INTRODUCTION

To Statistical Learning

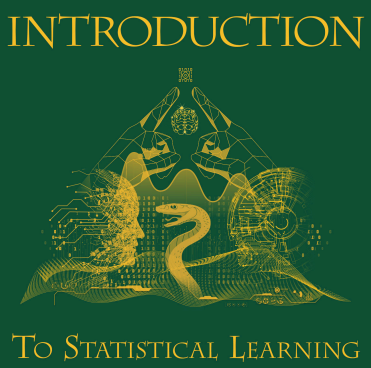
Introduction to Statistical Learning

Boosting - Class 10

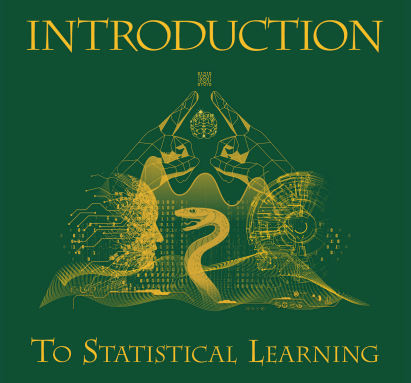
Giora Simchoni

gsimchoni@gmail.com and add #intro2sl in subject

Stat. and OR Department, TAU



AdaBoost



Instead of averaging

- Suppose $y \in \{-1, 1\}$
- Recall the final Random Forests model: $\hat{f}(x_0) = \text{sign}\left[\frac{1}{M}\sum_{m=1}^{M} T_m(x_0)\right]$
- Why not $\hat{f}(x_0) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m T_m(x_0)\right]$?
  - If M is the number of all possible trees T_m, then finding weights α_m is intractable!
- Boosting finds α_m sequentially, *adaptively*:
  - Take a weak classifier $G(x)$ with accuracy slightly better than random
    - for any dist. P_T of the training data: $Err_{P_T} = \mathbb{E}_{P_T}\left(\mathbb{I}\left[y_i \neq G(x_i)\right]\right) \leq \frac{1}{2} - \gamma$
  - Apply it sequentially over modified (weighted) versions of the training data
  - Each time selecting the best α_m to get: $\hat{f}(x_0) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x_0)\right]$

AdaBoost

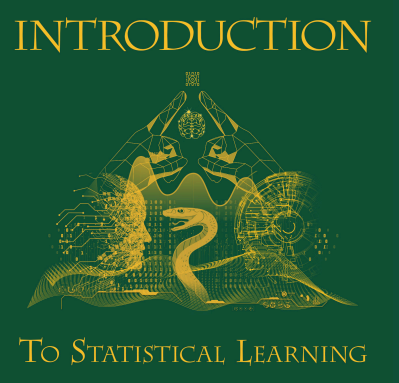
1. Initialize observations weights $w_i = 1/n$ for $i = 1, \ldots, n$
2. For $m = 1$ to M:
    a. Fit classifier $G_m(x)$ to the training data using **weights** w_i
    b. Compute weighted classification error:

$$err_m = \frac{\sum_{i=1}^{n} w_i \mathbb{I}\left[y_i \neq G_m(x_i)\right]}{\sum_{i=1}^{n} w_i}$$

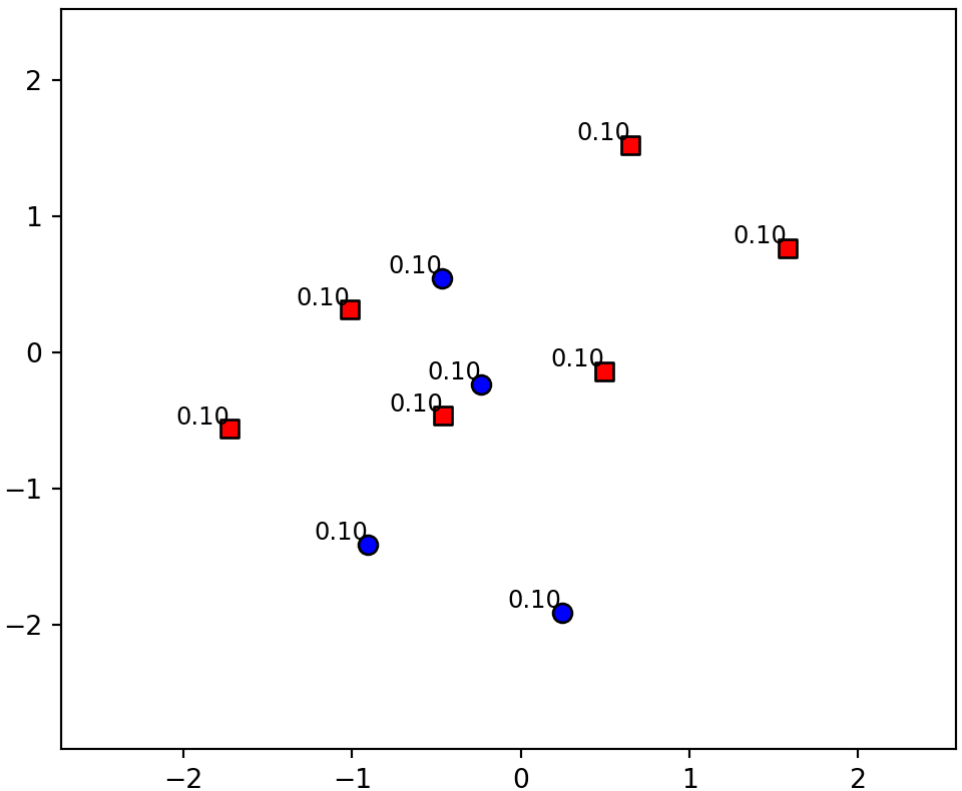
    c. Compute coefficient $\alpha_m = \ln\left[\frac{1-err_m}{err_m}\right]$
    d. Update weights: $w_i \leftarrow w_i \cdot \exp\left[\alpha_m \cdot \mathbb{I}\left[y_i \neq G_m(x_i)\right]\right]$
3. Output: $\hat{f}(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$

What does it mean training a classification tree with weighted data?

