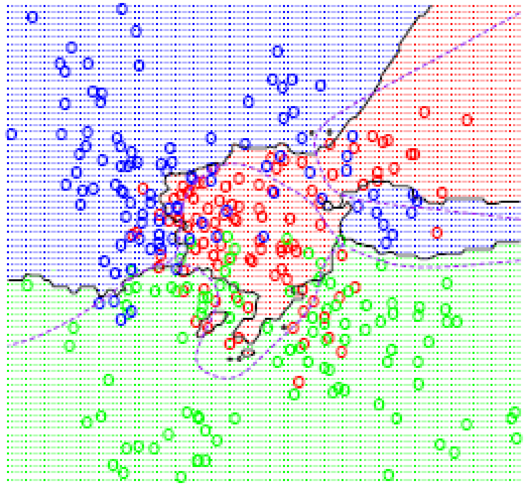
Machine Learning

Lecture 20: Boosting

The lectures are mainly offered on white board accompanied by some slides.



Hesam Montazeri
Department of Bioinformatics, IBB, University of Tehran

Azar 19, 1398

University of TEHRAN

Boosting

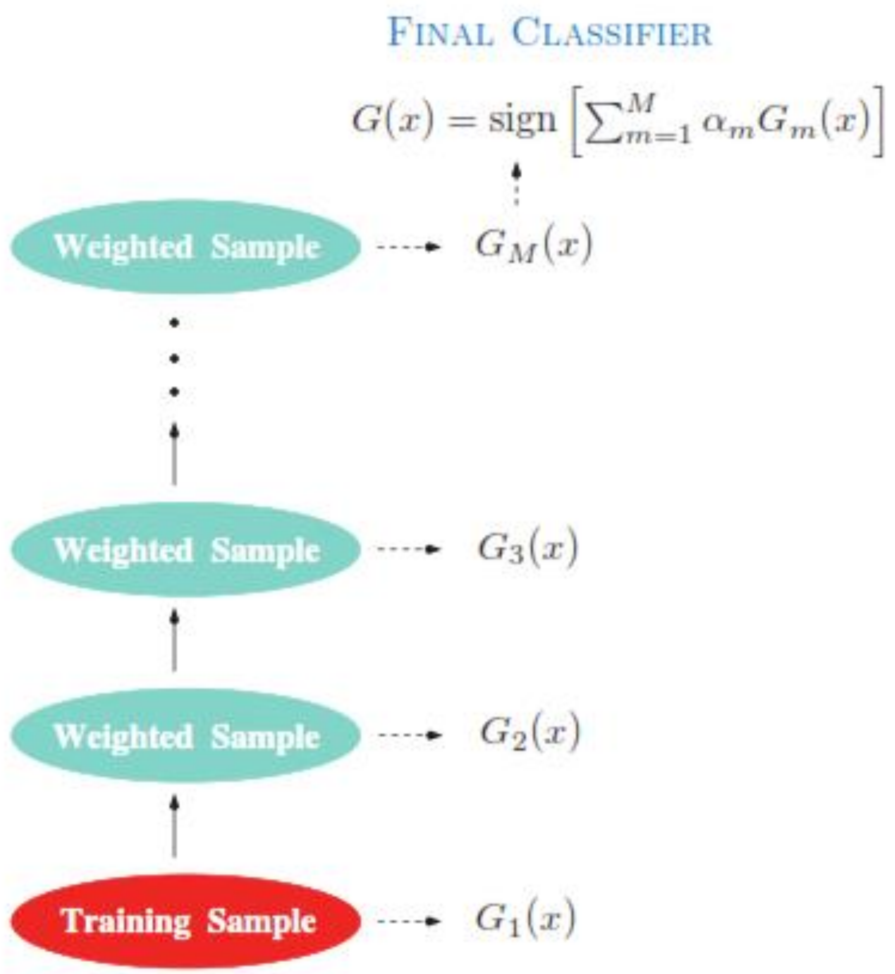
- Weak learner: a learner that slightly outperforms a random classifier
  - Weak learners have large bias

- A famous question: can a set of weak learners be combined to build a strong learner? (by Kearns 1988)
  - Yes! (Schapire 1990)

