10-301/601: Introduction to Machine Learning Lecture 28 – Boosting

Henry Chai

8/2/23

Front Matter

- Announcements
  - PA7 released 7/27, due 8/3 (tomorrow) at 11:59 PM
    - This is the last programming assignment!
  - Quiz 10: Ensemble Methods on 8/8
- Recommended Readings
  - Schapire, The Boosting Approach to Machine Learning: An Overview (2001)

Final Logistics

- Time and place:
  - Friday, 8/11 from 12 PM to 3 PM in POS 152 (here!)
- Closed book/notes
  - 1-page cheatsheet allowed, both back and front; can be typeset or handwritten

Final Coverage

- Lectures: 15 – 28 (through today's lecture)
  - Deep Learning
  - Learning Theory
  - Unsupervised Learning: Dimensionality Reduction, Clustering
  - Graphical Models: Naïve Bayes, Bayesian Networks, Hidden Markov Models
  - Reinforcement Learning
  - Ensemble Methods: Random Forests, Boosting
- The final is *not* cumulative: pre-midterm content may be referenced but will not be the primary focus of any question

Midterm Preparation

- Review final practice problems, posted to the course website (under Recitations)
- Attend the exam review recitation on 8/8 (after the quiz)
- Review this year's quizzes and study guides
- Consider whether you understand the "Key Takeaways" for each lecture / section
- Write your cheat sheet

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
  - High variance
    - Can be addressed via bagging → random forests
  - High bias (especially short trees, i.e., stumps)
    - Can be addressed via boosting

Boosting

- Another ensemble method (like bagging) that combines the predictions of multiple hypotheses.
- Aims to reduce the bias of a “weak” or highly biased model (can also reduce variance).

Ranking Classifiers (Caruana & Niculescu-Mizil, 2006)

Table 2. Normalized scores for each learning algorithm by metric (average over eleven problems)

MODEL	CAL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN	OPT-SEL
BST-DT	PLT	.843*	.779	**.939**	**.963**	**.938**	.929*	**.880**	**.896**	**.896**	**.917**
RF	PLT	.872*	.805	.934*	.957	.931	**.930**	.851	.858	.892	.898
BAG-DT	–	.846	.781	.938*	.962*	.937*	.918	.845	.872	.887*	.899
BST-DT	ISO	.826*	.860*	.929*	.952	.921	.925*	.854	.815	.885	.917*
RF	–	**.872**	.790	.934*	.957	.931	**.930**	.829	.830	.884	.890
BAG-DT	PLT	.841	.774	.938*	.962*	.937*	.918	.836	.852	.882	.895
RF	ISO	.861*	**.861**	.923	.946	.910	.925	.836	.776	.880	.895
BAG-DT	ISO	.826	.843*	.933*	.954	.921	.915	.832	.791	.877	.894
SVM	PLT	.824	.760	.895	.938	.898	.913	.831	.836	.862	.880
ANN	–	.803	.762	.910	.936	.892	.899	.811	.821	.854	.885
SVM	ISO	.813	.836*	.892	.925	.882	.911	.814	.744	.852	.882
ANN	PLT	.815	.748	.910	.936	.892	.899	.783	.785	.846	.875
ANN	ISO	.803	.836	.908	.924	.876	.891	.777	.718	.842	.884
BST-DT	–	.834*	.816	**.939**	**.963**	**.938**	.929*	.598	.605	.828	.851
KNN	PLT	.757	.707	.889	.918	.872	.872	.742	.764	.815	.837
KNN	–	.756	.728	.889	.918	.872	.872	.729	.718	.810	.830
KNN	ISO	.755	.758	.882	.907	.854	.869	.738	.706	.809	.844
BST-STMP	PLT	.724	.651	.876	.908	.853	.845	.716	.754	.791	.808
SVM	–	.817	.804	.895	.938	.899	.913	.514	.467	.781	.810
BST-STMP	ISO	.709	.744	.873	.899	.835	.840	.695	.646	.780	.810
BST-STMP	–	.741	.684	.876	.908	.853	.845	.394	.382	.710	.726
DT	ISO	.648	.654	.818	.838	.756	.778	.590	.589	.709	.774
DT	–	.647	.639	.824	.843	.762	.777	.562	.607	.708	.763
DT	PLT	.651	.618	.824	.843	.762	.777	.575	.594	.706	.761
LR	–	.636	.545	.823	.852	.743	.734	.620	.645	.700	.710
LR	ISO	.627	.567	.818	.847	.735	.742	.608	.589	.692	.703
LR	PLT	.630	.500	.823	.852	.743	.734	.593	.604	.685	.695
NB	ISO	.579	.468	.779	.820	.727	.733	.572	.555	.654	.661
NB	PLT	.576	.448	.780	.824	.738	.735	.537	.559	.650	.654
NB	–	.496	.562	.781	.825	.738	.738	.347	-.633	.481	.489

