ITCS 6156/8156 Fall 2024
Machine Learning

Ensemble Methods

Instructor: Hongfei Xue
Email: hongfei.xue@charlotte.edu
Class Meeting: Tue & Thu, 4:00 PM – 5:15 PM, WWH 130

UNIVERSITY OF NORTH CAROLINA CHARLOTTE

Some content in the slides is based on Dr. Raquel Urtasun's lecture

Ensemble Methods

- Typical application: classification
- Ensemble of classifiers is a set of classifiers whose individual decisions are combined in some way to classify new examples
- Simplest approach:
  1. Generate multiple classifiers
  2. Each votes on test instance
  3. Take majority as classification
- Classifiers are different due to different sampling of training data, or randomized parameters within the classification algorithm
- Aim: take simple mediocre algorithm and transform it into a super classifier without requiring any fancy new algorithm

Ensemble Methods

- Differ in training strategy, and combination method
  - Parallel training with different training sets
    1. Bagging (bootstrap aggregation) – train separate models on overlapping training sets, average their predictions
  - Sequential training, iteratively re-weighting training examples so current classifier focuses on hard examples: boosting
  - Parallel training with objective encouraging division of labor: mixture of experts
- Notes:
  - Also known as meta-learning
  - Typically applied to weak models, such as decision stumps (single-node decision trees), or linear classifiers

Variance-bias Tradeoff

- Minimize two sets of errors:
  1. Variance: error from sensitivity to small fluctuations in the training set
  2. Bias: erroneous assumptions in the model
- Variance-bias decomposition is a way of analyzing the generalization error as a sum of 3 terms: variance, bias and irreducible error (resulting from the problem itself)
- Based on one of two basic observations:
  1. Variance reduction: if the training sets are completely independent, it will always help to average an ensemble because this will reduce variance without affecting bias (e.g., bagging)
     - reduce sensitivity to individual data points
  2. Bias reduction: for simple models, average of models has much greater capacity than single model (e.g., hyperplane classifiers, Gaussian densities).
     - Averaging models can reduce bias substantially by increasing capacity, and control variance by fitting one component at a time (e.g., boosting)

Ensemble Methods: Justification

- Ensemble methods more accurate than any individual members if:
  - Accurate (better than guessing)
  - Diverse (different errors on new examples)
- Why?
- Independent errors: prob k of N classifiers (independent error rate ϵ) wrong:

$$P(\text{num errors} = k) = \binom{N}{k} \epsilon^k (1-\epsilon)^{N-k}$$

- Probability that majority vote wrong: error under distribution where more than $N/2$ wrong

