Boosting 2: Classification

-Statistical Machine Learning-

Lecturer: Darren Homrighausen, PhD

Additive models (for classification)

As squared error loss isn't quite right for classification, additive logistic regression is a popular approach

Suppose $Y \in \{-1, 1\}$

$$\log\left(\frac{\mathbb{P}(Y = 1|X)}{\mathbb{P}(Y = -1|X)}\right) = \sum_{j=1}^{p} h_j(x_j) = h(X)$$

This gets inverted in the usual way to acquire a probability estimate

$$\pi(X) = \mathbb{P}(Y = 1|X) = \frac{e^{h(X)}}{1 + e^{h(X)}}$$

($h(X) = X^\top \beta$ gives us (linear) logistic regression, with classifier $g(X) = \mathrm{sgn}(h(X))$)

These models are usually fit by numerically maximizing the binomial likelihood, and hence enjoy all the asymptotic optimality features of MLEs

Additive models (for classification)

Example: In R, this can be fit with the package gam

In the gam package there is a dataset kyphosis

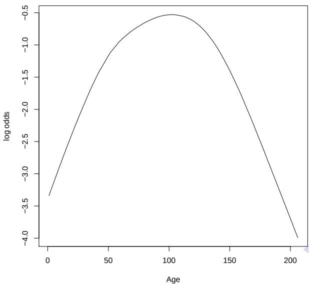
This dataset examines a disorder of the spine

Let's look at two possible covariates Age and Number

(Number refers to the number of vertebrae that were involved in a surgery)

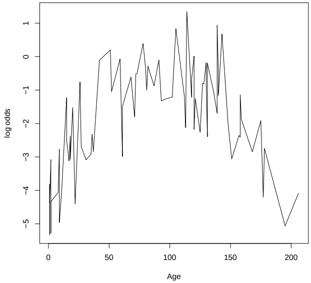
Additive models (for classification)

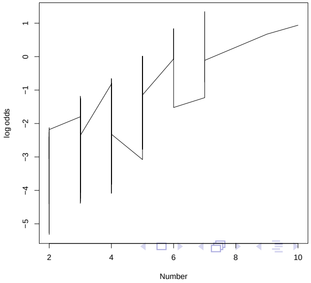
```
library(gam)
data(kyphosis)

out = gam(Kyphosis~s(Age,3),family=binomial,data=kyphosis)
out.pred = predict(out)
plot(sort(kyphosis$Age),out.pred[order(kyphosis$Age)],
       type='l',xlab='Age',ylab='log odds')
```



Additive models (for classification)

```
out = gam(Kyphosis ~ s(Age,3) + s(Number,3),
        family = binomial, data=kyphosis)
out.pred = predict(out)
plot(sort(kyphosis$Age),out.pred[order(kyphosis$Age)],
     type='l',xlab='Age',ylab='log odds')
plot(sort(kyphosis$Number),out.pred[order(kyphosis$Number)]
     type='l',xlab='Number',ylab='log odds')
```





Adaboost

AdaBoost outline

We give an overview of 'AdaBoost.M1.'

(Freund and Schapire (1997))

First, train the classifier as usual

(This is done by setting $w_i \equiv 1/n$)

At each step b, the misclassified observations have their weights increased

(Implicitly, this lowers the weight on correctly classified observations)

