CSE 417T: Introduction to Machine Learning

Lecture 17: Boosting

Henry Chai

10/30/18

Decision Tree / ID3 Cons

- The ID3 algorithm is greedy (it selects the feature w/ the highest information gain at every step) so no optimality guarantee
- Overfitting
  - Can be addressed via heuristics ("regularization") or pruning ("validation"):
- High variance
  - Can be addressed via bagging (random forests)

Bagging

- Short for **B**ootstrap **agg**regat**ing**
- Combines the prediction of many hypotheses to reduce variance

Bootstrapping

- A statistical method for estimating properties of a distribution, given (potentially a small number of) samples from that distribution

- Relies on resampling the samples ***with replacement*** many, many times

Aggregating

- Combining multiple hypotheses, $\{h_1, h_2, \ldots, h_m\}$, to arrive at a single hypothesis
- Regression: average the predictions $\left(\bar{h}(\vec{x}) = \frac{1}{m}\sum_{i=1}^{m} h_i(\vec{x})\right)$
- Classification: find the category that the most hypotheses predict (plurality vote)

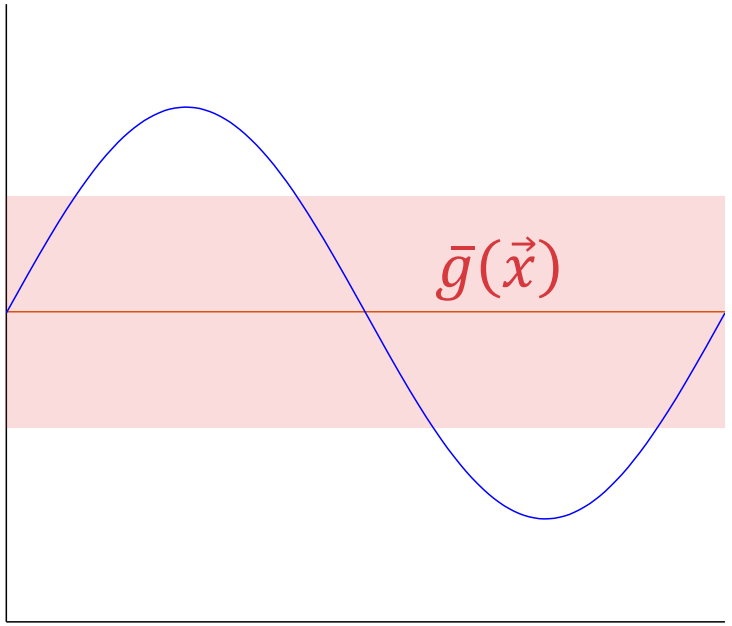
Random Forests

- Input: $\mathcal{D} = \{(\overrightarrow{x_1}, y_1), (\overrightarrow{x_2}, y_2), \dots, (\overrightarrow{x_n}, y_n)\}$, B, m
- For $b = 1, 2, \dots, B$
  - Create a dataset, $\mathcal{D}_b$, by sampling n points from $\mathcal{D}$ with replacement
  - Learn a decision tree, t_b, using $\mathcal{D}_b$ and the ID3 algorithm ***with split-feature randomization***
- Output: $\bar{t}$, the aggregated hypothesis

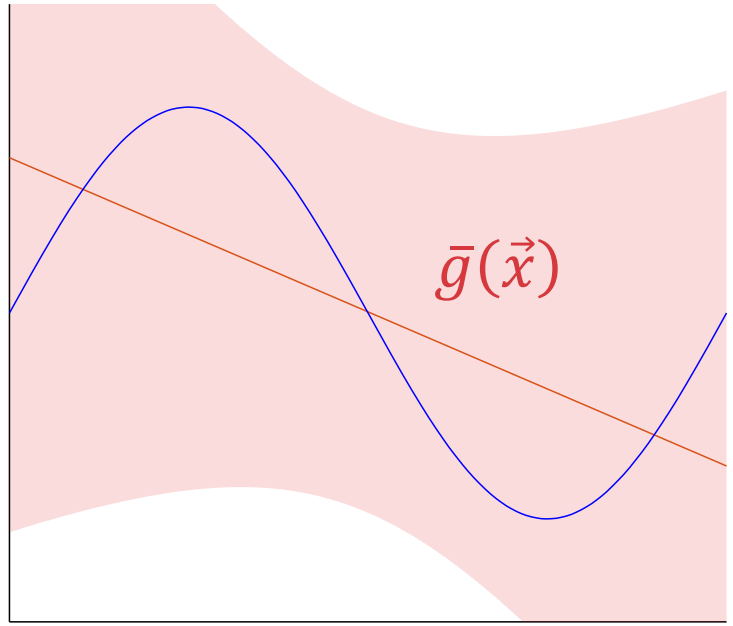
Decision Tree / ID3 Cons

- The ID3 algorithm is greedy (it selects the feature w/ the highest information gain at every step) so no optimality guarantee
- Overfitting
  - Can be addressed via heuristics ("regularization") or pruning ("validation"):
- High variance
  - Can be addressed via bagging (random forests)
- High bias (especially short trees i.e. stumps)

Bias-Variance Tradeoff (Example)

