10-301/601: Introduction to Machine Learning Lecture 30 – Boosting

Henry Chai

6/11/25

Boosting

- An *ensemble method* combines the predictions of multiple “weak” hypotheses to learn a single, more powerful classifier
- Boosting is a *meta-algorithm*: it can be applied to a variety of machine learning models
    - Commonly used with decision trees

Decision Trees: Pros & Cons

- Pros
  - Interpretable
  - Efficient (computational cost and storage)
  - Can be used for classification and regression tasks
  - Compatible with categorical and real-valued features
- Cons
  - Learned greedily: each split only considers the immediate impact on the splitting criterion
    - Not guaranteed to find the smallest (fewest number of splits) tree that achieves a training error rate of 0.
  - Prone to overfit
  - Highly variable
    - Can be addressed via bagging → random forests
  - Limited expressivity (especially short trees, i.e., stumps)
    - Can be addressed via boosting

AdaBoost

- Intuition: iteratively reweight inputs, giving more weight to inputs that are difficult-to-predict correctly
- Analogy:
  - You all have to take a test (😱) …
  - … 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: $\mathcal{D}\left(y^{(n)} \in \{-1,+1\}\right), T$
- Initialize data point weights: $\omega_0^{(1)}, \dots, \omega_0^{(N)} = \frac{1}{N}$
- For $t = 1, \dots, T$
  1. Train a weak learner, h_t, by minimizing the *weighted* training error
  2. Compute the *weighted* training error of h_t:
     $$\epsilon_t = \sum_{n=1}^{N} \omega_{t-1}^{(n)} \mathbb{1}\left(y^{(n)} \neq h_t\left(\boldsymbol{x}^{(n)}\right)\right)$$
  3. Compute the **importance** of h_t:
     $$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$
  4. Update the data point weights:

$$\omega_t^{(n)} = \frac{\omega_{t-1}^{(n)}}{Z_t} \times \begin{cases} e^{-\alpha_t} \text{ if } h_t\left(\boldsymbol{x}^{(n)}\right) = y^{(n)} \\ e^{\alpha_t} \quad \text{if } h_t\left(\boldsymbol{x}^{(n)}\right) \neq y^{(n)} \end{cases}$$

- Output: an aggregated hypothesis

$$g_T(\boldsymbol{x}) = \operatorname{sign}\left(H_T(\boldsymbol{x})\right)$$

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

Setting α_t

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

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

Intuition: we want good weak learners to have high importances

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

How does the importance of a very bad/mostly incorrect weak learner compare to the importance of a very good/mostly correct weak learner?