Ensemble Methods: Justification

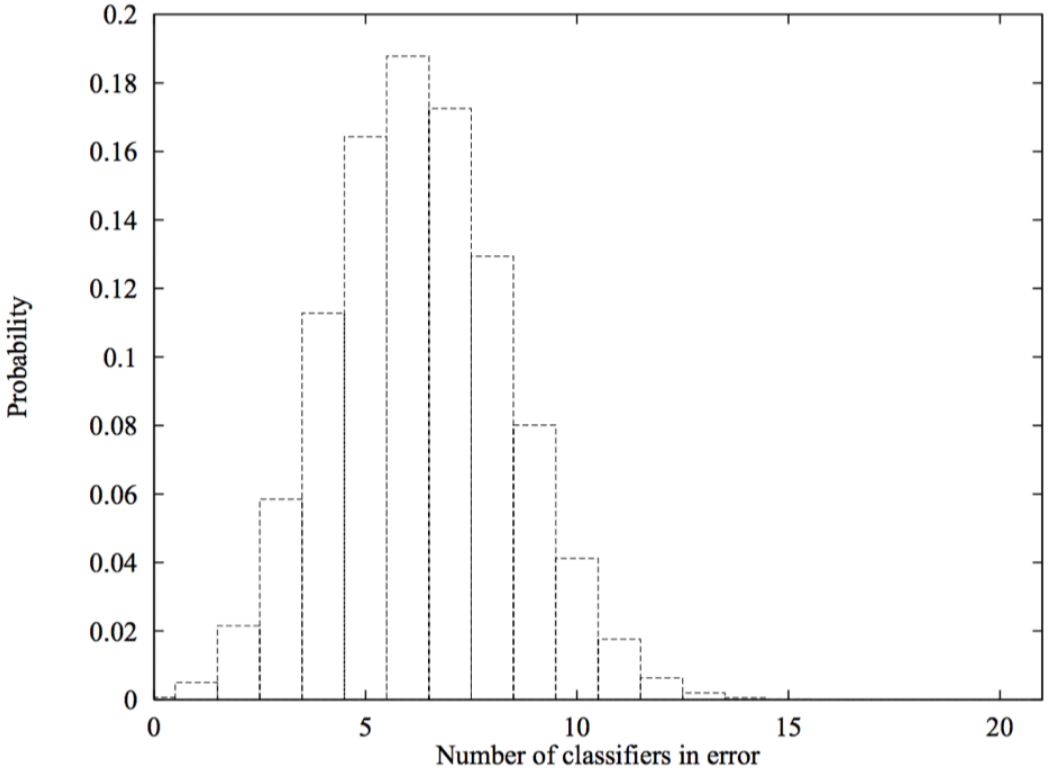


Figure : Example: The probability that exactly K (out of 21) classifiers will make an error assuming each classifier has an error rate of $\epsilon = 0.3$ and makes its errors independently of the other classifier. The area under the curve for 11 or more classifiers being simultaneously wrong is 0.026 (much less than ϵ).

Ensemble Methods: Justification

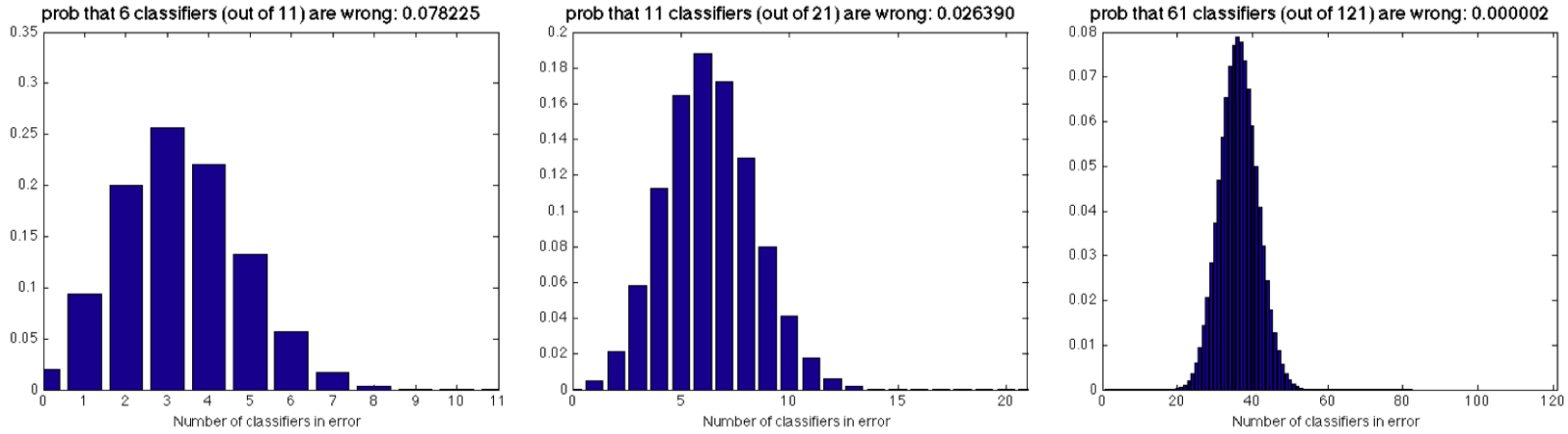


Figure : $\epsilon = 0.3$: **(left)** $N = 11$ classifiers, **(middle)** $N = 21$, **(right)** $N = 121$.