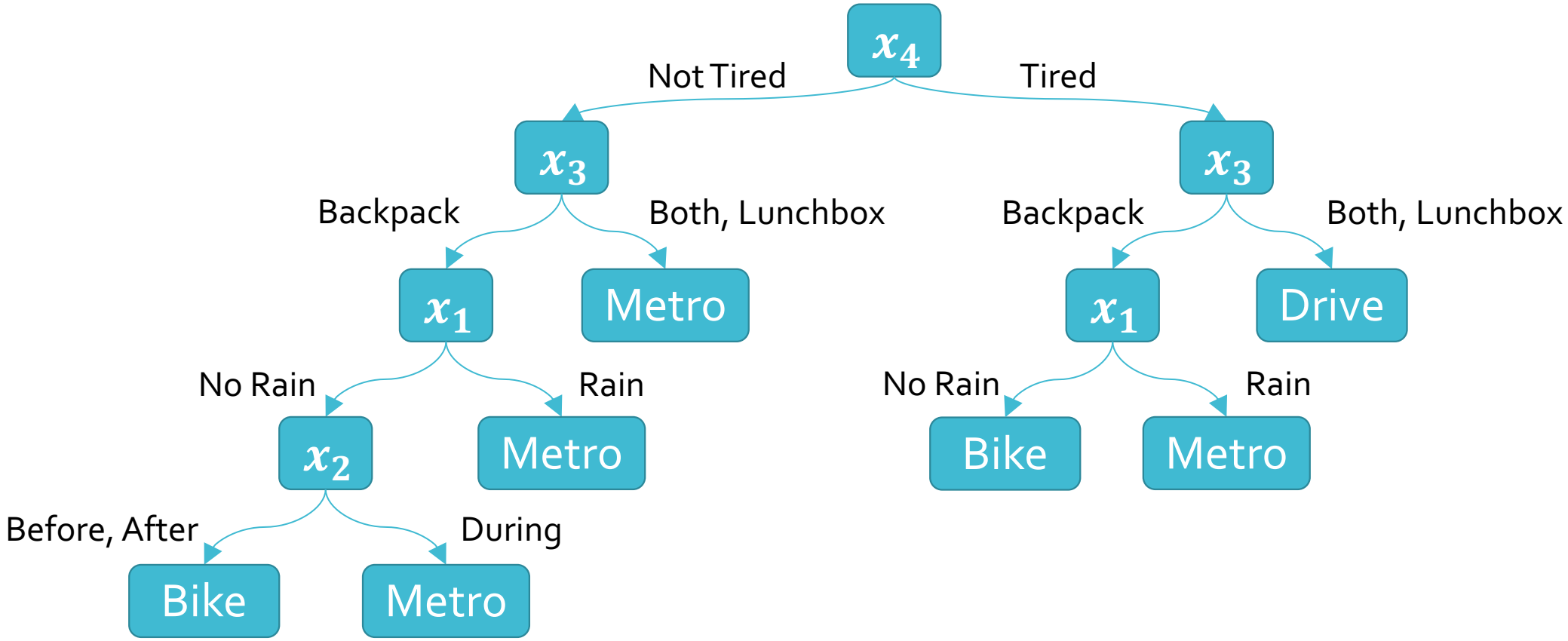


$\mathcal{H}_0$

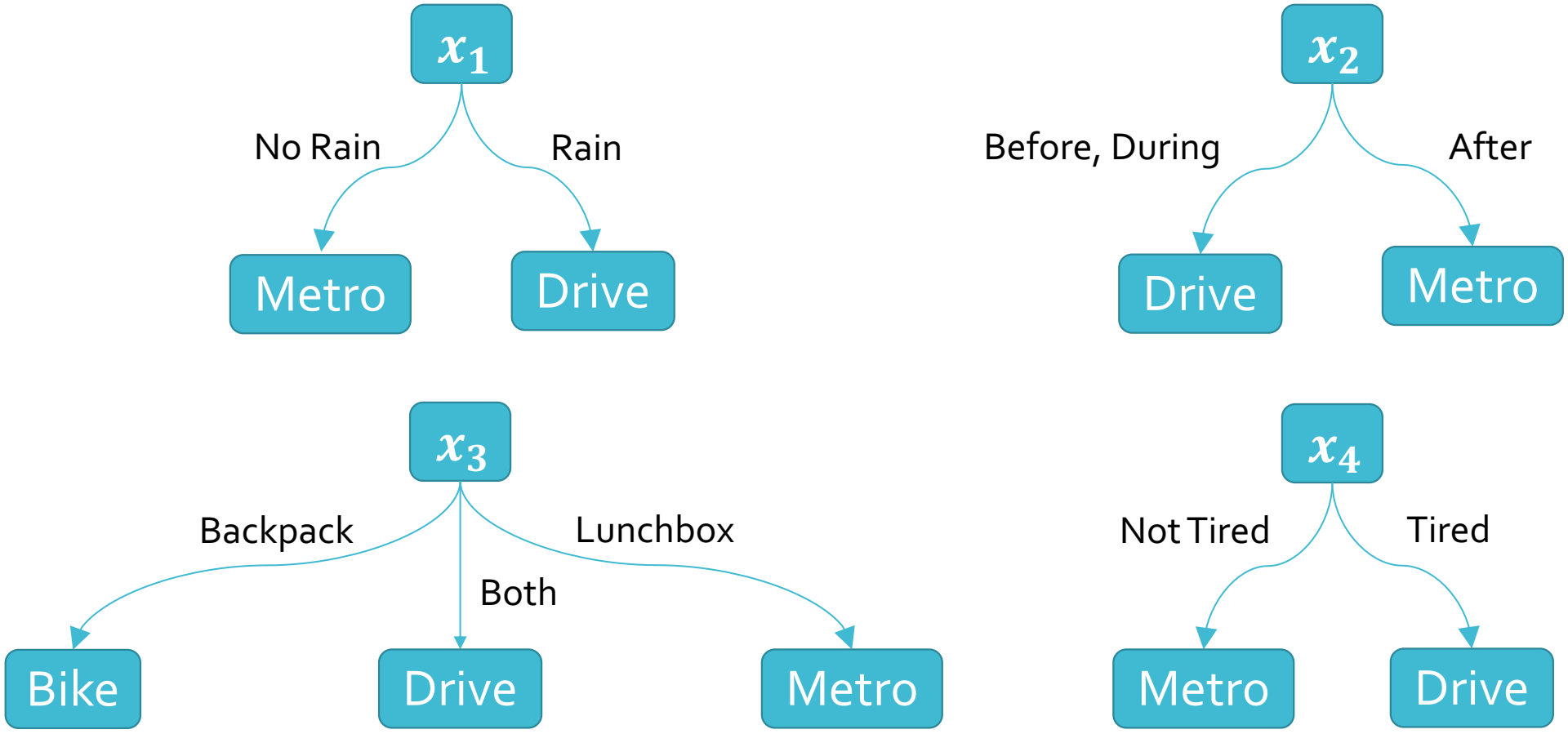


$\mathcal{H}_1$

$$\mathbb{E}_{\mathcal{D}}[E_{out}(g_{\mathcal{D}})] = \mathbb{E}_{\vec{x}}[\text{Variance of } g_{\mathcal{D}}(\vec{x}) + \text{Bias of } \bar{g}(\vec{x})]$$



Decision Tree: Example



Decision Stumps: Example

Boosting

- Another ensemble method (like bagging) that combines the predictions of multiple hypotheses

- Aims to reduce the bias of a "weak" or highly biased hypothesis set (can also reduce variance)

AdaBoost

- Intuition: iteratively reweight inputs, giving more weight to inputs that are difficult-to-predict correctly
- Analogy:
  - You all have to take the midterm again (😱) …
  - … but you're going to be taking it one at a time.
  - After you finish, you get to tell the next person the questions you struggled with.
  - Hopefully, they can cover for you because…
  - … if "enough" of you get a question right, you'll all receive full credit for that problem

AdaBoost

- Input: $\mathcal{D}$ ($\mathcal{Y} = \{-1, +1\}$), T
- Initialize input weights: $\omega_1^{(0)}, \ldots, \omega_n^{(0)} = \frac{1}{n}$
- For $t = 1, \ldots, T$
  1. Train a weak learner (hypothesis), h_t, by minimizing the weighted training error
