursday
- RRR Week (no labs/class/office hours), three sessions:
  1. Raaz's office hours on Tuesday: 1:00-2:00 PM Evans 444
  2. Review Session on Thursday: 8-9:30 AM VLSB 2040.
  3. Bin's office hours on Friday: 1:30-2:30 PM Evans 409–her office
- Final's week: Usual office hours:
  1. Raaz: Monday 10:30-11:30 AM and Thursday 9:30-10:30 AM Evans 444
  2. Yuansi: Tuesday 1:00-3:00 PM Evans 444
  3. Bin: Tuesday 9:30-10:30 AM and Wednesday 1:30-2:30 PM Evans 409–her office
- Final Exam May 16th: 7-10 PM, Li Ka Sching 245. Seat map will be uploaded soon.
- For students with conflicts: 4-7 PM Evans 344.

Additive Model

- Recall OLS and Kernel regression:

$$f(\mathbf{x}) = \beta^\top \mathbf{x} = \sum_{j=1}^{d} \beta_j x_j,$$

$$f(\mathbf{x}) = \beta^\top \phi(\mathbf{x}) = \sum_{j=1}^{p} \beta_j \phi_j(\mathbf{x})$$

  where we fit β but features $\phi(\mathbf{x})$ are specified and not fitted.
- What if we can also learn the features $\phi(\mathbf{x})$?
- **Notation:**
  1. $(y_i, \mathbf{x}_i)$ denotes i-th sample point and for a given feature $\mathbf{x}_i$ or $\mathbf{x}$, we use x_{ij} or x_j to denote the j-th coordinate of that feature.
  2. Given predicted value $\widehat{y}$ for an observation y, the **zero-one loss** or the **classification error** is denoted as

$$\mathbb{I}[\widehat{y} \neq y] = \begin{cases} 0 & \text{if } \widehat{y} = y \\ 1 & \text{if } \widehat{y} \neq y \end{cases}$$

Additive Model

- In additive model, we try to fit a model of the form:

$$f(\mathbf{x}) = \sum_{m=1}^{M} \beta_m b(\mathbf{x}; \gamma_m)$$

such that

$$y_i \approx f(\mathbf{x}_i) \qquad \text{for regression}$$
$$y_i \approx \text{sign}(f(\mathbf{x}_i)) \quad \text{for classification}$$

- Here β_m are the coefficients and $b(\mathbf{x}; \gamma)$ are usually simple functions parameterized by a parameter γ. e.g., if it is a stump then γ denotes the variable and the threshold.
- For OLS, we can see γ as just the index of the feature $\mathbf{x}$ and $b(\mathbf{x}, j) = x_j$.

Additive Model

Optimization Problem

Given a loss function $\mathcal{L}$, we want to solve the following problem:

$$\min \sum_{i=1}^{n} \mathcal{L}(y_i, f(\mathbf{x}_i)) = \min_{(\beta_m, \gamma_m)_{m=1}^{M}} \sum_{i=1}^{n} \mathcal{L}(y_i, \sum_{m=1}^{M} \beta_m b(\mathbf{x}_i; \gamma_m))$$

- We wish to build a combination of M-learners but in general joint optimization is complex and hard.
- Nonetheless building one learner is relatively easy as we choose the class of such learners"

$$\min_{(\beta,\gamma)} \sum_{i=1}^{n} \mathcal{L}(y_i, \beta b(\mathbf{x}_i; \gamma)) \quad \text{Base Model/Learner} \tag{1}$$

Complex learners: Bagging, Random Forest, Adaboost

- **Bagging:** Create M-bootstrapped datasets; fit a base learner on each datasets and **simply average** the learners.
  - Averaging leads to smoother boundaries
  - But the models have correlation since they have similar samples and use all the features to build the models.
- **Random Forest:** De-correlated bagging. Create M-bootstrapped datasets; fit a base learner using only a random subset of features on each datasets and **simply average** the learners.
- In bagging and random forest, the M-learners can be learned independent of one another, i.e., the process can be parallelized.
- **Adaboost**: **Sequential training** with emphasis on previous mistakes (weighted retraining), and models are combined using **weighted average** at the end

Adaboost: Sequential Reweighted Smart "Bagging"