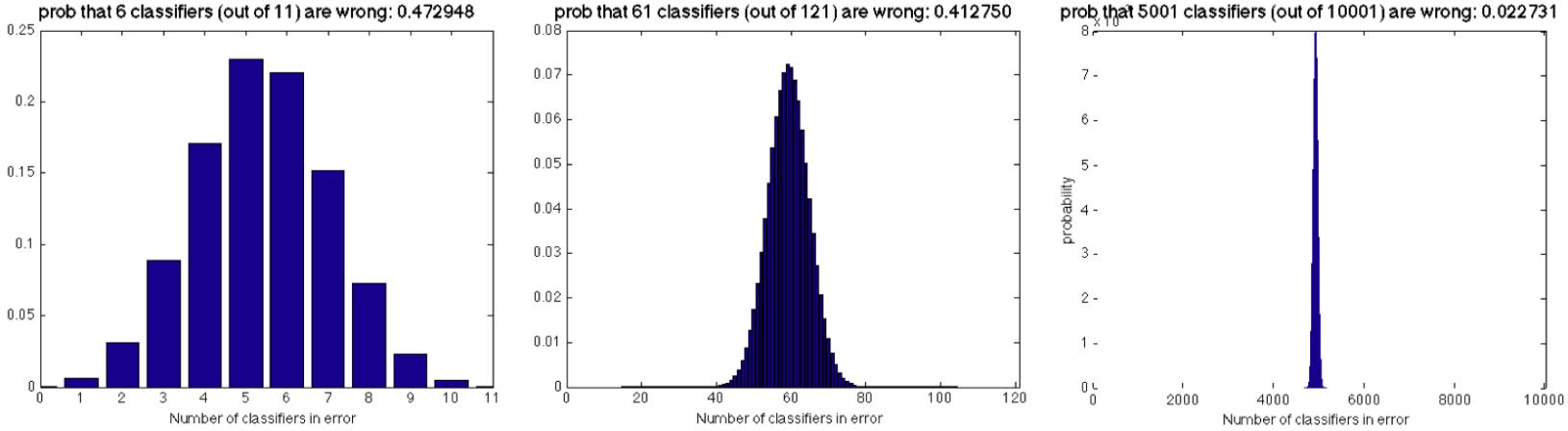


Figure : $\epsilon = 0.49$: **(left)** $N = 11$, **(middle)** $N = 121$, **(right)** $N = 10001$.

Netflix Prize 2007

- The Netflix Prize was an open competition for the best collaborative filtering algorithm to predict user ratings for films, based on previous ratings without any other information about the users or films.

- Rewarded $50,000 in 2007.

- Original progress prize winner (BellKor) was ensemble of 107 models!
  - *"Our experience is that most efforts should be concentrated in deriving substantially different approaches, rather than refining a simple technique."*
  - *"We strongly believe that the success of an ensemble approach depends on the ability of its various predictors to expose different complementing aspects of the data. Experience shows that this is very different than optimizing the accuracy of each individual predictor."*

Bootstrap Estimation

- Repeatedly draw n samples from D
- For each set of samples, estimate a statistic
- The bootstrap estimate is the mean of the individual estimates
- Used to estimate a statistic (parameter) and its variance
- Bagging: bootstrap aggregation (Breiman 1994)

Bagging

- Simple idea: generate M bootstrap samples from your original training set. Train on each one to get y_m, and average them

$$y_{bag}^{M}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x})$$

- For regression: average predictions
- For classification: average class probabilities (or take the majority vote if only hard outputs available)
- Bagging approximates the Bayesian posterior mean. The more bootstraps the better, so use as many as you have time for
- Each bootstrap sample is drawn with replacement, so each one contains some duplicates of certain training points and leaves out other training points completely

Random/Decision Forests

- Definition: Ensemble of decision trees
- Algorithm:
  - Divide training examples into multiple training sets (bagging)
  - Train a decision tree on each set (can randomly select subset of variables to consider)
  - Aggregate the predictions of each tree to make classification decision (e.g., can choose mode vote)

Boosting (AdaBoost)

- Also works by manipulating training set, but classifiers trained sequentially
- Each classifier trained given knowledge of the performance of previously trained classifiers: focus on hard examples
- Final classifier: weighted sum of component classifiers