AdaBoost Example

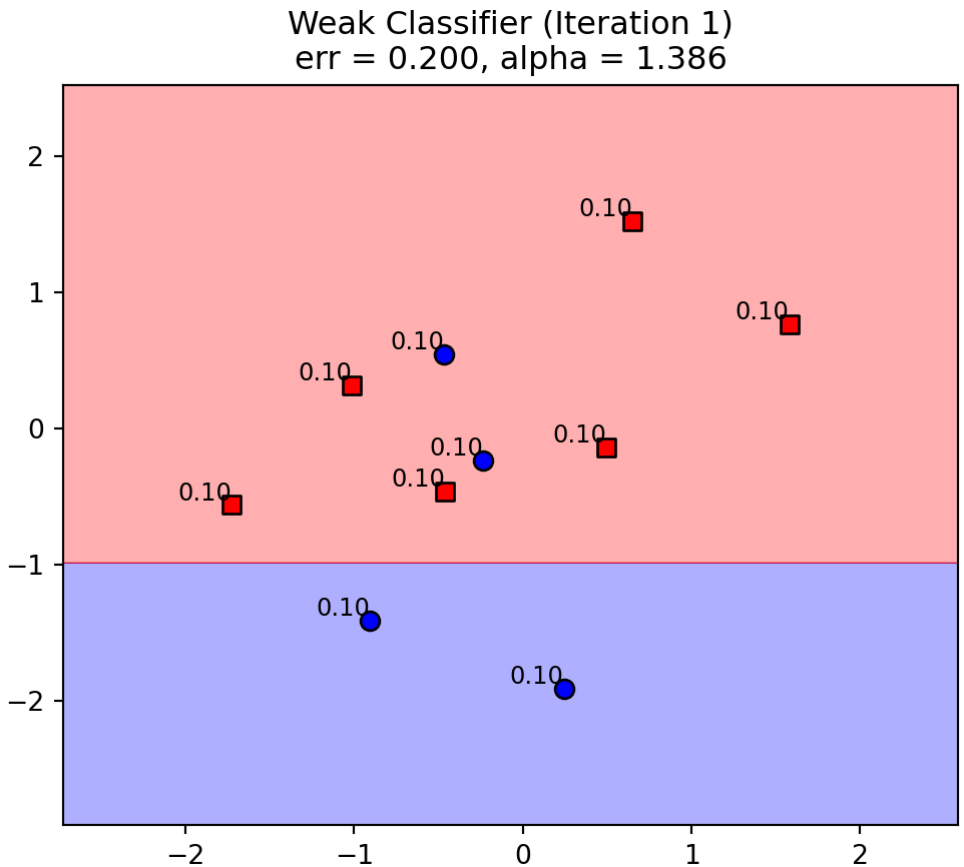


AdaBoost with Tree Stumps

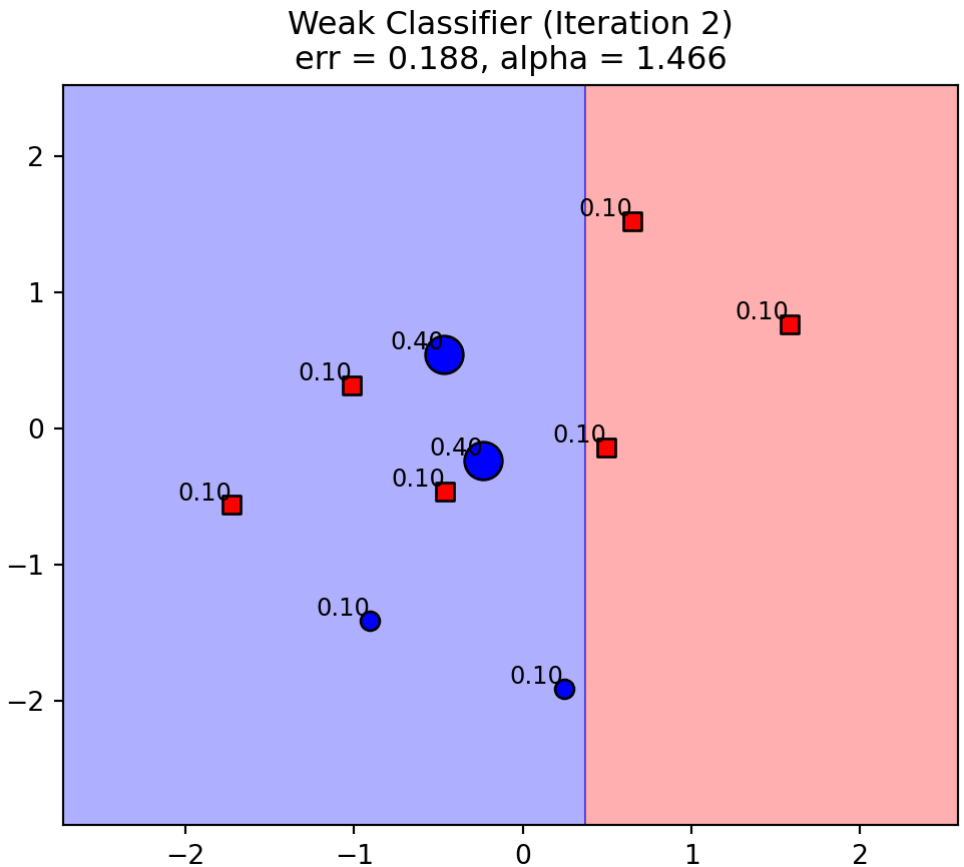


Can a tree stump get to $\overline{err} = 0$? Can a deep decision tree?