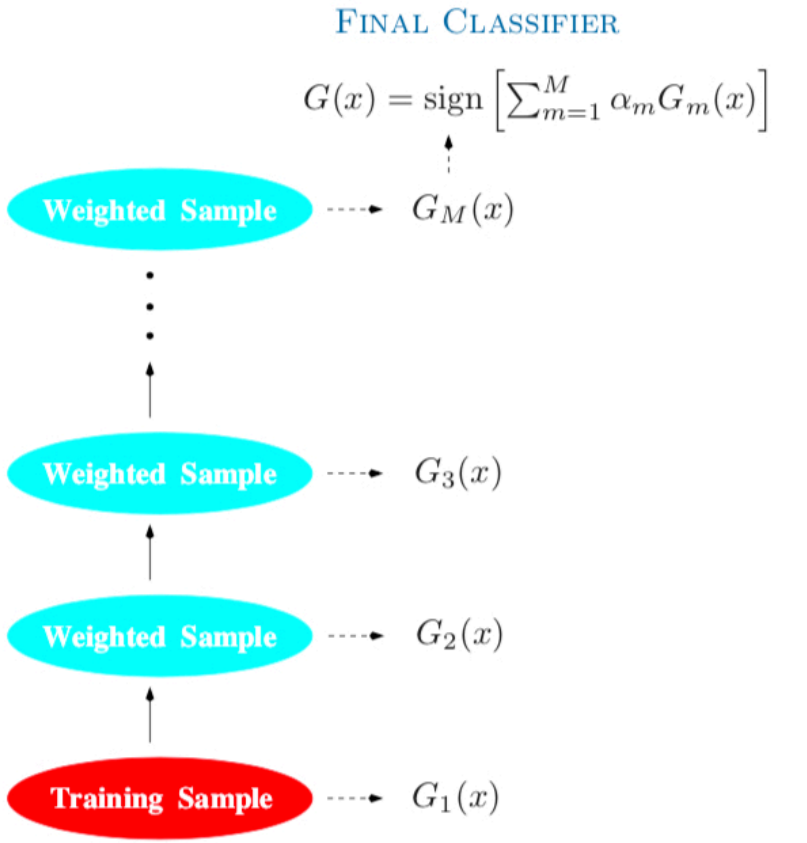


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted v sions of the dataset, and then combined to produce a final prediction.*

1

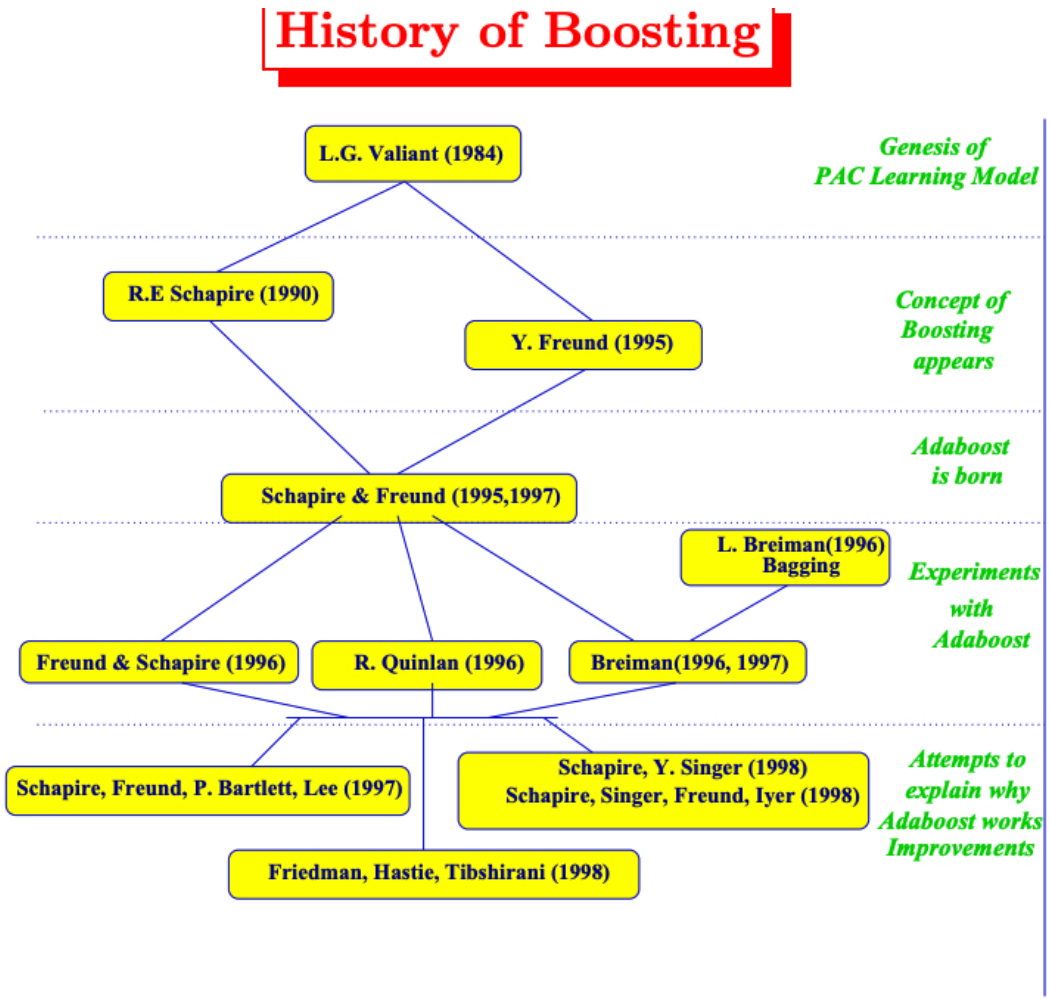
[1] Also show slides 6-21 of the notes:
http://www.csc.kth.se/utbildning/kth/kurser/DD3364/Lectures/Lecture9.pdf