Similar magnitude, same sign

0%

Similar magnitude, different sign

0%

Different magnitude, same sign

0%

Different magnitude, different sign

0%

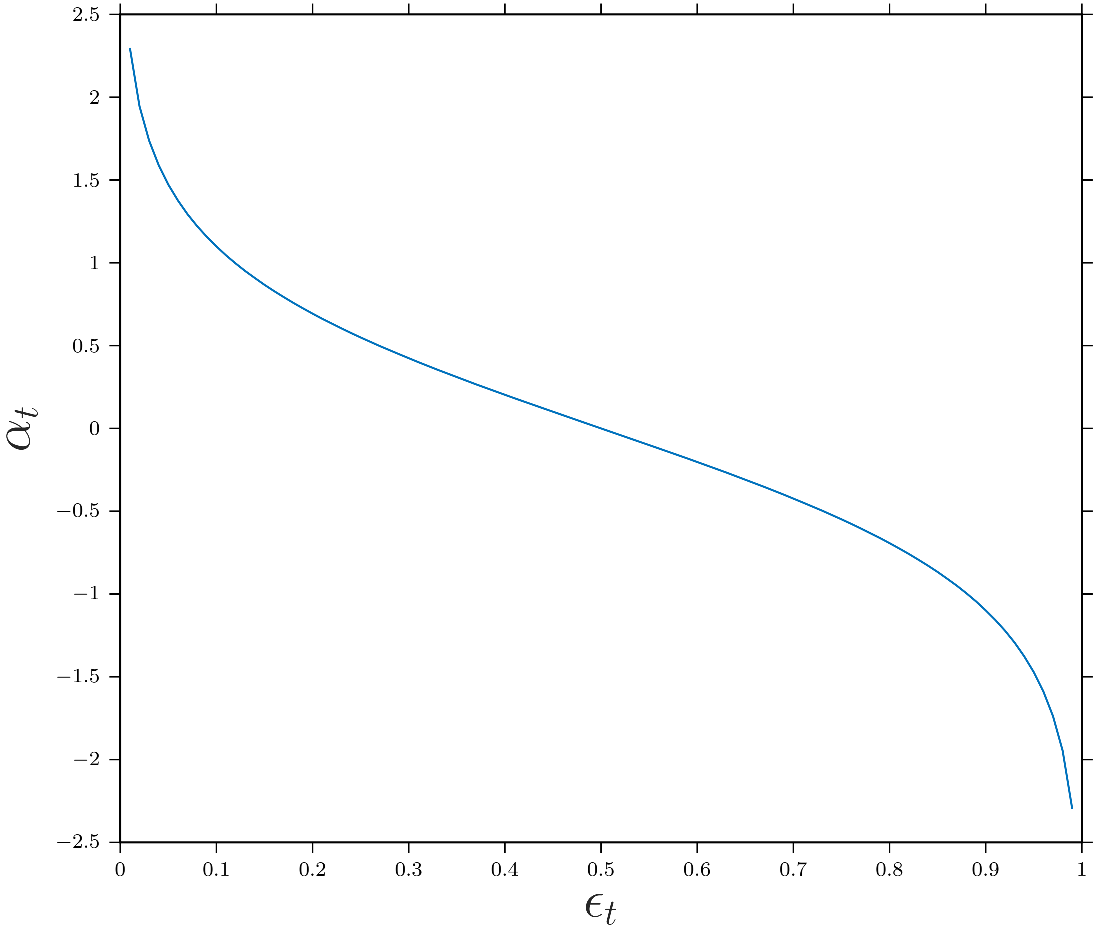
Setting α_t

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

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

Intuition: we want good weak learners to have high importances

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



Updating $\omega^{(n)}$

- Intuition: we want incorrectly classified inputs to receive a higher weight in the next round

$$\omega_t^{(n)} = \frac{\omega_{t-1}^{(n)}}{Z_t} \times \underbrace{\begin{cases} e^{-\alpha_t} \text{ if } h_t\left(\boldsymbol{x}^{(n)}\right) = y^{(n)} \\ e^{\alpha_t} \text{ if } h_t\left(\boldsymbol{x}^{(n)}\right) \neq y^{(n)} \end{cases}}_{} = \frac{\omega_{t-1}^{(n)} e^{-\alpha_t y^{(n)} h_t\left(\boldsymbol{x}^{(n)}\right)}}{Z_t}$$

- if $\epsilon_t < \frac{1}{2} \Rightarrow \frac{1-\epsilon_t}{\epsilon_t} > 1$

$\Rightarrow \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) = \alpha_t > 0$