A new classifier is trained which emphasizes the previous mistakes

AdaBoost algorithm

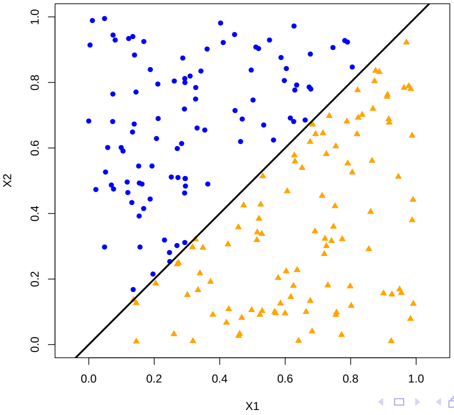
1. Initialize $w_i \equiv 1/n$
2. For $b = 1, \ldots, B$
   2.1 Fit $g_b(X)$ on $\mathcal{D}$, weighted by w_i
   2.2 Compute
   $$R_b = \frac{\sum_{i=1}^n w_i \mathbf{1}(Y_i \neq g_b(X_i))}{\sum_{i=1}^n w_i}$$
   2.3 Find $\beta_b = \log((1 - R_b)/R_b)$
   2.4 Set $w_i \leftarrow w_i \exp\{\beta_b \mathbf{1}(Y_i \neq g_b(X_i))\}$
3. Output: $g(X) = \mathrm{sgn}\left(\sum_{b=1}^B \beta_b g_b(X)\right)$

Some supporting simulations

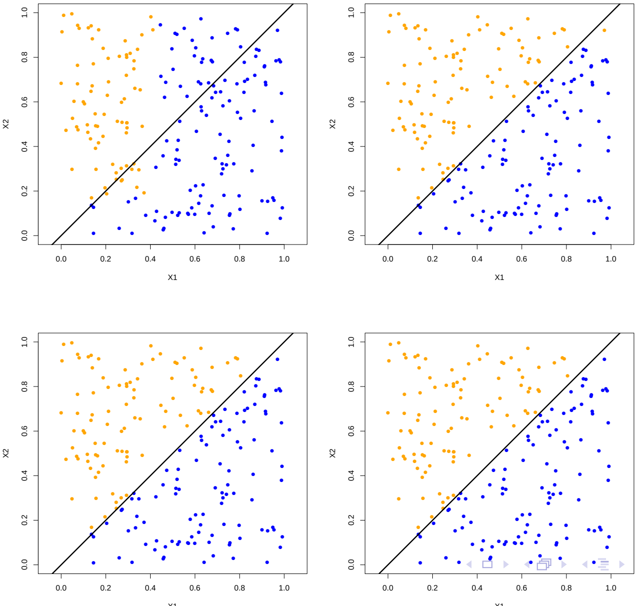
AdaBoost: Simulation

Let's use the classifier trees, but with 'depth 2-stumps'

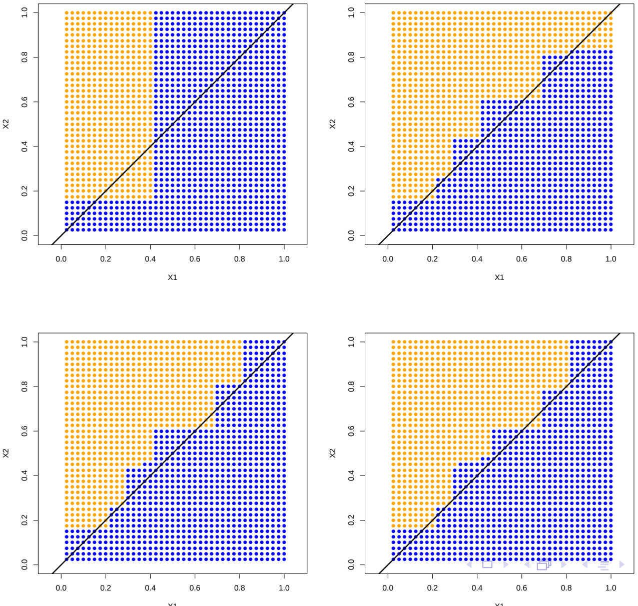
These are trees, but constrained to have no more than 4 terminal nodes



