Ensemble Learning

Richa Singh

Slides are prepared from several information sources on the web and books: Kamalika Chaudhury and Kilien Weinberger

Decision Tree – Limitations and Advantages

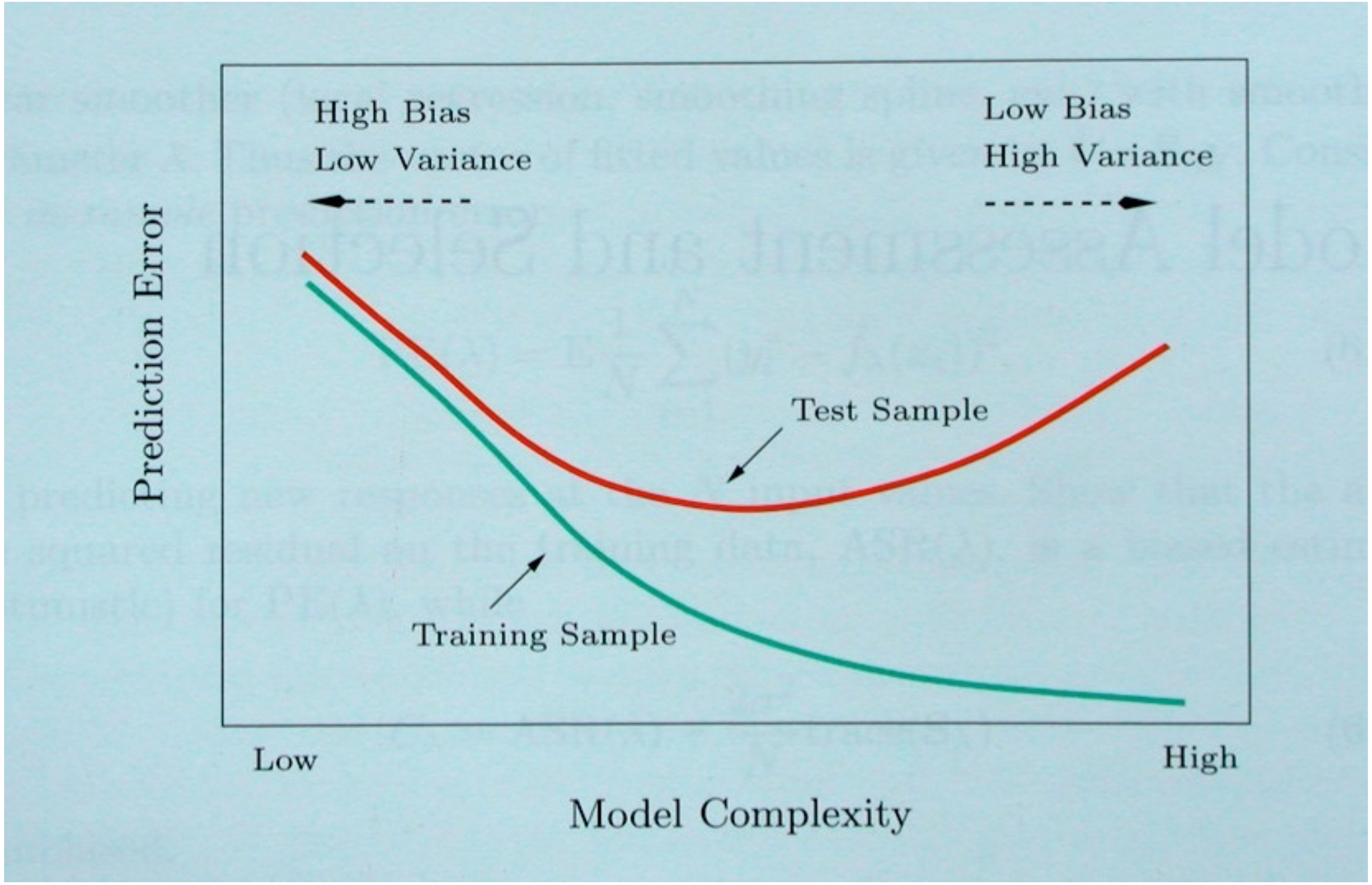
- Trees are flexible → good expressiveness
- Trees are flexible → poor generalization

- Options:
  - Pruning
  - Early stopping

- CART: Classification and Regression Trees

- Can we combine information from multiple decision tress to improve the performance?

Bias/Variance Tradeoff



Hastie, Tibshirani, Friedman "Elements of Statistical Learning" 2001

Bias and Variance

Variance: Captures how much your classifier changes if you train on a different training set. How "over-specialized" is your classifier to a particular training set (overfitting)? If we have the best possible model for our training data, how far off are we from the average classifier?

Bias: What is the inherent error that you obtain from your classifier even with infinite training data? This is due to your classifier being "biased" to a particular kind of solution (e.g. linear classifier). In other words, bias is inherent to your model.

Ensemble Learning

- Simple learners:

- Instead of learning a single (weak) classifier, learn many weak classifiers that are good at different parts of the input space

- How to combine multiple classifiers into a single one
- Works well if the classifiers are complementary

- Two types of ensemble methods:

  Bagging

  Boosting

Reduce Variance Without Increasing Bias

- Averaging reduces variance:

$$Var(\bar{X}) = \frac{Var(X)}{n}$$

(when predictions are **independent**)

Average models to reduce model variance

One problem:

only one training set

where do multiple models come from?

Bagging: Bootstrap Aggregation

- Leo Breiman (1994)
- Take repeated bootstrap samples from training set D.
- *Bootstrap sampling*: Given set D containing N training examples, create D' by drawing N examples at random with replacement from D.

- Bagging:
  - Create k bootstrap samples $D_1 \ldots D_k$.
  - Train the classifier on each D_i.
  - Classify new instance by majority vote / average.

Bagging (Bootstrap AGgregatING)

Input: n labelled training examples $(x_i, y_i), i = 1,..,n$

Algorithm:

Repeat k times:

Select m samples out of n **with replacement** to get training set S_i

Train classifier (decision tree, k-NN, perceptron, etc) h_i on S_i

Output: Classifiers $h_1, .., h_k$

Classification: On test example x, output majority $(h_1, .., h_k)$

Example

Input: n labelled training examples $(x_i, y_i), i = 1,..,n$

Algorithm:

Repeat k times:

Select m samples out of n **with replacement** to get training set S_i

Train classifier (decision tree, k-NN, perceptron, etc) h_i on S_i

How to pick m?

Popular choice: $m = n$

Still different from working with entire training set. Why?

Bagging

Input: n labelled training examples S = $\{(x_i, y_i)\}, i = 1,..,n$

Suppose we select n samples out of n **with replacement** to get training set S_i

