Foundations of Machine Learning

# Ensemble Classifiers

Oct 2021

Vineeth N Balasubramanian

आई आई टी हैदराबाद
IIT Hyderabad

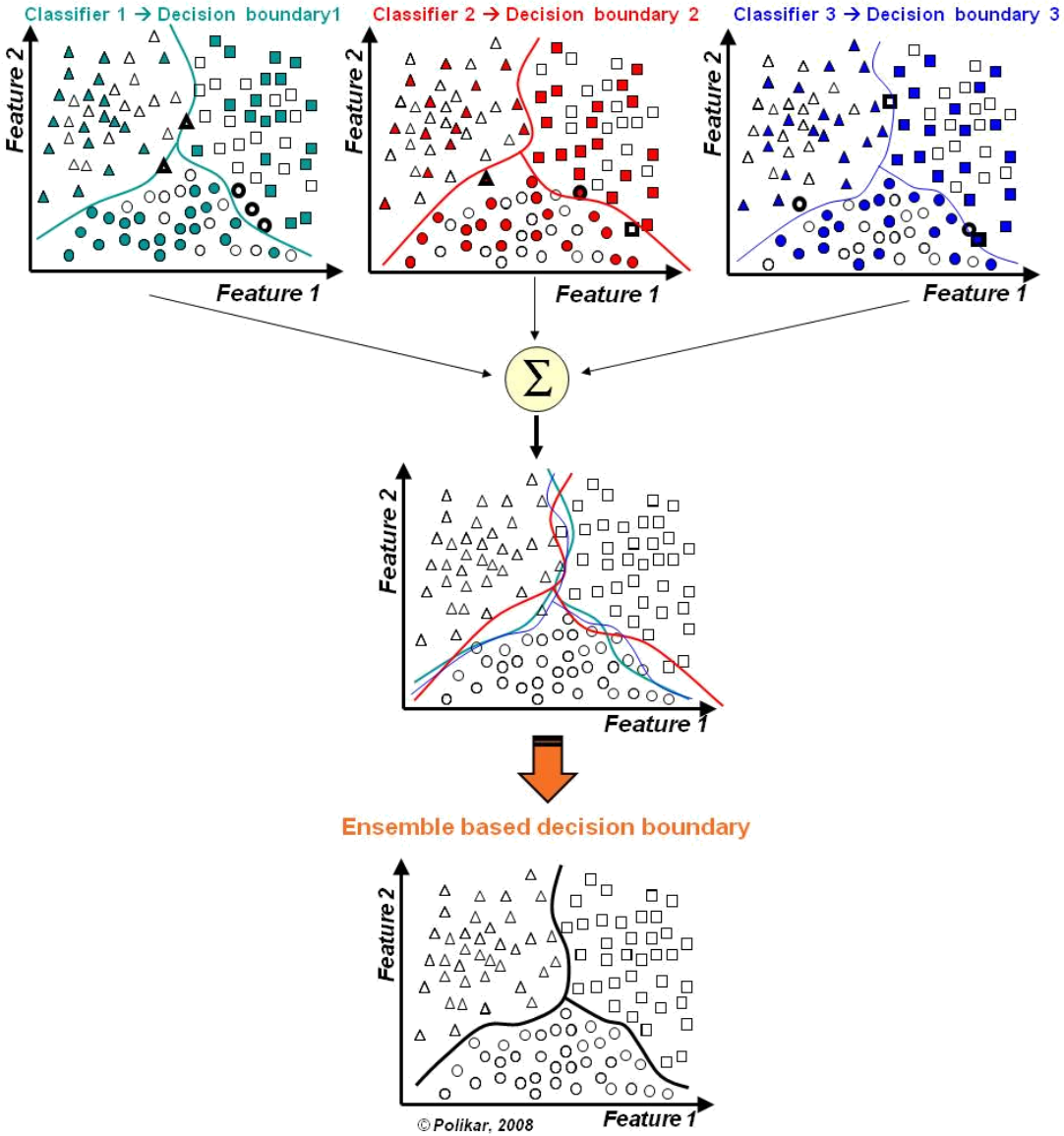
# Classification Methods

- k-Nearest Neighbors
- Decision Trees
- Naïve Bayes
- Support Vector Machines
- Logistic Regression
- Neural Networks
- Ensemble Methods (Boosting, Random Forests)

# Ensemble Learning

- Ensemble classification combines multiple classifiers to improve accuracy
- Use multiple methods to obtain better performance than any of the individual method
- Can combine outputs