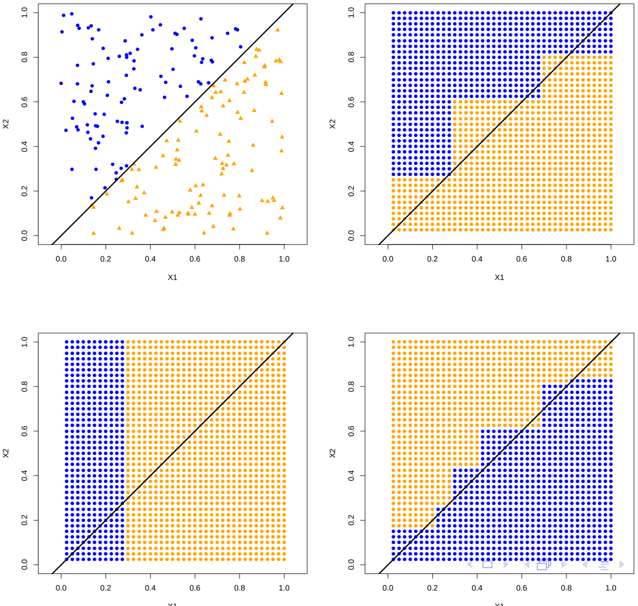
AdaBoost: Increasing B (train)



AdaBoost: Increasing B (test)

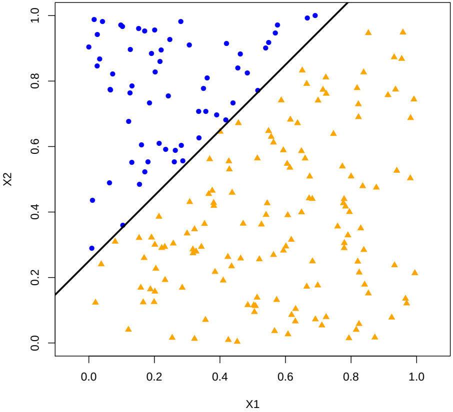


AdaBoost: Train vs. Test

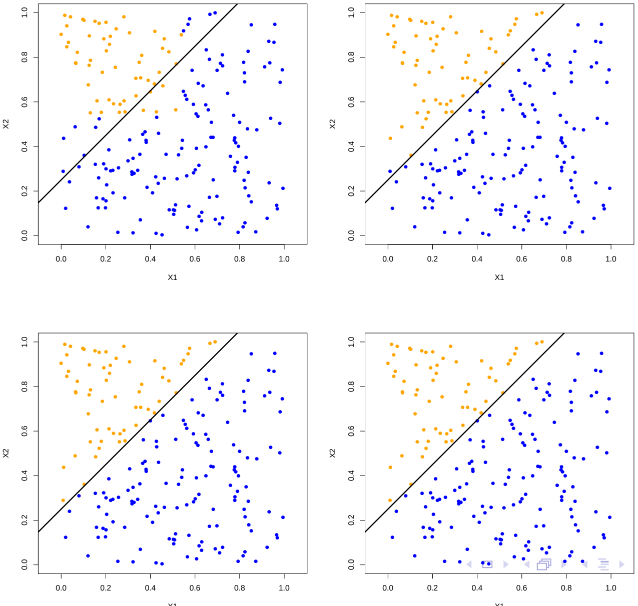


AdaBoost: Simulation

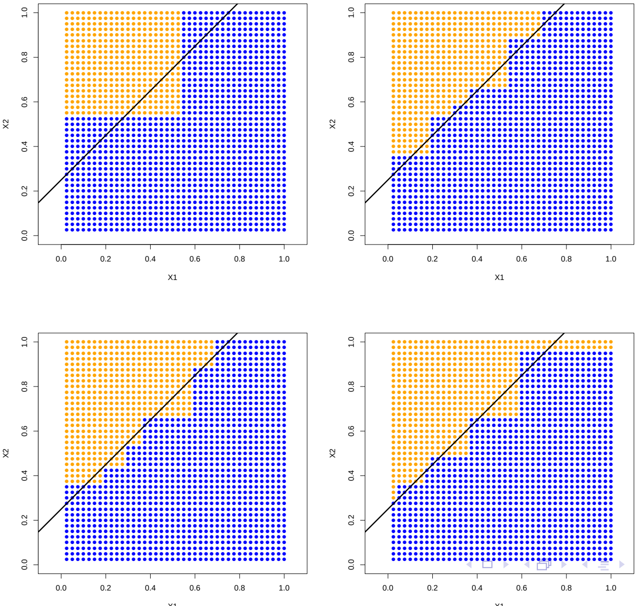
Let's change the simulation so that the class probabilities aren't the same