  2. Compute the weighted training error of h_t:

$$\epsilon_t = \sum_{i=1}^{n} \omega_i^{(t-1)} [\![h_t(\vec{x_i}) \neq y_i]\!]$$

  3. Compute the "importance" of h_t:

$$\alpha_t = \frac{1}{2} \log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$$

  4. Update the weights:

$$\omega_i^{(t)} = \frac{\omega_i^{(t-1)}}{Z_t} \times \begin{cases} e^{-\alpha_t} \text{ if } h_t(\vec{x_i}) = y_i \\ e^{\alpha_t} \text{ if } h_t(\vec{x_i}) \neq y_i \end{cases} =$$

- Output: an aggregated hypothesis

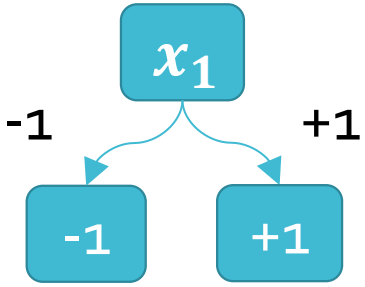
$$g_T(\vec{x}) = \text{sign}\left(H_T(\vec{x})\right)$$

$$= \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\vec{x})\right)$$

Minimizing ϵ_t (w/ decision stumps)

- Decision stumps are simple so just minimize the weighted training error using brute force.

$x_1 \in \{-1, +1\}$	$x_2 \in \mathbb{R}$	y	w
-1	4	+1	0.1
-1	5	+1	0.1
-1	5.3	-1	0.2
-1	5.5	-1	0.05
+1	6.1	+1	0.1
+1	7	-1	0.2
+1	7.2	+1	0.1
+1	8	+1	0.15

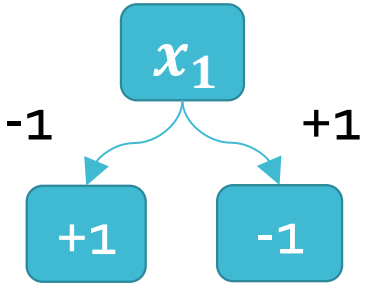


- $\epsilon_t = 0.1 + 0.1 + 0.2 = 0.4$

Minimizing ϵ_t (w/ decision stumps)

- Decision stumps are simple so just minimize the weighted training error using brute force.

