# 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 for binary classification problems

# 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 “business opportunity” 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?**

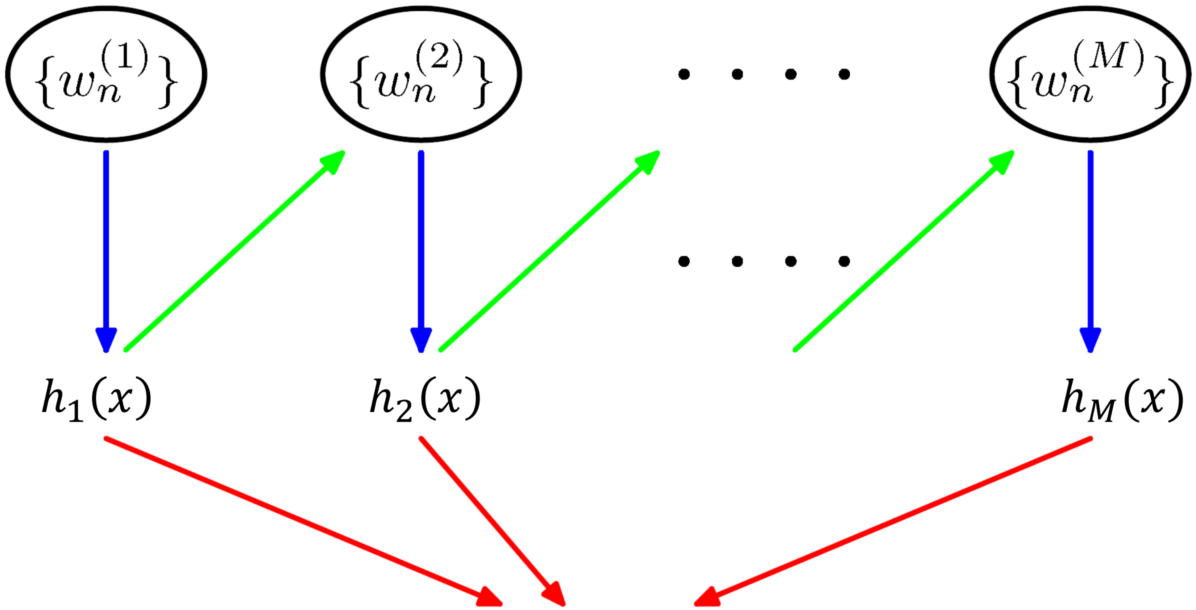
# Boosting

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

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 for the first round as $w_1^{(1)}, \dots, w_1^{(M)} = \frac{1}{M}$
2. For $t = 1, \dots, T$
   a) Select a classifier $h_t$ for the $T^{th}$ round by minimizing the weighted error

$$\epsilon_t = \sum_m w_t^{(m)} 1_{h_t(x^{(m)}) \neq y^{(m)}}$$

   b) Compute

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

   c) Update the weights

$$w_{t+1}^{(m)} = \frac{w_t^{(m)} \exp\left(-y^{(m)} h_t\left(x^{(m)}\right) \alpha_t\right)}{2\sqrt{\epsilon_t \cdot (1 - \epsilon_t)}}$$

# AdaBoost

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

$$\epsilon_t = \sum_m w_t^{(m)} 1_{h_t(x^{(m)}) \neq y^{(m)}}$$

   b) Compute

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

Weighted number of incorrect classifications of the $t^{th}$ classifier

   c) Update the weights

$$w_{t+1}^{(m)} = \frac{w_t^{(m)} \exp\left(-y^{(m)} h_t\left(x^{(m)}\right) \alpha_t\right)}{2\sqrt{\epsilon_t \cdot (1-\epsilon_t)}}$$

# AdaBoost

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

$$\epsilon_t = \sum_m w_t^{(m)} 1_{h_t(x^{(m)}) \neq y^{(m)}}$$

   b) Compute

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

$\epsilon_t \to 0$

$\alpha_t \to \infty$

   c) Update the weights

$$w_{t+1}^{(m)} = \frac{w_t^{(m)} \exp\left(-y^{(m)} h_t\left(x^{(m)}\right)\alpha_t\right)}{2\sqrt{\epsilon_t \cdot (1 - \epsilon_t)}}$$