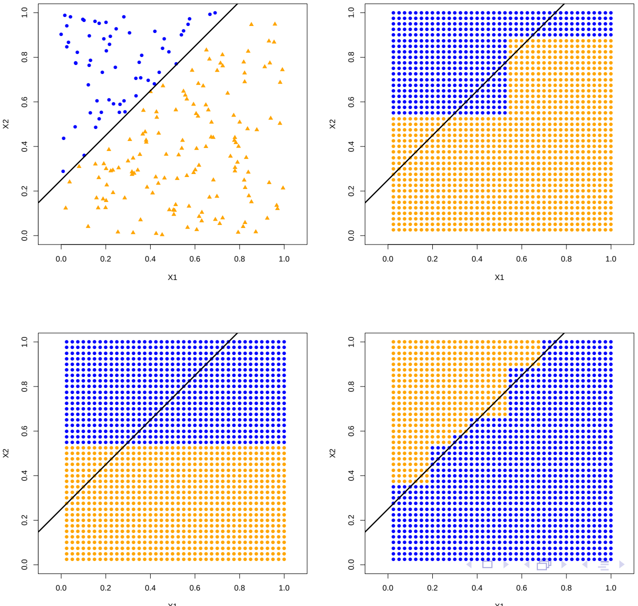
AdaBoost: Increasing B (train)



AdaBoost: Increasing B (test)



AdaBoost: Train vs. Test



Back to Algorithms

AdaBoost

This algorithm became known as 'discrete AdaBoost'

(This is due to the base classifier returning a discrete label)

This was adapted to real-valued predictions in Real AdaBoost

(In particular, probability estimates)

This terminology was introduced in Friedman's seminal paper on Functional Gradient Boosting (2001)

Real AdaBoost

1. Initialize $w_i \equiv 1/n$
2. For $b = 1, \ldots, B$
   2.1 Fit the classifier on $\mathcal{D}$, weighted by w_i and produce $p_b(X) = \hat{P}_w(Y = 1|X)$
   2.2 Set $h_b(X) \leftarrow \frac{1}{2} \log(p_b(X)/(1 - p_b(X)))$
   2.3 Set $w_i \leftarrow w_i \exp\{-Y_i h_b(X_i)\}$
3. Output: $g(X) = \mathrm{sgn}\left(\sum_{b=1}^{B} h_b(X)\right)$

This is referred to as Real AdaBoost and it used the class probability estimates to construct the contribution of the b^{th} classifier, instead of the estimated label

(The distinction between Discrete/Real AdaBoost is reminiscent of 1 vs. 1 and 1 vs. All multiclass classification)

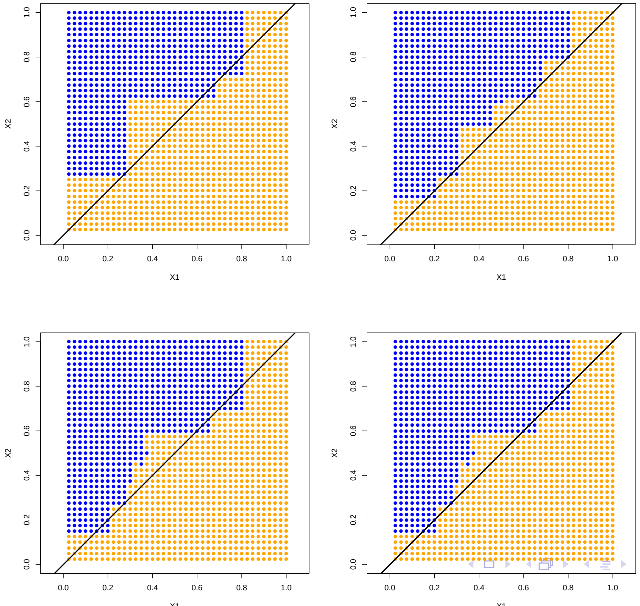
Real AdaBoost

1. Initialize $w_i \equiv 1/n$
2. For $b = 1, \ldots, B$
   2.1 Fit the classifier on $\mathcal{D}$, weighted by w_i and produce $p_b(X) = \hat{P}_w(Y = 1|X)$
   2.2 Set $h_b(X) \leftarrow \frac{1}{2}\log(p_b(X)/(1 - p_b(X)))$
   2.3 Set $w_i \leftarrow w_i \exp\{-Y_i h_b(X_i)\}$
3. Output: $g(X) = \mathrm{sgn}\left(\sum_{b=1}^{B} h_b(X)\right)$