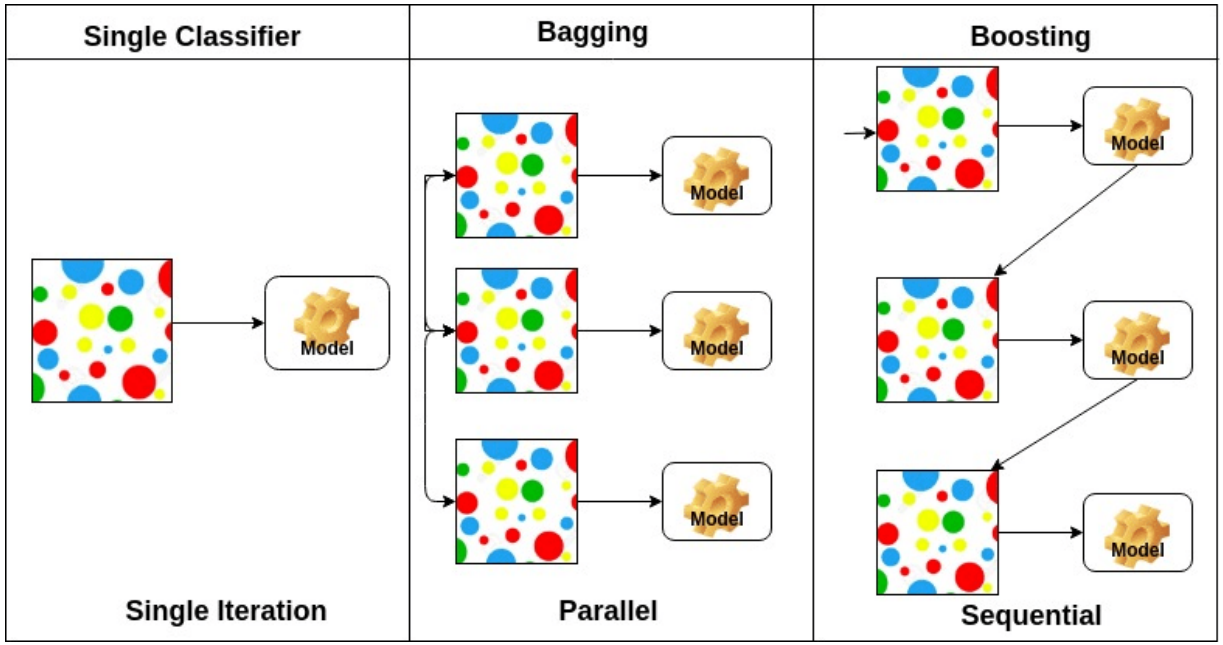


Image source: https://www.datacamp.com/tutorial/adaboost-classifier-python

Making Weak Learners Stronger

- Suppose you have a weak learning module (a base classifier) that can always get $(0.5 + \epsilon)$ correct when given a two-way classification task
  - That seems like a weak assumption but beware!
- Can you apply this learning module many times to get a strong learner that can get close to zero error rate on the training data?
  - Theorists showed how to do this and it actually led to an effective new learning procedure (Freund & Shapire, 1996)

Boosting (AdaBoost)

- First train the base classifier on all the training data with equal importance weights on each case.
- Then re-weight the training data to emphasize the hard cases and train a second model.
  - How do we re-weight the data?
- Keep training new models on re-weighted data
- Finally, use a weighted committee of all the models for the test data.
  - How do we weight the models in the committee?

Train Each Classifier

- Input: $\mathbf{x}$, Output: $y(\mathbf{x}) \in \{1, -1\}$
- Target $t \in \{-1, 1\}$
- Weight on example n for classifier m: $\mathbf{w}_n^m$
- Cost function for classifier m

$$J_m = \sum_{n=1}^{N} w_n^m \underbrace{[y_m(\mathbf{x}^n) \neq t^{(n)}]}_{\text{1 if error, 0 o.w.}} = \sum \text{weighted errors}$$

Weight each training case for classifier m

- Recall cost function is

$$J_m = \sum_{n=1}^{N} w_n^m \underbrace{[y_m(\mathbf{x}^n) \neq t^{(n)}]}_{1 \text{ if error, } \quad 0 \text{ o.w.}} = \sum \text{weighted errors}$$

- Weighted error rate of a classifier

$$\epsilon_m = \frac{J_m}{\sum w_n^m}$$

- The quality of the classifier is

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

It is zero if the classifier has weighted error rate of 0.5 and infinity if the classifier is perfect

- The weights for the next round are then

$$w_n^{m+1} = \exp\left(-\frac{1}{2} t^{(n)} \sum_{i=1}^{m} \alpha_i y_i(\mathbf{x}^{(n)})\right) = w_n^m \exp\left(-\frac{1}{2} t^{(n)} \alpha_m y_m(\mathbf{x}^{(n)})\right)$$

Make predictions using a committee of classifiers

- Weight the binary prediction of each classifier by the quality of that classifier:

$$y_M(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} \frac{1}{2}\alpha_m y_m(\mathbf{x})\right)$$

- This is how to do inference, i.e., how to compute the prediction for each new example.

AdaBoost Algorithm

- Input: $\{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^{N}$, and WeakLearn: learning procedure, produces classifier $y(\mathbf{x})$
- Initialize example weights: $w_n^m(\mathbf{x}) = 1/N$
- For m=1:M
  - $y_m(\mathbf{x}) = \mathit{WeakLearn}(\{\mathbf{x}\}, \mathbf{t}, \mathbf{w})$, fit classifier by minimizing

$$J_m = \sum_{n=1}^{N} w_n^m [y_m(\mathbf{x}^n) \neq t^{(n)}]$$

  - Compute unnormalized error rate

$$\epsilon_m = \frac{J_m}{\sum w_n^m}$$

  - Compute classifier coefficient $\alpha_m = \ln \frac{1-\epsilon_m}{\epsilon_m}$
  - Update data weights

