# Ensemble Methods: Boosting

Nicholas Ruozzi
University of Texas at Dallas

Based on the slides of Vibhav Gogate and Rob Schapire

# Last Time

- Variance reduction via bagging
  - Generate "new" training data sets by sampling with replacement from the empirical distribution
  - Learn a classifier for each of the newly sampled sets
  - Combine the classifiers for prediction
- Today: how to reduce bias

# Boosting

- How to translate rules of thumb (i.e., good heuristics) into good learning algorithms

- For example, if we are trying to classify email as spam or not spam, a good rule of thumb may be that emails containing “Nigerian prince” or “Viagara” are likely to be spam most of the time

# Boosting

- Freund & Schapire
  - Theory for "weak learners" in late 80's
- Weak Learner: performance on ***any*** training set is slightly better than chance prediction
  - Intended to answer a theoretical question, not as a practical way to improve learning
  - Tested in mid 90's using not-so-weak learners
  - Works anyway!

# PAC Learning

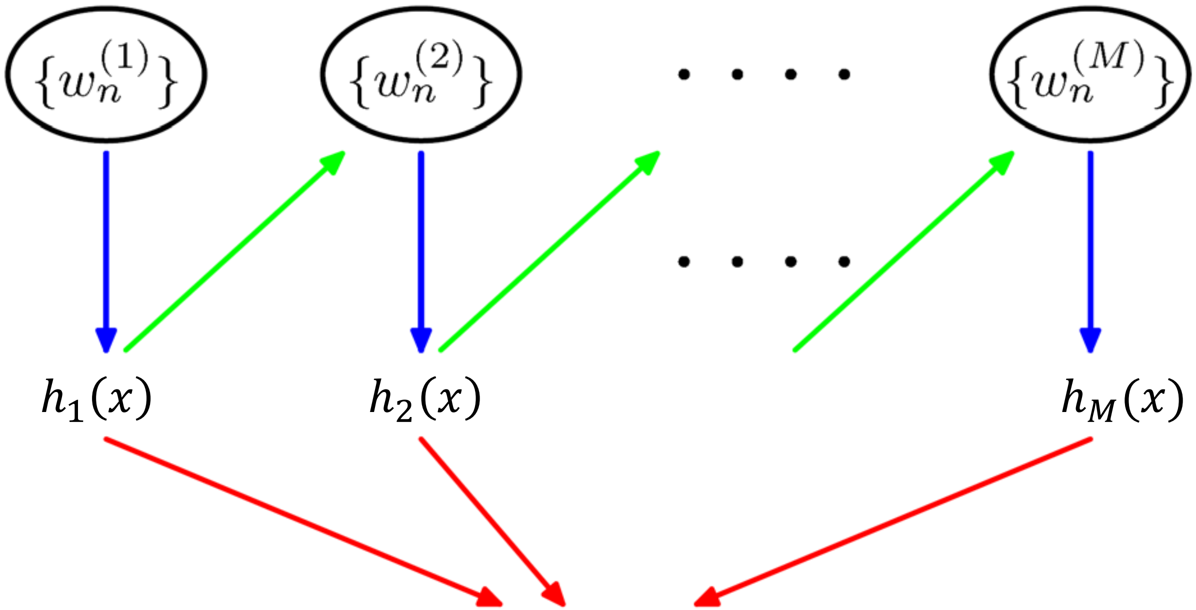
- Given i.i.d samples from an unknown, arbitrary distribution
  - “Strong” PAC learning algorithm
    - For any distribution with high probability given polynomially many samples (and polynomial time) can find classifier with arbitrarily small error
  - “Weak” PAC learning algorithm
    - Same, but error only needs to be slightly better than random guessing (e.g., accuracy only needs to exceed 50% for binary classification)
    - Does weak learnability imply strong learnability?