History of Random Forests

Bagging+Random selection of features

- Bagging introduced for variance reduction by Breiman in 1994: "Bagging Predictors"
- Random selection of features introduced by Tin Kam Ho in 1995 "Random Decision Forests" and independently by Amit and Geman in 1997: " Shape quantization and recognition with randomized trees"
- The two ideas were combined and popularized by Leo Breiman in 2001: "Random Forests"

History of Adaboost (Slide from Hastie's notes)



Boosting: Pros and Cons

- Works pretty well in practice, is often robust to moderate amount of noise; usually better than random forest if tuned well
- Test error may go down even after the training error has hit zero
- May have to tune several hyper-parameters in gradient boosting: depth of tree (usually shallow), number of trees, learning rate (can use cross-validation)
- Sequential training may make the learning process slow
- Loss of interpretability (but relative variable importances can be computed by looking at how many times a given variable contributes to a split across all trees) [See ESL book 10.13 or `http://www.stat.cmu.edu/~ryantibs/datamining/lectures/25-boost.pdf`]

Random Forest: Pros and Cons

- Works pretty well in practice, harder to overfit the data
- Relatively lesser number of hyper-parameters as the trees are often grown to purity: number of trees and number of features to be selected at each node (can use cross-validation)
- Can be parallelized, so training can be fast if you have many machines
- Loss of interpretability (but relative variable importances can be computed by looking at how many times a given variable contributes to a split across all trees) [see ESL book 15.3.2 or `http://www.stat.cmu.edu/~ryantibs/datamining/lectures/25-boost.pdf`]
- Real time prediction might be slow because of a large number of deep trees
- When the number of variables is large, but the fraction of relevant variables small, random forests are likely to perform poorly with small number of trees

Adaboost: Example

Example, ESL Page 399: $n = 1000$ points drawn from the model:

$$Y_i = \begin{cases} 1 & \text{if } \sum_{j=1}^{10} x_{ij}^2 > \chi_{10}^2(0.5) \\ -1 & \text{otherwise} \end{cases}$$

where each $X_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$ for $i \in [n], j \in [10]$.

- How will a stump classifier perform on independent validation data?
- How will a deep decision tree perform on independent validation data?
- How about boosting?