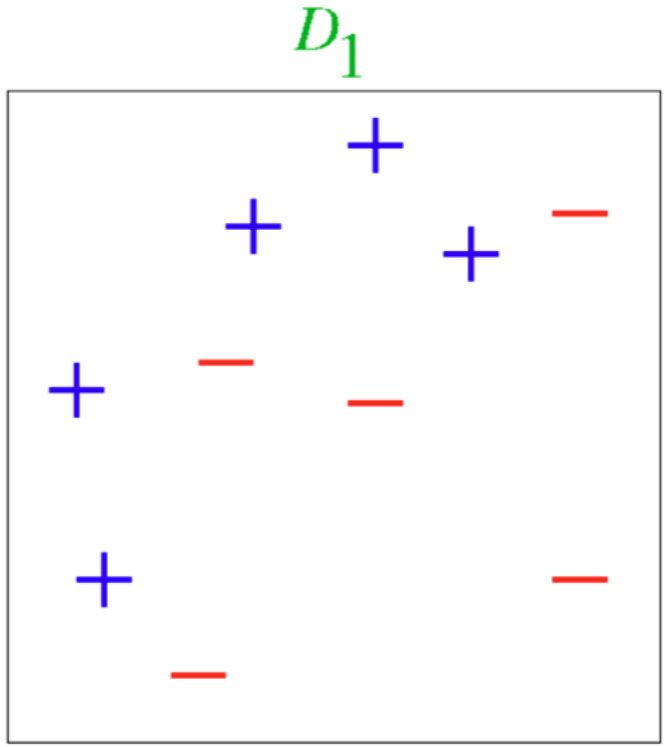
$$w_n^{m+1} = w_n^m \exp\left(-\frac{1}{2} t^{(n)} \alpha_m y_m(\mathbf{x}^{(n)})\right)$$

- Final model

$$Y(\mathbf{x}) = \mathit{sign}(y_M(\mathbf{x})) = \mathit{sign}(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}))$$