$x_1 \in \{-1, +1\}$	$x_2 \in \mathbb{R}$	y	w
-1	4	+1	0.1
-1	5	+1	0.1
-1	5.3	-1	0.2
-1	5.5	-1	0.05
+1	6.1	+1	0.1
+1	7	-1	0.2
+1	7.2	+1	0.1
+1	8	+1	0.15

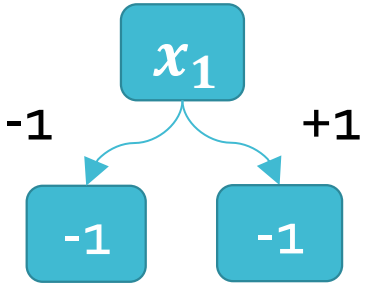


- $\epsilon_t = 0.2 + 0.05 + 0.1 + 0.1 + 0.15 = 0.6$

Minimizing ϵ_t (w/ decision stumps)

- Decision stumps are simple so just minimize the weighted training error using brute force.

$x_1 \in \{-1, +1\}$	$x_2 \in \mathbb{R}$	y	w
-1	4	+1	0.1
-1	5	+1	0.1
-1	5.3	-1	0.2
-1	5.5	-1	0.05
+1	6.1	+1	0.1
+1	7	-1	0.2
+1	7.2	+1	0.1
+1	8	+1	0.15

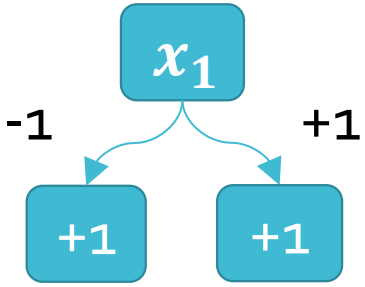


- $\epsilon_t = 0.1 + 0.1 + 0.1 + 0.1 + 0.15 = 0.55$

Minimizing ϵ_t (w/ decision stumps)

- Decision stumps are simple so just minimize the weighted training error using brute force.

$x_1 \in \{-1, +1\}$	$x_2 \in \mathbb{R}$	y	w
-1	4	+1	0.1
-1	5	+1	0.1
-1	5.3	-1	0.2
-1	5.5	-1	0.05
+1	6.1	+1	0.1
+1	7	-1	0.2
+1	7.2	+1	0.1
+1	8	+1	0.15

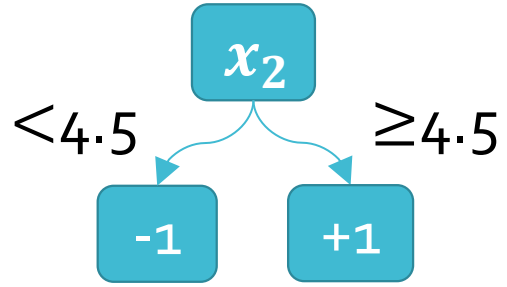


- $\epsilon_t = 0.2 + 0.05 + 0.2 = 0.45$

Minimizing ϵ_t (w/ decision stumps)

- Decision stumps are simple so just minimize the weighted training error using brute force.

$x_1 \in \{-1, +1\}$	$x_2 \in \mathbb{R}$	y	w
-1	4	+1	0.1
-1	5	+1	0.1
-1	5.3	-1	0.2
-1	5.5	-1	0.05
+1	6.1	+1	0.1
+1	7	-1	0.2
+1	7.2	+1	0.1
+1	8	+1	0.15

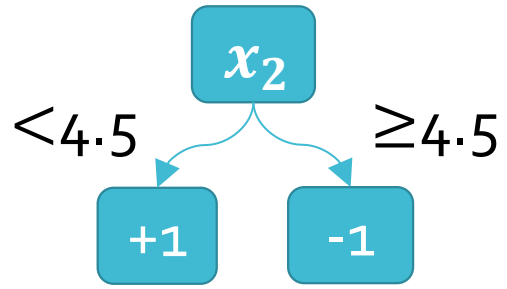


- $\epsilon_t = 0.1 + 0.2 + 0.05 + 0.2 = 0.55$

Minimizing ϵ_t (w/ decision stumps)

- Decision stumps are simple so just minimize the weighted training error using brute force.

$x_1 \in \{-1, +1\}$	$x_2 \in \mathbb{R}$	y	w
-1	4	+1	0.1
-1	5	+1	0.1
-1	5.3	-1	0.2
-1	5.5	-1	0.05
+1	6.1	+1	0.1
+1	7	-1	0.2
+1	7.2	+1	0.1
+1	8	+1	0.15

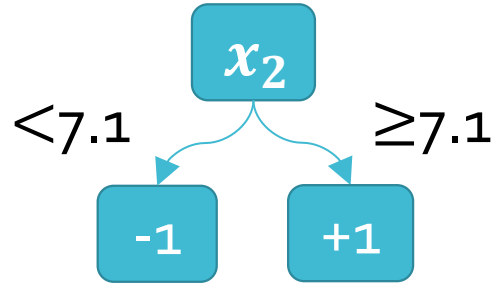


- $\epsilon_t = 0.1 + 0.1 + 0.1 + 0.15 = 0.45$

Minimizing ϵ_t (w/ decision stumps)

- Decision stumps are simple so just minimize the weighted training error using brute force.