# Boosting

1. Weight all training samples equally
2. Train model on training set
3. Compute error of model on training set
4. Increase weights on training cases model gets wrong
5. Train new model on re-weighted training set
6. Re-compute errors on weighted training set
7. Increase weights again on cases model gets wrong

- Repeat until tired (100+ iterations)
- Final model: weighted prediction of each model

# Boosting: Graphical Illustration




$$h(x) = \text{sign}\left(\sum_m \alpha_m h_m(x)\right)$$

# AdaBoost

1. Initialize the data weights $w_1, \dots, w_N$ for the first round as $w_1^{(1)}, \dots, w_N^{(1)} = \frac{1}{N}$
2. For $m = 1, \dots, M$
   a) Select a classifier $h_m$ for the $m^{th}$ round by minimizing the weighted error

   $$\sum_i w_i^{(m)} 1_{h_m(x^{(i)}) \neq y_i}$$

   b) Compute

   $$\epsilon_m = \sum_i w_i^{(m)} 1_{h_m(x^{(i)}) \neq y_i}$$

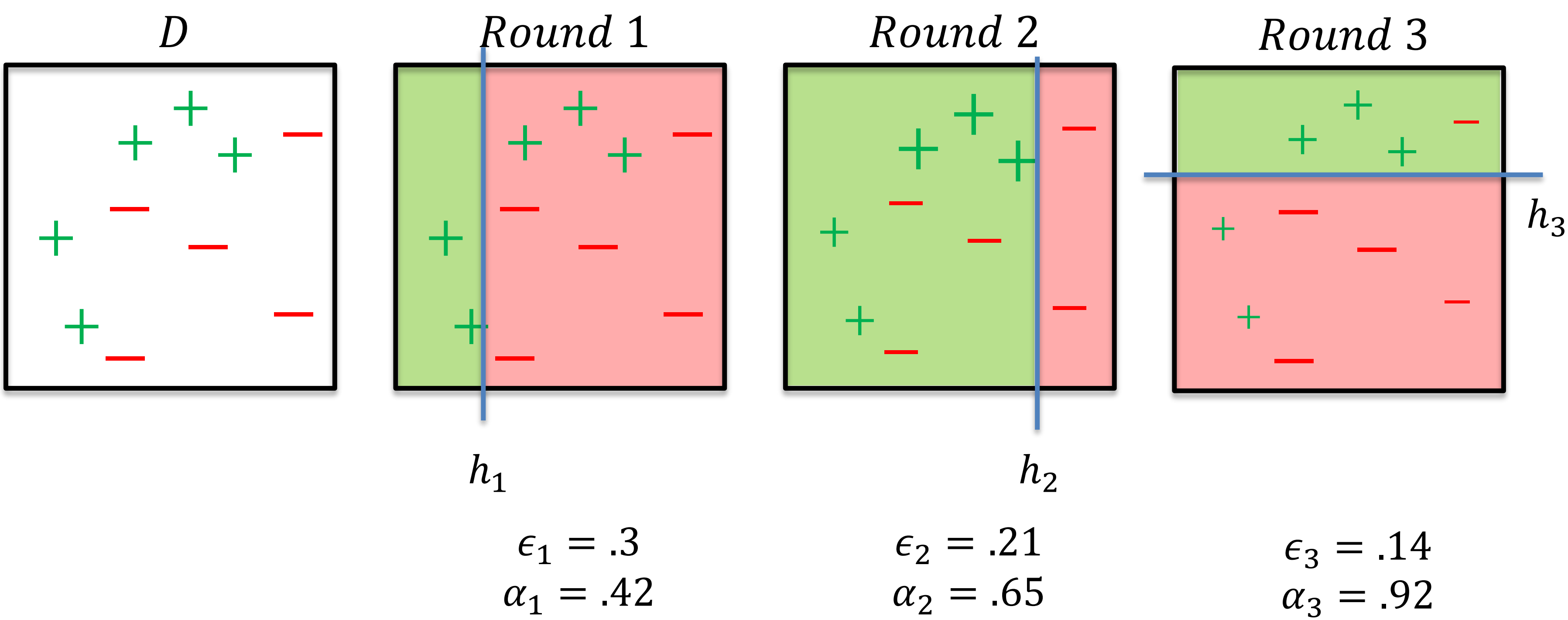
   $$\alpha_m = \frac{1}{2} \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$$