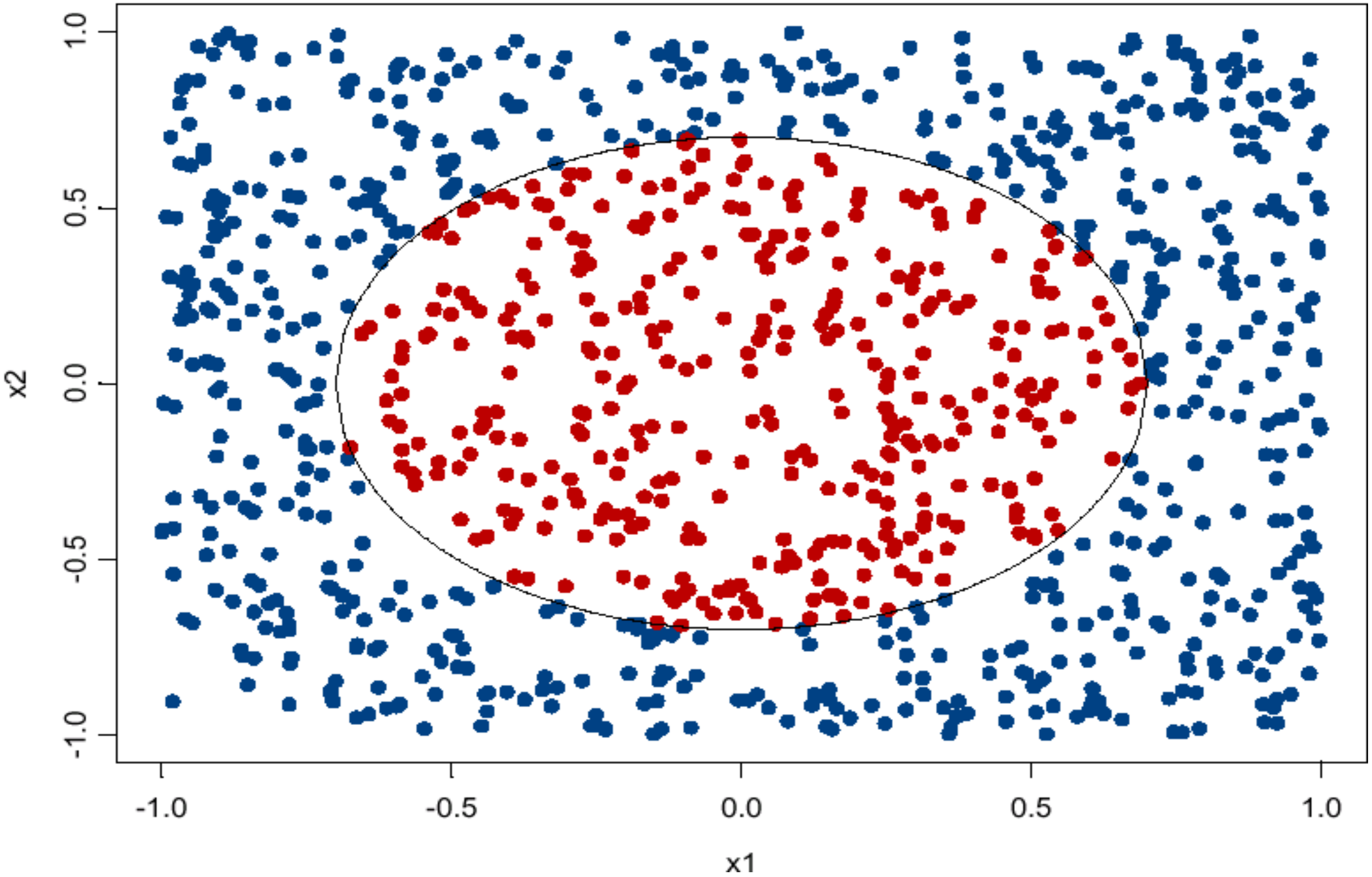
Still different from working with entire training set. Why?

$$\Pr(S_i = S) = \frac{n!}{n^n} \quad \text{(tiny number, exponentially small in n)}$$

$$\Pr((x_i, y_i) \text{ not in } S_i) = \left(1 - \frac{1}{n}\right)^n \approxeq e^{-1}$$

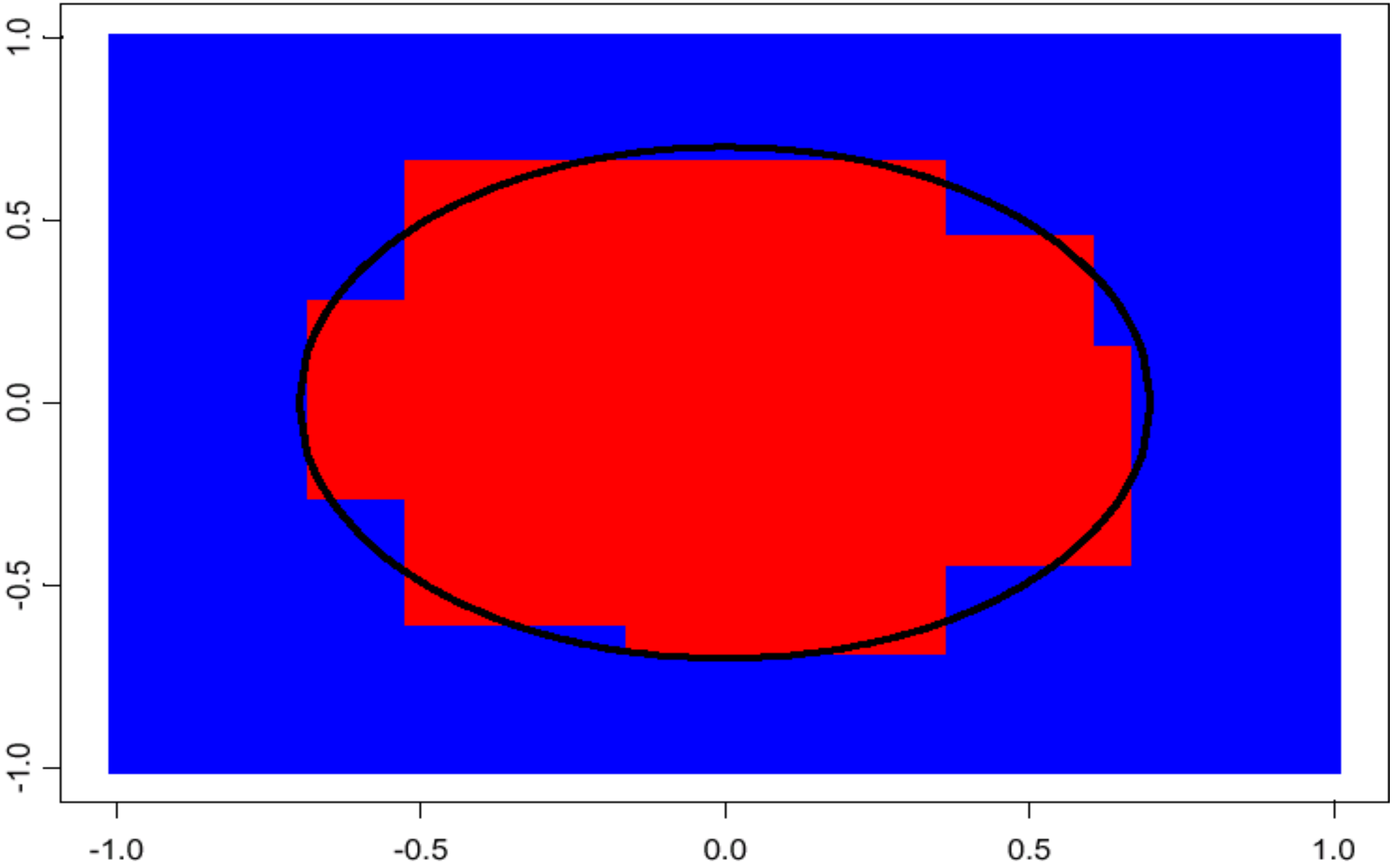
For large data sets, about 37% of the data set is left out!