Adaboost: Example

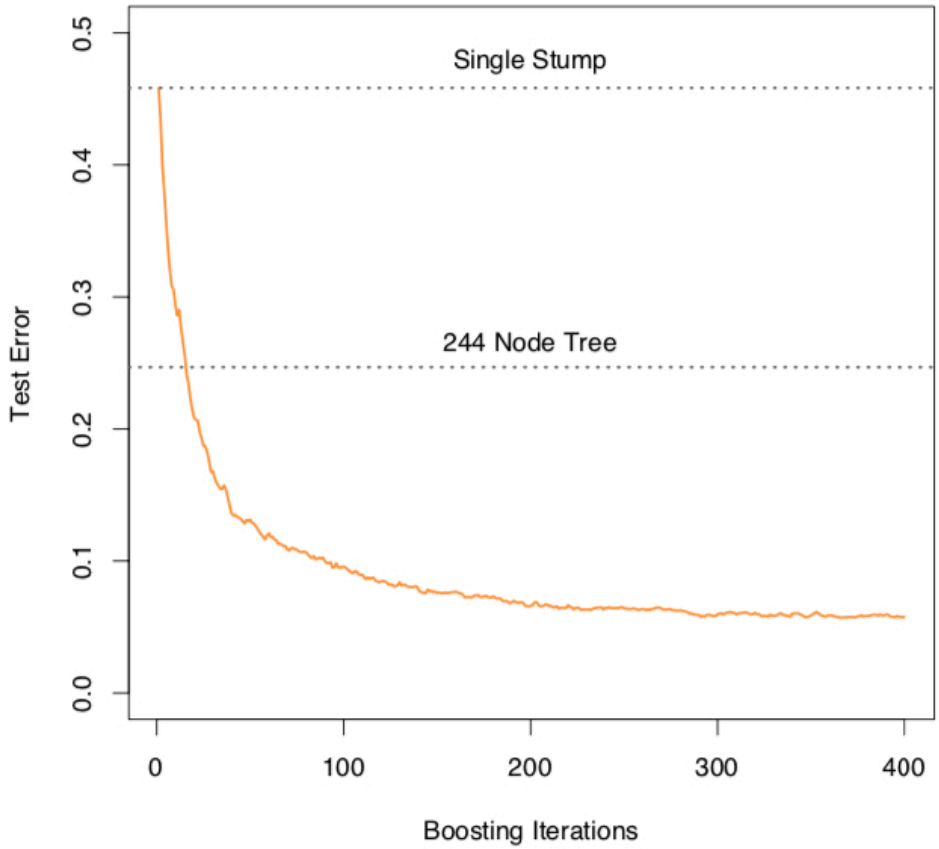


FIGURE 10.2. *Simulated data (10.2): test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 244-node classification tree.*

From Adaboost to General Boosting

- Adaboost was originally introduced as a smart bagging algorithm which made use of training on re-weighted samples based on the previous mistakes and then taking a *weighted* average.
- Later on its connection to a more general framework was found: "forward stagewise additive modeling".
- We will now derive the Adaboost update using the general framework.

Boosting in the context of additive models

Recall the optimization problem:

$$\min_{(\beta_m,\gamma_m)_{m=1}^M} \sum_{i=1}^{n} \mathcal{L}(y_i, \sum_{m=1}^{M} \beta_m b(\mathbf{x}_i; \gamma_m))$$

Boosting = Forward stagewise additive modeling

Learn one model at a time sequentially and use the previous model while learning the new model.

Boosting in the context of additive models

Recall the optimization problem:

$$\min_{(\beta_m,\gamma_m)_{m=1}^M} \sum_{i=1}^n \mathcal{L}(y_i, \sum_{m=1}^M \beta_m b(\mathbf{x}_i;\gamma_m))$$

Boosting = Forward stagewise additive modeling

$$\widehat{\beta}_1, \widehat{\gamma}_1 \leftarrow \arg\min_{(\beta,\gamma)} \sum_{i=1}^n \mathcal{L}(y_i, \beta b(\mathbf{x}_i;\gamma))$$

$$\widehat{\beta}_2, \widehat{\gamma}_2 \leftarrow \arg\min_{(\beta,\gamma)} \sum_{i=1}^n \mathcal{L}(y_i, \widehat{\beta}_1 b(\mathbf{x}_i;\widehat{\gamma}_1) + \beta b(\mathbf{x}_i;\gamma))$$

$$\dots$$

$$\widehat{\beta}_M, \widehat{\gamma}_M \leftarrow \arg\min_{(\beta,\gamma)} \sum_{i=1}^n \mathcal{L}(y_i, \sum_{m=1}^{M-1} \widehat{\beta}_m b(\mathbf{x}_i;\widehat{\gamma}_m) + \beta b(\mathbf{x}_i;\gamma))$$

$$\widehat{f}_M(x) = \sum_{m=1}^M \widehat{\beta}_m b(\mathbf{x}_i;\widehat{\gamma}_m)$$

Adaboost as forward stagewise additive modeling

To proceed further, we need to specify two things:

Loss function

We use the exponential loss function:

$$\mathcal{L}(y, f(\mathbf{x})) = e^{-yf(\mathbf{x})}$$

Note that

$$\mathcal{L}(y, f(\mathbf{x}) + g(\mathbf{x})) = e^{-yf(\mathbf{x})} \cdot e^{-yg(\mathbf{x})}$$

Base learners

The choice of base learner is up to us. Usually weak learners are preferred so that each step is cheap. Popular examples: shallow decision tree or stump. For two-class classification, we assume $b(\mathbf{x}; \gamma) \in \{-1, 1\}$.