$x_1 \in \{-1, +1\}$	$x_2 \in \mathbb{R}$	y	w
-1	4	+1	0.1
-1	5	+1	0.1
-1	5.3	-1	0.2
-1	5.5	-1	0.05
+1	6.1	+1	0.1
+1	7	-1	0.2
+1	7.2	+1	0.1
+1	8	+1	0.15



- $\epsilon_t = 0.1 + 0.1 + 0.1 = 0.3$

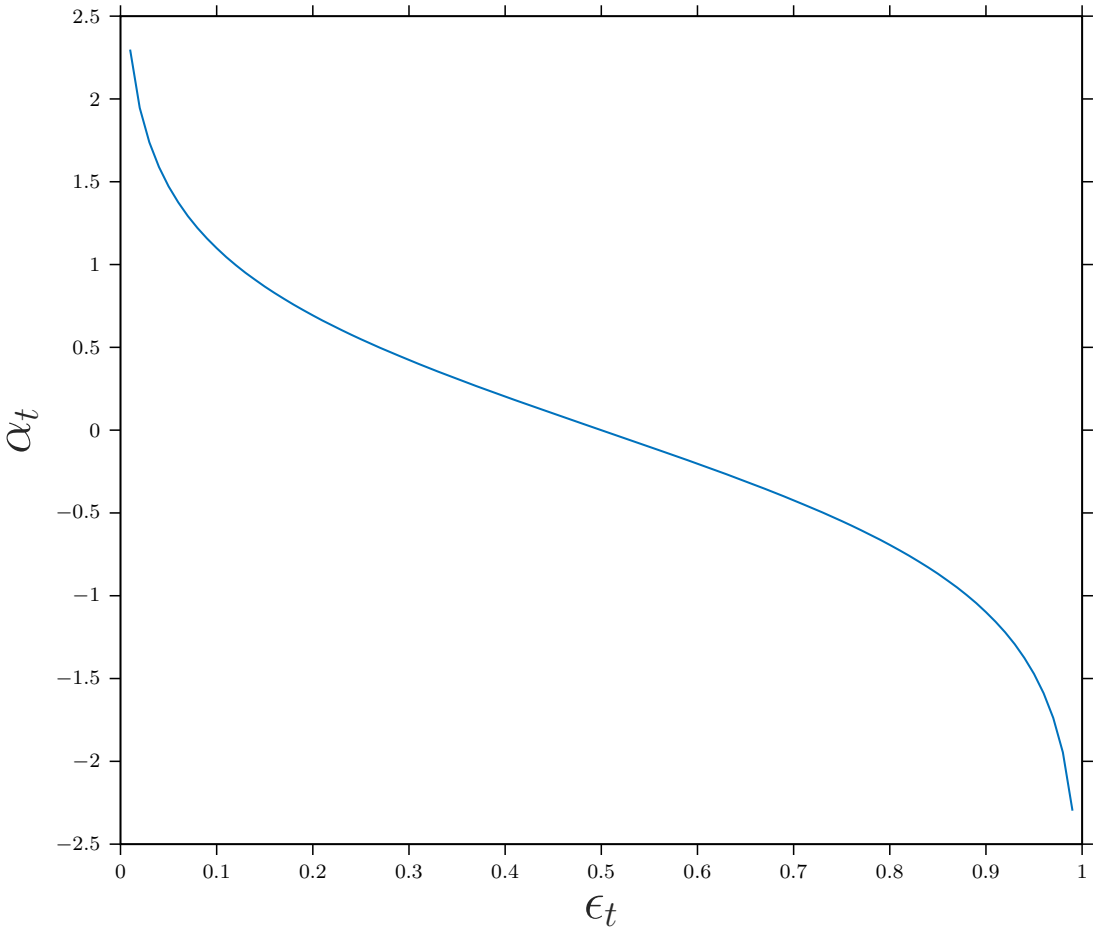
Setting α_t

α_t determines the contribution of h_t to the final, aggregated hypothesis:

$$g(\vec{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\vec{x})\right)$$

Intuitively, we want good hypotheses to have high weights

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$



Updating ω_i

- Intuitively, we want incorrectly classified inputs to receive a higher weight in the next round

$$\omega_i^{(t)} = \frac{\omega_i^{(t-1)}}{Z_t} \times \begin{cases} e^{-\alpha_t} \text{ if } h_t(\overrightarrow{x_i}) = y_i \\ e^{\alpha_t} \text{ if } h_t(\overrightarrow{x_i}) \neq y_i \end{cases} = \frac{\omega_i^{(t-1)} e^{-\alpha_t y_i h_t(\overrightarrow{x_i})}}{Z_t}$$

- If $\epsilon_t < \frac{1}{2}$ then $\frac{1-\epsilon_t}{\epsilon_t} > 1$
- If $\frac{1-\epsilon_t}{\epsilon_t} > 1$ then $\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$
- If $\alpha_t > 0$ then $e^{-\alpha_t} < 1$ and $e^{\alpha_t} > 1$

Normalizing ω_i

$$Z_t = \sum_{i=1}^{n} \omega_i^{(t-1)} e^{-\alpha_t y_i h_t(\vec{x_i})}$$

$$= \sum_{y_i = h_t(\vec{x_i})} \omega_i^{(t-1)} e^{-\alpha_t} + \sum_{y_i \neq h_t(\vec{x_i})} \omega_i^{(t-1)} e^{\alpha_t}$$

$$= e^{-\alpha_t} \sum_{y_i = h_t(\vec{x_i})} \omega_i^{(t-1)} + e^{\alpha_t} \sum_{y_i \neq h_t(\vec{x_i})} \omega_i^{(t-1)}$$

$$= e^{-\alpha_t}(1 - \epsilon_t) + e^{\alpha_t}\epsilon_t$$

$$= e^{-\frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)}(1 - \epsilon_t) + e^{\frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)}\epsilon_t$$

$$= \sqrt{\epsilon_t(1 - \epsilon_t)} + \sqrt{\epsilon_t(1 - \epsilon_t)} = 2\sqrt{\epsilon_t(1 - \epsilon_t)}$$

Why AdaBoost?