Example



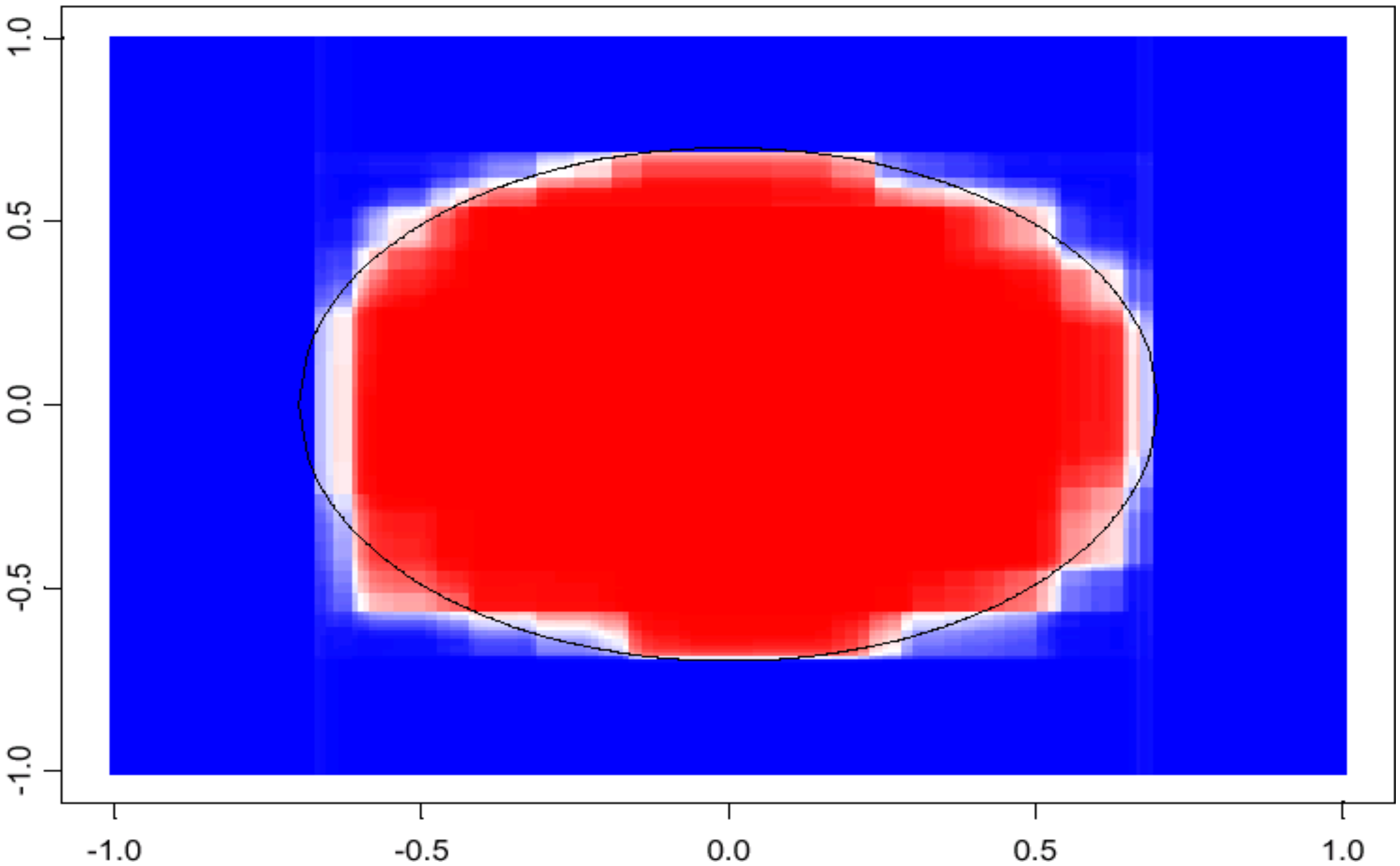
decision tree learning algorithm; very similar to ID3

CART decision boundary



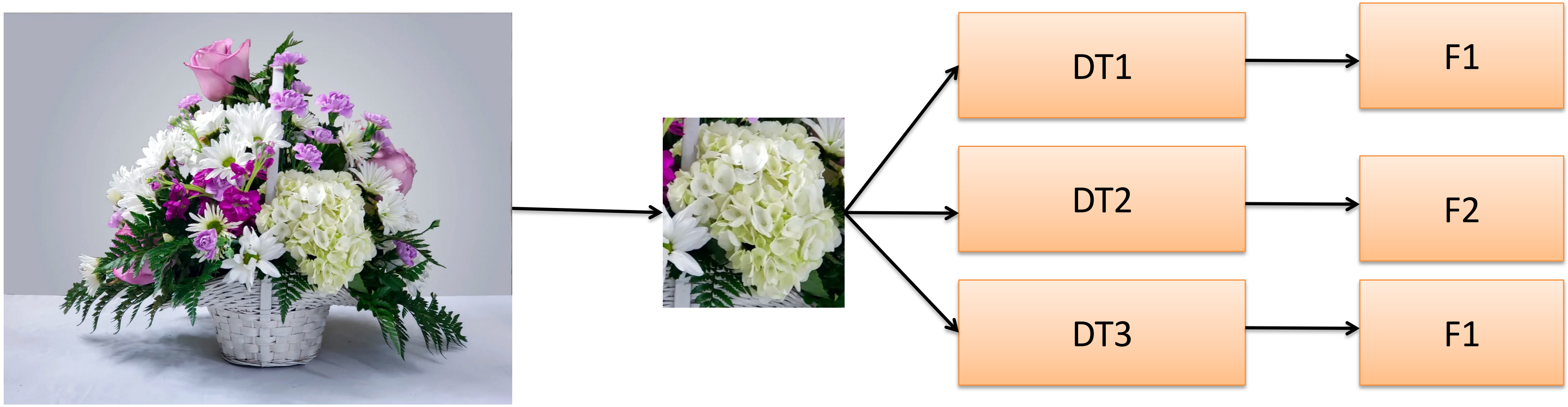
8

100 bagged trees



shades of blue/red indicate strength of vote for particular classification

Random Decision Forest



Majority voting: F1

Bias and Variance

Classification error = Bias +Variance

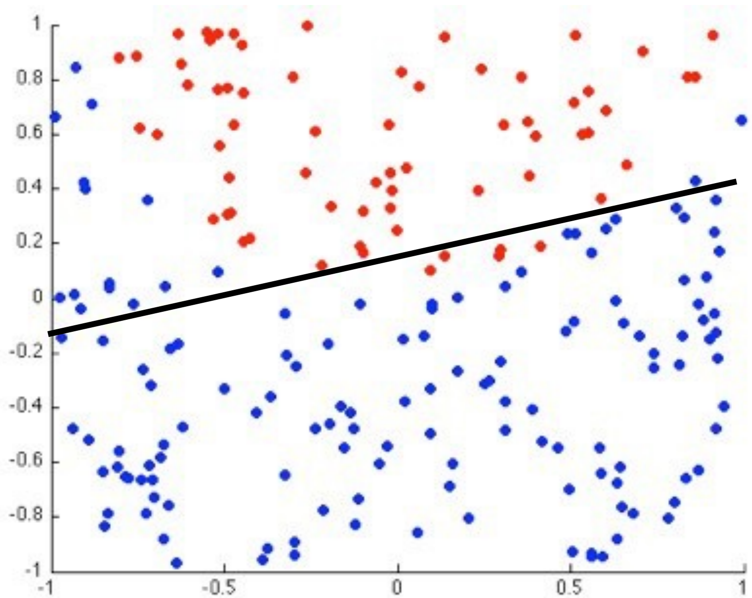
Bias and Variance

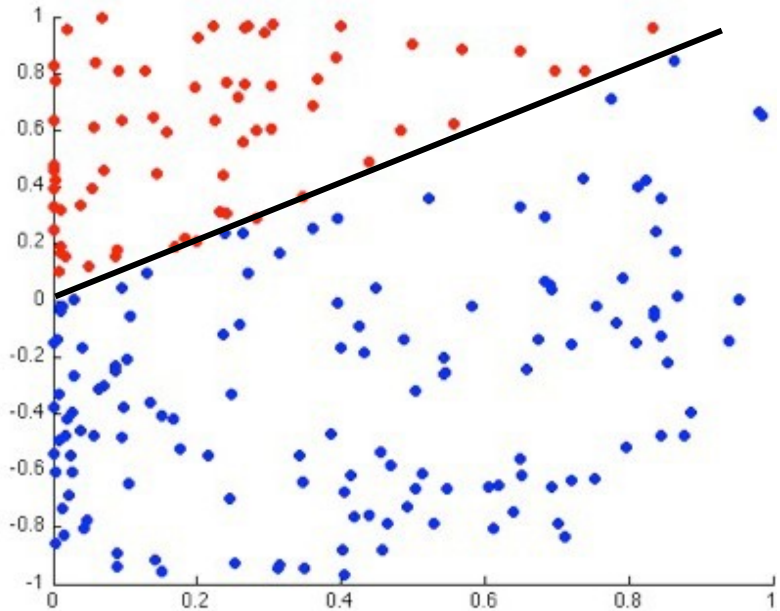
Classification error = Bias +Variance

Bias is the true error of the best classifier in the concept class (e.g, best linear separator, best decision tree on a fixed number of nodes).

Bias is high if the concept class cannot model the true data distribution well, and does not depend on training set size.