Source: https://www.cs.cornell.edu/~caruana/ctp/ct.papers/caruana.icml06.pdf

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

- Output: an aggregated hypothesis

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

Setting α_t

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

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

Intuition: we want good weak learners to have high importances

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

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
- Similar magnitude, different sign
- Different magnitude, same sign
- Different magnitude, different sign

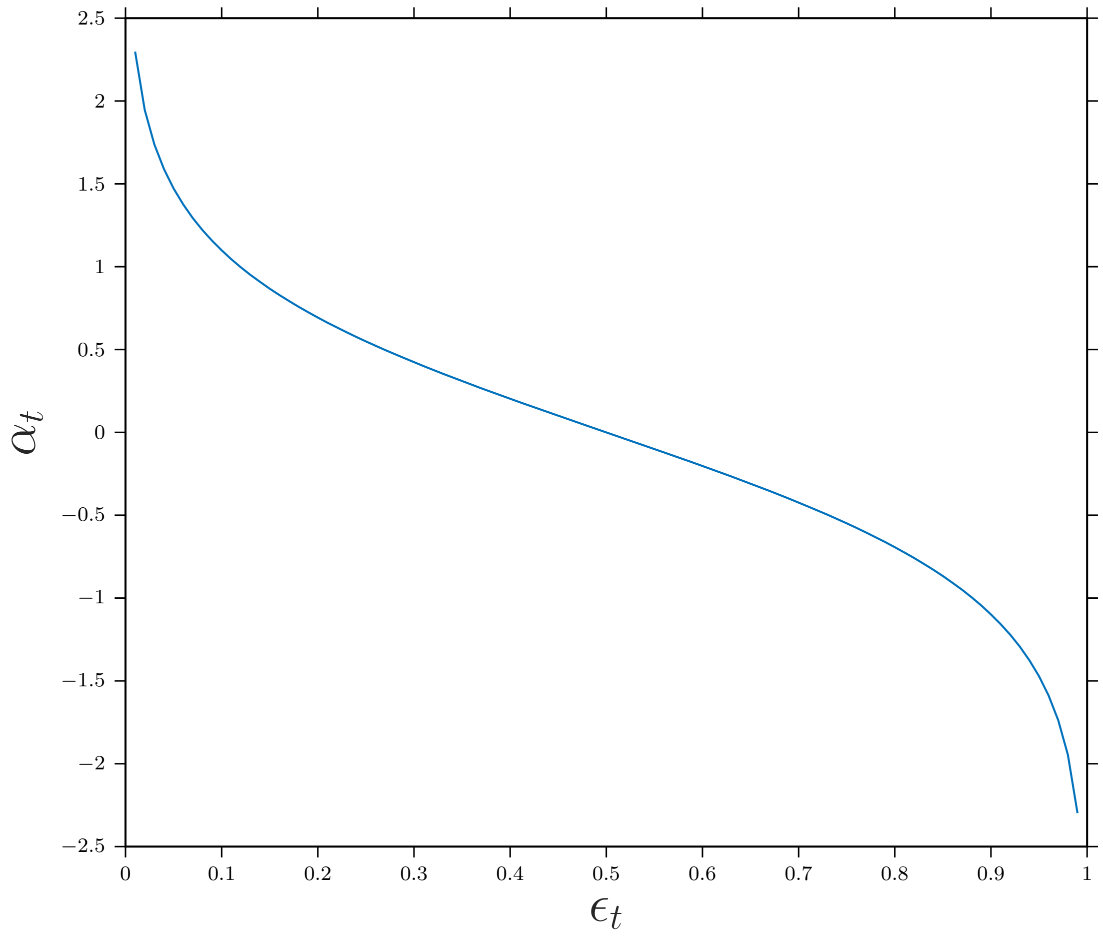
Setting α_t

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

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



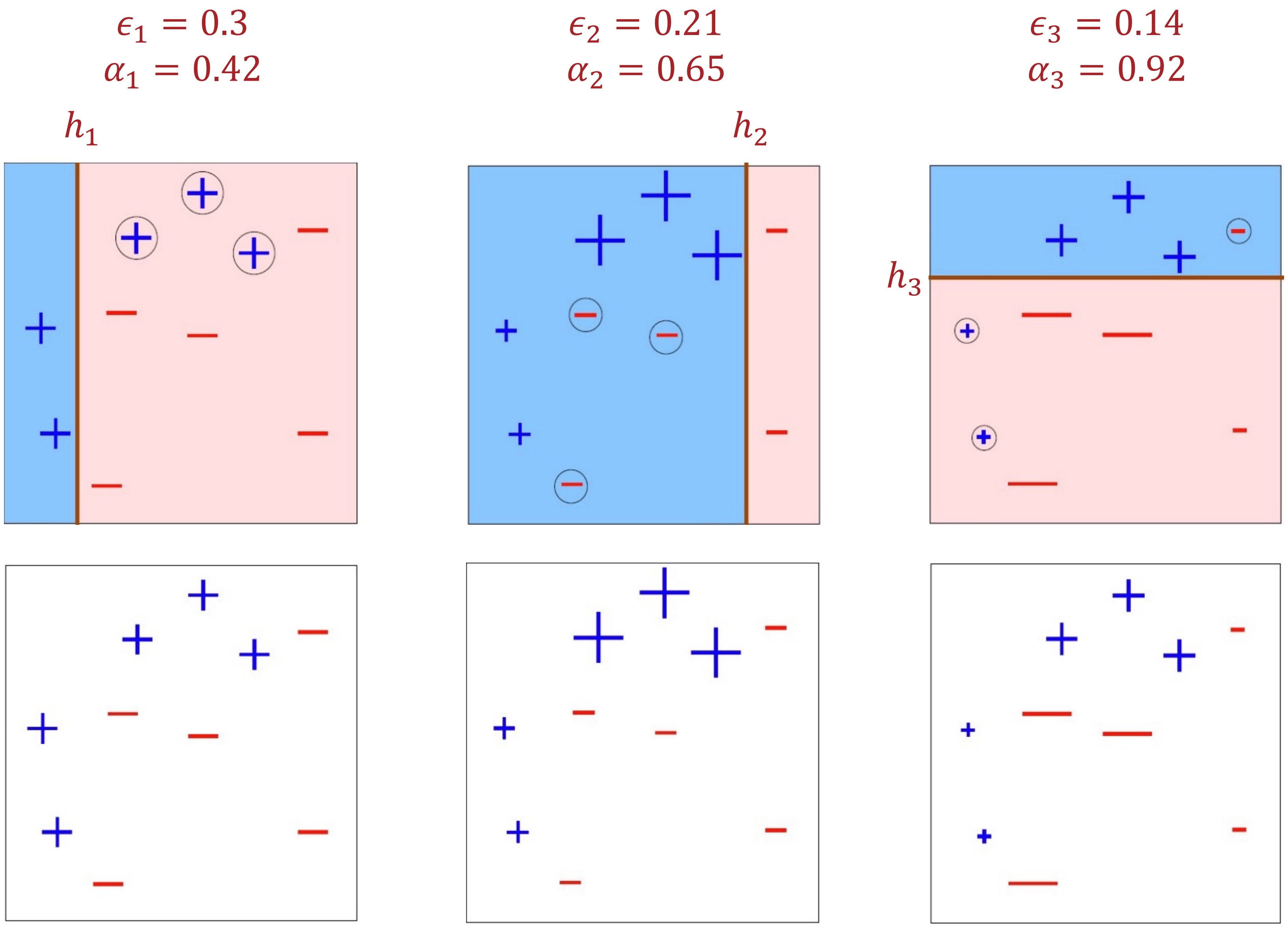
Updating $\omega^{(n)}$