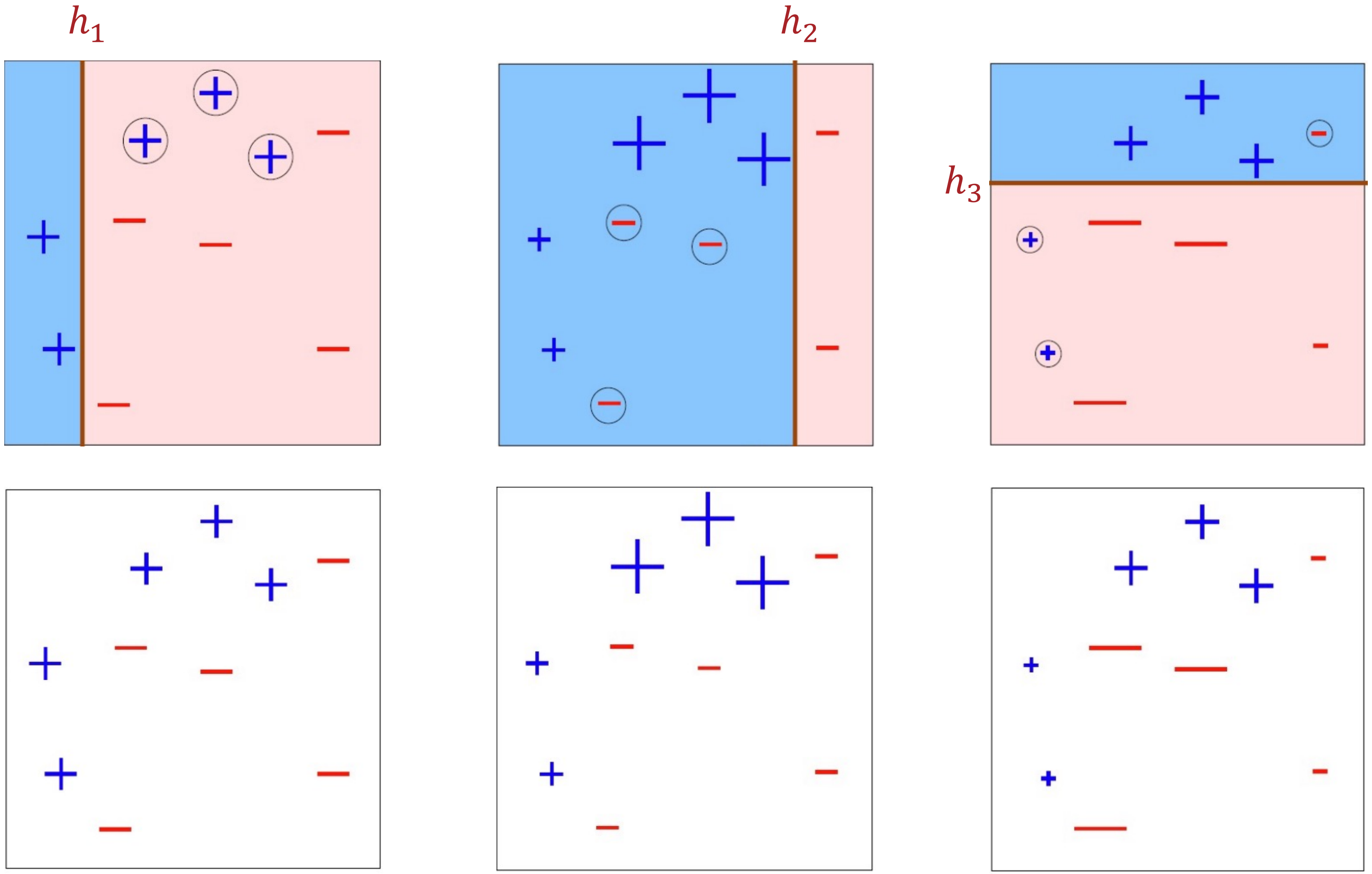
$\Rightarrow e^{\alpha_t} > 1$ and $e^{-\alpha_t} < 1$