Bias





High Bias **Low Bias**

Underfitting: when you have high bias

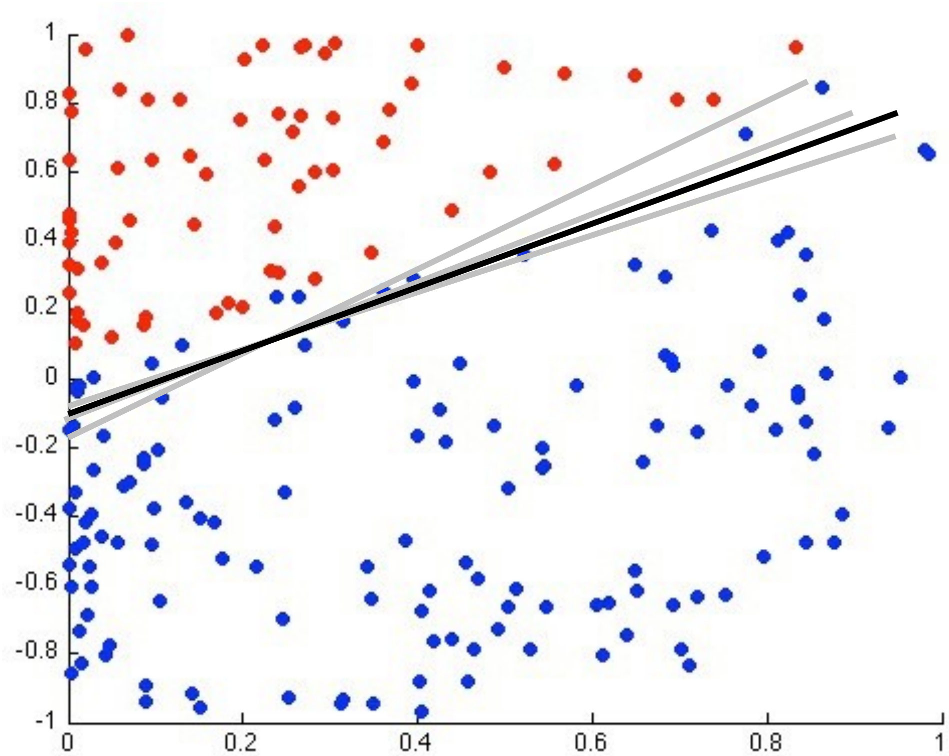
Bias and Variance

Classification error = Bias +Variance

Variance is the error of the trained classifier with respect to the best classifier in the concept class.

Variance depends on the training set size. It decreases with more training data, and increases with more complicated classifiers.

Variance



Overfitting: when you have extra high variance

Bias and Variance

Classification error = Bias + Variance

If you have high bias, both training and test error will be high

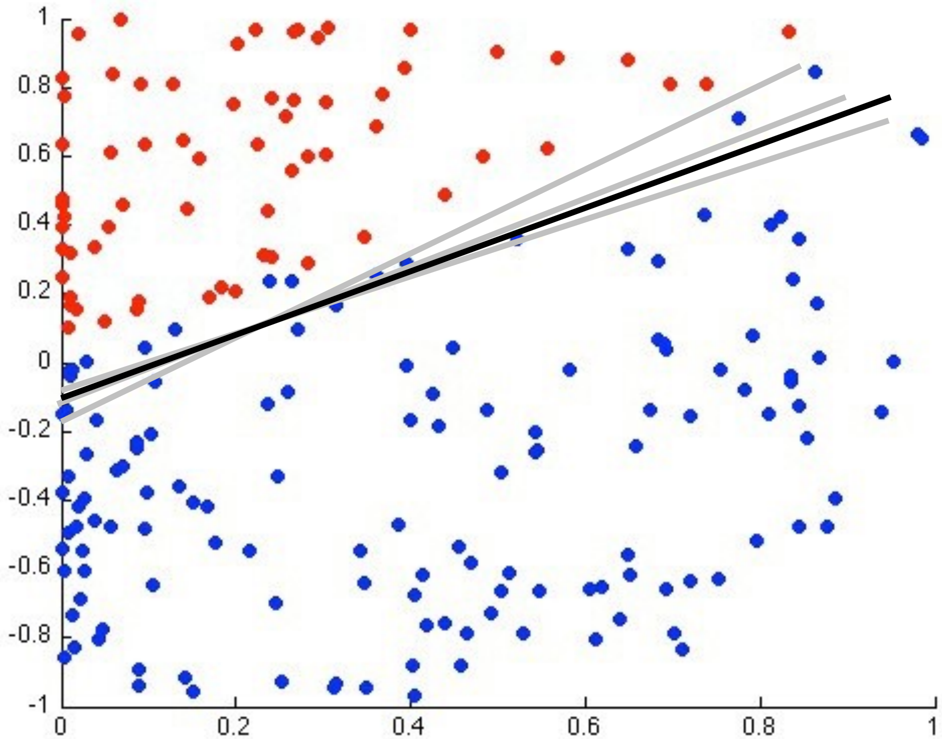
If you have high variance, training error will be low, and test error will be high

Bias Variance Tradeoff

If we make the concept class more complicated (e.g, linear classification to quadratic classification, or increase number of nodes in the decision tree), then bias decreases but variance increases.

Thus there is a bias-variance tradeoff

Why is Bagging useful?



Bagging reduces the variance of the classifier, doesn't help much with bias

Questions?

Ensemble Learning

How to combine multiple classifiers into a single one

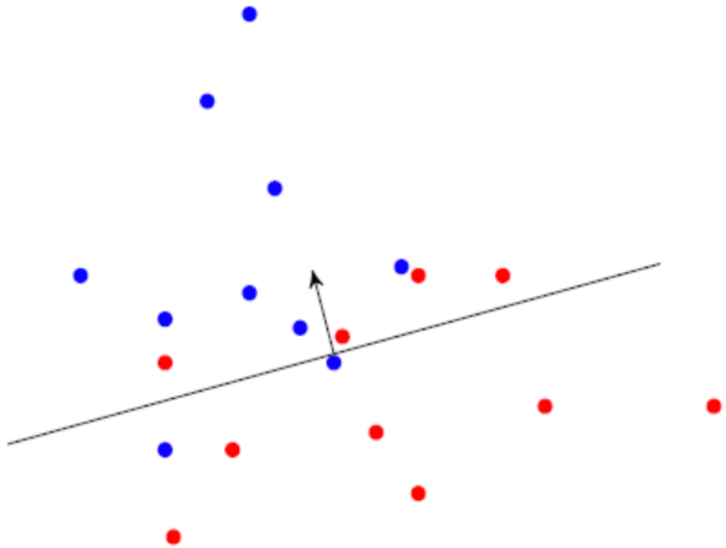
Works well if the classifiers are complementary

This class: two types of ensemble methods:

- Bagging
- Boosting

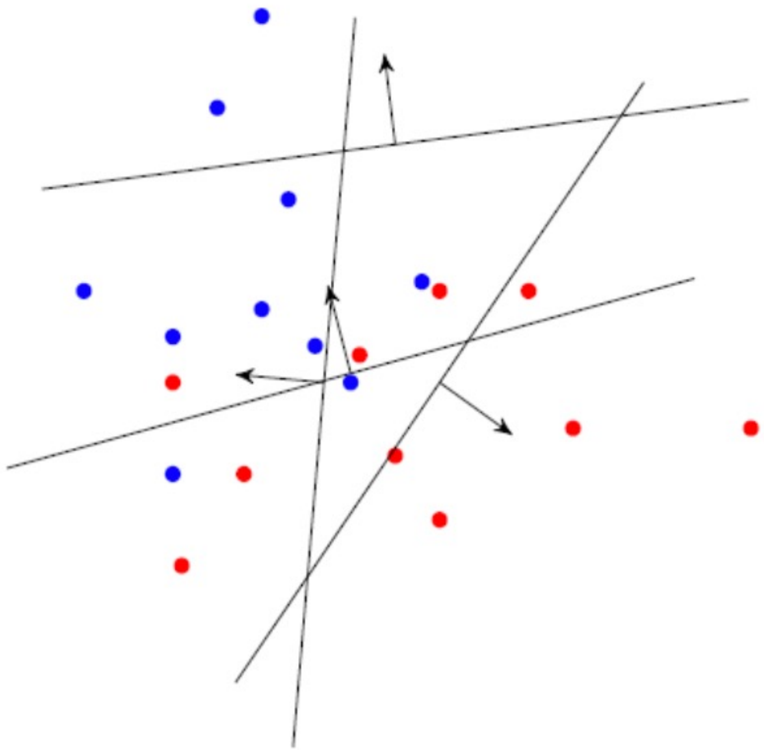
Ensembles

A randomly chosen hyperplane classifier has an expected error of 0.5 (i.e. 50%).



Ensembles

A randomly chosen hyperplane classifier has an expected error of 0.5 (i.e. 50%).



Voting

- Decision by majority vote
  - m individuals (or classifiers) take a vote. m is an odd number
  - They decide between two choices; one is correct, one is wrong.
  - After everyone has voted, a decision is made by simple majority.