# Ensemble Learning

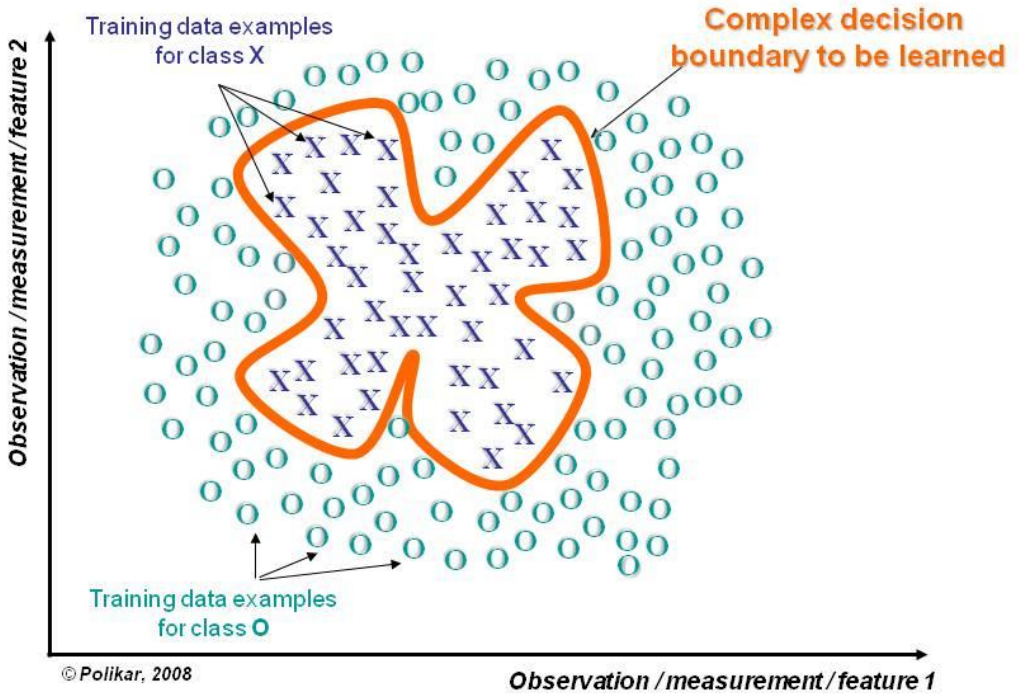



http://www.scholarpedia.org/wiki/images/8/82/Combining_classifiers2.jpg

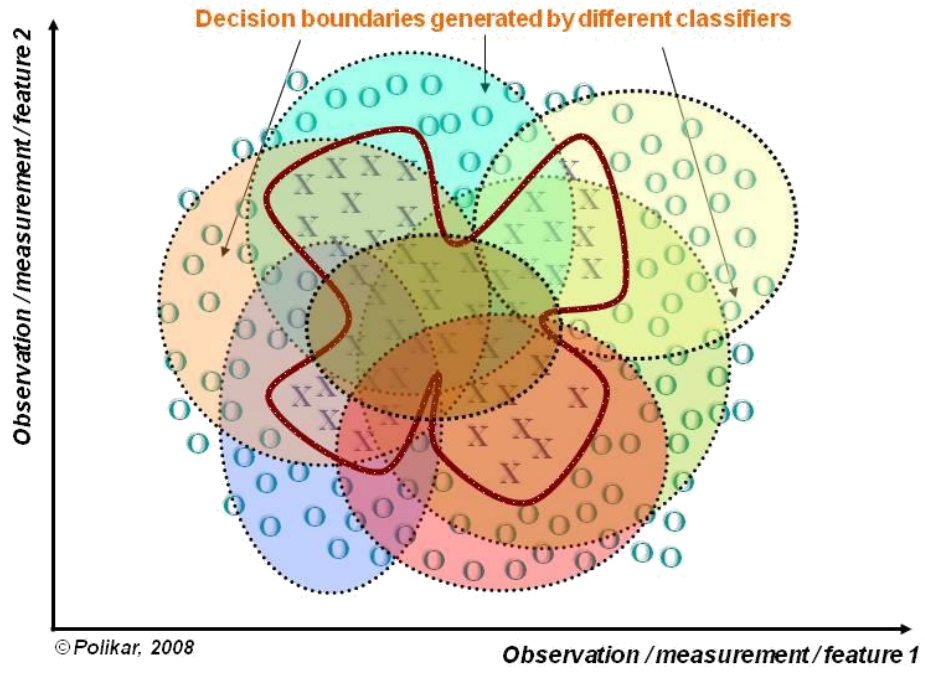
# Advantages

- Large datasets
  - May be too large for training single classifier
  - Can use different subsets of data to train multiple classifiers
- Small datasets
  - Can handle them with bootstrapping (random sampling)
- Some problems too complicated to solve with a single classifier
  - Eg. Complex separation between classes

# Divide and Conquer




http://www.scholarpedia.org/article/Image:Figure1a.jpg




http://www.scholarpedia.org/wiki/images/a/ab/Figure1b.jpg

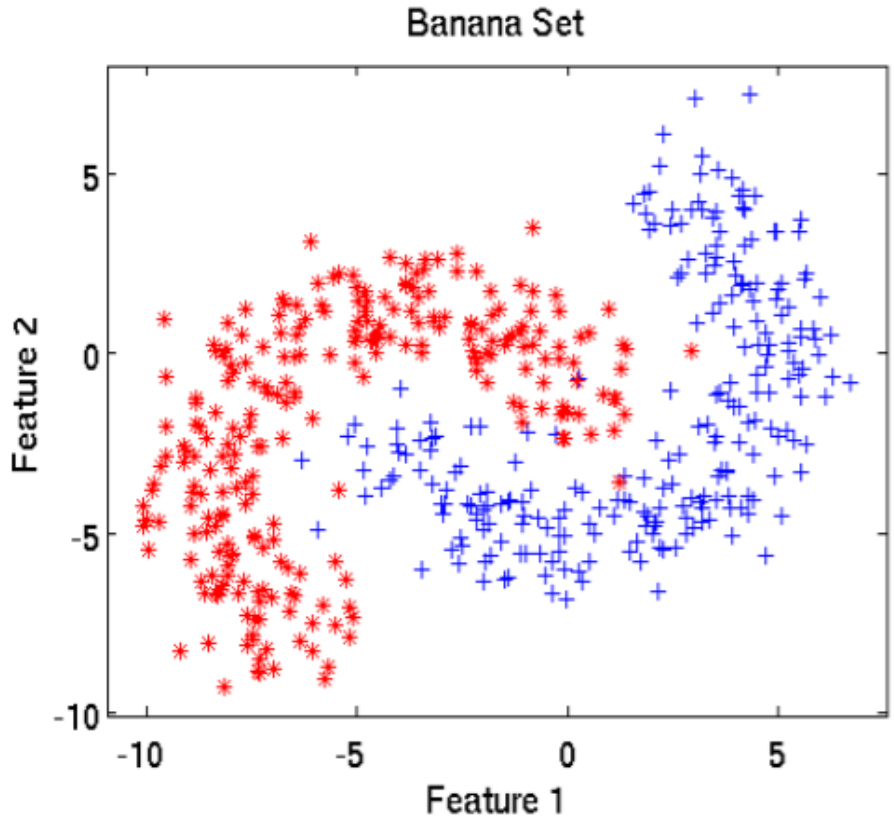
# Types of Ensemble Classifiers

- Bagging (bootstrap aggregating)
  - Train several models using bootstrapped datasets
  - The majority classification is selected
- Boosting
  - Use several weak classifiers to create a strong classifier
  - Resample previously misclassified points
- Stacking (stacked generalization)
  - Train multiple tiers of classifiers
  - Higher tiers can correct lower tiers