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

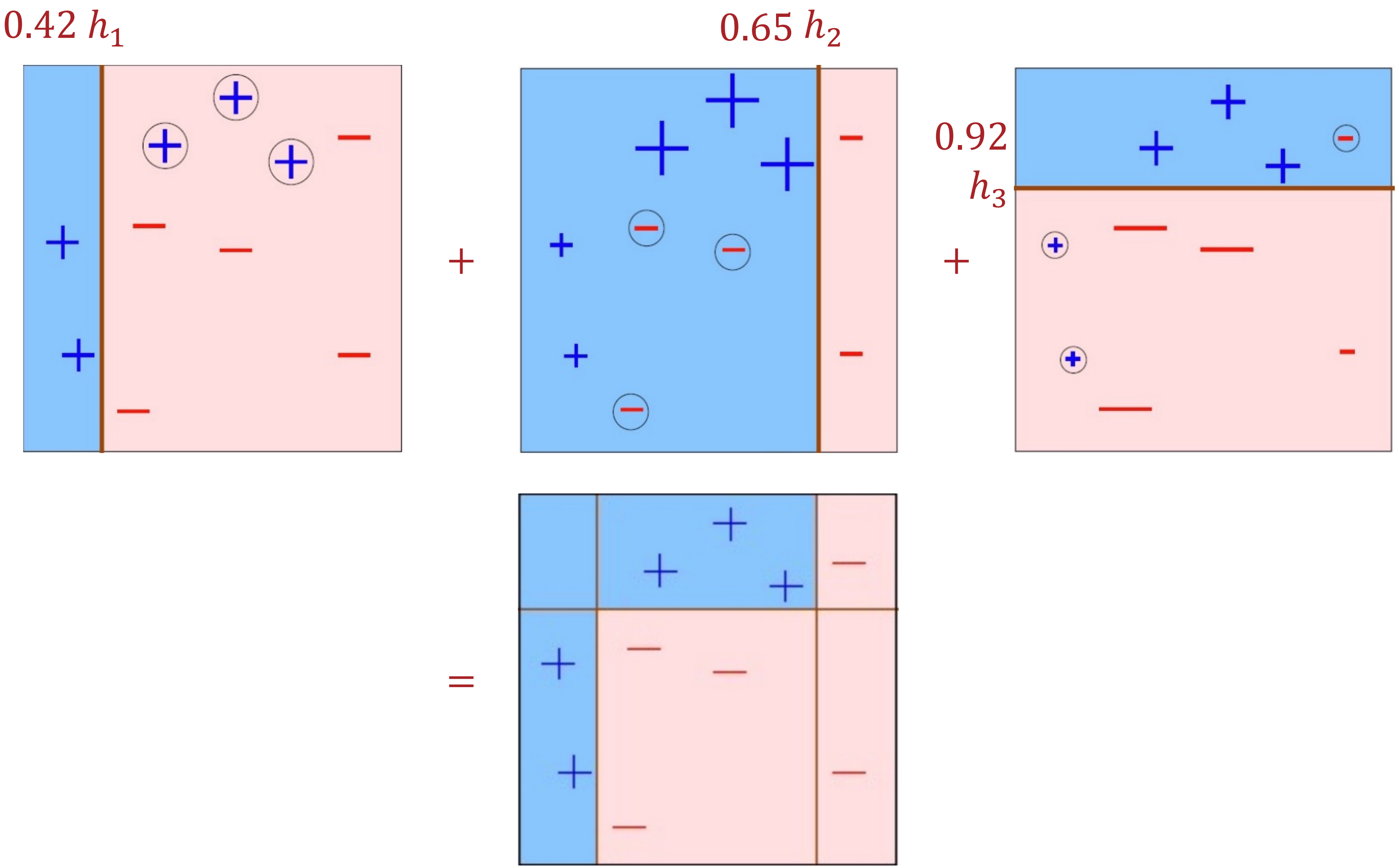
$$\omega_t^{(n)} = \frac{\omega_{t-1}^{(n)}}{Z_t} \times \begin{cases} e^{-\alpha_t} \text{ if } h_t\left(\boldsymbol{x}^{(n)}\right) = y^{(n)} \\ e^{\alpha_t} \quad \text{if } h_t\left(\boldsymbol{x}^{(n)}\right) \neq y^{(n)} \end{cases} = \frac{\omega_{t-1}^{(n)} e^{-\alpha_t y^{(n)} h_t\left(\boldsymbol{x}^{(n)}\right)}}{Z_t}$$