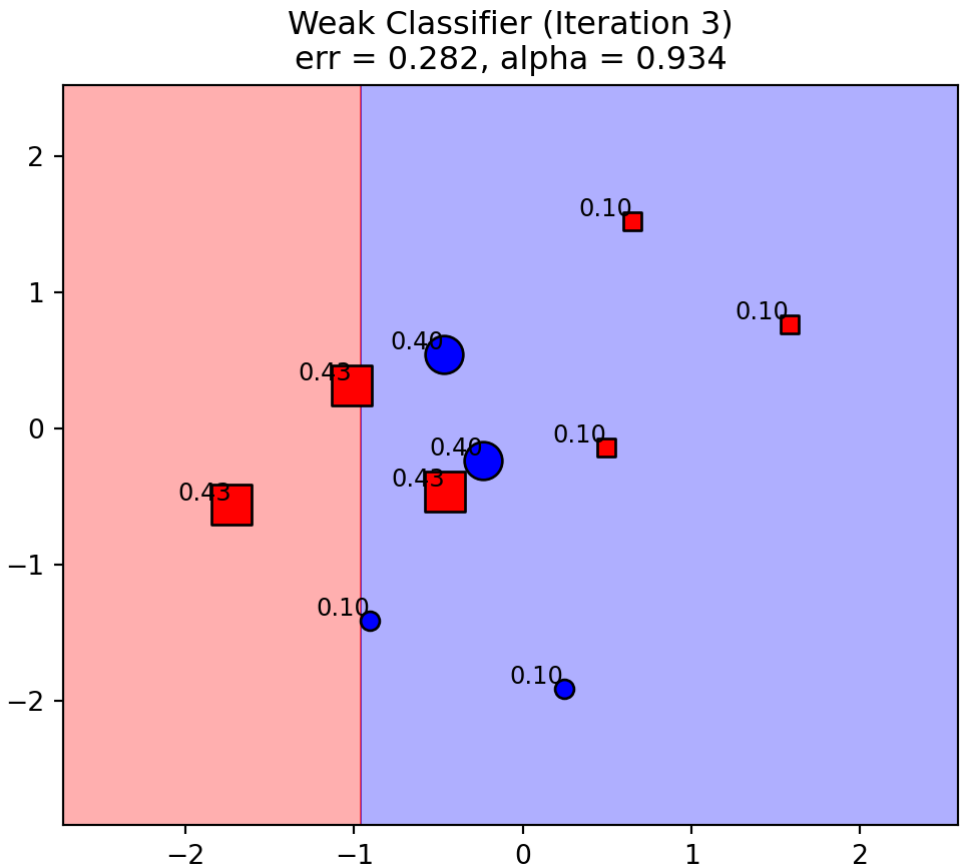
AdaBoost: iteration 1



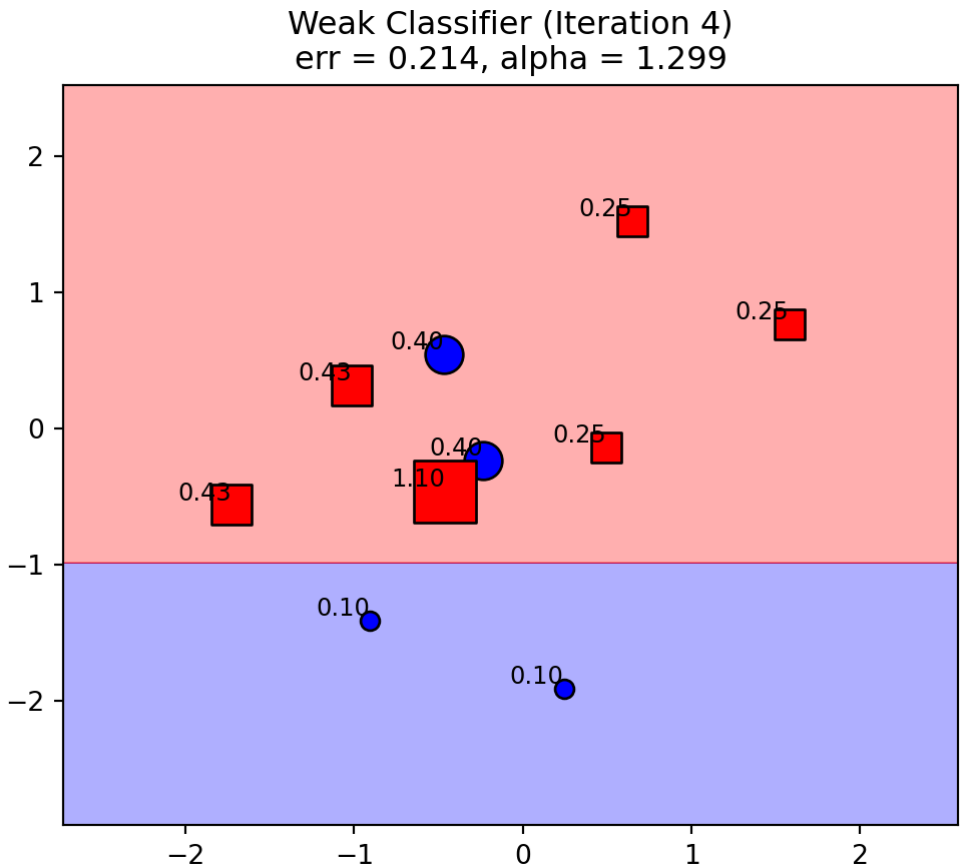
AdaBoost: iteration 2



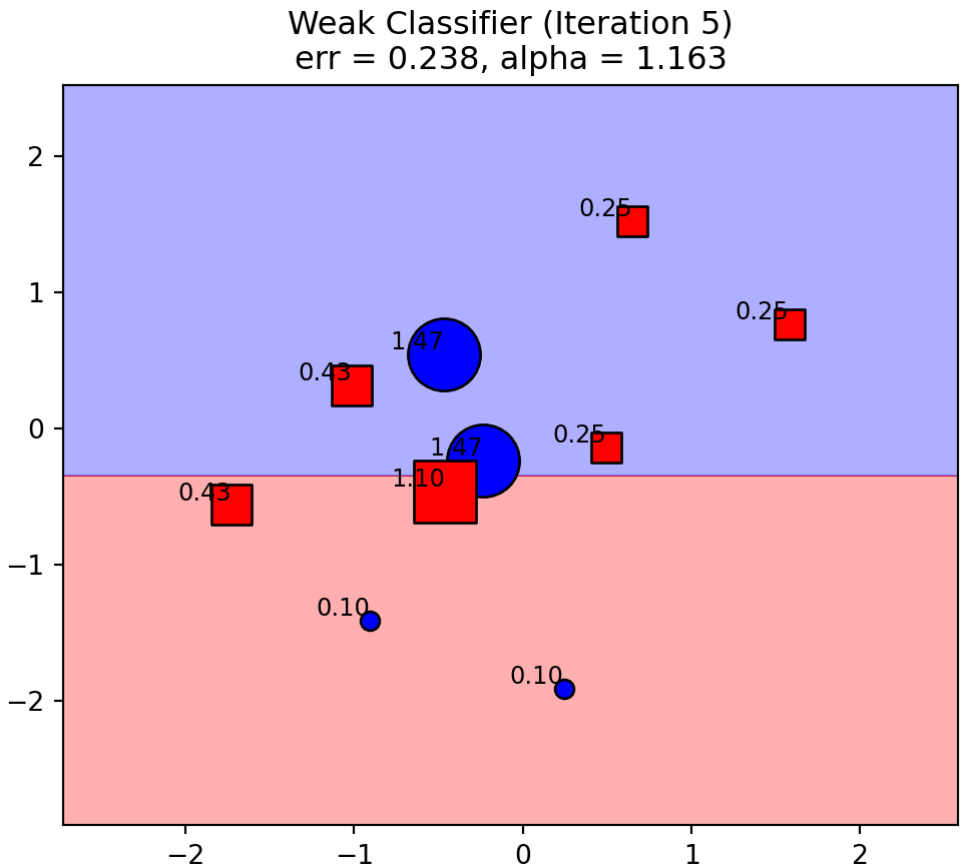
AdaBoost: iteration 3



AdaBoost: iteration 4



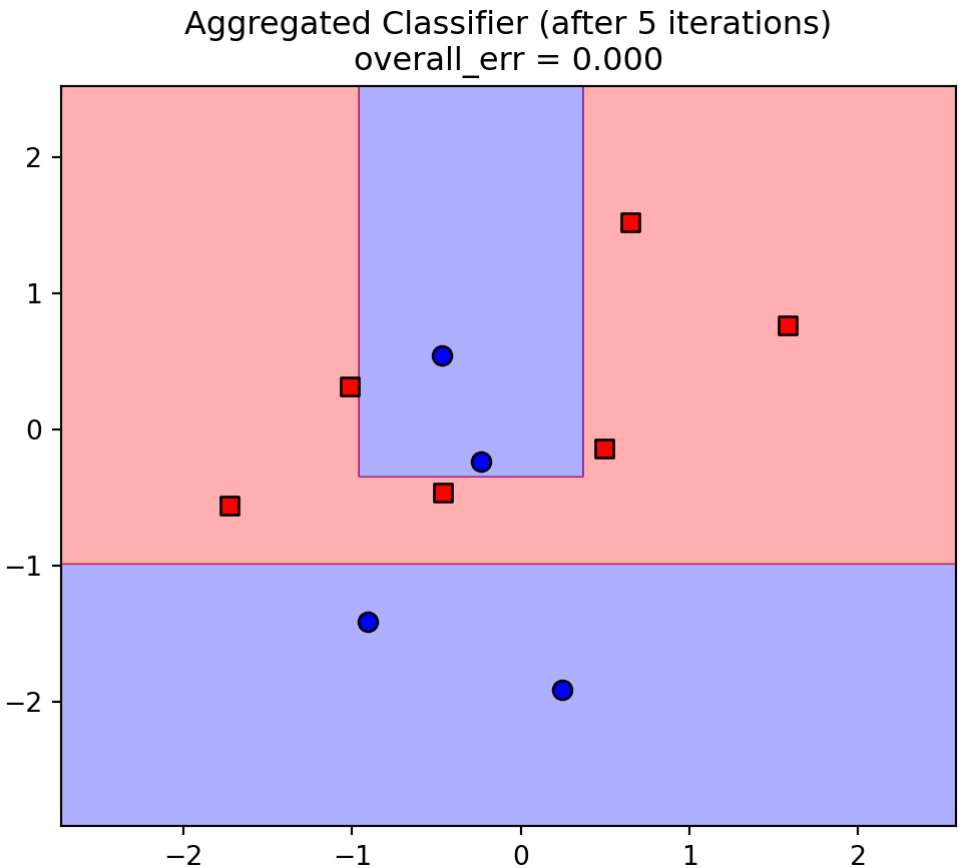
AdaBoost: iteration 5



AdaBoost: aggregate

$$\hat{f}(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right] =$$

$$= \text{sign}\left[1.386 G_1(x) + 1.466 G_2(x) + 0.934 G_3(x) + 1.299 G_4(x) + 1.126 G_5(x)\right]$$