Note: For two-class classifiers $f1, \ldots, fm$ (with output ±1):

$$\text{majority vote} = sgn\left(\sum_{j=1}^{m} f_j\right)$$

Voting - Likelihoods

We make some simplifying assumptions:

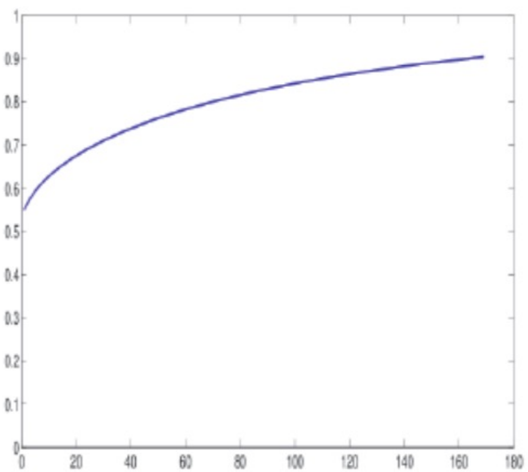
- Each individual makes the right choice with probability $p \in [0, 1]$
- The votes are independent, i.e. stochastically independent when regarded as random outcomes.
- Given n voters, the probability the majority makes the right choice:

$$\text{Pr(majority correct)}_j = \sum_{j=\frac{m+1}{2}}^{m} \frac{m!}{j!(m-j)!} p^j (1-p)^{m-j}$$

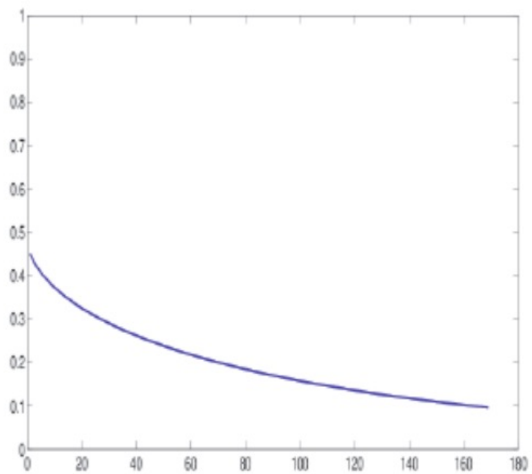
This formula is known as Condorcet's jury theorem

Voting - Likelihoods

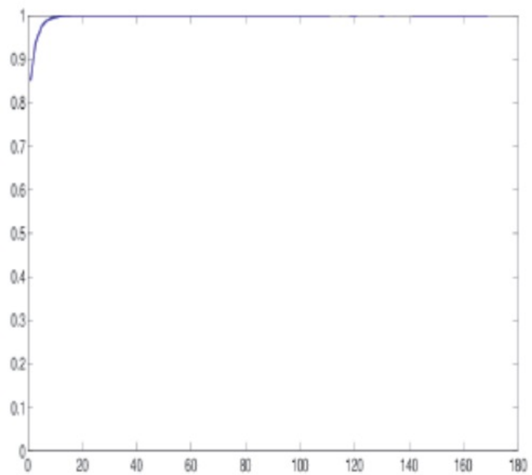
$$\Pr(\text{majority correct})_j = \sum_{j=\frac{m+1}{2}}^{m} \frac{m!}{j!(m-j)!} p^j (1-p)^{m-j}$$



$p = 0.55$



$p = 0.45$



$p = 0.85$

Ensemble Methods

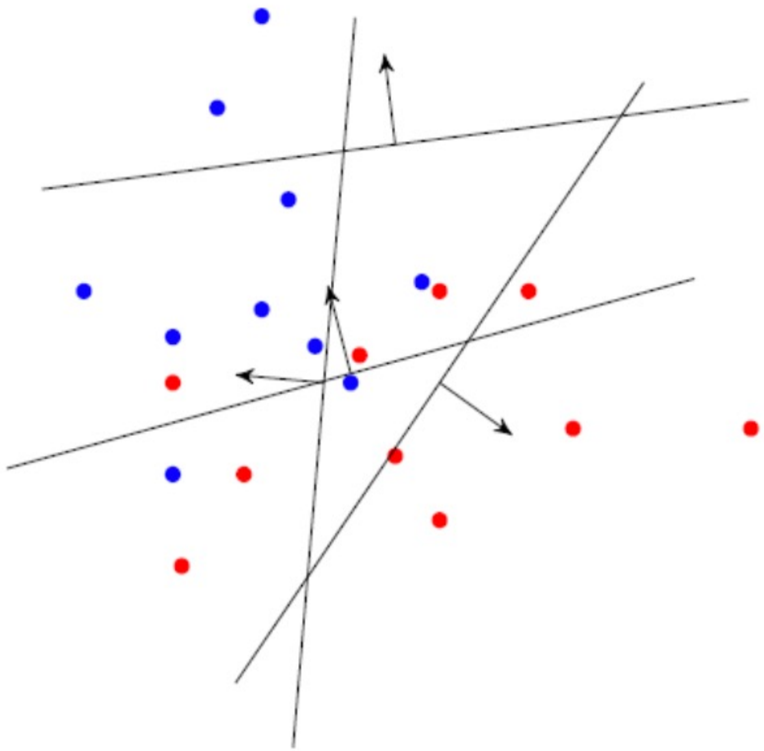
- An ensemble method makes a prediction by combining the predictions of many classifiers into a single vote.
- The individual classifiers are usually required to perform only slightly better than random.
- For two classes, this means slightly more than 50% of the data are classified correctly. Such a classifier is called a weak learner.

- Are good: Low variance, don't usually overfit
- Are bad: High bias, can't solve hard learning problems

- Can we make weak learners always good?
  - No!!! But often yes…

Ensembles

A randomly chosen hyperplane classifier has an expected error of 0.5 (i.e. 50%).



Boosting

Goal: Determine if an email is spam or not based on text in it

From: Yuncong Chen
Text: 151 homeworks are all graded...

Not Spam

From: Work from home solutions
Text: Earn money without working!

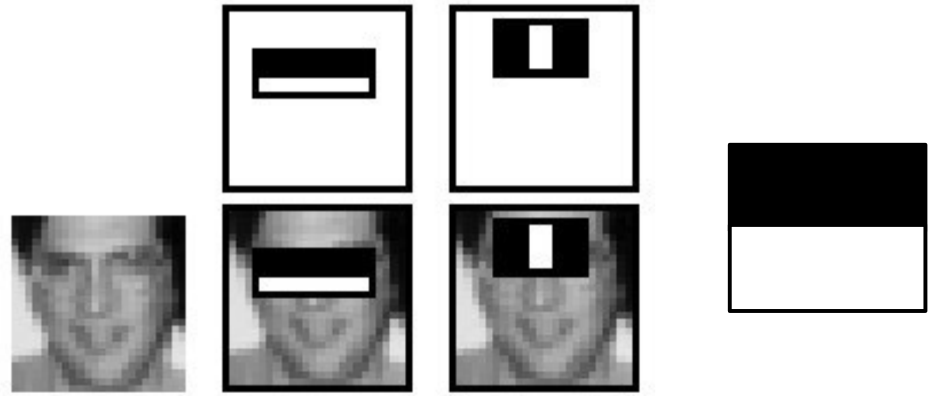
Spam

Sometimes it is:

- Easy to come up with simple rules-of-thumb classifiers,
- Hard to come up with a single high accuracy rule

Boosting

Goal: Detect if an image contains a face in it



= Is the black region darker on average than the white?

Sometimes it is:

- Easy to come up with simple rules-of-thumb classifiers,
- Hard to come up with a single high accuracy rule

Boosting

Weak Learner: A simple rule-of-the-thumb classifier that doesn’t necessarily work very well

Strong Learner: A good classifier

Boosting: How to combine many weak learners into a strong learner?

Boosting

Procedure:

1. Design a method for finding a good rule-of-thumb
2. Apply method to training data to get a good rule-of-thumb
3. Modify the training data to get a 2nd data set
4. Apply method to 2nd data set to get a good rule-of-thumb
5. Repeat T times...