- If $\epsilon_t < \frac{1}{2}$, 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 $e^{\alpha_t} > 1$

AdaBoost: Example



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:

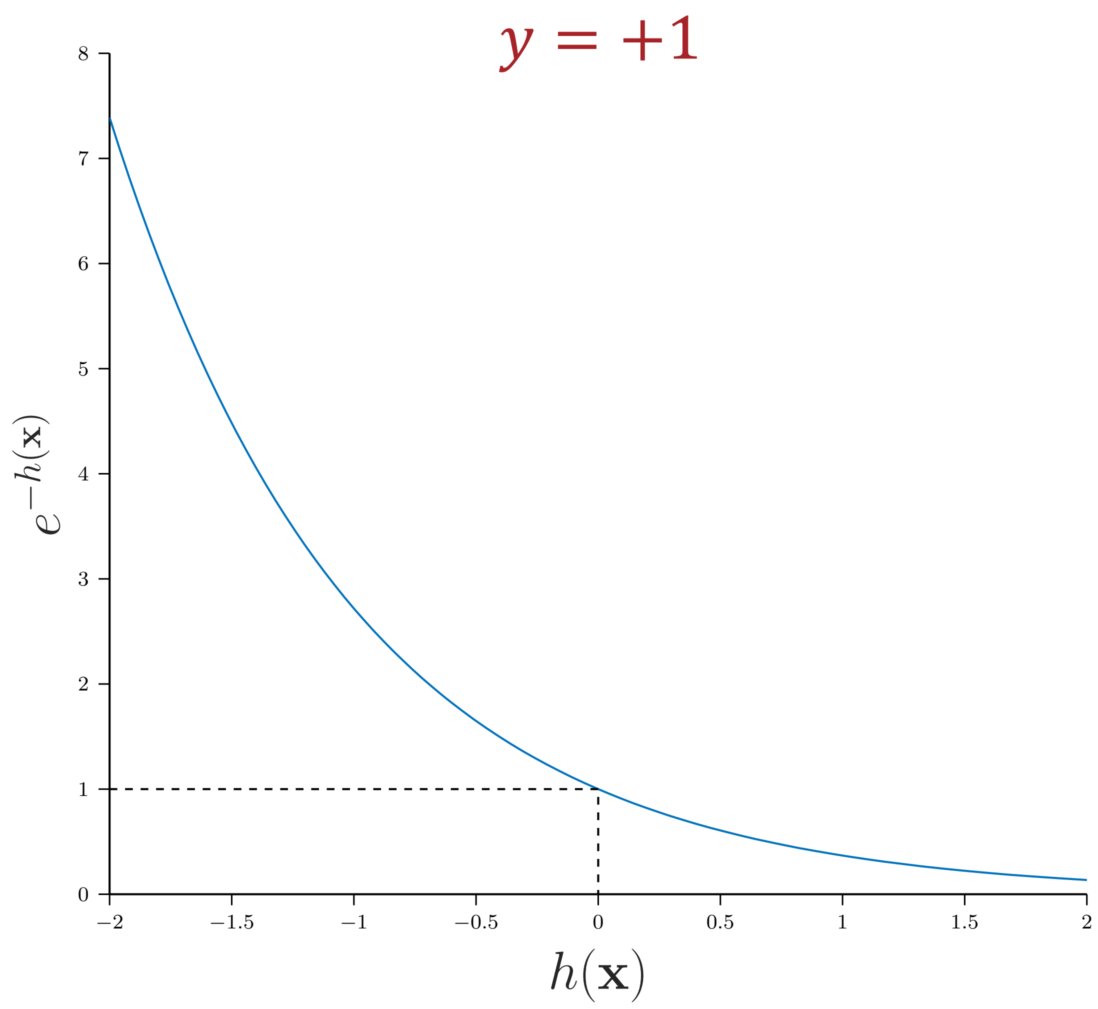
$$e(h(\boldsymbol{x}), y) = e^{(-yh(\boldsymbol{x}))}$$