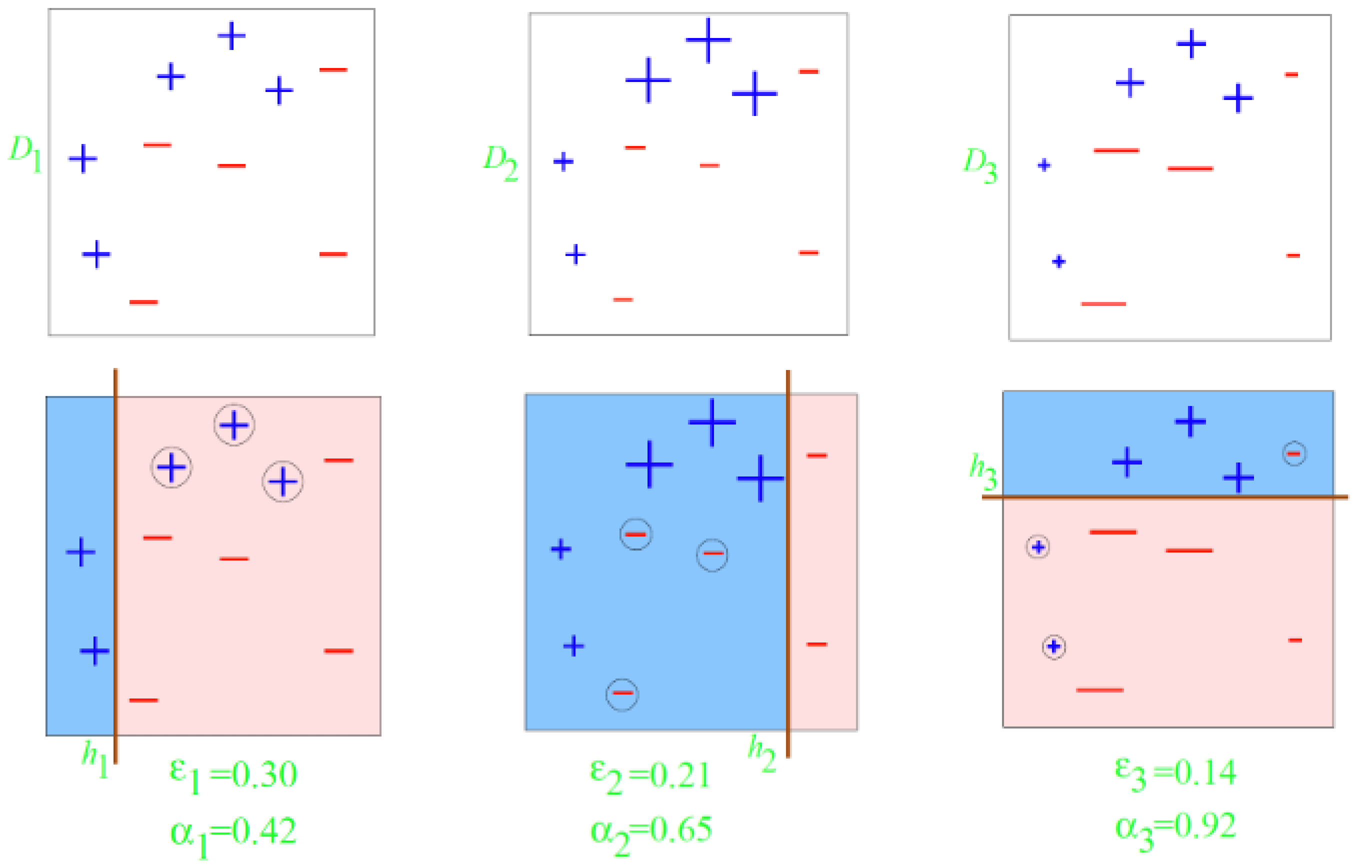
- Boosting low-depth trees are among strong ML algorithms.