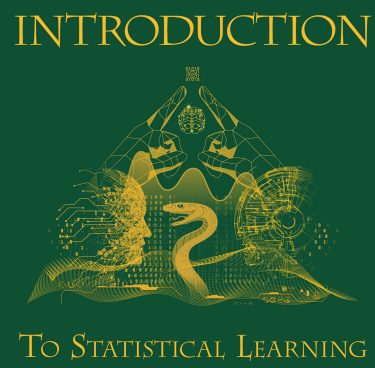
AdaBoost Guarantee

Theorem:

$$\overline{err} = \sum_{i=1}^{n} \mathbb{I}\left[y_i \neq \hat{f}(x_i)\right] \leq \left(1 - 4\gamma^2\right)^{M/2} \leq e^{-2M\gamma^2}$$

- This means for a large enough M we can get $\overline{err}$ as low as we want (on training data)!
- Specifically, to make the upper bound $e^{-2M\gamma^2} < \frac{1}{n}$ (i.e. $\overline{err} = 0$) $\Rightarrow M > \frac{\log(n)}{2\gamma^2}$
- A weak classifier which can only guarantee up to 40% error at each iteration ($\gamma = 0.1$):
  - after $M = 100$ iterations will be boosted to give $\overline{err} = (1 - 4 \cdot 0.1^2)^{(100/2)} = 0.96^{50} \approx 0.13$ error
  - if $n = 1000$, need $M > \log(1000)/2 \cdot 0.1^2 \approx 345$ iterations to get $\overline{err} = 0$
- Really nice proof in Freund & Schapire (1997)

AdaBoost as Additive Stagewise Modeling



Questions from AdaBoost

- Does it optimize a specific loss? Can we plug-in a different loss?
- What about regression?
- What about K-class classification?
- Is there a probabilistic justification? A likelihood-based approach?

AdaBoost sure sounds familiar

Recall: **Forward stagewise regression**

0. Standardize all features, input some $\tau_{thresh} \in (0, 1)$ and $\varepsilon > 0$ step size
1. Residual $\mathbf{r} = \mathbf{y} - \bar{y}$, $\beta_1, \ldots, \beta_p = 0$
2. Find the predictor $\mathbf{x}_j$ most correlated with $\mathbf{r}$, and let $\tau = Corr(\mathbf{r}, \mathbf{x}_j)$
3. While $|\tau| > \tau_{thresh}$:
    i. Update $\beta_j \leftarrow \beta_j + \delta_j$, where $\delta_j = \varepsilon \cdot \text{sign}(\tau)$
    ii. Update $\mathbf{r} \leftarrow \mathbf{r} - \delta_j \mathbf{x}_j$
    iii. Find the predictor $\mathbf{x}_j$ most correlated with $\mathbf{r}$, and let $\tau = Corr(\mathbf{r}, \mathbf{x}_j)$

A general forward stagewise additive model (FSAM)

1. Initialize $f_0(x) = 0$
2. For $m = 1$ to M:
    a. Compute:

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{n} L\left(y_i, f_{m-1}(x_i) + \beta b(x_i, \gamma)\right)$$

    b. Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$

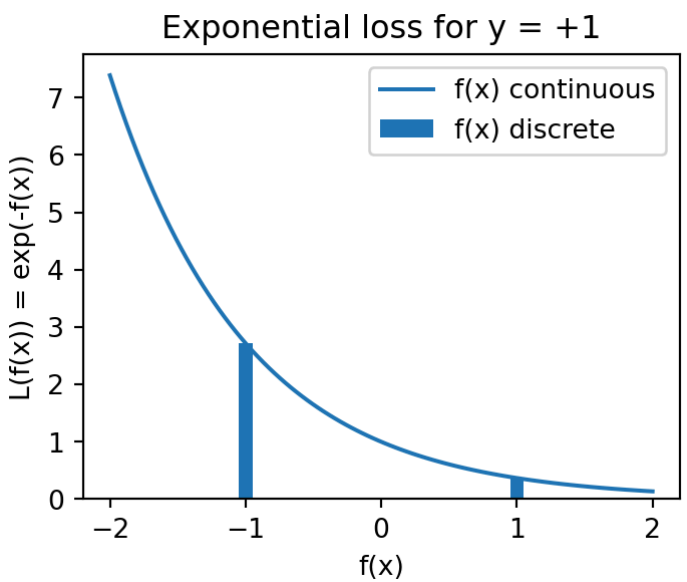
3. Output: $\hat{f}(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m)$

If L is the squared loss, and finding γ_m is finding the j-th predictor $\mathbf{x}_j$ most correlated with the residual

$\Rightarrow$ this is forward stagewise regression!

Claim: AdaBoost is also a FSAM

- If L is the exponential loss: $L(y, f(x)) = \exp(-yf(x))$
- And, $b(x; \gamma_m) = Gm(x)$
- We get AdaBoost!



Fast forward: plug-in *other* loss functions!