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:

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

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

Exponential Loss

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

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

- Proof:

$$\omega_0^{(n)} = \frac{1}{N},\ \omega_1^{(n)} = \frac{e^{-\alpha_1 y^{(n)} h_1\left(\boldsymbol{x}^{(n)}\right)}}{N Z_1},\ \omega_2^{(n)} = \frac{e^{-\alpha_1 y^{(n)} h_1\left(\boldsymbol{x}^{(n)}\right)} e^{-\alpha_2 y^{(n)} h_2\left(\boldsymbol{x}^{(n)}\right)}}{N Z_1 Z_2}$$

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

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

Exponential Loss

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

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

- 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.

Greedy Exponential Loss Minimization

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

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

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

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

Greedy Exponential Loss Minimization

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

$$\frac{\partial Z_t}{\partial a} = -e^{-a}(1 - \epsilon_t) + e^{a}\epsilon_t \Longrightarrow -e^{-\hat{a}}(1 - \epsilon_t) + e^{\hat{a}}\epsilon_t = 0$$

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

$$\Longrightarrow e^{2\hat{a}} = \frac{1 - \epsilon_t}{\epsilon_t}$$

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

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

Normalizing $\omega^{(n)}$

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

$$= \sum_{y^{(n)} = h_t\left(\boldsymbol{x}^{(n)}\right)} \omega_{t-1}^{(n)} e^{-\alpha_t} + \sum_{y^{(n)} \neq h_t\left(\boldsymbol{x}^{(n)}\right)} \omega_{t-1}^{(n)} e^{\alpha_t}$$

$$= e^{-\alpha_t} \sum_{y^{(n)} = h_t\left(\boldsymbol{x}^{(n)}\right)} \omega_{t-1}^{(n)} + e^{\alpha_t} \sum_{y^{(n)} \neq h_t\left(\boldsymbol{x}^{(n)}\right)} \omega_{t-1}^{(n)}$$