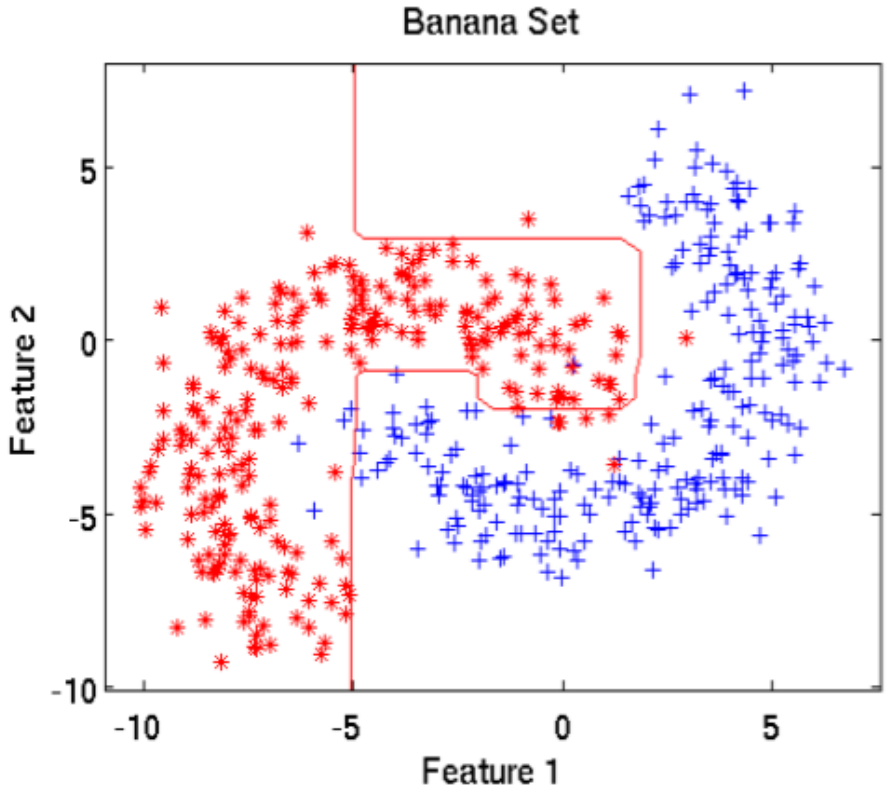
# Bagging

- **Problem:** we only have one dataset.
- **Solution:** generate new ones of size n by bootstrapping, i.e. sampling it with replacement
- Bagging works because it reduces variance by voting/averaging
- Usually, the more classifiers the better
- Some candidates:
  - Decision tree, decision stump, SVMs
  - Can do this with regression too: Regression tree, linear regression