# AdaBoost

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

$$\epsilon_t = \sum_m w_t^{(m)} 1_{h_t(x^{(m)}) \neq y^{(m)}}$$

   b) Compute

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

$\epsilon_t \to .5$

$\alpha_t \to 0$

   c) Update the weights

$$w_{t+1}^{(m)} = \frac{w_t^{(m)} \exp\left(-y^{(m)} h_t\left(x^{(m)}\right)\alpha_t\right)}{2\sqrt{\epsilon_t \cdot (1 - \epsilon_t)}}$$

# AdaBoost

1. Initialize the data weights for the first round as $w_1^{(1)}, \dots, w_1^{(M)} = \frac{1}{M}$
2. For $t = 1, \dots, T$
   a) Select a classifier $h_t$ for the $T^{th}$ round by minimizing the weighted error
   $$\epsilon_t = \sum_m w_t^{(m)} 1_{h_t(x^{(m)}) \neq y^{(m)}}$$
   b) Compute
   $$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$
   $\epsilon_t \to 1$
   $\alpha_t \to -\infty$
   c) Update the weights
   $$w_{t+1}^{(m)} = \frac{w_t^{(m)} \exp\left(-y^{(m)} h_t\left(x^{(m)}\right)\alpha_t\right)}{2\sqrt{\epsilon_t \cdot (1-\epsilon_t)}}$$

# AdaBoost

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

$$\epsilon_t = \sum_m w_t^{(m)} 1_{h_t(x^{(m)}) \neq y^{(m)}}$$

   b) Compute

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

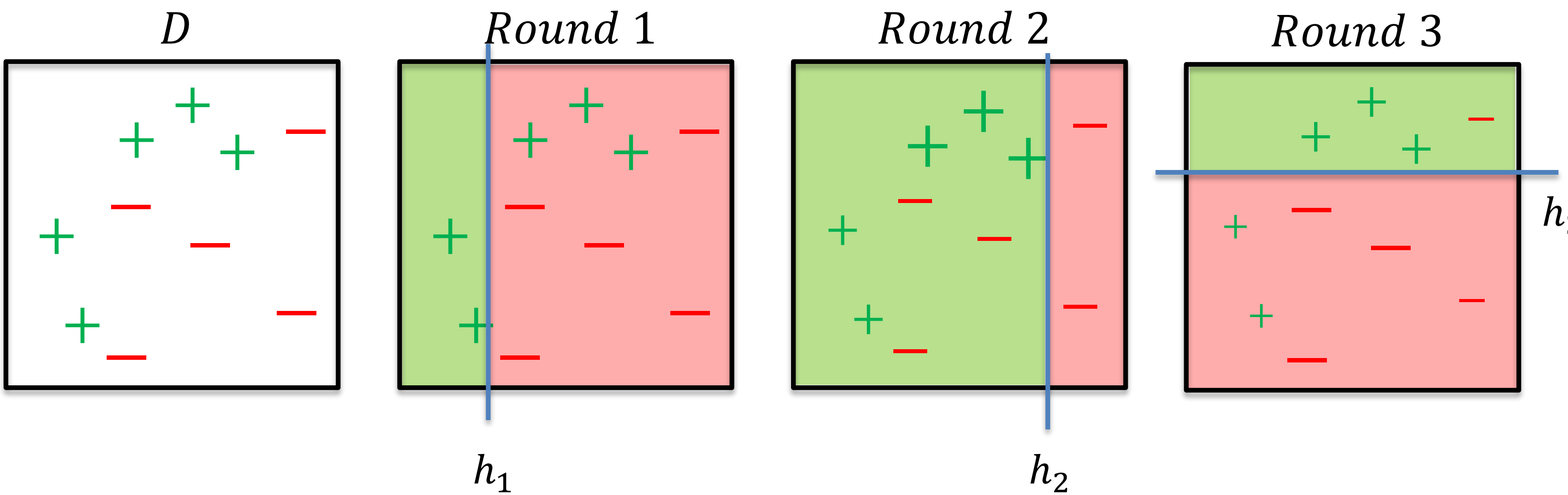
   c) Update the weights

$$w_{t+1}^{(m)} = \frac{w_t^{(m)} \exp\left(-y^{(m)} h_t\left(x^{(m)}\right)\alpha_t\right)}{2\sqrt{\epsilon_t \cdot (1-\epsilon_t)}}$$