Adaboost as forward stagewise additive modeling with exponential loss

- Given the classifier $\widehat{f}_t$ at iteration t, we need to solve

$$\widehat{\beta}_{t+1}, \widehat{\gamma}_{t+1} \leftarrow \arg\min_{(\beta,\gamma)} \sum_{i=1}^{n} \mathcal{L}(y_i, \underbrace{\sum_{m=1}^{t} \widehat{\beta}_m b(\mathbf{x}_i; \widehat{\gamma}_m)}_{\widehat{f}_t} + \beta b(\mathbf{x}_i; \gamma))$$

  where each term $b(\mathbf{x}; \gamma)$ denotes a simple classifier.
- Using simpler notation $b(\mathbf{x}; \gamma) = g(\mathbf{x})$, we have

$$\widehat{\beta}_{t+1}, \widehat{g}_{t+1} \leftarrow \arg\min_{\beta,g} \sum_{i=1}^{n} \mathcal{L}(y_i, \widehat{f}_t(\mathbf{x}_i) + \beta g(\mathbf{x}_i))$$

$$\widehat{\beta}_{t+1}, \widehat{g}_{t+1} \leftarrow \arg\min_{\beta,g} \sum_{i=1}^{n} e^{-y_i \widehat{f}_t(\mathbf{x}_i)} e^{-y_i \beta g(\mathbf{x}_i)}.$$

  How should we solve the problem?

Adaboost: Boosting with Exponential Loss

- Simplify the objective:

$$\widehat{\beta}_{t+1}, \widehat{g}_{t+1} \leftarrow \arg\min_{\beta, g} \sum_{i=1}^{n} \underbrace{e^{-y_i \widehat{f}_t(\mathbf{x}_i)}}_{w_i^{(t)}} e^{-y_i \beta g(\mathbf{x}_i)}.$$

- Note that $yg \in \{-1, 1\}$ and hence for any g and β, we can collect terms and further simplify the loss as

$$\sum_{i=1}^{n} w_i^{(t)} e^{-y_i \beta g(\mathbf{x}_i)} = \sum_{y_i = g(\mathbf{x}_i)} w_i^{(t)} e^{-\beta} + \sum_{y_i \neq g(\mathbf{x}_i)} w_i^{(t)} e^{\beta}$$

$$= (e^{\beta} - e^{-\beta}) \underbrace{\sum_{i=1}^{n} w_i^{(t)} \mathbb{I}(y_i \neq g(\mathbf{x}_i))}_{\text{Weighted classification error of } g \text{ on the data}}$$

$$+ e^{-\beta} \sum_{i=1}^{n} w_i^{(t)}$$

Adaboost: Boosting with Exponential Loss

- Thus the problem reduces to finding

$$\arg\min_{\beta,g}(e^{\beta}-e^{-\beta})\underbrace{\sum_{i=1}^{n}w_i^{(t)}\mathbb{I}(y_i\neq g(\mathbf{x}_i))}_{E_g}+e^{-\beta}\underbrace{\sum_{i=1}^{n}w_i^{(t)}}_{W}$$

$$=\arg\min_{\beta,g}\quad W\left((e^{\beta}-e^{-\beta})\frac{E_g}{W}+e^{-\beta}\right)$$

- Homework: For a given g, solving for β gives $\widehat{\beta}=\frac{1}{2}\log\frac{1-E_g/W}{E_g/W}$ and plugging back this β we see that optimal g satisfies

$$\widehat{g}=\arg\min_{g}\sqrt{\frac{E_g}{W}(1-\frac{E_g}{W})}$$

$$=\arg\min_{g}E_g=\arg\min_{g}\sum_{i=1}^{n}w_i^{(t)}\mathbb{I}(y_i\neq g(\mathbf{x}_i))$$