Schematic of AdaBoost

- Classifiers are sequentially trained on weighted versions of the training dataset

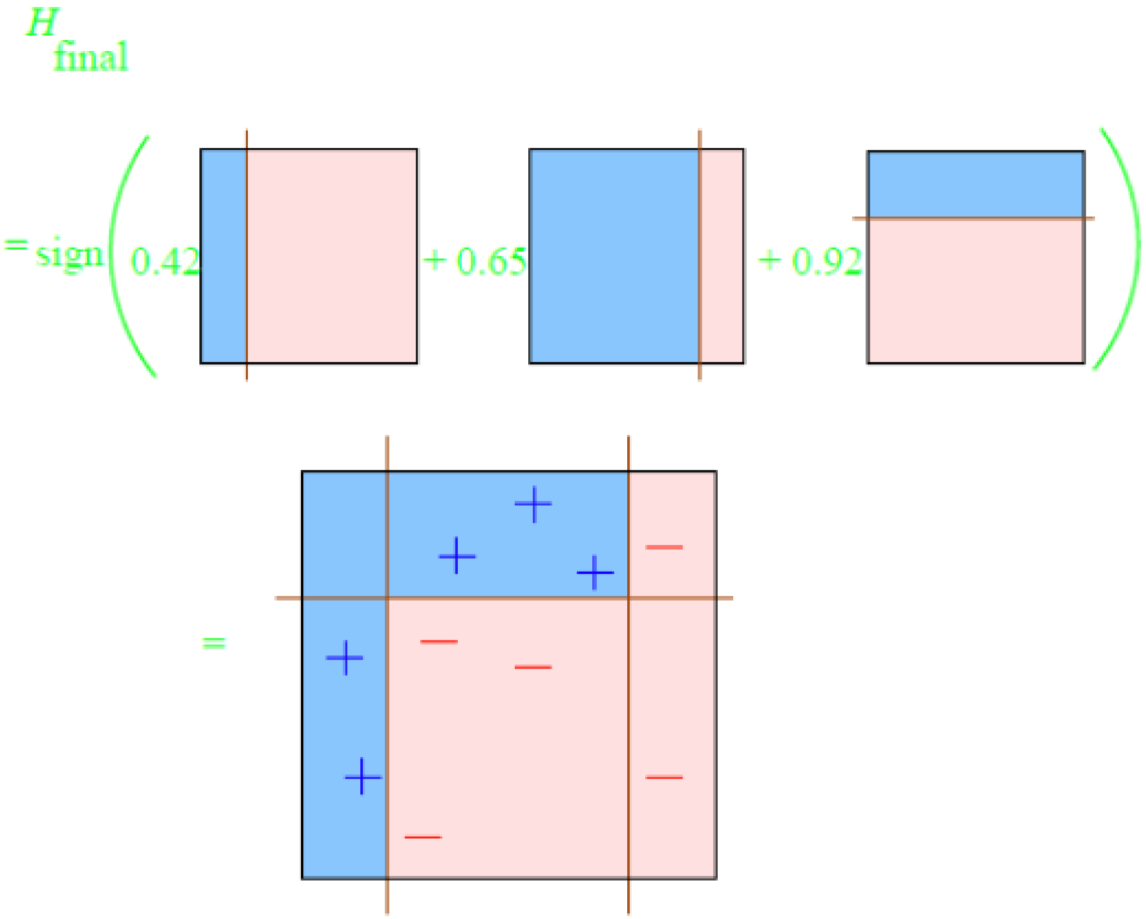


Example



https://www.cs.cmu.edu/~aarti/Class/10701/

Example 2



https://www.cs.cmu.edu/~aarti/Class/10701/

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \ldots, N$.
3. Output $G(x) = \mathrm{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