Normalization constant

# Example

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

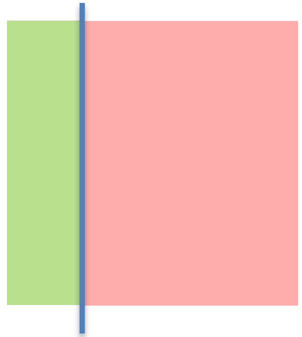
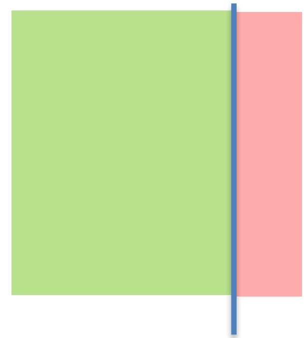
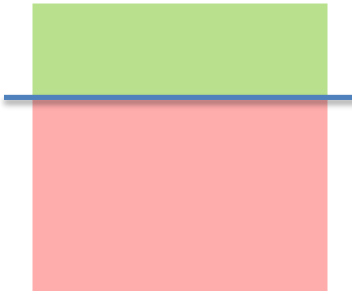



$\epsilon_1 = .3$, $\alpha_1 = .42$

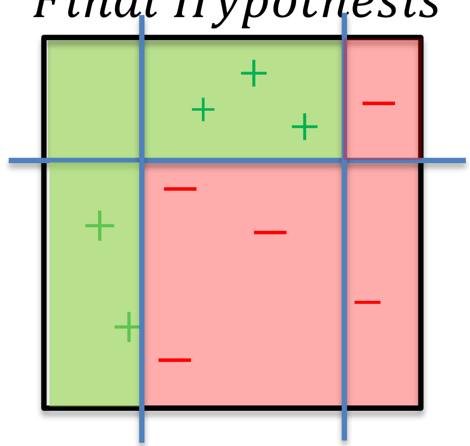
$\epsilon_2 = .21$, $\alpha_2 = .65$

$\epsilon_3 = .14$, $\alpha_3 = .92$

# Final Hypothesis

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

*Final Hypothesis*



# Boosting

**Theorem:** Let $Z_t = 2\sqrt{\epsilon_t \cdot (1 - \epsilon_t)}$ and $\gamma_t = \frac{1}{2} - \epsilon_t$.

$$\frac{1}{M} \sum_m 1_{h(x^{(m)}) \neq y^{(m)}} \leq \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2}$$

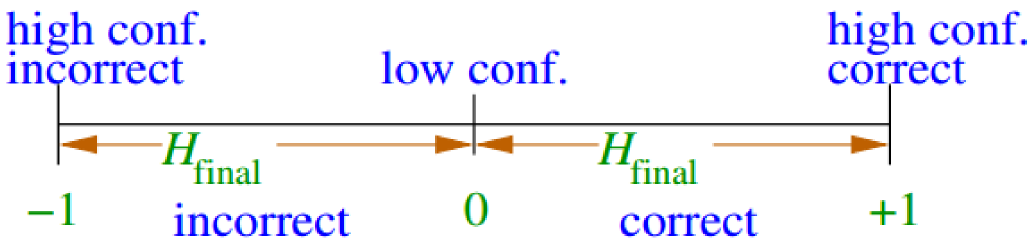
So, even if all of the $\gamma$'s are small positive numbers (i.e., can always find a weak learner), the training error goes to zero as $T$ increases