AdaBoost: Example

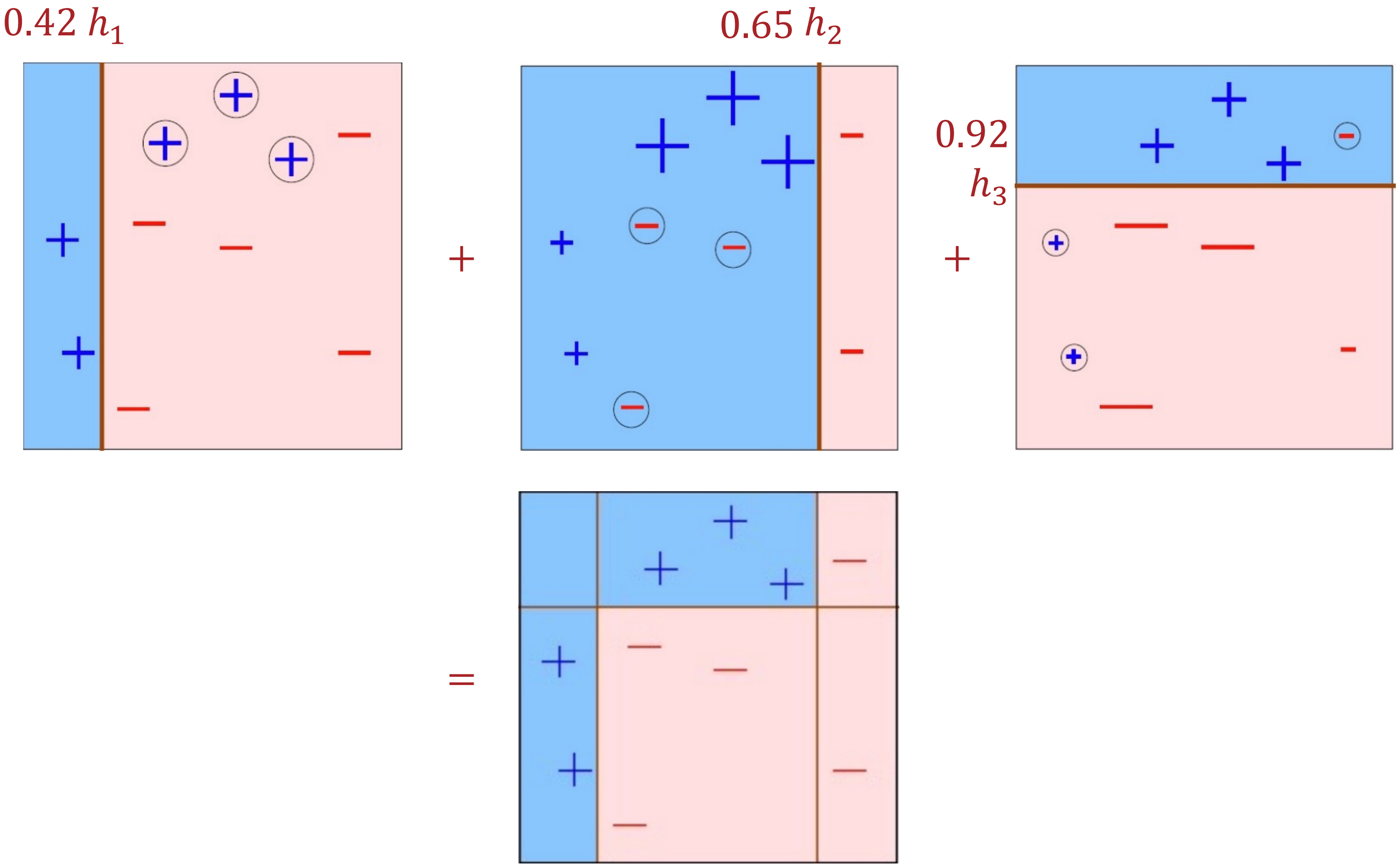
$\epsilon_1 = 0.3$
$\alpha_1 = 0.42$

$\epsilon_2 = 0.21$
$\alpha_2 = 0.65$

$\epsilon_3 = 0.14$
$\alpha_3 = 0.92$



AdaBoost: Example



Why AdaBoost?

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

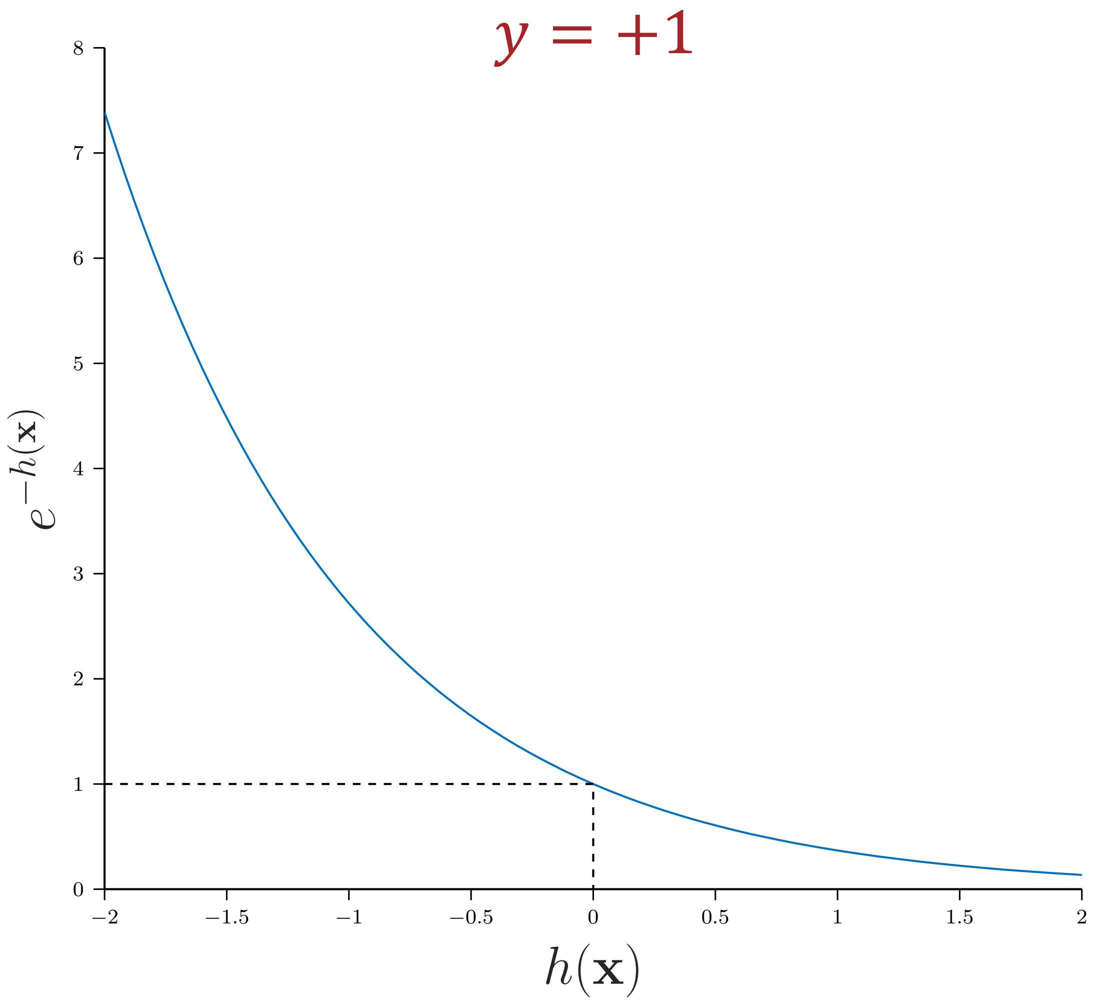
$$e(h(\boldsymbol{x}), y) = e^{(-yh(\boldsymbol{x}))}$$