This is referred to as Real AdaBoost and it used the class probability estimates to construct the contribution of the b^{th} classifier, instead of the estimated label

(The distinction between Discrete/Real AdaBoost is reminiscent of 1 vs. 1 and 1 vs. All multiclass classification)

Real AdaBoost: Increasing B (test)



AdaBoost intuition

Question: Why does this work?

One answer: Boosting fits an additive model

$$G_B(X) = \sum_{b=1}^{B} \beta_b \phi(X, \theta_b)$$

where

- β are weights
- ϕ is some base learner that depends on parameters θ

  (Example: Trees with all of its splits and terminal node values)

Overall: Both discrete and real AdaBoost can be interpreted as stage wise estimation procedures for fitting additive logistic regression models

(Discrete) AdaBoost interpretation

Forward stagewise additive modeling:

(Using a general likelihood ℓ)

1. $\beta_b, \theta_b = \text{argmin}_{\beta,\theta} \sum_{i=1}^{n} \ell(Y_i, G_{b-1}(X_i) + \beta\phi(X_i, \theta))$
2. Set $G_b(X) = G_{b-1}(X) + \beta_b \phi(X; \theta_b)$

AdaBoost implicitly does this by use of the exponential loss function

$$\ell(Y, G) = \exp\{-YG(X)\}$$

and basis functions $\phi(x, \theta) = g_b(X)$

AdaBoost intuition

Suppose we minimize exponential loss in a forward stagewise manner

Doing the forward selection for this loss, we get

$$(\beta_b, g_b) = \underset{\beta, g}{\operatorname{argmin}} \sum_{i=1}^{n} \exp\{-Y_i(G_{b-1}(X_i) + \beta g(X_i))\}$$

AdaBoost intuition

Rewriting:

$$
\begin{aligned}
(\beta_b, g_b) &= \underset{\beta,g}{\text{argmin}} \sum_{i=1}^{n} \exp\{-Y_i(G_{b-1}(X_i) + \beta g(X_i))\} \\
&= \underset{\beta,g}{\text{argmin}} \sum_{i=1}^{n} \exp\{-Y_i G_{b-1}(X_i)\} \exp\{-Y_i \beta g(X_i))\} \\
&= \underset{\beta,g}{\text{argmin}} \sum_{i=1}^{n} w_i \exp\{-Y_i \beta g(X_i)\}
\end{aligned}
$$

Where

- Define $w_i = \exp\{-Y_i G_{b-1}(X_i)\}$

  (This is independent of β, g)

- $\sum_{i=1}^{n} w_i \exp\{-Y_i \beta g_b(X_i))\}$ needs to be optimized

AdaBoost intuition

Note that

$$\sum_{i=1}^{n} w_i \exp\{-\beta Y_i g(X_i))\} = e^{-\beta} \sum_{i:Y_i = g(X_i)} w_i + e^{\beta} \sum_{i:Y_i \neq g(X_i)} w_i$$

$$= (e^{\beta} - e^{-\beta}) \sum_{i=1}^{n} w_i \mathbf{1}(Y_i \neq g(X_i)) +$$

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

As long as $(e^{\beta} - e^{-\beta}) \geq 0$, we can find

$$g_b = \underset{g}{\operatorname{argmin}} \sum_{i=1}^{n} w_i \mathbf{1}(Y_i \neq g(X_i))$$

(Note: If $(e^{\beta} - e^{-\beta}) < 0$, then $\beta < 0$. However, as $\beta_b = \log((1 - R_b)/R_b)$, this implies $R > 1/2$. Hence, we would flip the labels and get $R \leq 1/2$.)