   c) Update the weights

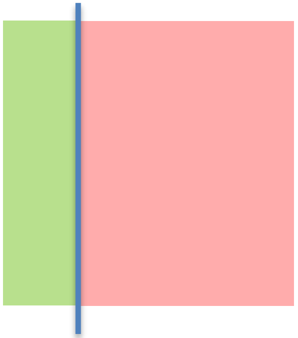
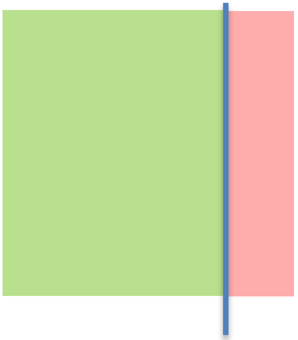
   $$w_i^{(m+1)} = \frac{w_i^{(m)} \exp(-y_i h_m(x_i)\alpha_m)}{2\sqrt{\epsilon_m \cdot (1-\epsilon_m)}}$$

# Example

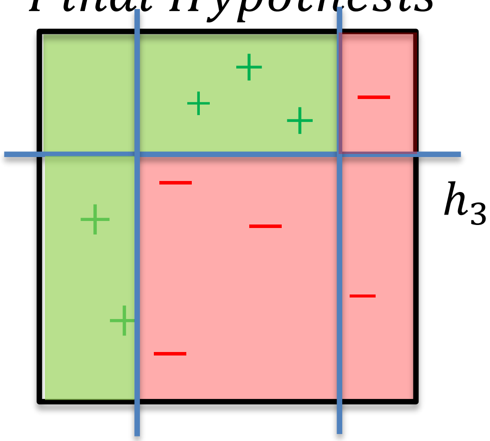
- Consider a classification problem where vertical and horizontal lines (and their corresponding half spaces) are the weak learners



# Final Hypothesis

$h(x) = sign$ [ .42  +.65  +.92  ]

*Final Hypothesis*



# Boosting

**Theorem:** Let $Z_m = 2\sqrt{\epsilon_m \cdot (1-\epsilon_m)}$ and $\gamma_m = \frac{1}{2} - \epsilon_m$.

$$\frac{1}{N}\sum_i 1_{h(x^{(i)}) \neq y_i} \leq \prod_m Z_m = \prod_m \sqrt{1 - 4\gamma_m^2}$$

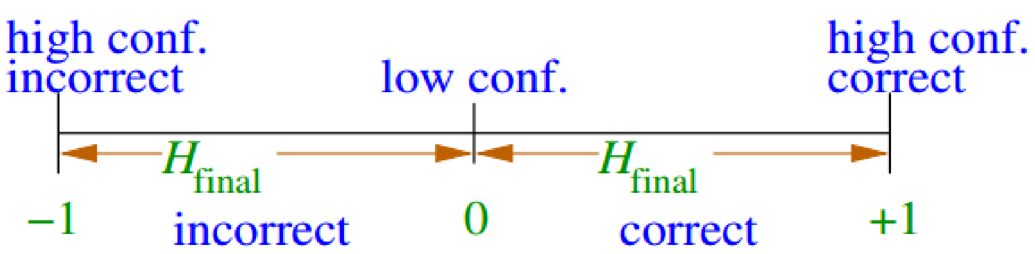
So, even if all of the $\gamma$'s are small positive numbers (i.e., every learner is a weak learner), the training error goes to zero as $M$ increases

# Margins & Boosting

- We can see that training error goes down, but what about test error?
  - That is, does boosting help us generalize better?
- To answer this question, we need to look at how confident we are in our predictions
  - How can we measure this?