University of TEHRAN

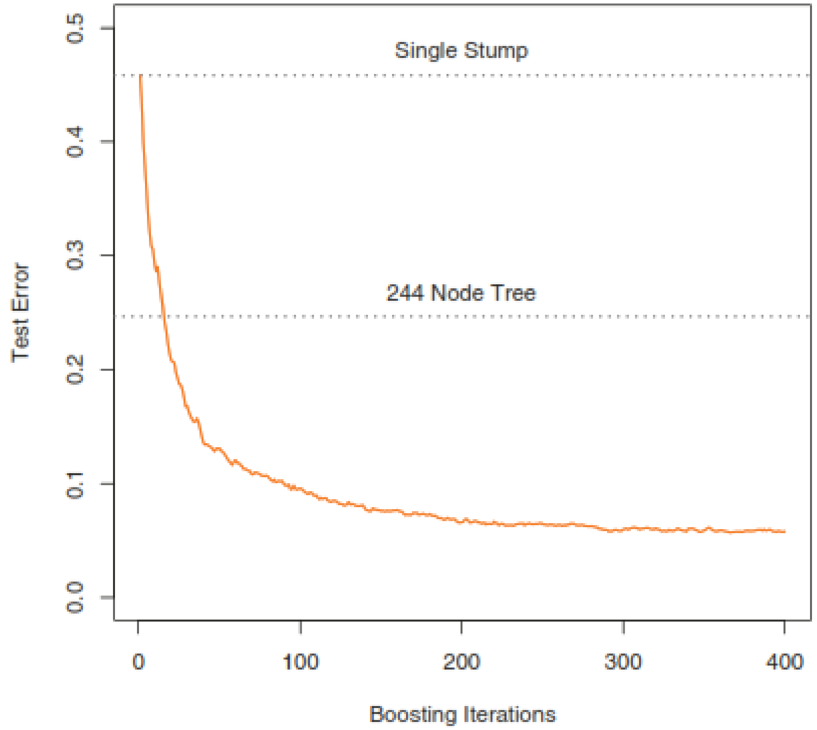
Simulated example

- The deterministic target Y is defined by

$$Y = \begin{cases} 1 & \text{if } \sum_{j=1}^{10} X_j^2 > \chi_{10}^2(0.5), \\ -1 & \text{otherwise.} \end{cases}$$

- 2000 training cases, approximate 1000 observations in each class

- Weak classifier is a stump (a two terminal-node classification tree).



Boosting fits an additive model

- Boosting is a way fitting an additive expansion in a set of elementary "basis" functions.
- Here the basis functions are the individual classifiers $G_m(x) \in \{-1, 1\}$

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

Expansion coefficients

Simple functions on x characterized by parameters γ_m

For trees, γ_m parameterizes the split variables and points, and the predictions at the terminal nodes

Boosting fits an additive model-2

- Traditionally parameters are jointly fit by minimizing a loss function

$$\min_{\{\beta_m,\gamma_m\}_1^M} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m b(x_i;\gamma_m)\right)$$

- Computationally expensive

- A simple alternative is to solve the subproblem of fitting just a single basis function

$$\min_{\beta,\gamma} \sum_{i=1}^{N} L\left(y_i, \beta b(x_i;\gamma)\right).$$

Boosting fits an additive model-3

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
   (a) Compute
   $$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).$$
   (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Example- squared-error loss

- For squared-error loss

$$L(y, f(x)) = (y - f(x))^2,$$

- We obtain

$$\begin{aligned} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)) &= (y_i - f_{m-1}(x_i) - \beta b(x_i; \gamma))^2 \\ &= (r_{im} - \beta b(x_i; \gamma))^2, \end{aligned}$$

- $r_{im} = y_i - f_{m-1}$ is simply the residual of the current model on the *i*th observation.

squared-error loss is not a good choice for classification!

AdaBoost-revisited

- AdaBoost.M1 is equivalent to forward stagewise additive modeling using the **exponential loss**

$$L(y, f(x)) = \exp(-y\, f(x)).$$

- For AdaBoost the basis functions are the classifiers $G_m(x) \in \{-1, 1\}$

$$(\beta_m, G_m) = \arg\min_{\beta, G} \sum_{i=1}^{N} \exp[-y_i(f_{m-1}(x_i) + \beta\, G(x_i))]$$

- Can be expressed as

$$(\beta_m, G_m) = \arg\min_{\beta, G} \sum_{i=1}^{N} w_i^{(m)} \exp(-\beta\, y_i\, G(x_i))$$

with $w_i^{(m)} = \exp(-y_i\, f_{m-1}(x_i)).$

w_i^m depends neither on β nor $G(x)$, hence can be regarded as a weight to each observation

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

$$(\beta_m, G_m) = \arg\min_{\beta, G} \sum_{i=1}^{N} w_i^{(m)} \exp(-\beta\, y_i\, G(x_i))$$

For any value $\beta > 0$, the solution is

$$G_m = \arg\min_{G} \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i)),$$

It is easy to see

$$e^{-\beta} \cdot \sum_{y_i = G(x_i)} w_i^{(m)} + e^{\beta} \cdot \sum_{y_i \neq G(x_i)} w_i^{(m)},$$

Can be written as

$$(e^{\beta} - e^{-\beta}) \cdot \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i)) + e^{-\beta} \cdot \sum_{i=1}^{N} w_i^{(m)}.$$

Solving for β, we obtain

$$\beta_m = \frac{1}{2} \log \frac{1 - \mathrm{err}_m}{\mathrm{err}_m},$$

where

$$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i^{(m)}}.$$

HW: complete the derivations. Explain each step in detail.