Reminder: AdaBoost

1. Initialize $w_i \equiv 1/n$
2. For $b = 1, \ldots, B$
   2.1 Fit $g_b(x)$ on $\mathcal{D}$, weighted by w_i

   (This step is finding the next best version of the classifier, trained on weighted data and added to the previous classifiers)

   2.2 Compute

   $$R_b = \frac{\sum_{i=1}^n w_i \mathbf{1}(Y_i \neq g_b(X_i))}{\sum_{i=1}^n w_i}$$

   2.3 Find $\beta_b = \log((1 - R_b)/R_b)$

   2.4 Set $w_i \leftarrow w_i \exp\{\beta_b \mathbf{1}(Y_i \neq g_b(X_i))\}$
3. Output: $g(X) = \text{sgn}\left(\sum_{b=1}^B \beta_b g_b(X)\right)$

Reminder: AdaBoost

1. Initialize $w_i \equiv 1/n$
2. For $b = 1, \ldots, B$
    2.1 Fit $g_b(x)$ on $\mathcal{D}$, weighted by w_i

    (This step is finding the next best version of the classifier, trained on weighted data and added to the previous classifiers)

    2.2 Compute

    $$R_b = \frac{\sum_{i=1}^n w_i \mathbf{1}(Y_i \neq g_b(X_i))}{\sum_{i=1}^n w_i}$$

    2.3 Find $\beta_b = \log((1 - R_b)/R_b)$

    2.4 Set $w_i \leftarrow w_i \exp\{\beta_b \mathbf{1}(Y_i \neq g_b(X_i))\}$
3. Output: $g(X) = \mathrm{sgn}\left(\sum_{b=1}^B \beta_b g_b(X)\right)$

AdaBoost intuition

Goal: Minimize

$$\sum_{i=1}^{n} w_i \exp\{-\beta Y_i g_b(X_i))\}$$

(Here, we have fixed $g = g_b$)

We showed this can be written

$$\sum_{i=1}^{n} w_i \exp\{-\beta Y_i g_b(X_i))\} = (e^{\beta} - e^{-\beta}) R_b W + e^{-\beta} W \quad (W = \sum w_i)$$

Take derivative with respect to β

$$(e^{\beta} + e^{-\beta}) R_b W - e^{-\beta} W \overset{set}{=} 0 \overset{set}{=} e^{\beta} R_b + e^{-\beta}(R_b - 1)$$

Solve for β to find $\beta_b = 1/2 \log[(1 - R_b)/R_b]$

Reminder: AdaBoost

1. Initialize $w_i \equiv 1/n$
2. For $b = 1, \ldots, B$
    2.1 Fit $g_b(x)$ on $\mathcal{D}$, weighted by w_i

    (This step is finding the next best version of the classifier, trained on weighted data and added to the previous classifiers)

    2.2 Compute

    $$R_b = \frac{\sum_{i=1}^n w_i \mathbf{1}(Y_i \neq g_b(X_i))}{\sum_{i=1}^n w_i}$$

    2.3 Find $\beta_b = \log((1 - R_b)/R_b)$

    2.4 Set $w_i \leftarrow w_i \exp\{\beta_b \mathbf{1}(Y_i \neq g_b(X_i))\}$
3. Output: $g(x) = \operatorname{sgn}\left(\sum_{b=1}^B \beta_b g_b(x)\right)$

Reminder: AdaBoost

1. Initialize $w_i \equiv 1/n$
2. For $b = 1, \ldots, B$
    2.1 Fit $g_b(x)$ on $\mathcal{D}$, weighted by w_i

    (This step is finding the next best version of the classifier, trained on weighted data and added to the previous classifiers)

    2.2 Compute

    $$R_b = \frac{\sum_{i=1}^{n} w_i \mathbf{1}(Y_i \neq g_b(X_i))}{\sum_{i=1}^{n} w_i}$$