# Margins & Boosting

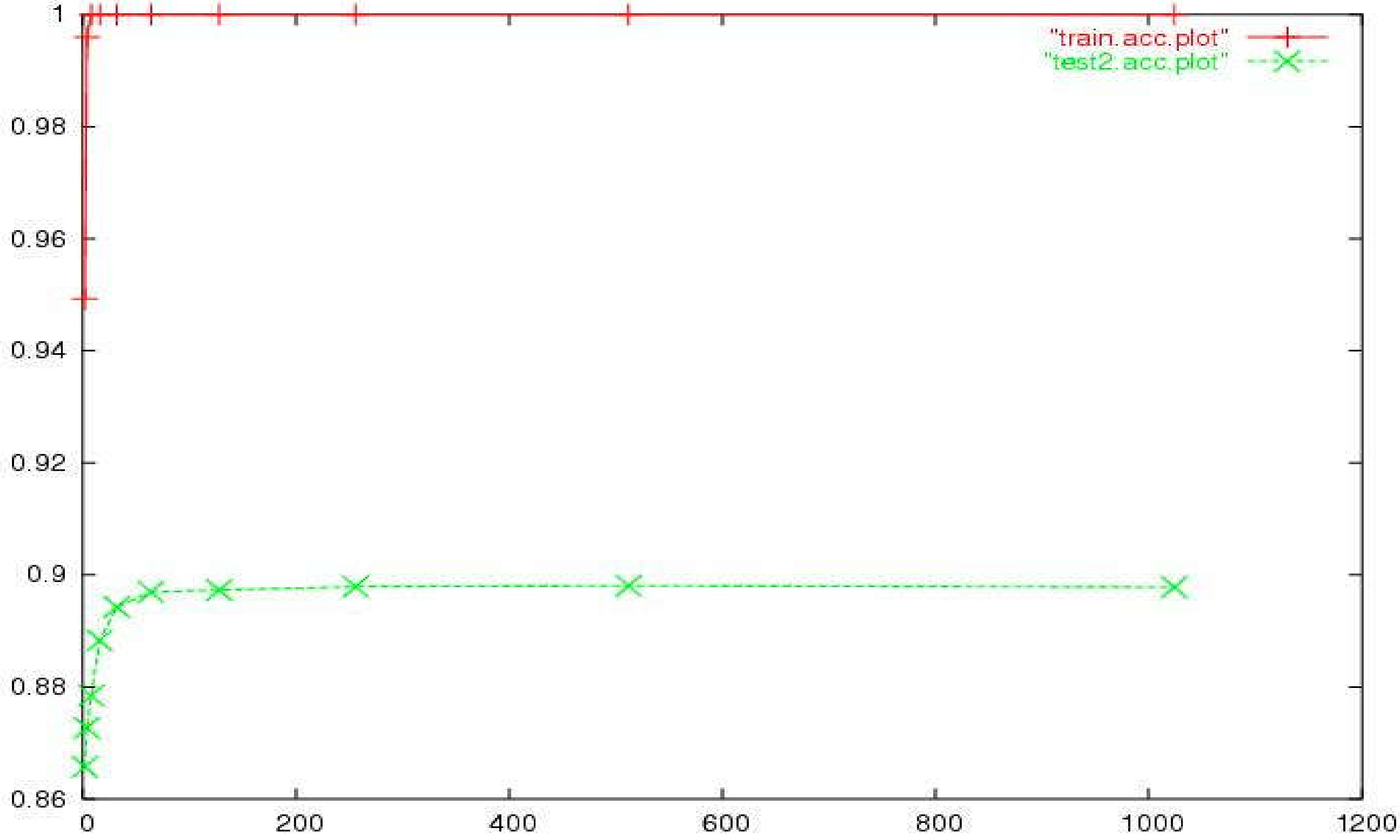
- We can see that training error goes down, but what about test error?
  - That is, does boosting help us generalize better?
- To answer this question, we need to look at how confident we are in our predictions
  - Margins!