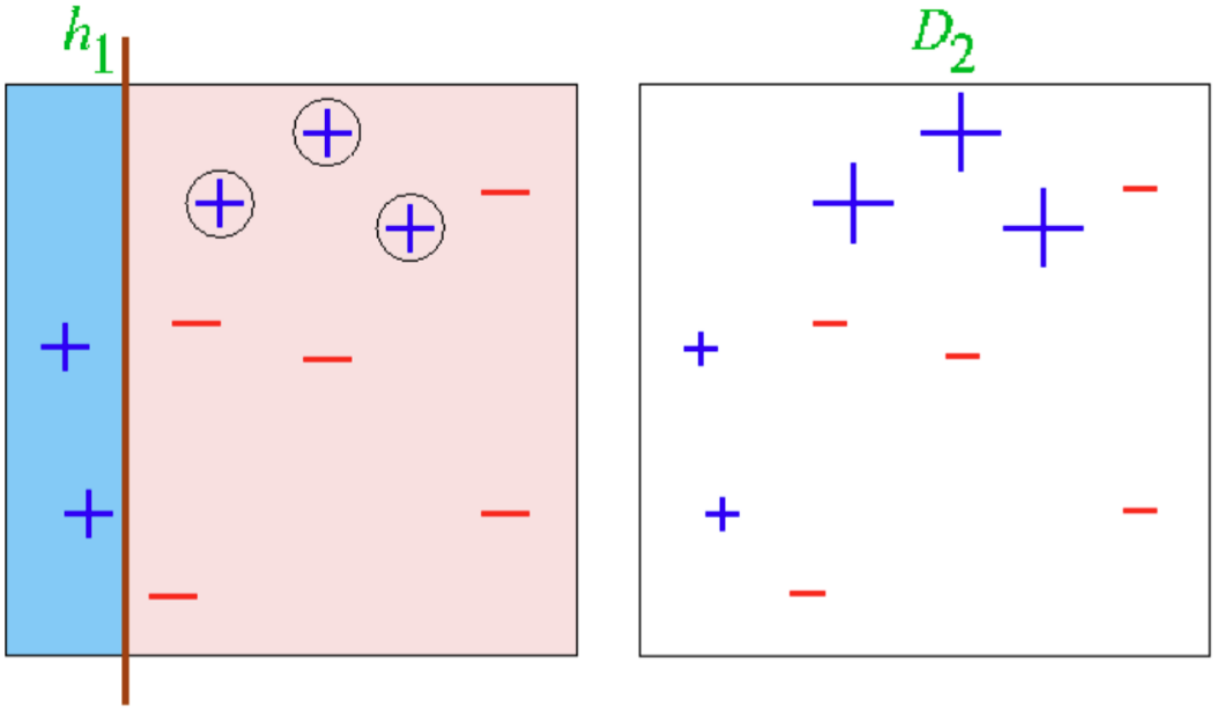
AdaBoost Example

- Training data



AdaBoost Example

- Round 1

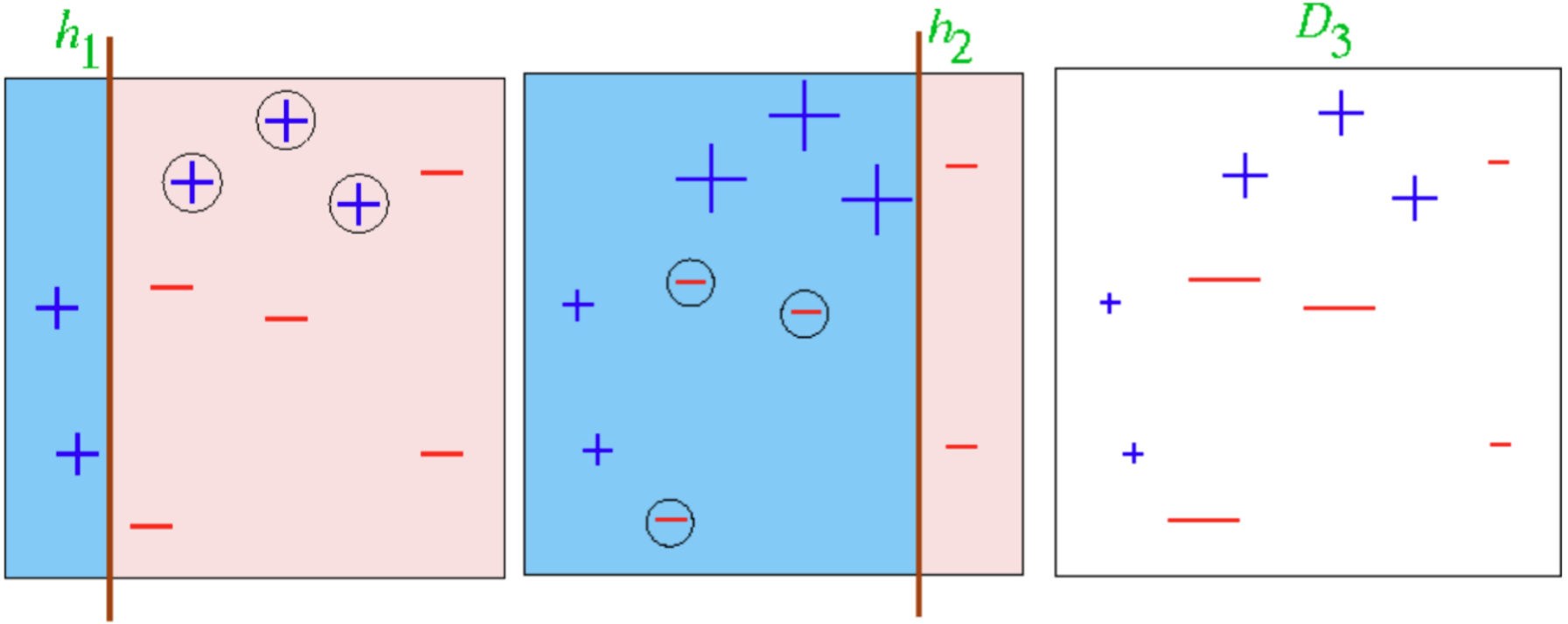


$\varepsilon_1 = 0.300$

$\alpha_1 = 0.847$

AdaBoost Example

- Round 2

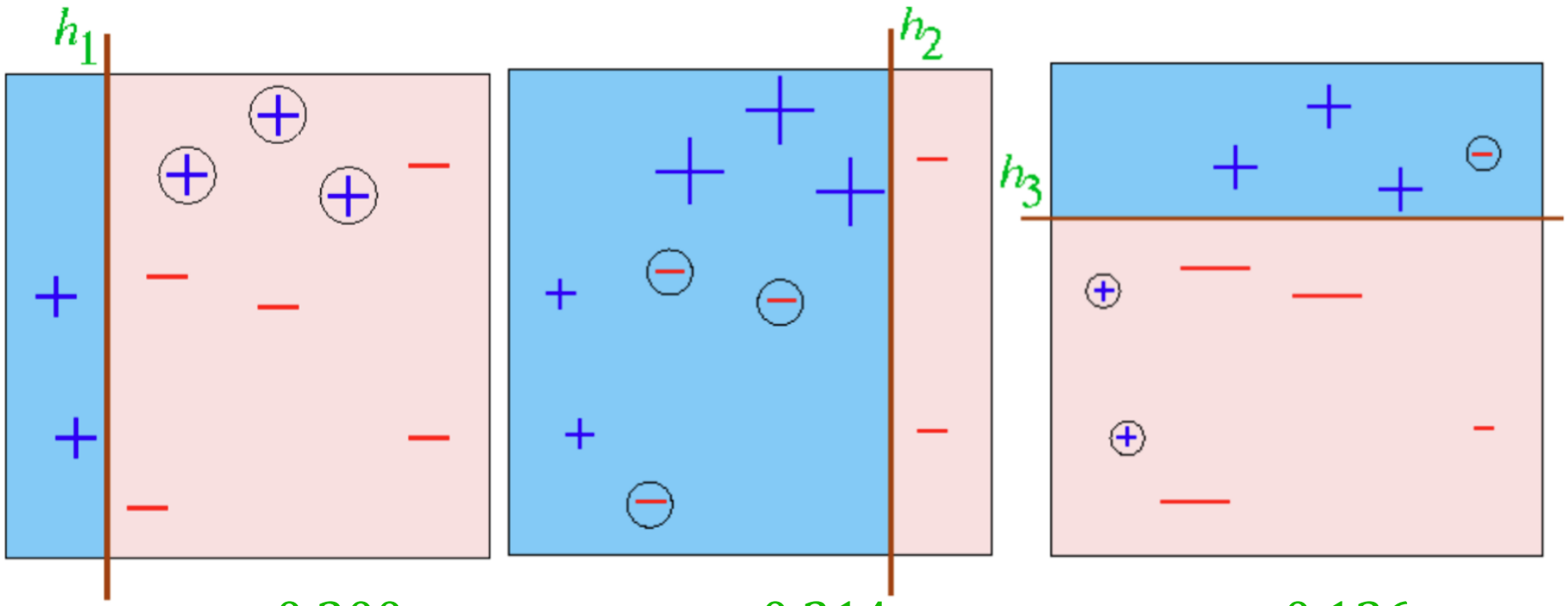