1. If you only have access to weak learners …
2. … and want your final hypothesis to be a weighted combination of weak learners, …
3. … then Adaboost greedily minimizes the exponential loss:

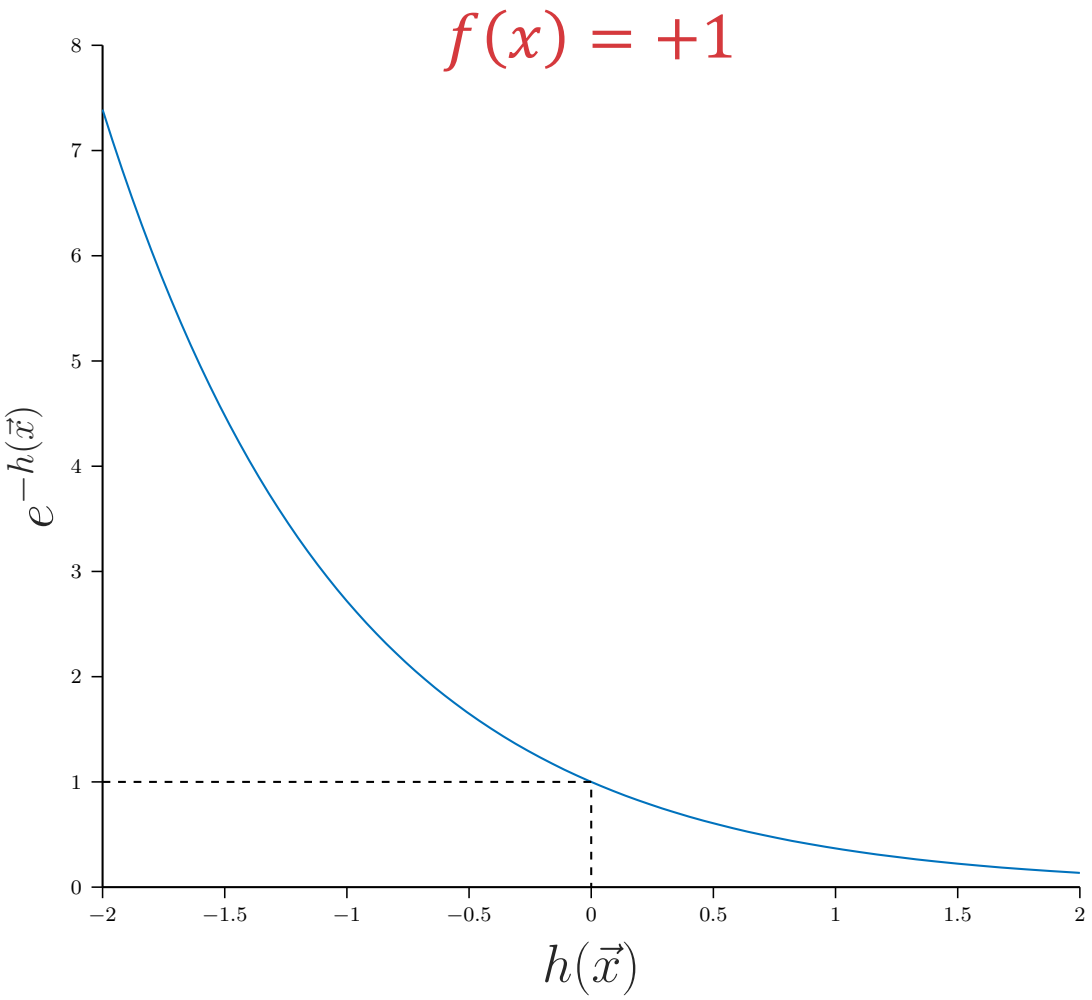
$$e(h, f, \vec{x}) = e^{(-f(\vec{x})h(\vec{x}))}$$

1. Because of computational constraints
2. Because weak learners are not great on their own
3. Because the exponential loss upper bounds binary error

Exponential Loss

Applies to binary classification ($f(\vec{x}) \in \{-1, +1\}$) with hypotheses of the form $g(\vec{x}) = \text{sign}\big(h(\vec{x})\big)$

The more $h(\vec{x})$ "agrees with" $f(\vec{x})$, the smaller the loss and similarly, the more $h(\vec{x})$ "disagrees with" $f(\vec{x})$, the greater the loss



Exponential Loss

- Claim: $e^{(-f(\vec{x})h(\vec{x}))} \geq [\![f(\vec{x}) \neq \text{sign}(h(\vec{x}))]\!]$
- Proof:
  - $f(\vec{x}) \neq \text{sign}(h(\vec{x})) \rightarrow [\![f(\vec{x}) \neq \text{sign}(h(\vec{x}))]\!] = 1$ and $f(\vec{x})h(\vec{x}) \leq 0 \rightarrow e^{(-f(\vec{x})h(\vec{x}))} \geq 1$
  - $f(\vec{x}) = \text{sign}(h(\vec{x})) \rightarrow [\![f(\vec{x}) \neq \text{sign}(h(\vec{x}))]\!] = 0$ and $e^{(-f(\vec{x})h(\vec{x}))} \geq 0 \; \forall \; \vec{x}$ ■
- Consequence:

$$\frac{1}{n}\sum_{i=1}^{n} e^{(-f(\vec{x_i})h(\vec{x_i}))} \geq \frac{1}{n}\sum_{i=1}^{n} [\![f(\vec{x}) \neq \text{sign}(h(\vec{x}))]\!]$$