# Bagging: Illustration

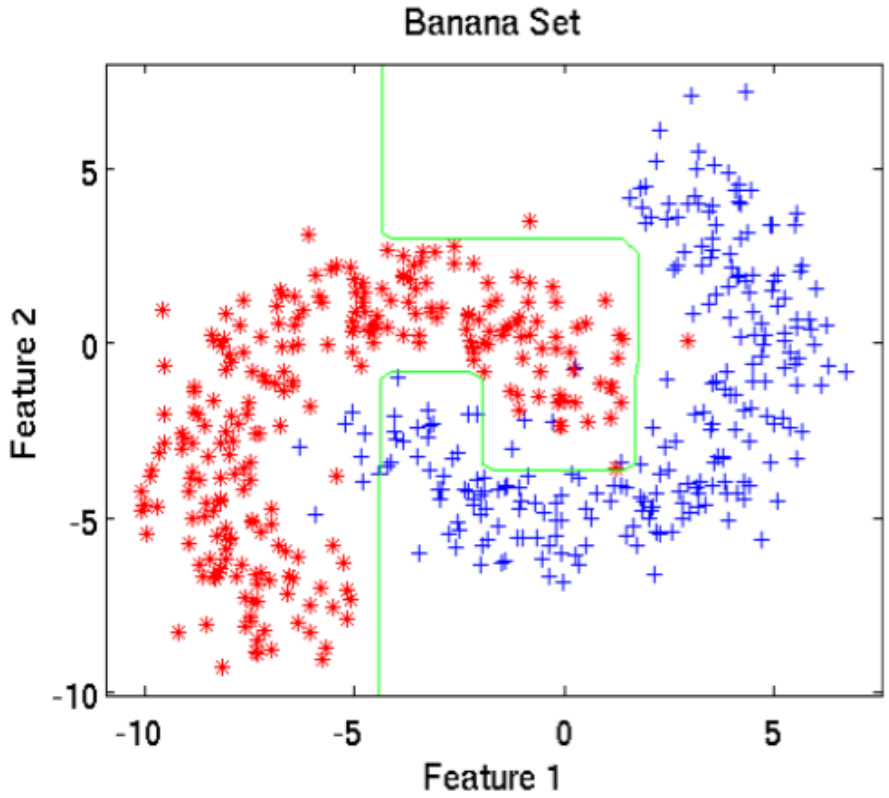



Training data

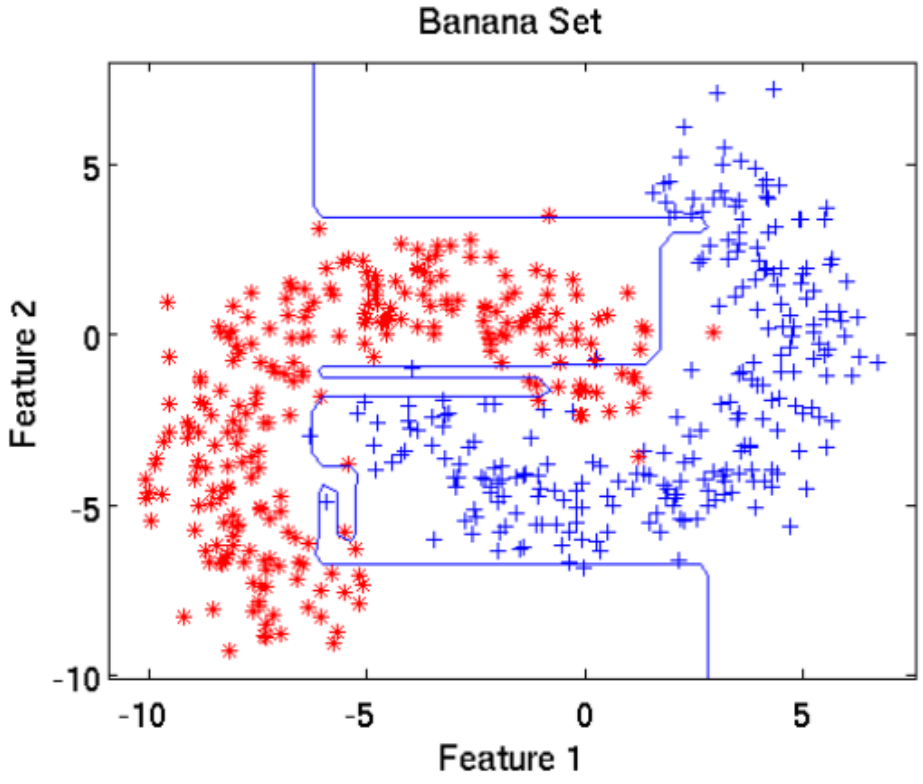



Decision boundary produced by one tree

# Bagging: Illustration

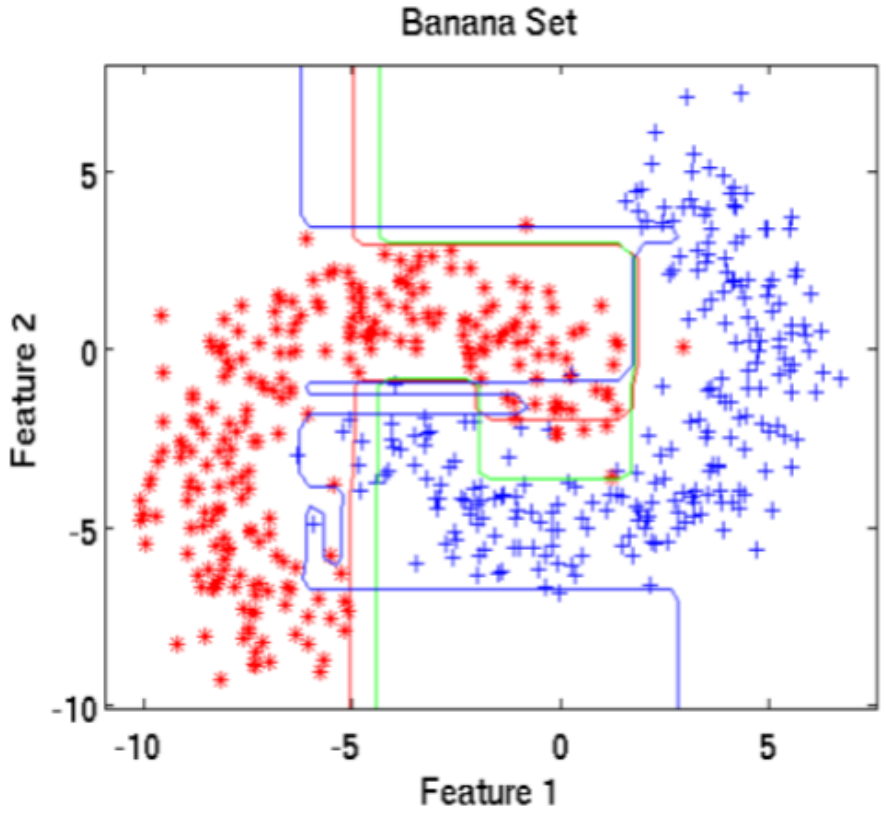



Decision boundary produced by a second tree

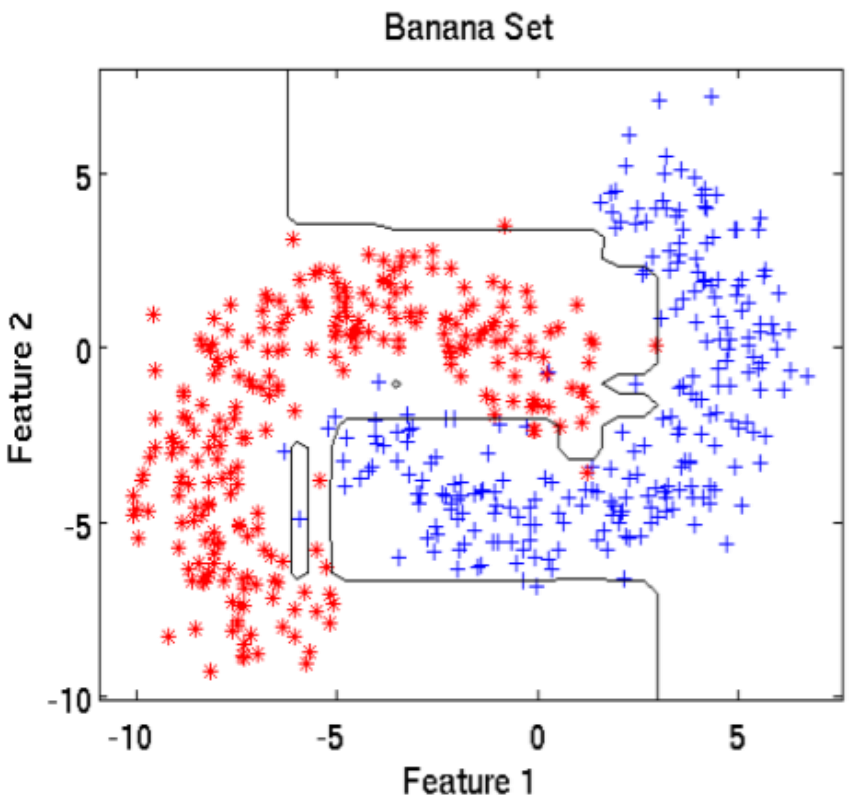



Decision boundary produced by a third tree

# Bagging: Illustration



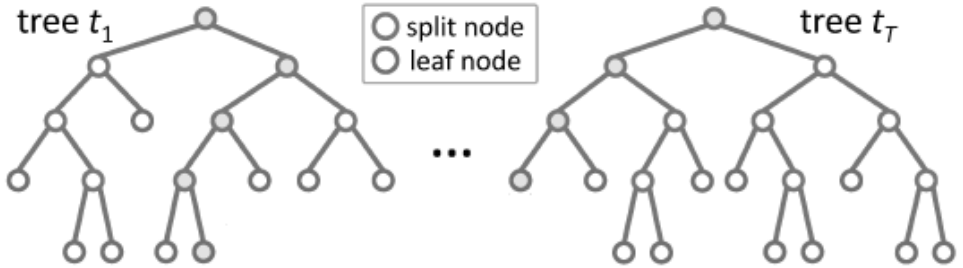

Three trees and final boundary overlaid




Final result from bagging all trees.

# Example: Random Forests

- Random forests (RF) are a combination of tree predictors
  - Variant of bagging, proposed by Breiman in 2001
- Extremely successful, especially on Kaggle challenges
- Each tree depends on the values of a random vector sampled independently
- The generalization error depends on the strength of the individual trees and the correlation between them
- Using a random selection of features yields results robust w.r.t. noise



# Random Forests: Algorithm

- Given a training set S
- For i = 1 to k do:
- Build subset $S_i$ by sampling with replacement from S
- Learn tree $T_i$ from $S_i$
- At each node:
- Choose best split from random subset of F features
- Each tree grows to the largest extent, and no pruning
- Make predictions according to majority vote of the set of k trees.

# Random Forests: Algorithm

- If there are M input variables, a number m is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
  - Depending upon the value of m, there are three slightly different systems:
    - Random splitter selection: m =1
    - Breiman's bagger: m = total number of predictor variables
    - Random forest: m << number of predictor variables. Breiman suggests three possible values for m: ½√m, √m, and 2√m
- Each tree is grown to the largest extent possible. There is no pruning.

# Bagging: When does it work?

- Can help if data is noisy
- If learning algorithm is unstable, i.e. if small changes to the training set cause large changes in the learned classifier

# Bagging: Why does it work?

- Let $S = \{(x_i, y_i), i = 1...N\}$ be the training dataset
- Let $\{S_k\}$ be a sequence of training sets containing a sub-set of *S*
- Let P be the underlying distribution of $S$.
- Bagging replaces the prediction of the model with the majority of the predictions given by the classifiers $S$

$$\varphi(x, P) = E_S(\varphi(x, S_k))$$

# Features of Random Forests

- One of the best in the business
- It runs efficiently on large data bases
- It can handle thousands of input variables without variable deletion/reduction