1. Because they’re low variance / computational constraints
2. Because weak learners are not great on their own
3. Because the exponential loss upper bounds binary error

Exponential Loss

$$e(h(\boldsymbol{x}), y) = e^{(-yh(\boldsymbol{x}))}$$

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



Exponential Loss

- Claim:

$$\frac{1}{N}\sum_{n=1}^{N} e^{\left(-y^{(n)}h\left(\boldsymbol{x}^{(n)}\right)\right)} \geq \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}\left(\text{sign}\left(h\left(\boldsymbol{x}^{(n)}\right)\right) \neq y^{(n)}\right)$$

- Consequence:

$$\frac{1}{N}\sum_{n=1}^{N} e^{\left(-y^{(n)}h\left(\boldsymbol{x}^{(n)}\right)\right)} \to 0$$

$$\Longrightarrow \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}\left(\text{sign}\left(h\left(\boldsymbol{x}^{(n)}\right)\right) \neq y^{(n)}\right) \to 0$$

Exponential Loss

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

aggregated $\Rightarrow H_T(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x})$

$$\frac{1}{N}\sum_{n=1}^{N} e^{\left(-y^{(n)} H_T\left(\boldsymbol{x}^{(n)}\right)\right)} = \prod_{t=1}^{T} Z_t$$

- Proof:

$$w_0^{(n)} = \frac{1}{N}, \quad w_1^{(n)} = \frac{e^{-\alpha_1 y^{(n)} h_1(\vec{x}^{(n)})}}{Z_1} w_0^{(n)} = \frac{e^{-\alpha_1 y^{(n)} h_1(\vec{x}^{(n)})}}{NZ}$$

$$w_2^{(n)} = \left(\frac{e^{-\alpha_1 y^{(n)} h_1(\vec{x}^{(n)})}}{NZ_1}\right)\left(\frac{e^{-\alpha_2 y^{(n)} h_2(\vec{x}^{(n)})}}{Z_2}\right) = \frac{e^{-\alpha_1 y^{(n)} h_1(\vec{x}^{(n)}) - \alpha_2 y^{(n)} h_2(\vec{x}^{(n)})}}{NZ_1 Z_2}$$

$$\vdots$$

$$w_T^{(n)} = \frac{e^{-\sum_{t=1}^{T} \alpha_t y^{(n)} h_t(\vec{x}^{(n)})}}{N \prod_{t=1}^{T} Z_t} \Rightarrow \sum_{n=1}^{N} w_T^{(n)} = \sum_{n=1}^{N} \frac{e^{-y^{(n)} \sum_{t=1}^{T} \alpha_t h_t(\vec{x}^{(n)})}}{N \prod_{t=1}^{T} Z_t} = 1$$

$$\Rightarrow \frac{1}{N} \sum_{n=1}^{N} e^{-y^{(n)} \sum_{t=1}^{T} \alpha_t h_t(\vec{x}^{(n)})} = \prod_{t=1}^{T} Z_t \quad ■$$

Exponential Loss

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

$$\frac{1}{N}\sum_{n=1}^{N} e^{\left(-y^{(n)}H_T\left(\boldsymbol{x}^{(n)}\right)\right)} = \prod_{t=1}^{T} Z_t$$

- Consequence: one way to minimize the exponential training loss is to greedily minimize Z_t, i.e., in each iteration, make the normalization constant as small as possible by tuning α_t.