Showing AdaBoost is FSAM

- Recall the goal in FSAM iteration m:

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{n} L\left(y_i, f_{m-1}(x_i) + \beta b(x_i, \gamma_)\right)$$

- For exponential loss:

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

$$= \arg\min_{\beta,G} \sum_{i=1}^{n} \exp\left[-y_i f_{m-1}(x_i)\right] \exp\left[\beta G(x_i)\right]$$

$$= \arg\min_{\beta,G} \sum_{i=1}^{n} w_i^{(m)} \exp\left[-\beta y_i G(x_i)\right]$$

Getting G_m

- Assuming $\beta > 0$, separate the criterion to "correct" + "incorrect" sums:

$$\sum_{i=1}^{n} w_i^{(m)} \exp\left[-\beta y_i G(x_i)\right] =$$
$$= e^{-\beta} \sum_{i=1}^{n} w_i^{(m)} \mathbb{I}\left[y_i = G(x_i)\right] + e^{\beta} \sum_{i=1}^{n} w_i^{(m)} \mathbb{I}\left[y_i \neq G(x_i)\right]$$

- Can also write as:

$$= (e^{\beta} - e^{-\beta}) \sum_{i=1}^{n} w_i^{(m)} \mathbb{I}\left[y_i \neq G(x_i)\right] + e^{-\beta} \sum_{i=1}^{n} w_i^{(m)}$$

- AdaBoost stage 2a: G_m minimizing the criterion is minimizing weighted error rate:

$$G_m = \arg\min_{G} \sum_{i=1}^{n} w_i^{(m)} \mathbb{I}\left[y_i \neq G(x_i)\right]$$

Getting α_m

$$\text{criterion} = (e^{\beta} - e^{-\beta}) \sum_{i=1}^{n} w_i^{(m)} \mathbb{I}\left[y_i \neq G(x_i)\right] + e^{-\beta} \sum_{i=1}^{n} w_i^{(m)}$$

$$\frac{\partial \text{criterion}}{\partial \beta} = (e^{\beta} + e^{-\beta}) \sum_{i=1}^{n} w_i^{(m)} \mathbb{I}\left[y_i \neq G(x_i)\right] - e^{-\beta} \sum_{i=1}^{n} w_i^{(m)} \rightarrow = 0$$

- Multiply by e^{β}:

$$(1 + e^{2\beta}) = \frac{\sum_{i=1}^{n} w_i^{(m)}}{\sum_{i=1}^{n} w_i^{(m)} \mathbb{I}[y_i \neq G(x_i)]} = \frac{1}{err_m}$$

- Take ln, we get AdaBoost stages 2b and 2c:

$$\beta_m = \frac{1}{2} \ln \left[\frac{1 - err_m}{err_m}\right] = \frac{1}{2} \alpha_m$$

Getting w_i

- FSAM says:

$$f_m(x) = f_{m-1}(x) + \beta_m b(x;\gamma_m) = f_{m-1}(x) + \tfrac{1}{2}\alpha_m G_m(x)$$

- Which means next iteration weights are:

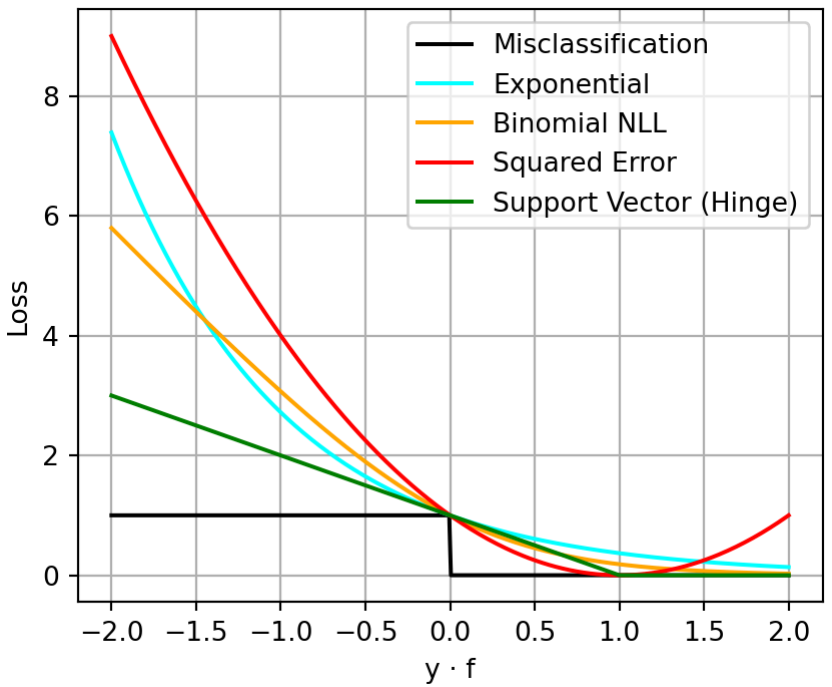
$$w_i^{(m+1)} = \exp\left[-y_i f_{m+1}(x_i)\right] = \exp\left[-y_i(f_m(x) + \tfrac{1}{2}\alpha_m G_m(x))\right] =$$
$$= w_i^{(m)} \cdot \exp\left[-\tfrac{1}{2}\alpha_m y_i G_m(x)\right]$$

- Given that $-y_i G_m(x_i) = 2 \cdot \mathbb{I}\left[y_i \neq G_m(x_i)\right] - 1$, we can write:

$$w_i^{(m+1)} = w_i^{(m)} \cdot \exp\left[\alpha_m \cdot \mathbb{I}\left[y_i \neq G_m(x_i)\right]\right] \cdot Const$$

- Equivalent to AdaBoost stage 2d.

Classification losses



The $f(x)$ minimizing exponential loss is also minimizing Binomial negative log-likelihood!