Adaboost: Summary of one-step

- Given the classifier $\widehat{f}_t$, define the weights $w_i^{(t)} = e^{-y_i \widehat{f}_t(\mathbf{x}_i)}$
- Then compute the next classifier as

$$\widehat{f}_{t+1} = \widehat{f}_t + \widehat{\beta}\widehat{g}$$

where

$$\widehat{g} = \arg\min_g \sum_{i=1}^n w_i^{(t)} \mathbb{I}(y_i \neq g(\mathbf{x}_i)) \quad \text{Reweighted training}$$

$$\widehat{\beta} = \frac{1}{2}\log\frac{1-e_{\widehat{g}}}{e_{\widehat{g}}}, \quad e_{\widehat{g}} = \frac{\sum_{i=1}^n w_i^{(t)} \mathbb{I}(y_i \neq \widehat{g}_{t+1}(\mathbf{x}_i))}{\sum_{i=1}^n w_i^{(t)}}$$

Weighted-error of $\widehat{g}$

Note that $\widehat{\beta} > 0 \iff e_{\widehat{g}} < 1/2$, which is true. Why? Because $\widehat{g}$ should perform better than random classifier. Why? Otherwise we can simply multiply it by -1.

Adaboost: Summary of one-step—New weights

- We can compute the new weights as follows:

$$
\begin{aligned}
w_i^{(t+1)} = e^{-y_i \widehat{f}_{t+1}(\mathbf{x}_i)} &= e^{-y_i(\widehat{f}_t(\mathbf{x}_i)+\widehat{\beta}\widehat{g}(\mathbf{x}_i))} \\
&= e^{-y_i \widehat{f}_t(\mathbf{x}_i)} \cdot e^{-y_i \widehat{\beta}\widehat{g}(\mathbf{x}_i))} \\
&= w_i^{(t)} \cdot \begin{cases} e^{\widehat{\beta}} & \text{if } y_i \neq \widehat{g}(\mathbf{x}_i) \\ e^{-\widehat{\beta}} & \text{if } y_i = \widehat{g}(\mathbf{x}_i) \end{cases} \\
&= w_i^{(t)} \cdot \begin{cases} \sqrt{\frac{1-e_{\widehat{g}}}{e_{\widehat{g}}}} & \text{if } y_i \neq \widehat{g}(\mathbf{x}_i) \\ \sqrt{\frac{e_{\widehat{g}}}{1-e_{\widehat{g}}}} & \text{if } y_i = \widehat{g}(\mathbf{x}_i). \end{cases}
\end{aligned}
$$

Thus the training samples that the new learner $\widehat{g}$ classifies incorrectly get weighted more for the next round and for the correctly classified the weight is reduced.

Adaboost: Data $\{\mathbf{x}_i, y_i\}_{i=1}^n$, Learners $\mathcal{G} = \{g : \mathbb{R}^d \to \{-1, 1\}\}$

1. Initialize weights $w_i^{(0)} = 1/n$ for $i = 1, \ldots, n$.
2. For $t = 0, \ldots, T$, do
   (i) Compute classifier on the **re-weighted** training data and its **weighted-error**:

$$\widehat{g}_{t+1} = \arg\min_{g \in \mathcal{G}} \sum_{i=1}^n w_i^{(t)} \mathbb{I}(y_i \neq g(\mathbf{x}_i)) \quad \text{and}$$

$$e_{\widehat{g}_{t+1}} = \frac{\sum_{i=1}^n w_i^{(t)} \mathbb{I}(y_i \neq \widehat{g}_{t+1}(\mathbf{x}_i))}{\sum_{i=1}^n w_i^{(t)}}$$

   (ii) Compute $\widehat{\beta}_{t+1} = \frac{1}{2} \log \frac{1 - e_{\widehat{g}_{t+1}}}{e_{\widehat{g}_{t+1}}}$ and the **new-weights**:

$$w_i^{(t+1)} = w_i^{(t)} \cdot \begin{cases} e^{\widehat{\beta}_{t+1}} & \text{if } y_i \neq \widehat{g}_{t+1}(\mathbf{x}_i) \\ e^{-\widehat{\beta}_{t+1}} & \text{if } y_i = \widehat{g}(\mathbf{x}_i). \end{cases}$$

3. Output $\widehat{f}_{\text{boosted}}(\mathbf{x}) = \text{sign}(\sum_{t=1}^T \widehat{\beta}_t \widehat{g}_t(\mathbf{x}))$

Why exponential loss

"The AdaBoost algorithm was originally motivated from a very different perspective and its equivalence to forward stage-wise additive modeling based on exponential loss was only discovered five years after its inception."