$$Z_t(\alpha_t) = \sum_{n=1}^{N} \omega_{t-1}^{(n)} e^{-\alpha_t y^{(n)} h_t(\vec{x}^{(n)})}$$

Greedy Exponential Loss Minimization

$$Z_t(a) = \sum_{n=1}^{N} \omega_{t-1}^{(n)} e^{-(a)y^{(n)}h_t(\boldsymbol{x}^{(n)})}$$

$$= \sum_{y^{(n)} = h_t(\vec{x}^{(n)})} \omega_{t-1}^{(n)} e^{-a} + \sum_{y^{(n)} \neq h_t(\vec{x}^{(n)})} \omega_{t-1}^{(n)} e^{a}$$

$$= e^{-a} \underbrace{\sum_{y^{(n)} = h_t(\vec{x}^{(n)})} \omega_{t-1}^{(n)}}_{1-\epsilon_t} + e^{a} \underbrace{\sum_{y^{(n)} \neq h_t(\vec{x}^{(n)})} \omega_{t-1}^{(n)}}_{\epsilon_t}$$

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

Greedy Exponential Loss Minimization

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

$$\frac{\partial Z_t}{\partial a} = -e^{-a}(1-\epsilon_t) + e^{a}\epsilon_t$$

$$\Rightarrow -e^{-\hat{a}}(1-\epsilon_t) + e^{\hat{a}}\epsilon_t = 0$$

$$\Rightarrow e^{\hat{a}}\epsilon_t = e^{-\hat{a}}(1-\epsilon_t)$$

$$\Rightarrow e^{2\hat{a}} = \frac{(1-\epsilon_t)}{\epsilon_t}$$

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

Normalizing $\omega^{(n)}$

$$Z_t = \sum_{n=1}^{N} \omega_{t-1}^{(n)} e^{-\alpha_t y^{(n)} h_t(\boldsymbol{x}^{(n)})}$$

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

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

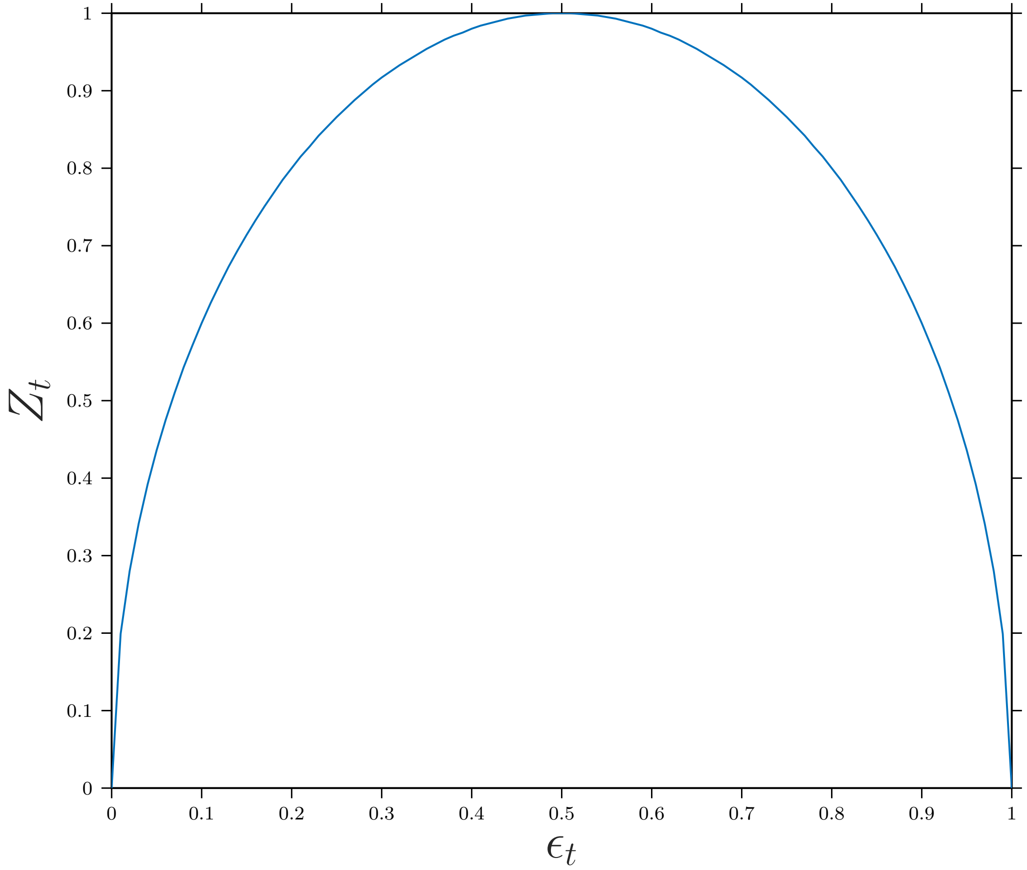
$$= e^{\log\sqrt{\frac{\epsilon_t}{1-\epsilon_t}}}(1-\epsilon_t) + e^{\log\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}\epsilon_t$$