# 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

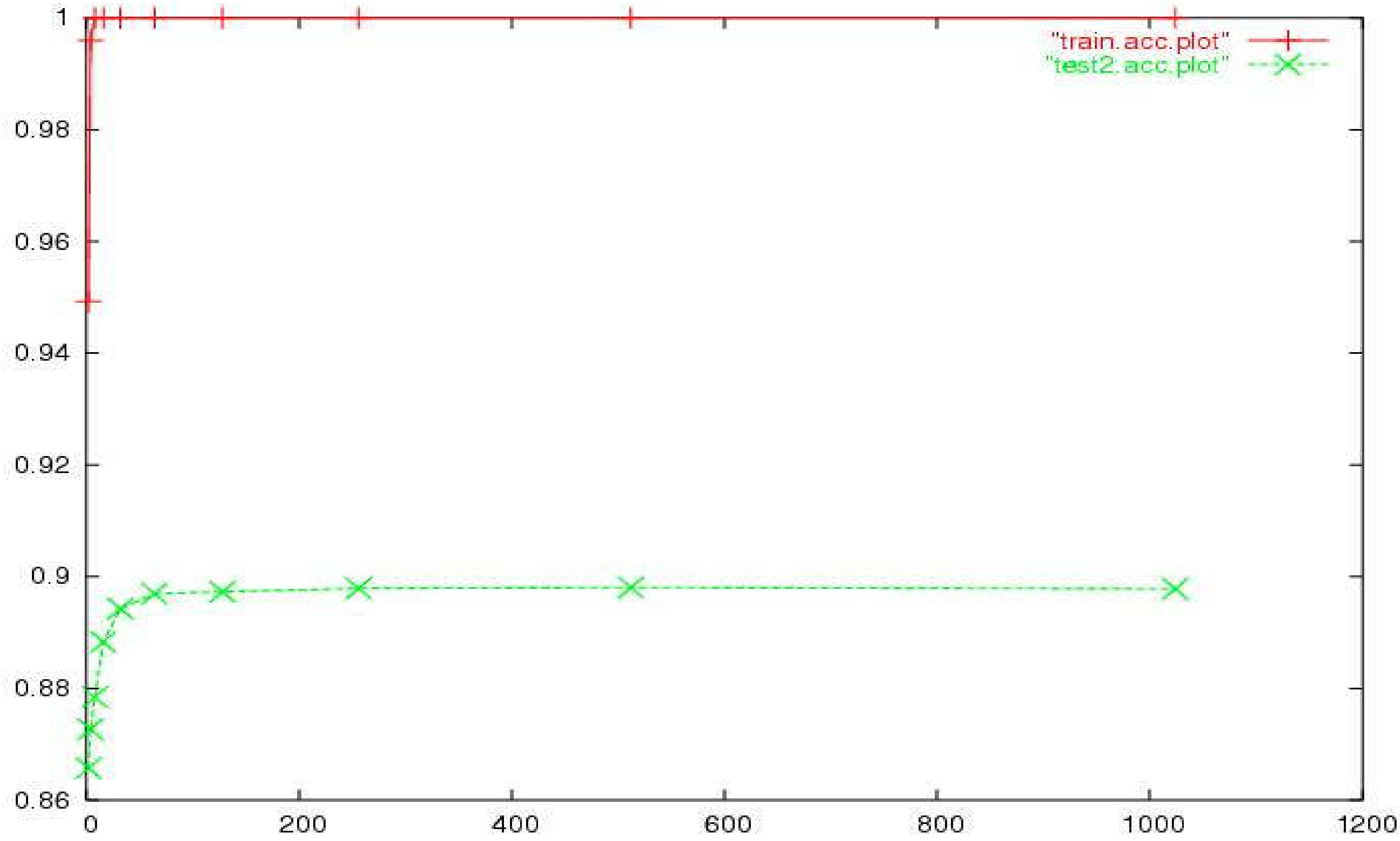
- We can see that training error goes down, but what about test error?
  - That is, does boosting help us generalize better?
- To answer this question, we need to look at how confident we are in our predictions
  - Margins!



# 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 could still result in poor generalization performance

# Boosting Performance



# Boosting as Optimization

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

$$\ell(\alpha_1, \dots, \alpha_T) = \sum_m \exp\left(-y^{(m)} \cdot \sum_t \alpha_t h_t(x^{(m)})\right)$$

  - 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 $\alpha_{t'}$

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

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

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

$$= 0$$

# Coordinate Descent

- Solving for $\alpha_{t'}$

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

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

# AdaBoost as Optimization

- Could derive an adaBoost algorithm for other types of loss functions!
- Important to note
  - Exponential loss is convex, but not strict (may have multiple global optima)
  - In practice, adaBoost can perform quite differently than other methods for minimizing this loss (e.g., gradient descent)

# Boosting in Practice

- Our description of the algorithm assumed that a set of possible hypotheses was given
  - In practice, the set of hypotheses can be built as the algorithm progress
- Example: build new decision tree at each iteration for the data set in which the $m^{th}$ example has weight $w_m^{(t)}$
  - When computing information gain, compute the empirical probabilities using the weights

# Boosting vs. Bagging

- Bagging doesn't work well with stable models
  - Boosting might still help
- Boosting might hurt performance on noisy datasets
  - Bagging doesn't have this problem
- On average, boosting improves classification accuracy more than bagging, but it is also more common for boosting to hurt performance
- Bagging is easier to parallelize
- Both ensemble methods have added overhead required to train multiple classifiers

# Boosting Beyond Binary Classification

- Slightly more complicated
  - Want to select weak learners that are better than random guessing, but there are many different ways to do better than random
  - A hypothesis space is boostable if there exists a baseline measure, that is slightly better than random, such that you can always find a hypothesis that outperforms the baseline
  - Can be boostable with respect to some baselines but not others