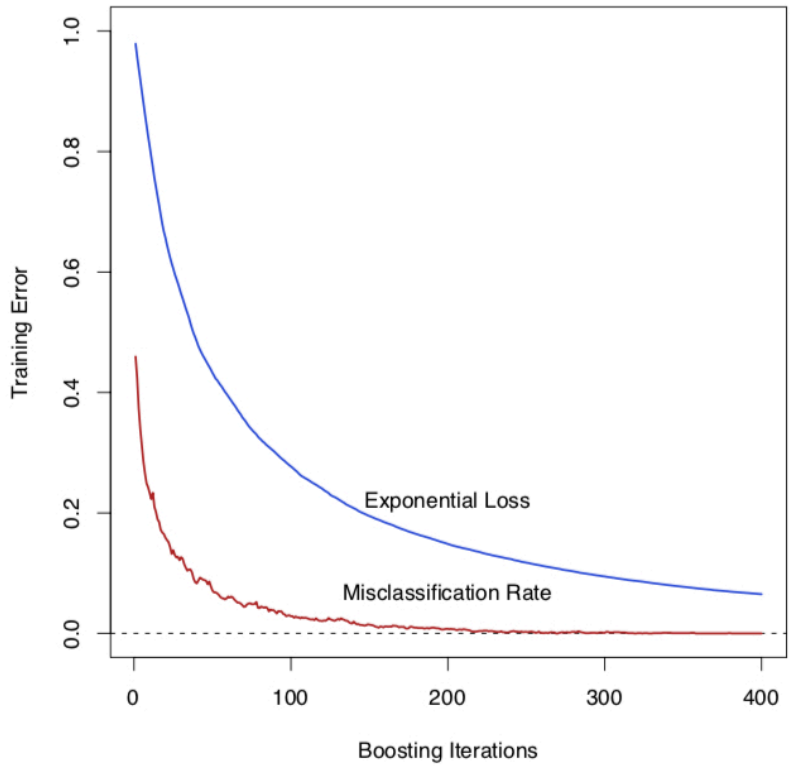


FIGURE 10.3. *Simulated data, boosting with stumps: misclassification error rate on the training set, and average exponential loss:* $(1/N)\sum_{i=1}^{N}\exp(-y_i f(x_i))$. *After about 250 iterations, the misclassification error is zero, while the exponential loss continues to decrease.*

Why exponential loss

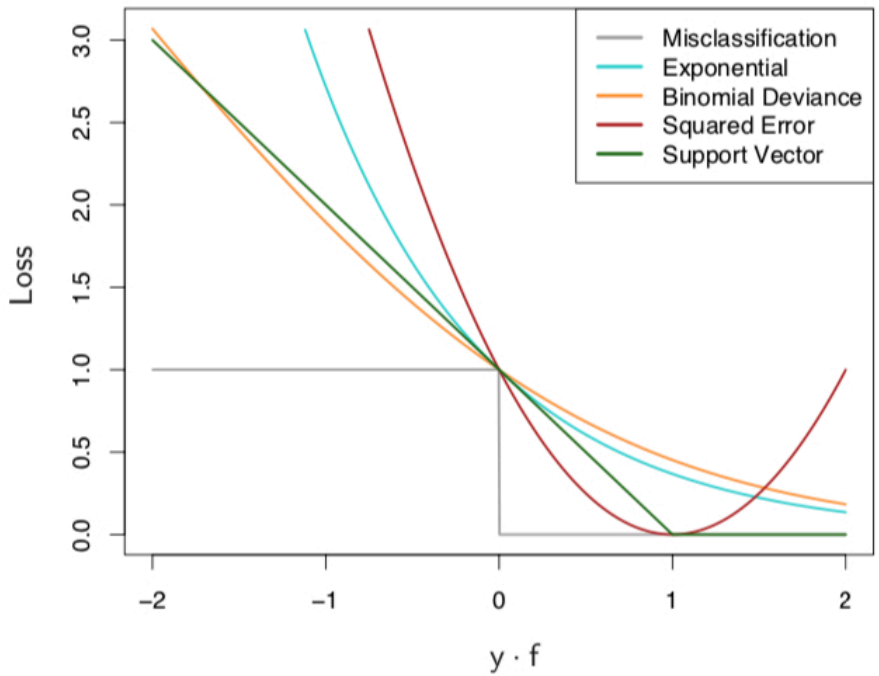


FIGURE 10.4. *Loss functions for two-class classification. The response is* $y = \pm 1$*; the prediction is* f*, with class prediction* $\text{sign}(f)$*. The losses are misclassification:* $I(\text{sign}(f) \neq y)$*; exponential:* $\exp(-yf)$*; binomial deviance:* $\log(1 + \exp(-2yf))$*; squared error:* $(y - f)^2$*; and support vector:* $(1 - yf)_+$ *(see Section 12.3). Each function has been scaled so that it passes through the point* $(0, 1)$*.*