Boosting

1. How to get a good rule-of-thumb?

   Depends on application e.g, single node decision trees

2. How to choose examples on each round?

   Focus on the **hardest examples** so far - namely, examples misclassified most often by previous rules of thumb

3. How to combine the rules-of-thumb to a prediction rule? Take a weighted majority of the rules

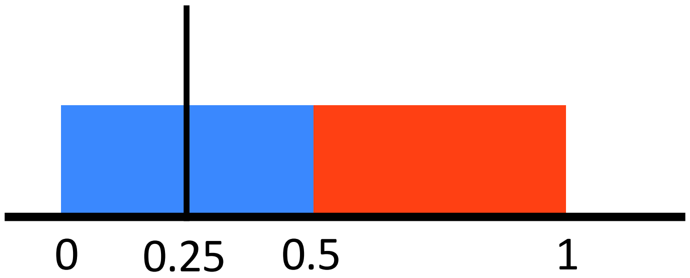
Some Notations

Let D be a distribution over examples, and h be a classifier
Error of h with respect to D is:

$$err_D(h) = Pr_{(X,Y)\sim D}(h(X) \neq Y)$$

Example:

Below X is uniform over [0, 1], and Y = 1 if X > 0.5, 0 otherwise



$err_D(h) = 0.25$

Some Notation

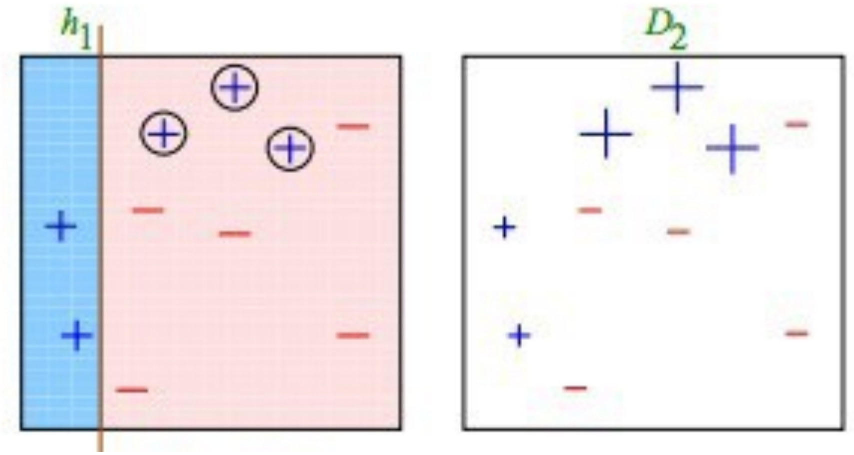
Let D be a distribution over examples, and h be a classifier
Error of h with respect to D is:

$$err_D(h) = Pr_{(X,Y)\sim D}(h(X) \neq Y)$$

h is called a **weak learner** if $err_D(h) < 0.5$

If you guess completely randomly, then the error is 0.5

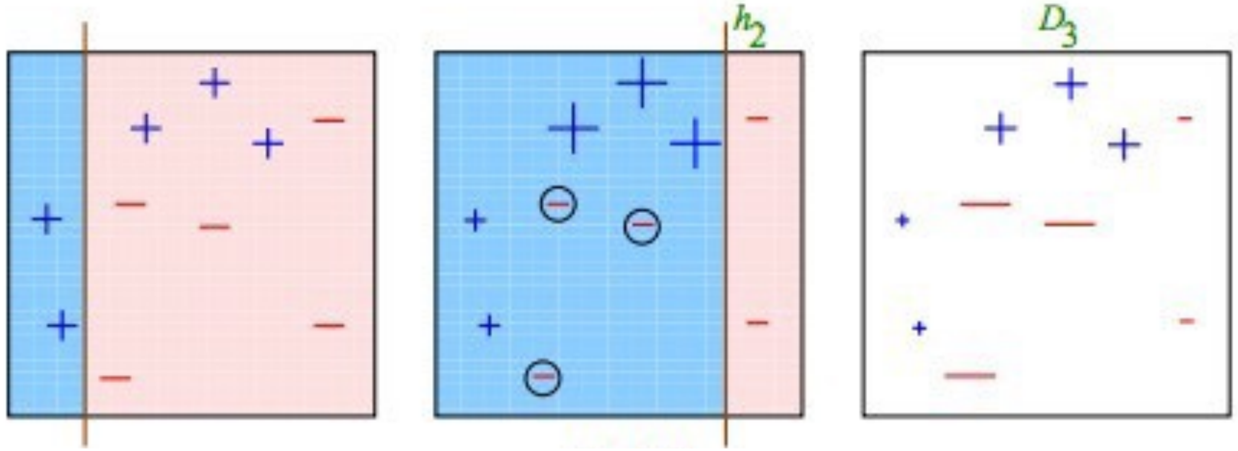
Boosting: Example



Schapire, 2011

weak classifiers: horizontal or vertical half-planes

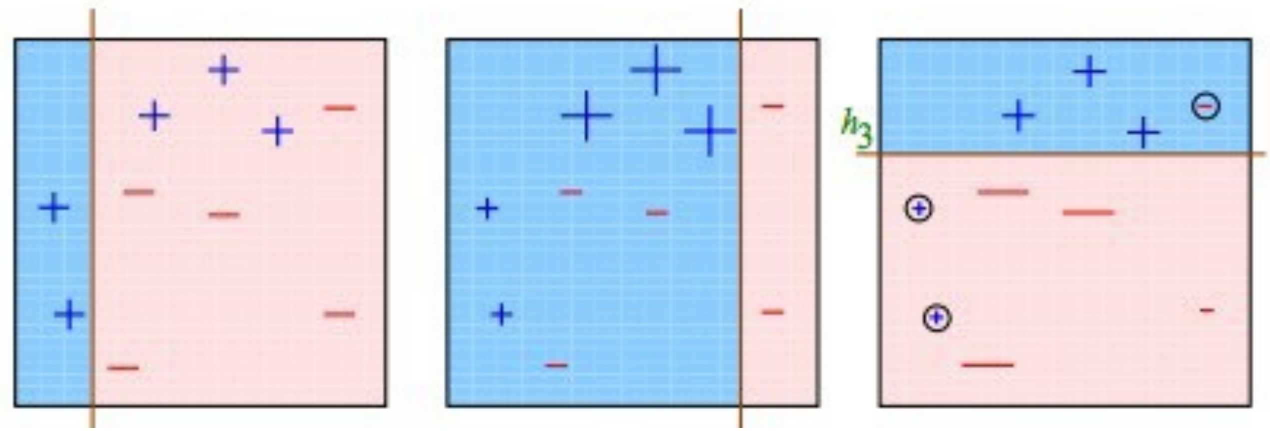
Boosting: Example



Schapire, 2011

weak classifiers: horizontal or vertical half-planes

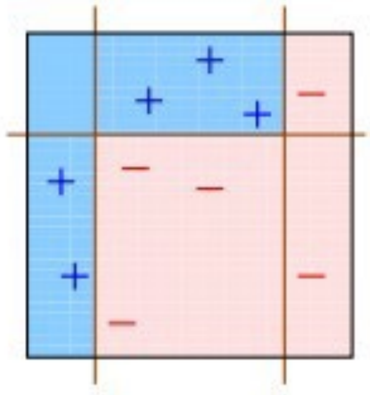
Boosting: Example



Schapire, 2011

weak classifiers: horizontal or vertical half-planes

The Final Classifier



Schapire, 2011

weak classifiers: horizontal or vertical half-planes

Some Notation

Given training examples $\{(x_i, y_i)\}$, $i = 1,..,n$, we can assign weights w_i to the examples. If the w_is sum to 1, then we can think of them as a distribution W over the examples.