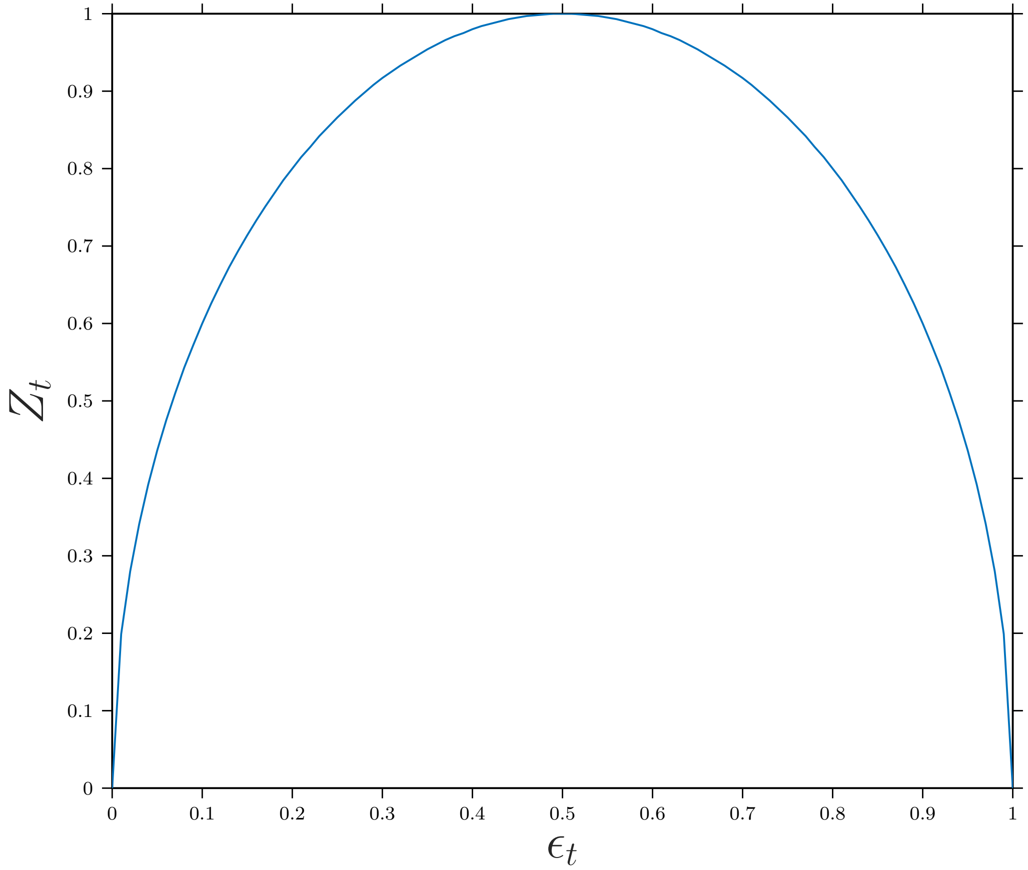
$$= e^{-\alpha_t}(1 - \epsilon_t) + e^{\alpha_t} \epsilon_t$$

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

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

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

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

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

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

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

True Error (Freund & Schapire, 1995)

- For AdaBoost, with high probability:

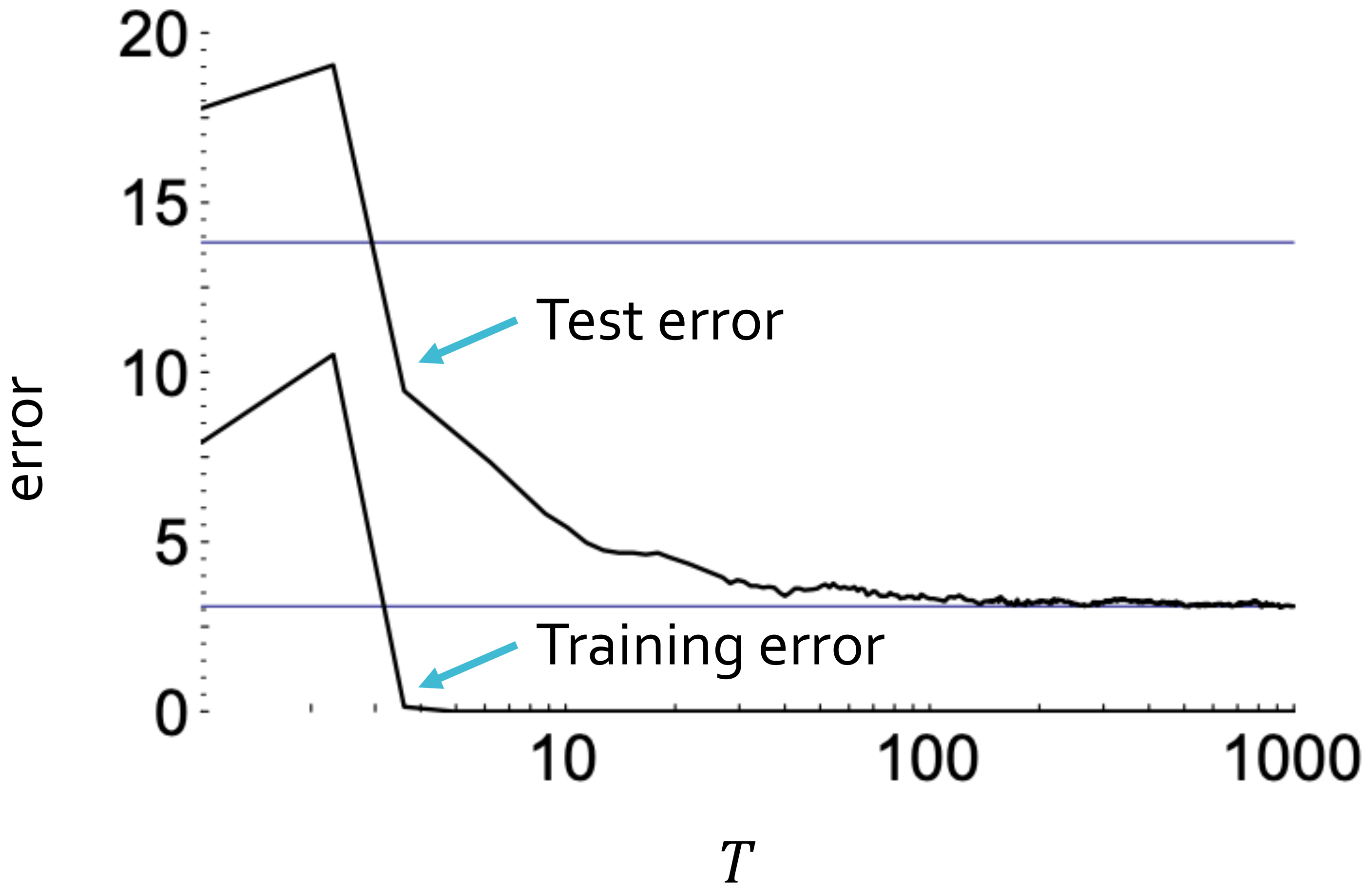
$$\text{True Error} \leq \text{Training Error} + \tilde{O}\left(\sqrt{\frac{d_{vc}(\mathcal{H})T}{N}}\right)$$

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)



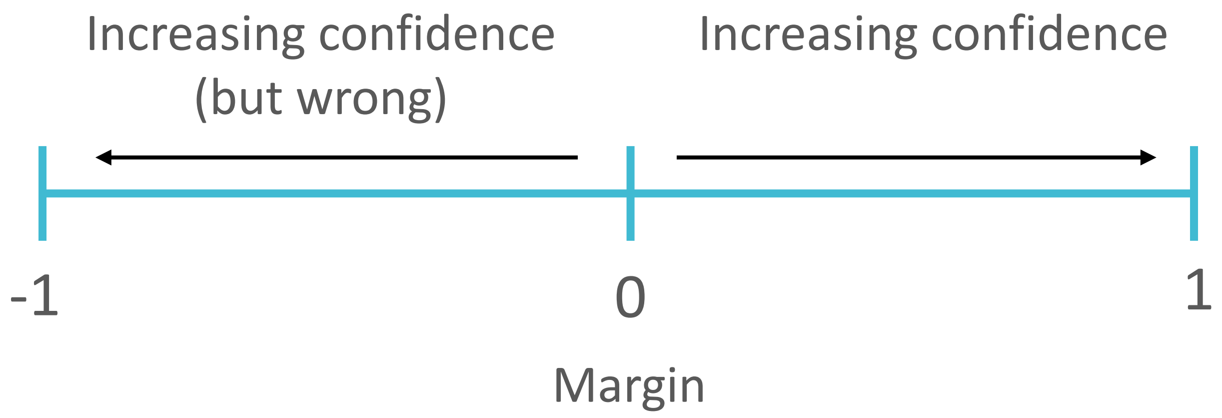
Source: http://rob.schapire.net/papers/msri.pdf

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:

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

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