- It gives estimates of what variables are important in the classification
- Does not overfit by design
- The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them

# Types of Ensemble Classifiers

- Bagging (bootstrap aggregating)
  - Train several models using bootstrapped datasets
  - The majority classification is selected
- Boosting
  - Use several weak classifiers to create a strong classifier
  - Resample previously misclassified points
- Stacking (stacked generalization)
  - Train multiple tiers of classifiers
  - Higher tiers can correct lower tiers

# Boosting

- Strong Learner →Objective of machine learning
  - Take labeled data for training
  - Produce a classifier which can be *arbitrarily accurate*
- Weak Learner
  - Take labeled data for training
  - Produce a classifier which is more accurate than random guessing

Can a set of **weak learners** create a single **strong learner** ?

# Boosting: Key Idea

- An algorithm for constructing a "strong" classifier as linear combination of "simple" "weak" classifier

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

- Final classification based on weighted vote of weak classifiers

# Boosting: History and Significance

- Considered to be one of the most significant developments in machine learning
- Finding many weak rules of thumb is easier than finding a single, highly prediction rule
- Schapire (1989):
  - first provable boosting algorithm
  - call weak learner three times on three modified distributions
  - get slight boost in accuracy
  - apply recursively
- Freund (1990)
  - "optimal" algorithm that "boosts by majority"
- Drucker, Schapire & Simard (1992):
  - first experiments using boosting
  - limited by practical drawbacks
- Freund & Schapire (1995) – **AdaBoost**
  - strong practical advantages over previous boosting algorithms

# Adaboost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$

Initialise $D_1(i) = \frac{1}{m}$.

For $t = 1, \ldots, T$:

- Find the classifier $h_t : X \to \{-1, +1\}$ that minimizes the error with respect to the distribution $D_t$:

  $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$, where $\epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$

- Prerequisite: $\varepsilon_t < 0.5$, otherwise stop.
- Choose $\alpha_t \in \mathbf{R}$, typically $\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$ where $\varepsilon_t$ is the weighted error rate of classifier $h_t$.
- Update:

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

where $Z_t$ is a normalisation factor (chosen so that $D_{t+1}$ will be a distribution).