# Margins & Boosting

- Intuition: larger margins lead to better generalization (same as SVMs)
- Theorem: with high probability, boosting increases the size of the margins
  - Note: boosting does NOT maximize the margin, so it can still have poor generalization performance

# Boosting Performance



# Boosting as Optimization

- AdaBoost can actually be interpreted as a coordinate descent method for a specific loss function!
- Let $\{h_1, \dots, h_T\}$ be the set of all weak learners
- Exponential loss

$$\ell(\alpha_1, \dots, \alpha_T) = \sum_i \exp\left(-y_i \cdot \sum_t \alpha_t h_t(x^{(i)})\right)$$

  – Convex in $\alpha_t$

  – AdaBoost minimizes this exponential loss

# Coordinate Descent

- Minimize the loss with respect to a single component of $\alpha$, let's pick $\alpha_{t'}$

$$\frac{d\ell}{d\alpha_{t'}} = -\sum_i y_i h_{t'}\left(x^{(i)}\right) \exp\left(-y_i \cdot \sum_t \alpha_t h_t\left(x^{(i)}\right)\right)$$

$$= \sum_{i: h_{t'}\left(x^{(i)}\right) = y_i} -\exp(-\alpha_{t'}) \exp\left(-y_i \cdot \sum_{t \neq t'} \alpha_t h_t\left(x^{(i)}\right)\right)$$

$$+ \sum_{i: h_{t'}\left(x^{(i)}\right) \neq y_i} \exp(\alpha_{t'}) \exp\left(-y_i \cdot \sum_{t \neq t'} \alpha_t h_t\left(x^{(i)}\right)\right)$$

$$= 0$$

# Coordinate Descent

- Solving for $\alpha_{t'}$

$$\alpha_{t'} = \frac{1}{2}\ln\frac{\sum_{i:h_{t'}(x^{(i)})=y_i}\exp\left(-y_i\cdot\sum_{t\neq t'}\alpha_t h_t\left(x^{(i)}\right)\right)}{\sum_{i:h_{t'}(x^{(i)})\neq y_i}\exp\left(-y_i\cdot\sum_{t\neq t'}\alpha_t h_t\left(x^{(i)}\right)\right)}$$

- This is similar to the adaBoost update!
  - The only difference is that adaBoost tells us in which order we should update the variables

# Coordinate Descent

- Start with $\alpha^{(1)} = 0$
- Let $r_i = \exp\left(-y_i \cdot \sum_{t \neq t'} \alpha_t h_t\left(x^{(i)}\right)\right) = 1$
- Choose $t'$ to minimize

$$\sum_{i:h_{t'}\left(x^{(i)}\right) \neq y_i} r_i = N \sum_i w_i^{(1)} 1_{h_{t'}\left(x^{(i)}\right) \neq y_i}$$

- For this choice of $t'$, minimize the objective with respect to $\alpha_{t'}$ gives

$$\alpha_{t'} = \frac{1}{2} \ln \frac{N \sum_i w_i^{(1)} 1_{h_{t'}\left(x^{(i)}\right) = y_i}}{N \sum_i w_i^{(1)} 1_{h_{t'}\left(x^{(i)}\right) \neq y_i}} = \frac{1}{2} \ln\left(\frac{1 - \epsilon_1}{\epsilon_1}\right)$$

- Repeating this procedure yields adaBoost

# adaBoost as Optimization

- Could derive an adaBoost algorithm for other types of loss functions!
- Important to note
  - Exponential loss is convex, but may have multiple global optima
  - In practice, adaBoost can perform quite differently than other methods for minimizing this loss (e.g., gradient descent)

# Summary: Boosting & Bagging

- Bagging doesn't work so well with stable models. Boosting might still help
- Boosting might hurt performance on noisy datasets
  - Bagging doesn't have this problem
- On average, boosting helps more than bagging, but it is also more common for boosting to hurt performance.
- Bagging is easier to parallelize

# Other Approaches

- Mixture of Experts (See Bishop, Chapter 14)
- Cascading Classifiers
- many others…