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

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

Z_t

$$Z_t = \sum_{n=1}^{N} \omega_{t-1}^{(n)} e^{-\alpha_t y^{(n)} h_t(\boldsymbol{x}^{(n)})} = 2\sqrt{\epsilon_t(1-\epsilon_t)} < 1 \text{ if } \epsilon_t < \frac{1}{2}$$



Training Error

$$\frac{1}{N}\sum_{n=1}^{N}\mathbb{1}\left(y^{(n)} \neq g_T\left(\boldsymbol{x}^{(n)}\right)\right) \leq \frac{1}{N}\sum_{n=1}^{N} e^{\left(-y^{(n)}H_T\left(\boldsymbol{x}^{(n)}\right)\right)}$$

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

$$= \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)} \to 0$$

as $T \to \infty$

if $\epsilon_t < \frac{1}{2} \ \forall t$

True Error (Freund & Schapire, 1995)

- For AdaBoost, with high probability:

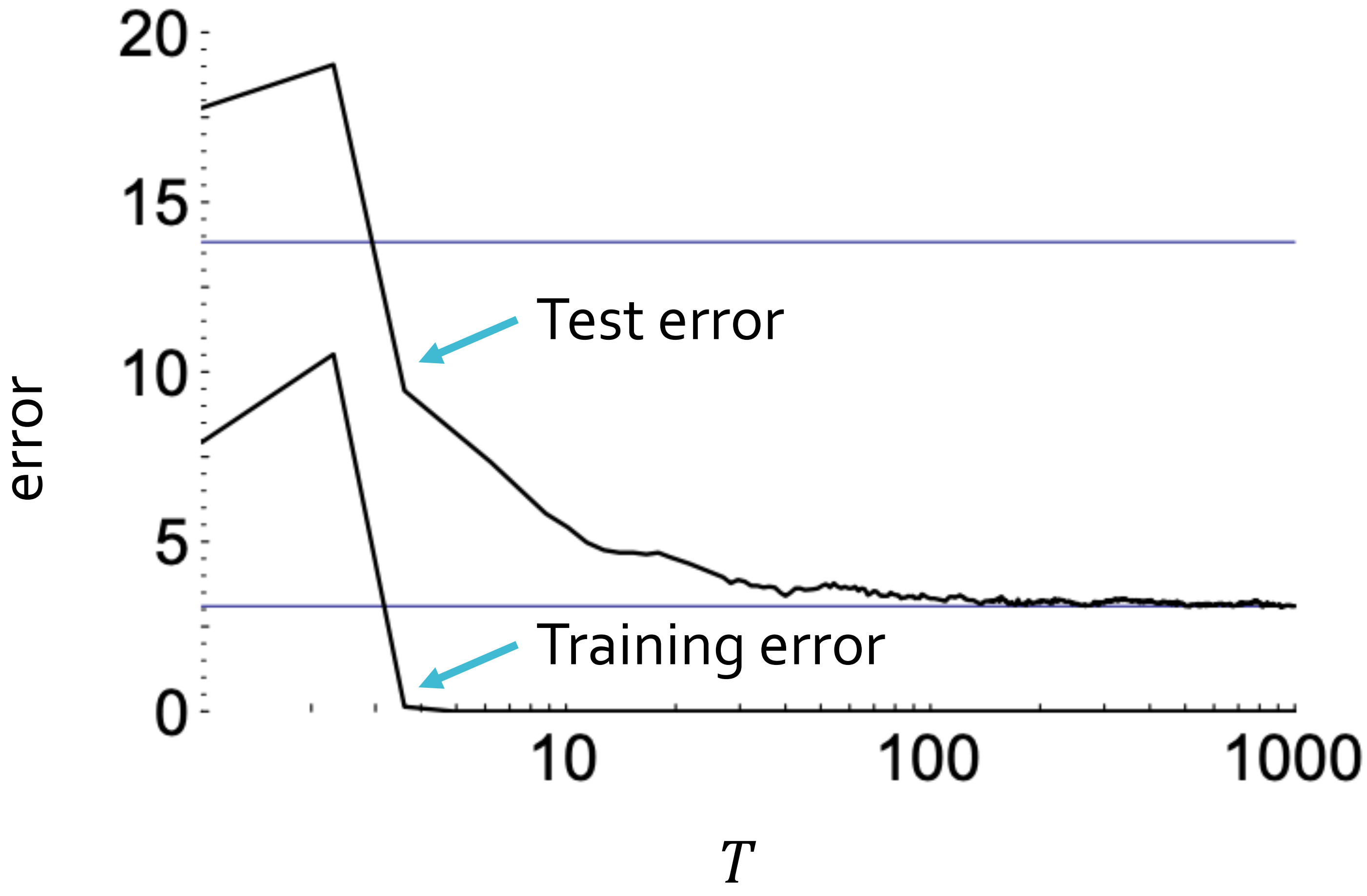
$$\text{True Error} \leq \text{Training Error} + \tilde{O}\left(\sqrt{\frac{d_{vc}(\mathcal{H})T}{N}}\right)$$