- Misclassification (0/1 loss):

$$L_{\text{misclass}}(y, f) = \begin{cases} 0 & \text{if } y \cdot f(x) > 0 \\ 1 & \text{otherwise} \end{cases}$$

- Exponential loss:

$$L_{\text{exp}}(y, f) = \exp(-y \cdot f(x))$$

- Binomial negative log-likelihood (cross-entropy):

$$L_{\text{bin}}(y, f) = \log(1 + \exp(-2 \cdot y \cdot f(x))) / \log(2)$$

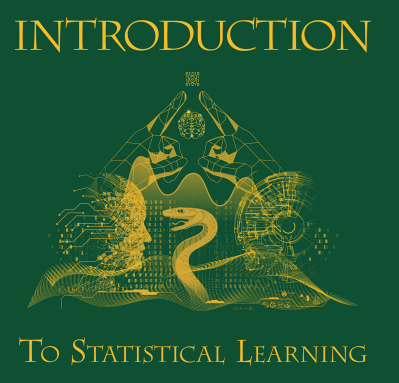
- Squared error loss:

$$L_{\text{sq}}(y, f) = (1 - y \cdot f(x))^2$$

- Support vector (hinge Loss):

$$L_{\text{hinge}}(y, f) = \max(0, 1 - y \cdot f(x))$$

Boosting for Regression



Recap

What the FSAM framework got us:

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{n} L\left(y_i, f_{m-1}(x_i) + \beta b(x_i, \gamma)\right)$$

- Classification:
  - $y \in \{-1, 1\}$, exponential loss, $b(x_i, \gamma)$ are weak learners $G(x) \Rightarrow$ AdaBoost
- Regression:
  - $y \in \mathbb{R}$, squared error loss, $b(x_i, \gamma)$ are single features $\mathbf{x}_j \Rightarrow$ Forward stagewise regression
- What if we want to boost weak learners $G(x)$ for regression?

Boosted trees for regression (I)

- Specifically, let us focus on boosting trees: $f(x) = \sum_{m=1}^{M} T(x, \Theta_m)$
- Where $T(x, \Theta) = \sum_{j=1}^{J} \gamma_j \mathbb{I}\left(X \in R_j\right)$
- With parameters $\Theta = \{R_j, \gamma_j\}_1^J$
- “Building a regression tree” is equivalent to:

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{i=1}^{n} \left(y_i - T(x_i, \Theta)\right)^2$$

Boosted trees for regression (II)

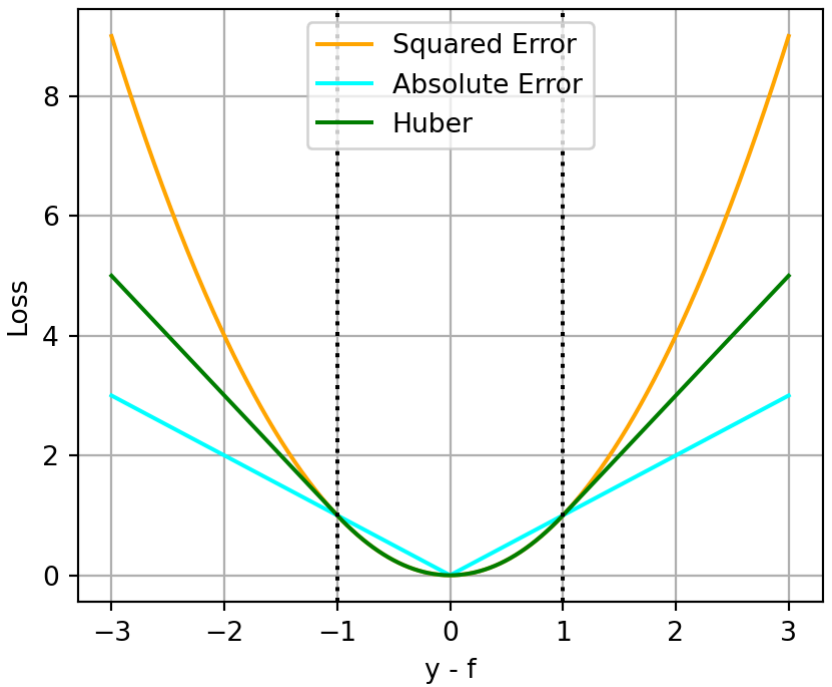
- So FSAM with squared loss, no problem:

$$\Theta_m = \arg\min_\Theta \sum_{i=1}^n L\left(y_i, f_{m-1}(x_i) + T(x_i, \Theta)\right) =$$
$$= \arg\min_\Theta \sum_{i=1}^n \left(y_i - f_{m-1}(x_i) - T(x_i, \Theta)\right)^2$$
$$= \arg\min_\Theta \sum_{i=1}^n \left(r_{im} - T(x_i, \Theta)\right)^2$$

- Because we know how to build such trees!
- To sum up: at each iteration m build standard regression tree $T(x, \Theta)$ which best fits the current residuals $\mathbf{r}_m$

At high-level, similar to AdaBoost!

What about more robust losses?