When exact minimizers don't admit a closed form

Gradient Boosting: Boosting+Gradient Descent

When at a given iteration $\widehat{\beta}_t, \widehat{\gamma}_t$ don't admit a closed form, one uses **Gradient descent** with a proper step size but in the **function space**—hence the name gradient boosting.

Bagging vs Random Forest vs Gradient Boosting on Spam dataset

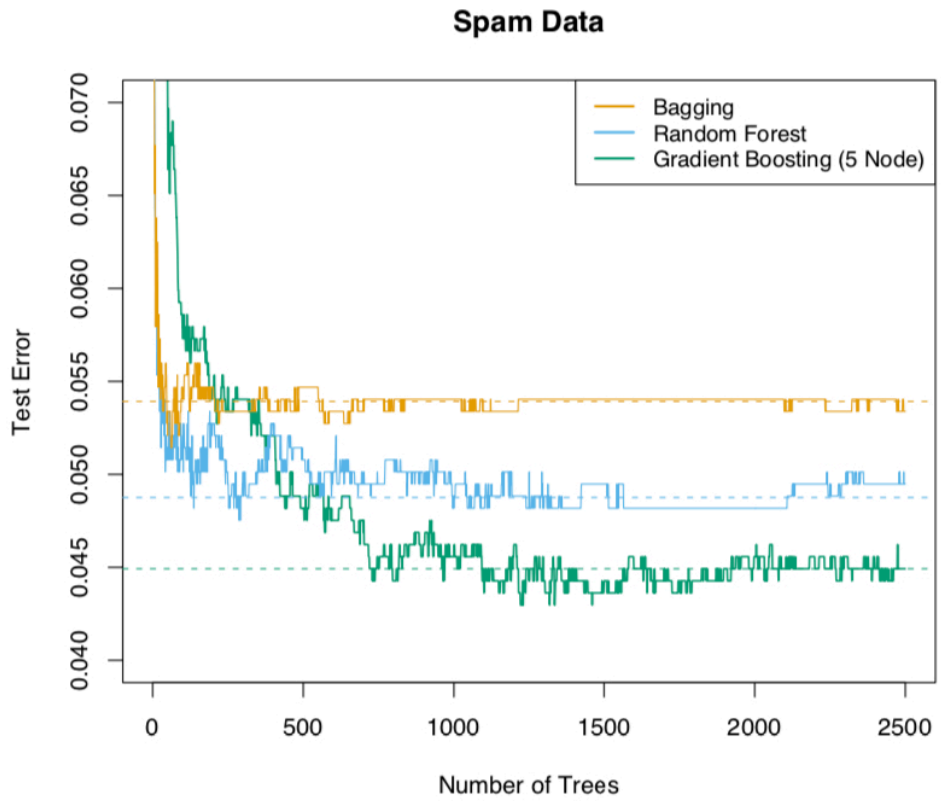


FIGURE 15.1. *Bagging, random forest, and gradient boosting, applied to the spam data. For boosting, 5-node trees were used, and the number of trees were chosen by 10-fold cross-validation (2500 trees). Each "step" in the figure corresponds to a change in a single misclassification (in a test set of 1536).*

Summary of additive modeling

- Building complex models using simple models:
  1. **Bagging**: Bootstrap samples (draw random subset), train again, and average / majority voting; smoother boundaries but correlation across models leads to flattening of error
  2. **Random Forest /** smart-bagging: Bootstrap samples, train with random subset of features (de-correlated training) and average / majority voting
  3. **Adaboost**: sequential training based on previous mistakes (weighted retraining), and models are combined using weighted average at the end

- Most of the images are taken from the ESL book.
- Reading ESL book: 10.1-10.5
- Reference for some variable importances: `http://www.stat.cmu.edu/~ryantibs/datamining/lectures/25-boost.pdf`