The error of a classifier h wrt W is:

$$err_W(h) = \sum_{i=1}^{n} w_i 1(h(x_i) \neq y_i)$$

Note: 1 here is the indicator function

Boosting

Given training set $S = \{(x_1, y_1), ..., (x_n, y_n)\}$, y in {-1, 1}

For t = 1,...,T

  Construct distribution D_t on the examples

  Find weak learner h_t which has small error $err_{D_t}(h_t)$ wrt D_t

Output final classifier

Initially, $D_1(i) = 1/n$, for all i (uniform)

Given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} exp(-\alpha_t y_i h_t(x_i))$$

Weight update rule

where:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - err_{D_t}(h_t)}{err_{D_t}(h_t)} \right)$$

Z_t = normalization constant

Boosting

Given training set $S = \{(x_1, y_1), ..., (x_n, y_n)\}$, y in $\{-1, 1\}$

For $t = 1, ..., T$

Construct distribution D_t on the examples

Find weak learner h_t which has small error $err_{D_t}(h_t)$ wrt D_t

Output final classifier

Initially, $D_1(i) = 1/n$, for all i (uniform)

Given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} exp(-\alpha_t y_i h_t(x_i))$$

where:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - err_{D_t}(h_t)}{err_{D_t}(h_t)} \right)$$

Z_t = normalization constant

$D_{t+1}(i)$ goes down if x_i is classified correctly by h_t, up otherwise

High $D_{t+1}(i)$ means hard example

Boosting

Given training set $S = \{(x_1, y_1), ..., (x_n, y_n)\}$, y in $\{-1, 1\}$

For $t = 1, ..., T$

Construct distribution D_t on the examples

Find weak learner h_t which has small error $err_{D_t}(h_t)$ wrt D_t

Output final classifier

Initially, $D_1(i) = 1/n$, for all i (uniform)

Given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} exp(-\alpha_t y_i h_t(x_i))$$

where:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - err_{D_t}(h_t)}{err_{D_t}(h_t)} \right)$$

Higher if h_t has low error wrt D_t, lower otherwise. >0 if $err_{D_t}(h_t) < 0.5$

Z_t = normalization constant

Boosting

Given training set $S = \{(x_1, y_1), ..., (x_n, y_n)\}$, y in {-1, 1}

For t = 1, ...,T

Construct distribution D_t on the examples

Find weak learner h_t which has small error $err_{D_t}(h_t)$ wrt D_t

Output final classifier

Initially, $D_1(i) = 1/n$, for all i (uniform)

Given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} exp(-\alpha_t y_i h_t(x_i))$$

where:

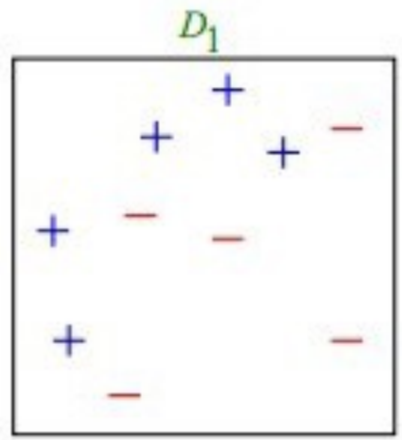
$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - err_{D_t}(h_t)}{err_{D_t}(h_t)} \right)$$

Z_t = normalization constant

Final classifier:

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

Boosting: Example



Schapire, 2011

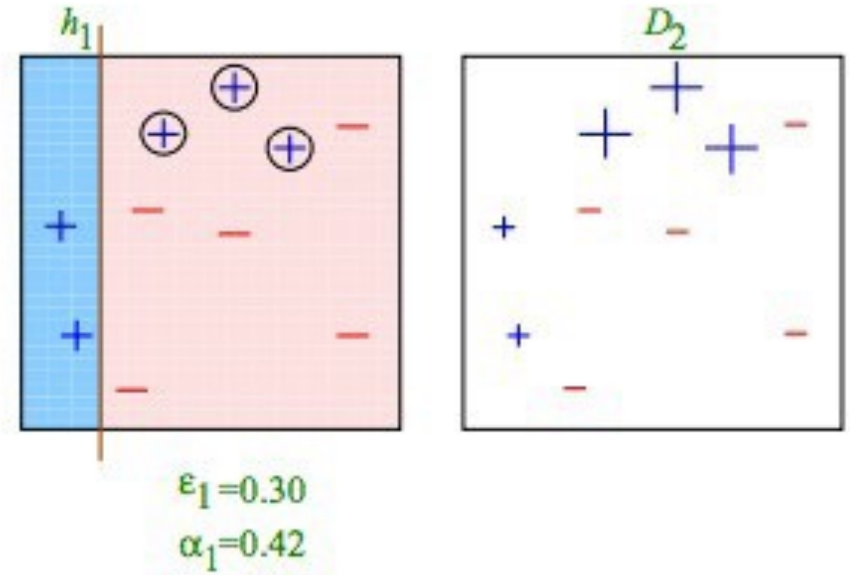
weak classifiers: horizontal or vertical half-planes

Boosting: Example

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} exp(-\alpha_t y_i h_t(x_i))$$

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - err_{D_t}(h_t)}{err_{D_t}(h_t)} \right)$$

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



Schapire, 2011

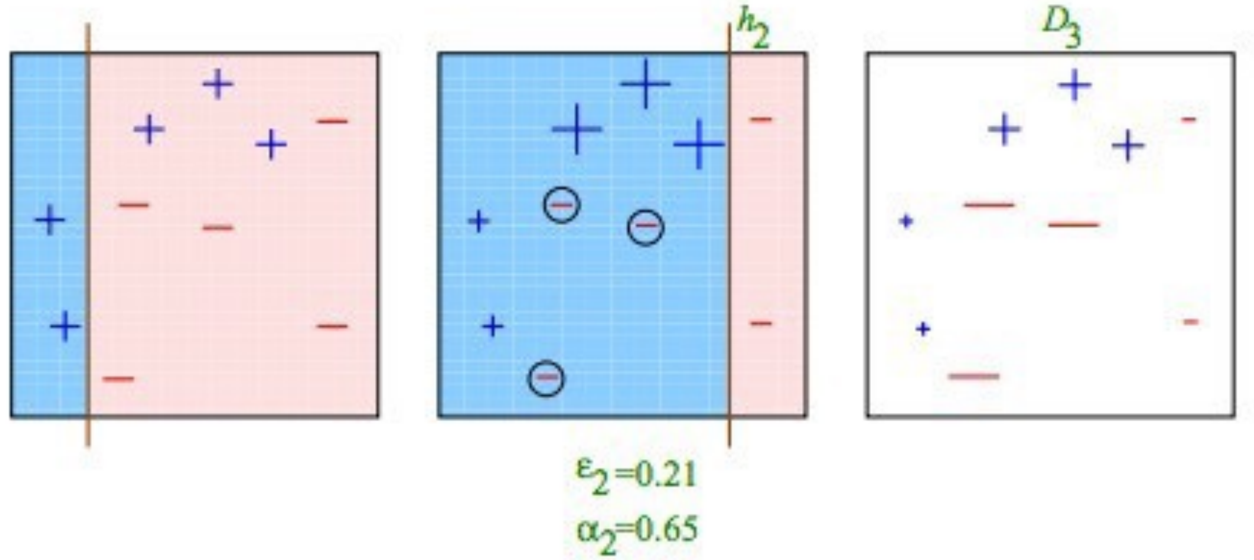
weak classifiers: horizontal or vertical half-planes

Boosting: Example

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} exp(-\alpha_t y_i h_t(x_i))$$

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - err_{D_t}(h_t)}{err_{D_t}(h_t)} \right)$$

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



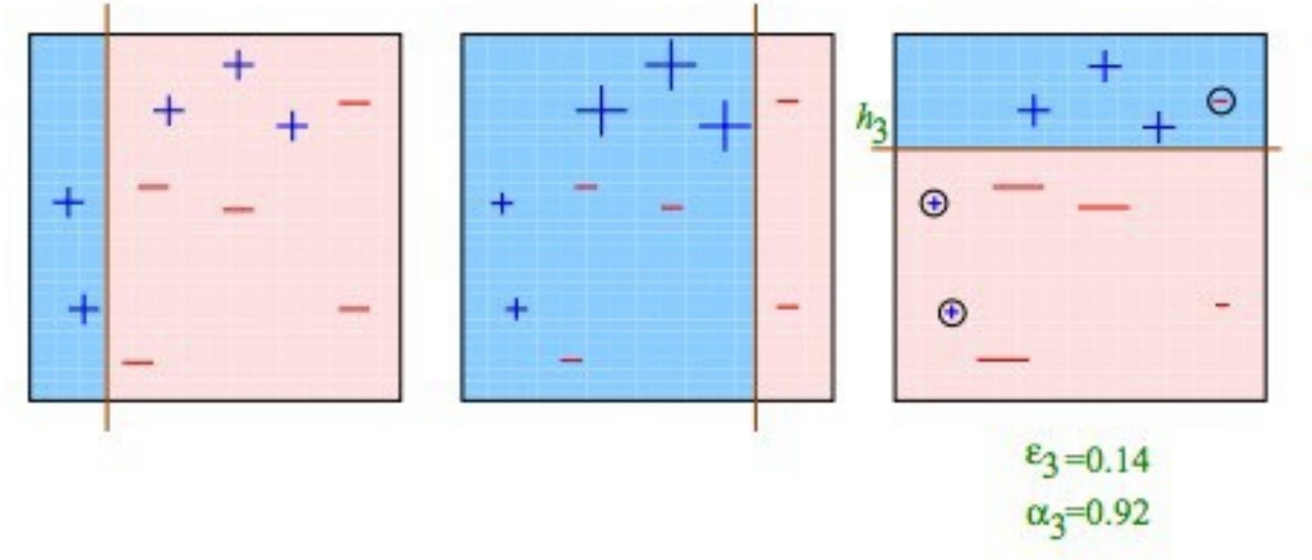
Schapire, 2011

weak classifiers: horizontal or vertical half-planes

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} exp(-\alpha_t y_i h_t(x_i))$$