where $d_{vc}(\mathcal{H})$ is the VC-dimension of the weak learners and T is the number of weak learners.

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

Source: http://rob.schapire.net/papers/FreundSc95.pdf

Test Error (Schapire, 1989)



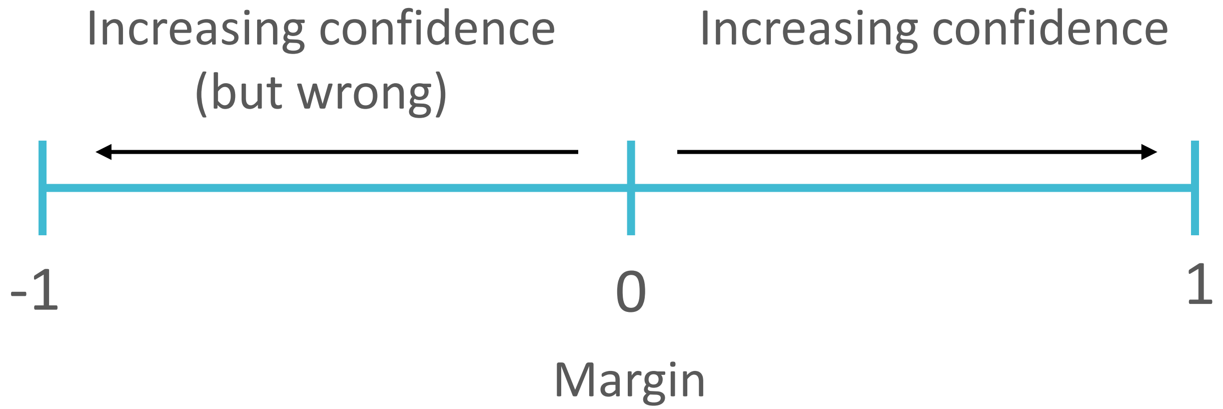
Source: http://rob.schapire.net/papers/msri.pdf

Margins

- The *margin* of training point $\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)$ is defined as:

$$m\left(\boldsymbol{x}^{(i)}, y^{(i)}\right) = \frac{y^{(i)} \sum_{t=1}^{T} \alpha_t h_t\left(\boldsymbol{x}^{(i)}\right)}{\sum_{t=1}^{T} \alpha_t}$$

- The margin can be interpreted as how confident g_T is in its prediction: the bigger the margin, the more confident.



True Error (Schapire, Freund et al., 1998)

- For AdaBoost, with high probability:

$$\text{True Error} \leq \frac{1}{N}\sum_{i=1}^{N}[\![m\left(\boldsymbol{x}^{(i)}, y^{(i)}\right) \leq \epsilon]\!] + \tilde{O}\left(\sqrt{\frac{d_{vc}(\mathcal{H})}{N\epsilon^2}}\right)$$

where $d_{vc}(\mathcal{H})$ is the VC-dimension of the weak learners and $\epsilon > 0$ is a tolerance parameter.

- Even after AdaBoost has driven the training error to 0, it continues to target the “training margin”

Source: http://rob.schapire.net/papers/SchapireFrBaLe98.pdf

Key Takeaways

- Boosting targets high bias models, i.e., weak learners
- Greedily minimizes the exponential loss, an upper bound of the classification error
- Theoretical (and empirical) results show resilience to overfitting by targeting training margin