$\varepsilon_1 = 0.300$
$\alpha_1 = 0.847$

$\varepsilon_2 = 0.214$
$\alpha_2 = 1.299$

AdaBoost Example

- Round 3



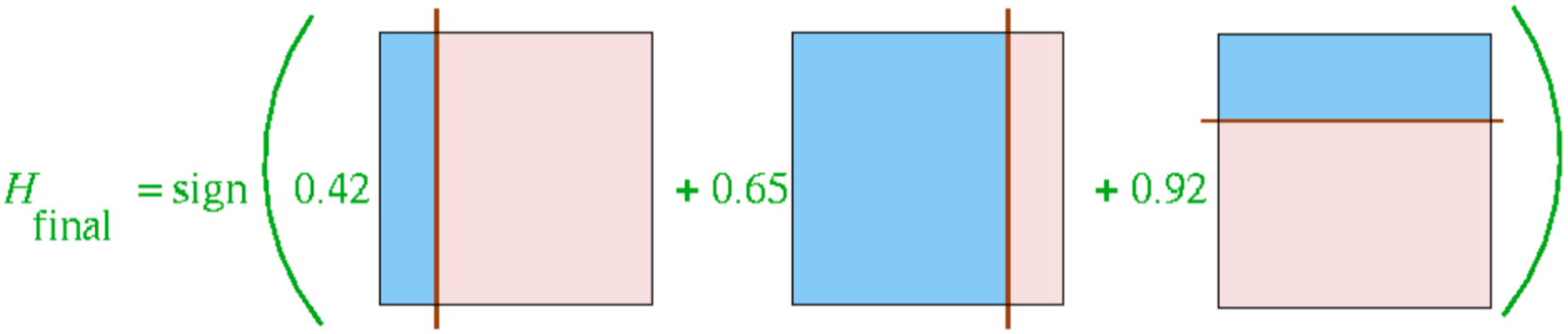
$\varepsilon_1 = 0.300$
$\alpha_1 = 0.847$

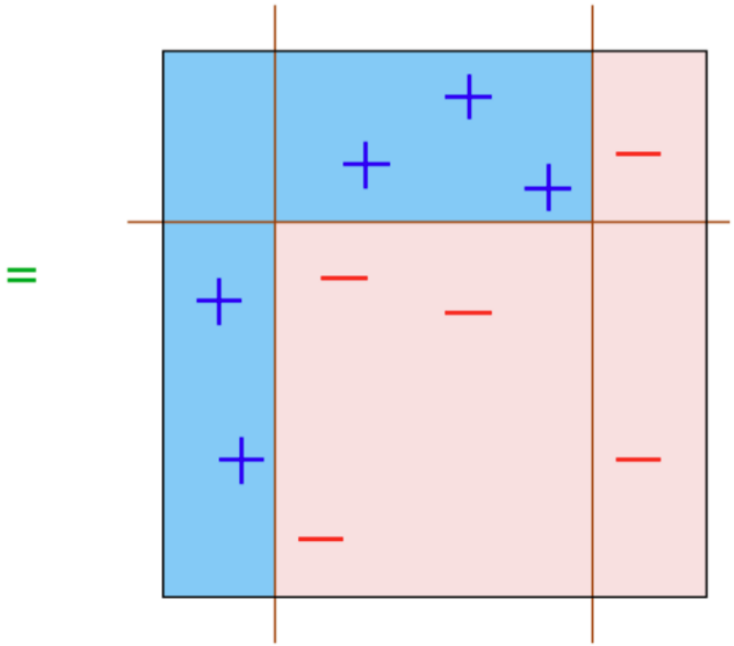
$\varepsilon_2 = 0.214$
$\alpha_2 = 1.299$