Boosting: Example

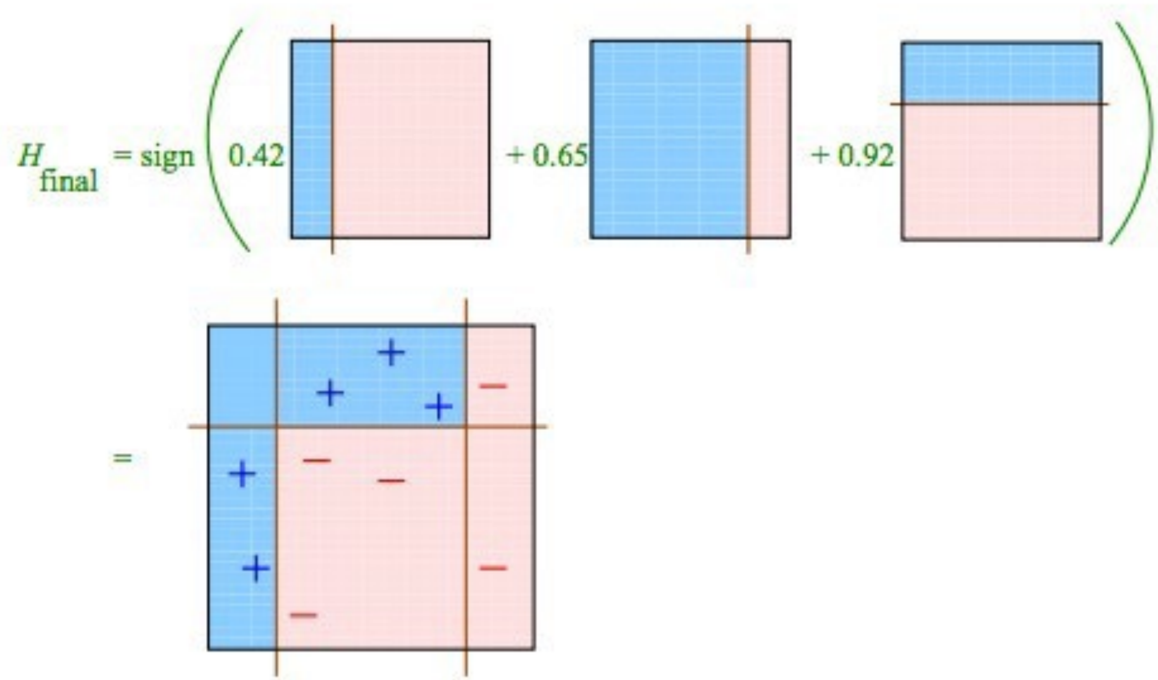
$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - err_{D_t}(h_t)}{err_{D_t}(h_t)} \right)$$



Schapire, 2011

weak classifiers: horizontal or vertical half-planes

The Final Classifier



Schapire, 2011

weak classifiers: horizontal or vertical half-planes

How to Find Stopping Time

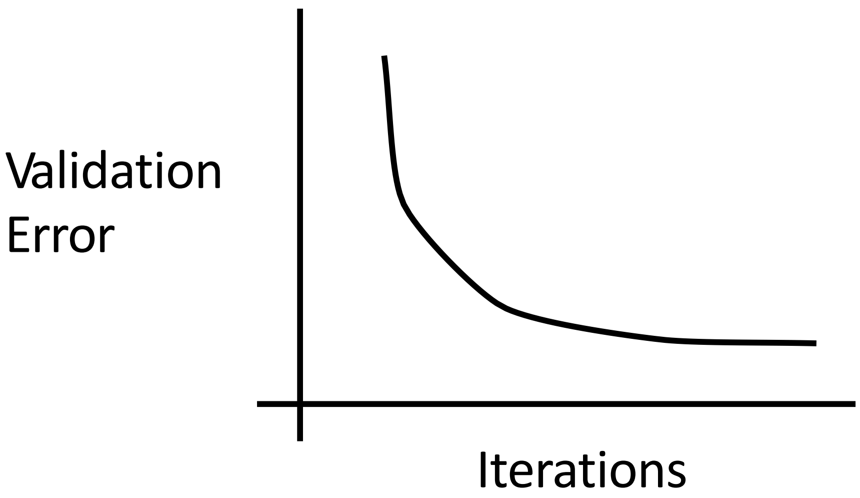
Given training set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, y in $\{-1, 1\}$

For $t = 1, \ldots, T$

Construct distribution D_t on the examples

Find weak learner h_t which has small error $err_{D_t}(h_t)$ wrt D_t

Output final classifier



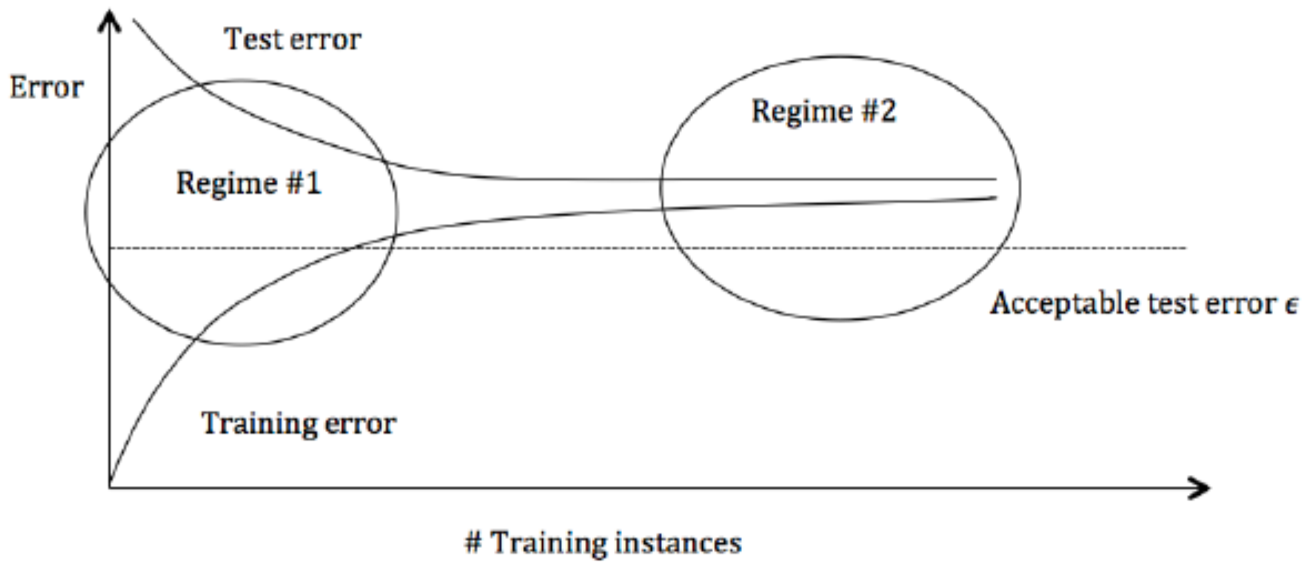
To find stopping time, use a **validation dataset**.
Stop when the error on the validation dataset stops getting better, or when you can't find a good rule of thumb.

Gradient Boosting

Adaboost

Boosted decision trees

Boosted Neural networks



	Regime 1 - High Variance	Regime 2 - High Bias
Symptoms	Training error is much lower than test error	Training error is lower than epsilon
	Training error is lower than epsilon	
	Test error is above epsilon	
Remedy	Add more training data	Add features
	Reduce model complexity -- complex Models are prone to high variance	Use more complex model (e.g. kernelize, use non-linear models)
	Bagging	Boosting

http://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html

Questions?