The approximation is then

$$f_m(x) = f_{m-1}(x) + \beta_m G_m(x),$$

Why exponential loss?

- The AdaBoost.M1 was originally motivated from a very different perspective.
- Its equivalence to forward stagewise additive modeling based on exponential loss was discovered five years after its inception.
- It has been shown

$$f^*(x) = \arg\min_{f(x)} \mathrm{E}_{Y|x}(e^{-Yf(x)}) = \frac{1}{2}\log\frac{\Pr(Y=1|x)}{\Pr(Y=-1|x)},$$

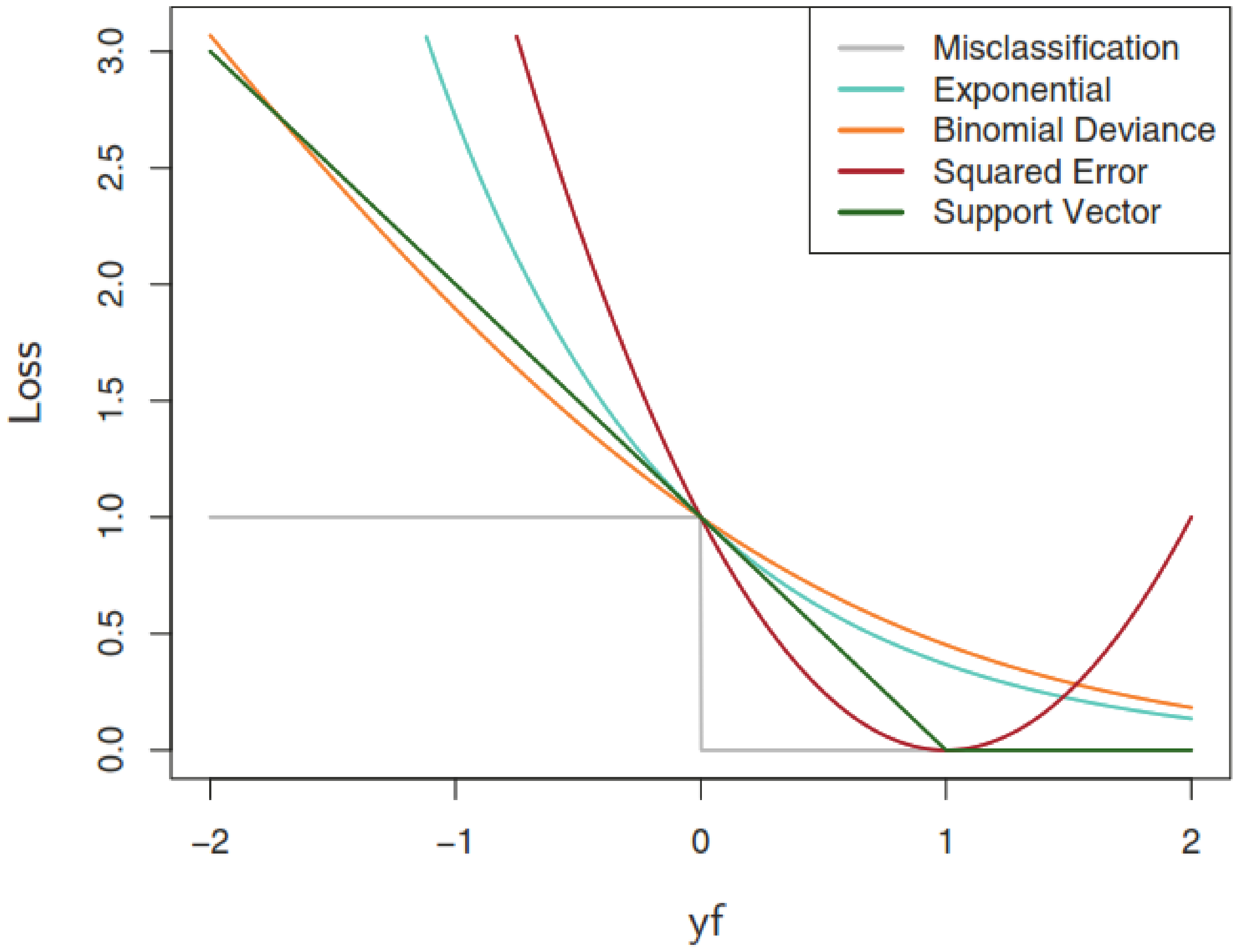
Half the log-odds of $P(Y = 1 \mid x)$

- This justifies using the sign function in the algorithm.

- Exponential loss is a robust loss function!

University of TEHRAN

Loss functions for two-class classification

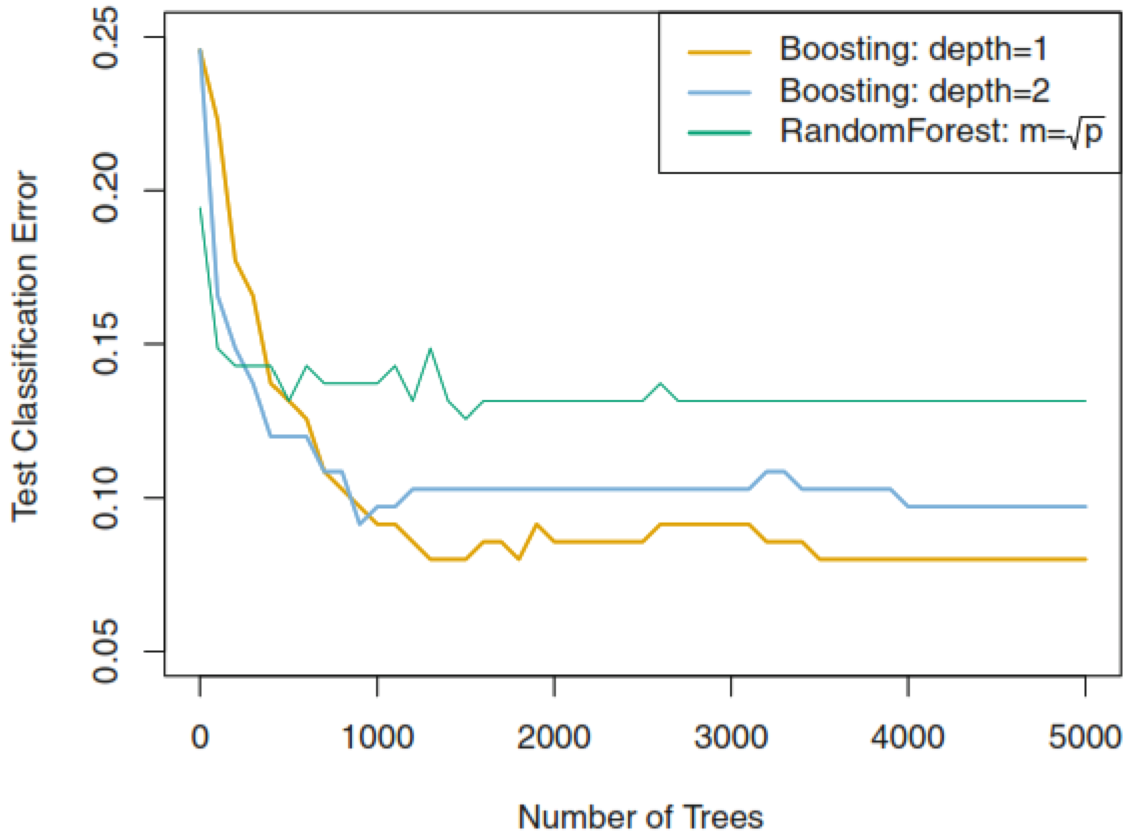


Boosting for regression trees [ISL]

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$
   (c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$
3. Output the boosted model,
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

University of TEHRAN

15-class gene expression data set



The test error for a single tree is 24%

References and Acknowledgement

- References
  - The Elements of Statistical Learning by Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie
  - An Introduction to Statistical Learning, with applications in R, 2013

- Acknowledgement
  - Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani. Few slides are also adjusted from theirs.

University of TEHRAN