    2.3 Find $\beta_b = \log((1 - R_b)/R_b)$

    2.4 Set $w_i \leftarrow w_i \exp\{\beta_b \mathbf{1}(Y_i \neq g_b(X_i))\}$
3. Output: $g(x) = \mathrm{sgn}\left(\sum_{b=1}^{B} \beta_b g_b(x)\right)$

AdaBoost intuition

The approximation is updated

$$G_b(X) = G_{b-1}(X) + \beta_b g_b(X)$$

This causes the weights

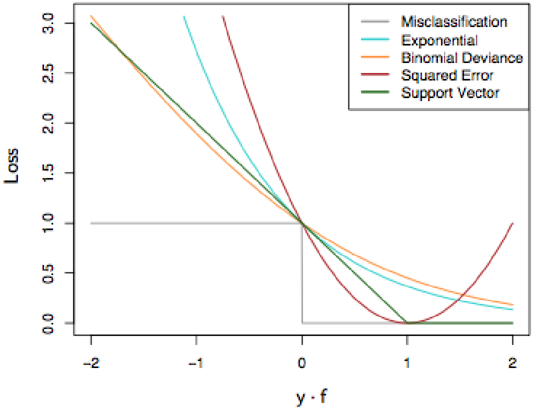
$$w_i^{(b+1)} = \exp\{-Y_i G_b(X_i)\} = w_i^{(b)} \exp\{-\beta_b Y_i g_b(X_i)\}$$

Using $-Y_i g_b(X_i) = 2\mathbf{1}(Y_i \neq g_b(X_i)) - 1$, this becomes

$$w_i^{(b+1)} \propto w_i^{(b)} \exp\{\beta_b \mathbf{1}(Y_i \neq g_b(X_i))\}$$

where $\beta_b \leftarrow 2\beta_b$, giving the last step of the algorithm

Other loss functions



(Hastie et al (2009))

AdaBoost: The controversy

Claim: Boosting is another version of bagging

The early versions of Boosting involved (weighted) resampling

Therefore, it was initially speculated that a connection with bagging explained its performance

However, boosting continues to work well when

- The algorithm is trained on weighted data rather than on sampling with weights

  (This removes the randomization component that is essential to bagging)

- Weak learners are used that have high bias and low variance

  (This is the opposite of what is prescribed for bagging)

AdaBoost: The controversy

Claim: Boosting fits an adaptive additive model which explains its effectiveness

The previous results appeared in Friedman et al. (2000) and claimed to have 'solved' the mystery of boosting

A crucial property of boosting is that is essentially never over fits

However, the additive model view really should translate into intuition of 'over fitting is a major concern,' as it is with additive models

AdaBoost: The controversy

As adaBoost fits an additive model in the base classifier, it cannot have higher order interactions than the base classifier

For instance, a stump would provide a purely additive fit
(It only splits on one variable. In general, the complexity of a tree can be interpreted as the number of included interactions)

It stands to reason, then, if the Bayes' rule is additive in a similar fashion, stumps should perform well in Boosting

AdaBoost: The controversy

A recent paper investigating this property did substantial simulations using underlying purely additive models
(Mease, Wyner (2008))

Here is an example figure from their paper:

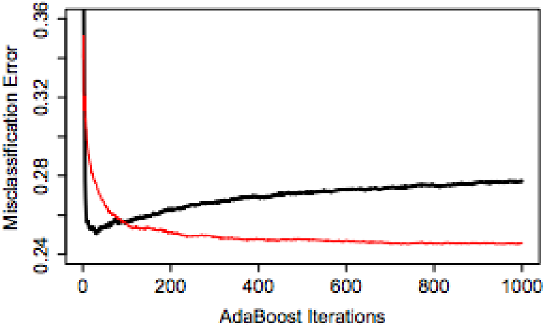


Figure: Black, bold line: Stumps. Red, thin line: 8-node trees

AdaBoost: The controversy continues

Ultimately, interpretations are just modes of human comprehension

The value of the insight is whether it provides fruitful thought about the idea

From this perspective, AdaBoost fits an additive model.

However, many of the other connections are still of debatable value

(For example, LogitBoost)

Next lectures

Discuss two current, popular algorithms and their R implementations

- GBM
- XGBoost