Exponential Loss

- Claim: $e^{(-f(\vec{x})h(\vec{x}))} \geq [\![f(\vec{x}) \neq \text{sign}(h(\vec{x}))]\!]$
- Proof:
  - $f(\vec{x}) \neq \text{sign}(h(\vec{x})) \rightarrow [\![f(\vec{x}) \neq \text{sign}(h(\vec{x}))]\!] = 1$ and $f(\vec{x})h(\vec{x}) \leq 0 \rightarrow e^{(-f(\vec{x})h(\vec{x}))} \geq 1$
  - $f(\vec{x}) = \text{sign}(h(\vec{x})) \rightarrow [\![f(\vec{x}) \neq \text{sign}(h(\vec{x}))]\!] = 0$ and $e^{(-f(\vec{x})h(\vec{x}))} \geq 0 \; \forall \; \vec{x}$ ■
- Consequence:

$$\frac{1}{n}\sum_{i=1}^{n} e^{(-f(\vec{x_i})h(\vec{x_i}))} \rightarrow 0 \Rightarrow \frac{1}{n}\sum_{i=1}^{n} [\![f(\vec{x}) \neq \text{sign}(h(\vec{x}))]\!] \rightarrow 0$$

Exponential Loss

- Claim: if $g_T = \text{sign}(H_T)$ is the Adaboost hypothesis, then

$$\frac{1}{n}\sum_{i=1}^{n} e^{\left(-y_i H_T(\vec{x_i})\right)} = \prod_{t=1}^{T} Z_t$$

- Proof:

$$\omega_i^{(0)} = \frac{1}{n}, \omega_i^{(1)} = \frac{e^{-\alpha_1 y_i h_1(\vec{x_i})}}{nZ_1}, \omega_i^{(2)} = \frac{e^{-\alpha_1 y_i h_1(\vec{x_i})} e^{-\alpha_2 y_i h_2(\vec{x_i})}}{nZ_1 Z_2}$$

$$\omega_i^{(T)} = \frac{\prod_{t=1}^{T} e^{-\alpha_t y_i h_t(\vec{x_i})}}{n\prod_{t=1}^{T} Z_t} = \frac{e^{-y_i \sum_{t=1}^{T} \alpha_t h_t(\vec{x_i})}}{n\prod_{t=1}^{T} Z_t} = \frac{e^{-y_i H_T(\vec{x_i})}}{n\prod_{t=1}^{T} Z_t}$$

$$\sum_{i=1}^{n} \omega_i^{(T)} = \sum_{i=1}^{n} \frac{e^{-y_i H_T(\vec{x_i})}}{n\prod_{t=1}^{T} Z_t} = 1 \Rightarrow \frac{1}{n}\sum_{i=1}^{n} e^{-y_i H_T(\vec{x_i})} = \prod_{t=1}^{T} Z_t \blacksquare$$

Exponential Loss

- Claim: if $g_T = \text{sign}(H_T)$ is the Adaboost hypothesis, then

$$\frac{1}{n}\sum_{i=1}^{n} e^{(-y_i H_T(\vec{x_i}))} = \prod_{t=1}^{T} Z_t$$

- Consequence: one way to minimize the in-sample exponential loss is to greedily minimize Z_t i.e. at iteration t, make Z_t as small as possible

Greedy Exponential Loss Minimization

$$Z_t = \sum_{i=1}^{n} \omega_i^{(t-1)} e^{-(w) y_i h_t(\vec{x_i})}$$

$$= \sum_{y_i = h_t(\vec{x_i})} \omega_i^{(t-1)} e^{-w} + \sum_{y_i \neq h_t(\vec{x_i})} \omega_i^{(t-1)} e^{w}$$

$$= e^{-w} \sum_{y_i = h_t(\vec{x_i})} \omega_i^{(t-1)} + e^{w} \sum_{y_i \neq h_t(\vec{x_i})} \omega_i^{(t-1)}$$

$$= e^{-w}(1 - \epsilon_t) + e^{w} \epsilon_t$$

Greedy Exponential Loss Minimization

$$Z_t = e^{-w}(1-\epsilon_t) + e^{w}\epsilon_t$$

$$\frac{\delta Z_t}{\delta w} = -e^{-w}(1-\epsilon_t) + e^{w}\epsilon_t \Rightarrow -e^{-w^*}(1-\epsilon_t) + e^{w^*}\epsilon_t = 0$$