$\varepsilon_3 = 0.136$
$\alpha_3 = 1.846$

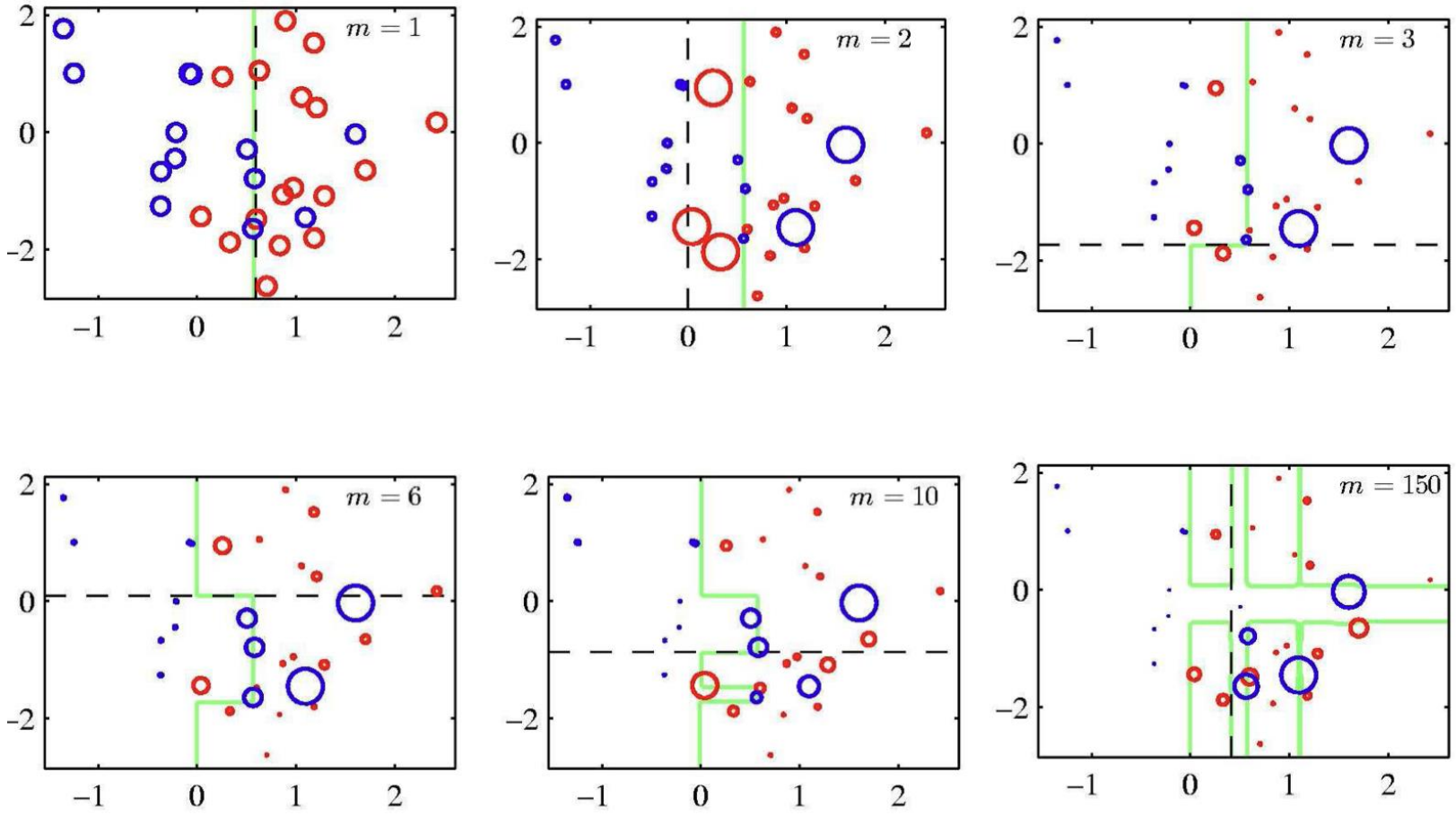
AdaBoost Example

- Final classifier





AdaBoost Example



- Each figure shows the number m of base learners trained so far, the decision of the most recent learner (dashed black), and the boundary of the ensemble (green)

Questions?

UNIVERSITY OF NORTH CAROLINA CHARLOTTE