- Squared Error Loss:

$$L_{\text{sq}}(y, f) = (y - f(x))^2$$

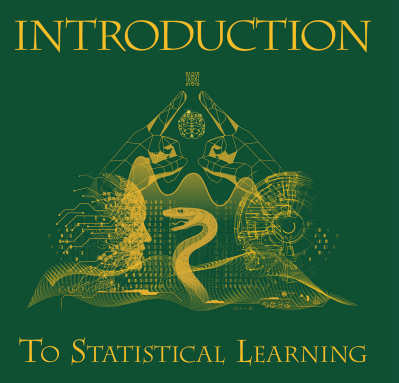
- Absolute Error Loss:

$$L_{\text{abs}}(y, f) = |y - f(x)|$$

- Huber Loss:

$$L_{\text{H}}(y, f) = \begin{cases} (y - f(x))^2 & |y - f(x)| \leq \delta \\ 2\delta|y - f(x)| - \delta^2 & \text{otherwise.} \end{cases}$$

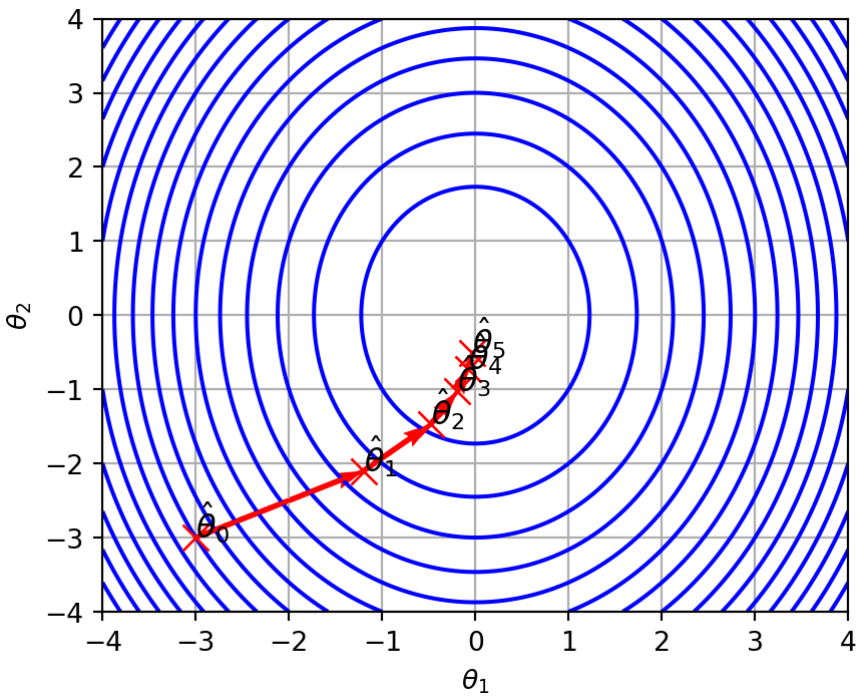
Gradient Boosting



Gradient descent algorithms

Minimize a function $J(\theta)$ by moving in the opposite direction of the gradient:

$$\hat{\theta}_{i+1} = \hat{\theta}_i - \varepsilon \frac{\partial J(\theta)}{\partial \theta}$$



Gradient boosting machines

- Consider any loss $L(y, \mathbf{f})$ a function of n data points $\mathbf{f} = (f(x_1), \ldots, f(x_n))$
- At each iteration m, calculate the gradient of L by $\mathbf{f}$, evaluated at $\mathbf{f}_{m-1}$:

$$\mathbf{g}_m = \frac{\partial L(y, \mathbf{f})}{\partial \mathbf{f}}\bigg|_{\mathbf{f}=\mathbf{f}_{m-1}}$$

- Move a small step ε down this gradient:

$$\mathbf{f}_m = \mathbf{f}_{m-1} - \varepsilon \mathbf{g}_m$$

- Technically, ε can also be optimized at each iteration m to minimize

$$\varepsilon_m = \arg\min_{\varepsilon} L(y, \mathbf{f}_{m-1} - \varepsilon \mathbf{g}_m)$$

Gradient boosting machines

- Eventually we would get:

$$\mathbf{f}_M = \sum_{i=1}^{M} \varepsilon_m(-\mathbf{g}_m)$$

- But we are not interested in $\mathbf{f}_M$ for our training data, we wanted a model! What about x_0?
  - E.g. $f(x) = \sum_{m=1}^{M} \varepsilon T(x, \Theta_m)$
- Solution: at each iteartion m approximate the negative gradient $-\mathbf{g}_m$ with a weak learner or regression tree!

$$\Theta_m = \arg\min_{\Theta} \sum_{i=1}^{n} (-g_{im} - T(x_i, \Theta))^2$$

Gradients of common loss functions

setting	loss	gradient
Regression	$\frac{1}{2}(y_i - f(x_i))^2$	$y_i - f(x)$
Regression	$\lvert y_i - f(x_i)\rvert$	$\text{sign}(y_i - f(x))$
Regression	Huber	$\begin{cases} y_i - f(x) & \text{if } \lvert y - f(x)\rvert \le \delta \\ \delta\text{sign}(y_i - f(x)) & \text{otherwise} \end{cases}$
Classification*	NLL	$\mathbb{I}\left[y_i = 1\right] - \hat{p}_i$

*Here $y \in \{0, 1\}$ and $\hat{p}_i = \frac{\exp(\hat{y}_i)}{1+\exp(\hat{y}_i)} = \frac{\exp(f_{m-1}(x_i))}{1+\exp(f_{m-1}(x_i))}$

Notice for (half) squared loss the gradient is the residuals, as we got earlier.

Gradient boosting trees (simplified)

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^n L(y_i, \gamma)$ (e.g. $f_0(x) = \bar{y}$)
2. For $m = 1$ to M:

    a. Compute pseudo-residuals:

$$\mathbf{r}_m = -\left[\frac{\partial L(y, \mathbf{f})}{\partial \mathbf{f}}\right]_{\mathbf{f}=\mathbf{f}_{m-1}}$$

    b. Fit a **regression tree** to data $(\mathbf{x}, \mathbf{r}_m)$, giving $T(x, \Theta_m)$

    c. Update: $f_m(x) = f_{m-1}(x) + \varepsilon T(x, \Theta_m)$ (or ε_m)

3. Output: $\hat{f}(x) = \sum_{m=1}^M \varepsilon T(x, \Theta_m)$

What parameters need tuning?

Example: credit data