$$\Rightarrow e^{w^*}\epsilon_t = e^{-w^*}(1-\epsilon_t)$$

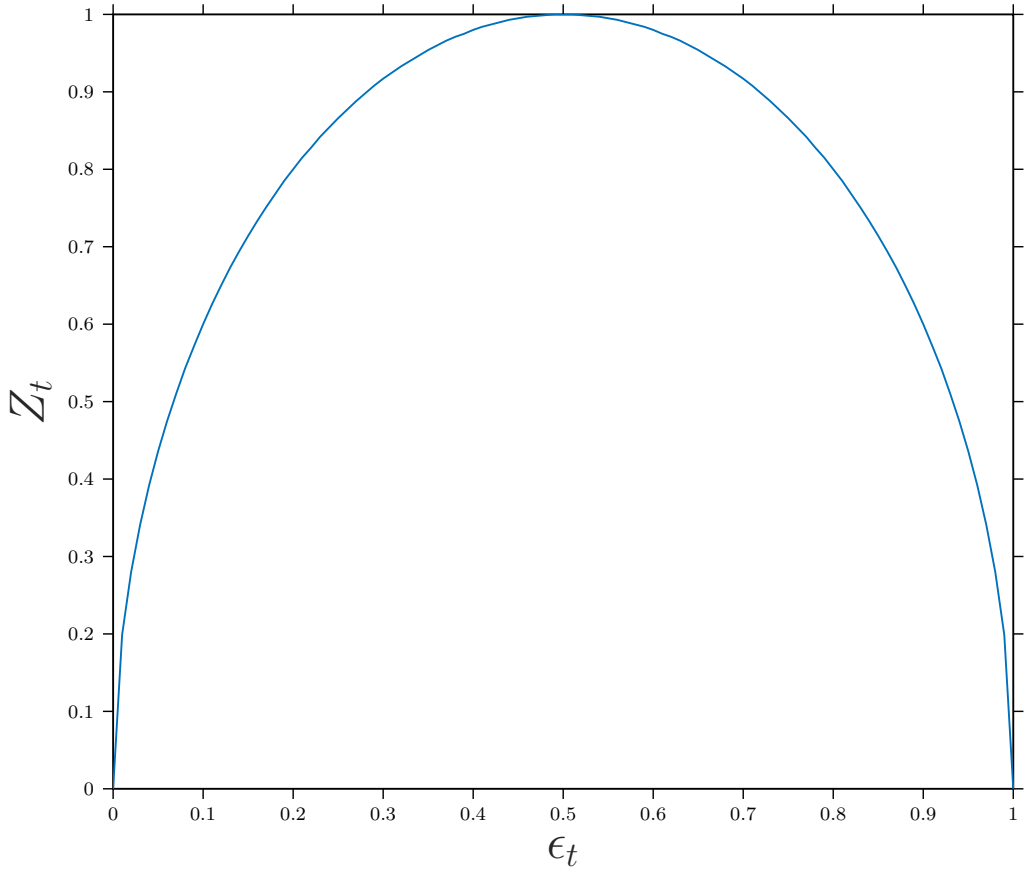
$$\Rightarrow e^{2w^*} = \frac{(1-\epsilon_t)}{\epsilon_t}$$

$$\Rightarrow w^* = \frac{1}{2}\log\left(\frac{(1-\epsilon_t)}{\epsilon_t}\right) = \alpha_t$$

$$\frac{\delta^2 Z_t}{\delta w^2} = e^{-w}(1-\epsilon_t) + e^{w}\epsilon_t > 0$$

Z_t

$$Z_t = \sum_{i=1}^{n} \omega_i^{(t-1)} e^{-\alpha_t y_i h_t(\vec{x_i})} = 2\sqrt{\epsilon_t(1-\epsilon_t)} < 1 \text{ if } \epsilon_t < \frac{1}{2}$$



In-sample Error

$$\frac{1}{n}\sum_{i=1}^{n}[\![y_i \neq g_T(\vec{x_i})]\!] \leq \frac{1}{n}\sum_{i=1}^{n} e^{\left(-y_i H_T(\vec{x_i})\right)}$$

$$= \prod_{t=1}^{T} Z_t$$

$$= \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)} \rightarrow 0 \text{ as } \mathrm{T} \rightarrow \infty$$

$$\left(\text{as long as } \epsilon_t < \frac{1}{2}\ \forall\ t\right)$$

Out-of-sample Error

- For AdaBoost, with high probability:

$$E_{out}(g) \leq E_{in}(g) + \tilde{O}\left(\sqrt{\frac{d_{vc}(\mathcal{H})T}{n}}\right)$$

where $d_{vc}(\mathcal{H})$ is the VC-dimension of the weak learners,

T is the number of weak learners,

n is the number of training data points

- Empirical results indicate that increasing T does not lead to overfitting as this bound would suggest

Margins

- After running Adaboost for T rounds, the returned hypothesis is

$$g_T(\vec{x}) = \text{sign}\big(H_T(\vec{x})\big) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\vec{x})\right)$$

- The margin of training point $(\overrightarrow{x_i}, y_i)$ is defined as:

$$m(\overrightarrow{x_i}, y_i) = \frac{y_i H_T(\vec{x})}{\sum_{t=1}^{T} \alpha_t}$$

- The margin of $(\overrightarrow{x_i}, y_i)$ is positive if g_T is correct at predicting $\overrightarrow{x_i}$
- The margin can be interpreted as how confident g_T is in its prediction: the bigger the margin, the more confident

Out-of-sample Error

- For AdaBoost, with high probability:

$$E_{out}(g) \leq \frac{1}{n}\sum_{i=1}^{n}[\![m(\overrightarrow{x_i}, y_i) \leq \theta]\!] + \tilde{O}\left(\sqrt{\frac{d_{vc}(\mathcal{H})}{n\theta^2}}\right)$$

where $d_{vc}(\mathcal{H})$ is the VC-dimension of the weak learners,

n is the number of training data points,

$\theta > 0$ is an arbitrary parameter