Output the final classifier:

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

Each training sample has a weight, which determines the probability of being selected for training the component classifier

# Reweighting

**Effect on the training set**

Reweighting formula:

$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{exp(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i))}{m \prod_{q=1}^{t} Z_q}$$

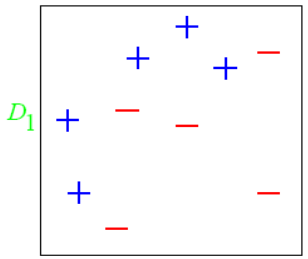
$$exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}$$

y * h(x) = 1

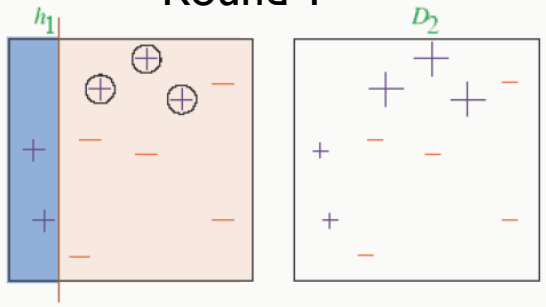
y * h(x) = -1

$\Rightarrow$ Increase (decrease) weight of wrongly (correctly) classified examples

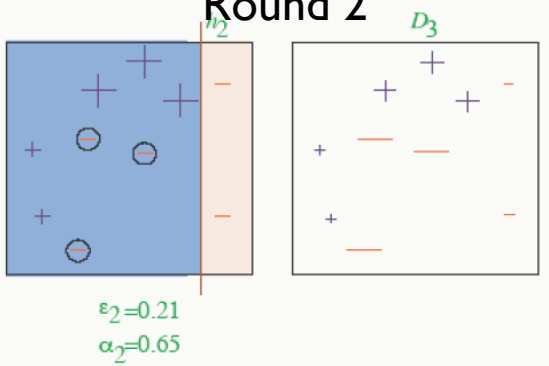
# Simple Example



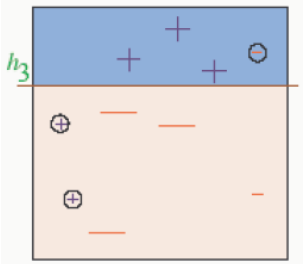

Round 1



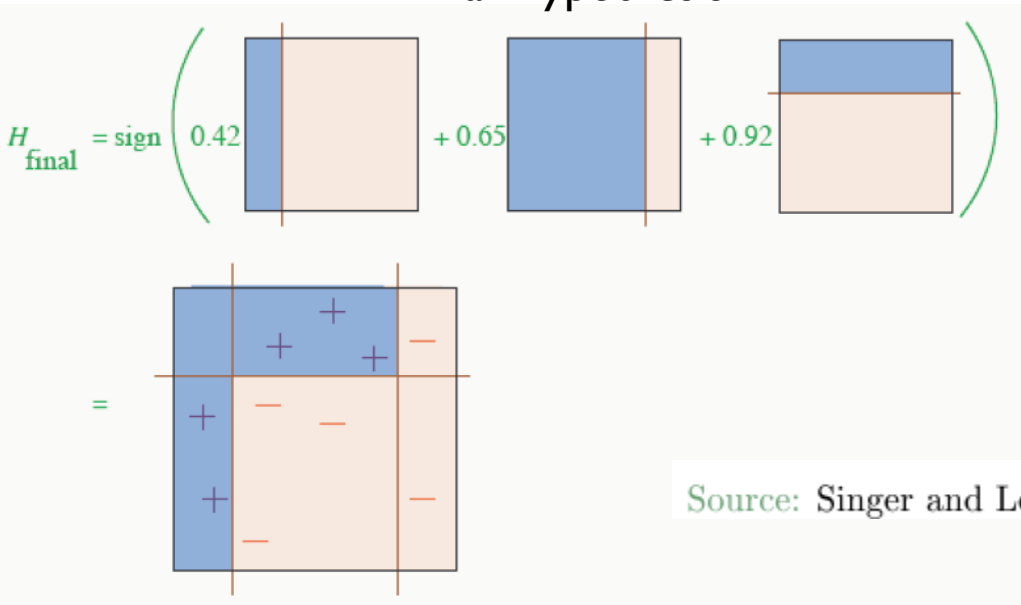

Round 2




Round 3



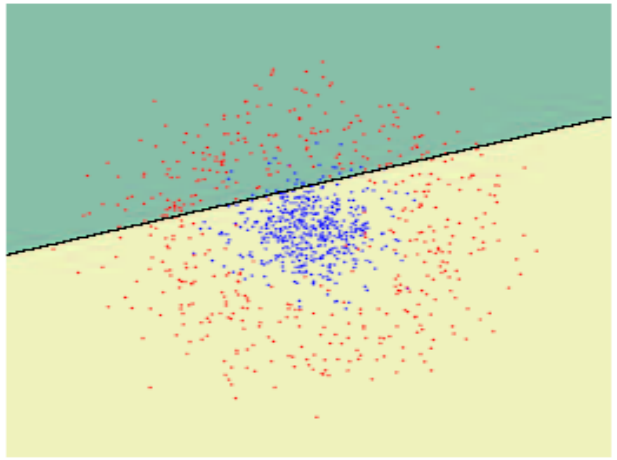

Final Hypothesis



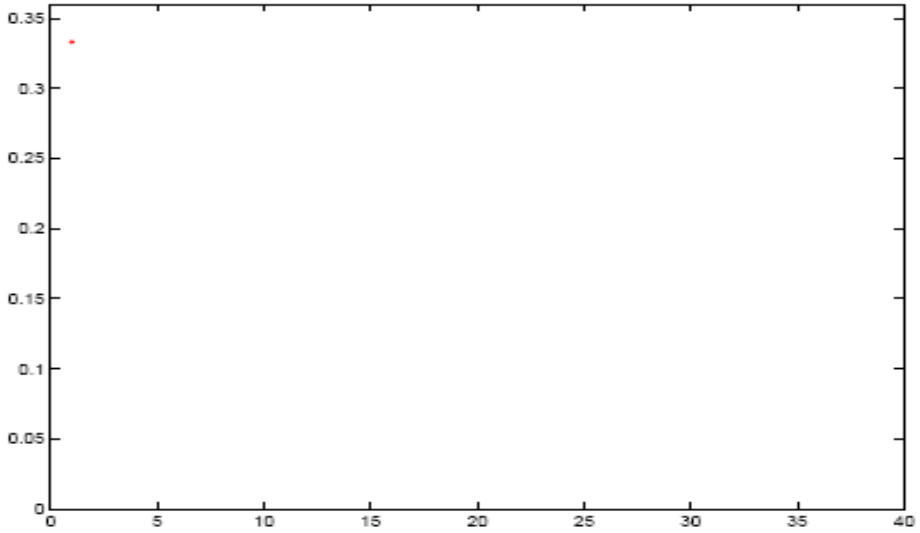

Source: Singer and Lewis Tutorial

# Illustration

$t = 1$

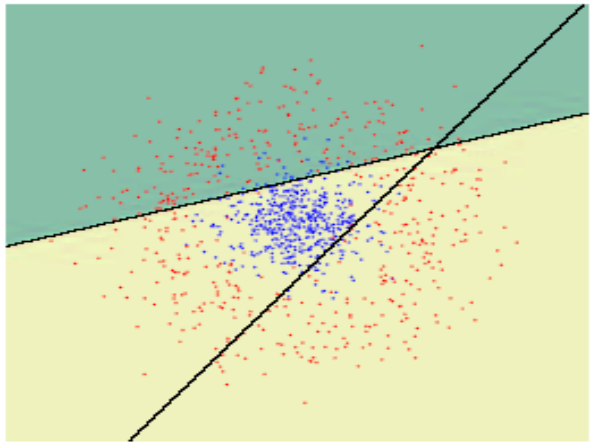


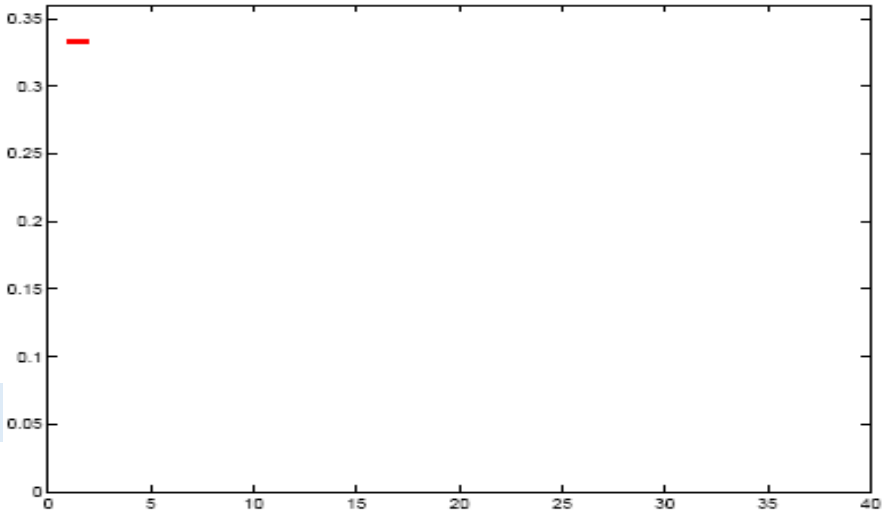
Sample plot of validation error



# Illustration

$t = 2$

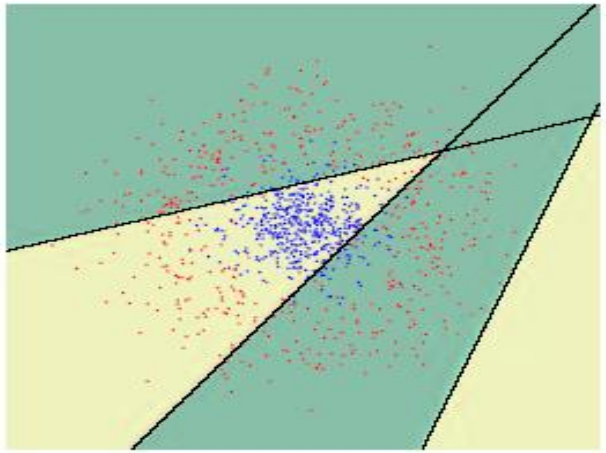


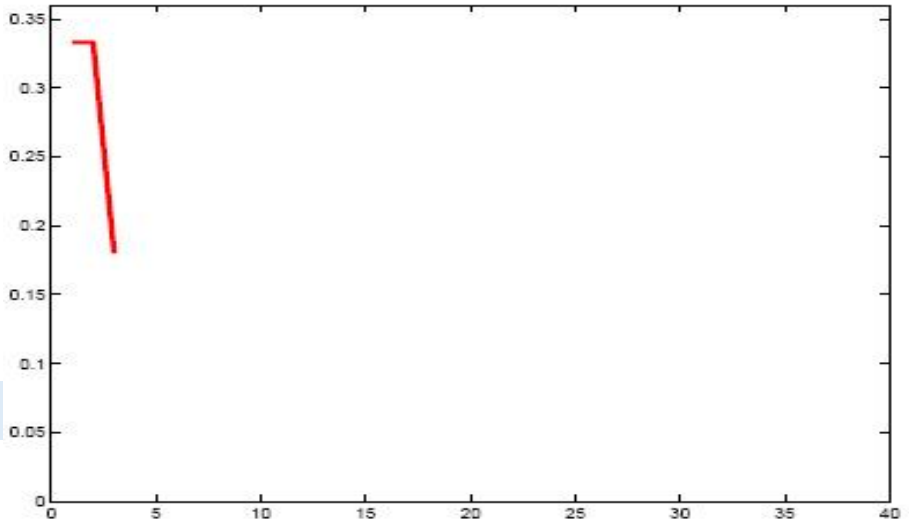



Sample plot of validation error

# Illustration

$t = 3$

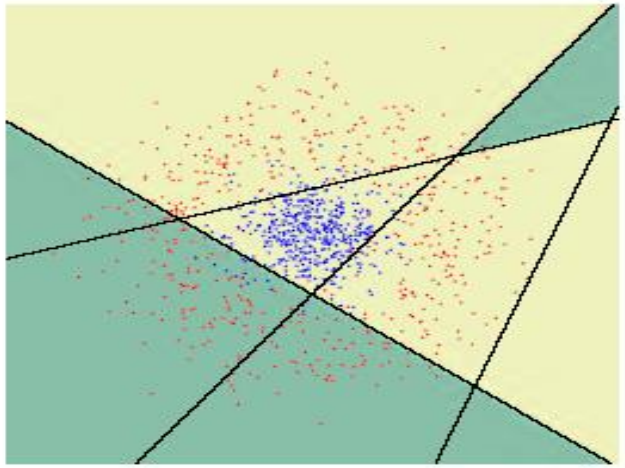


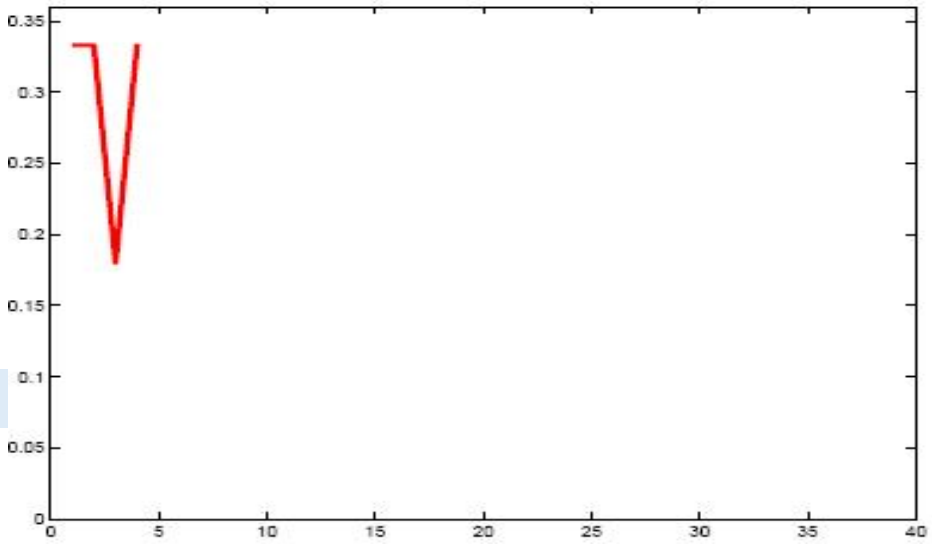



Sample plot of validation error

# Illustration

$t = 4$

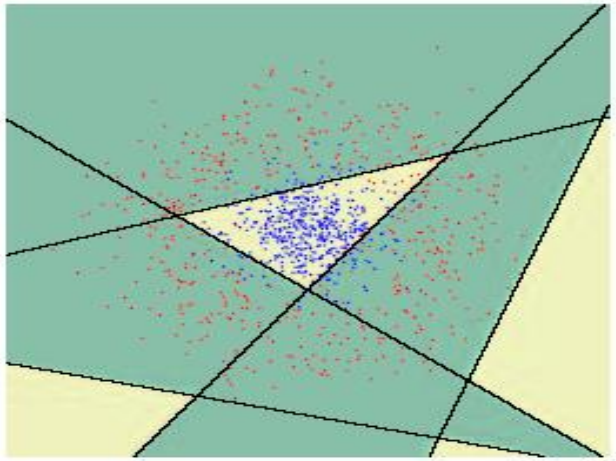


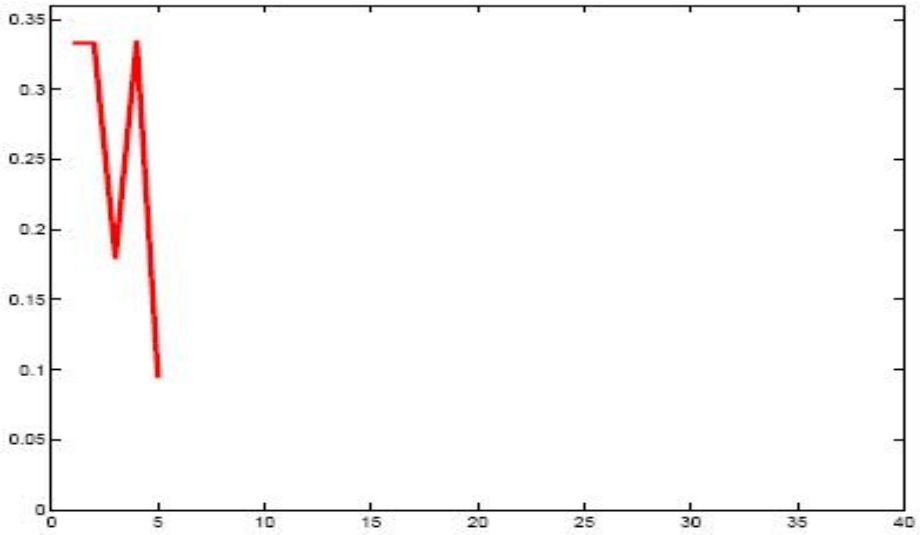



Sample plot of validation error

# Illustration

$t = 5$

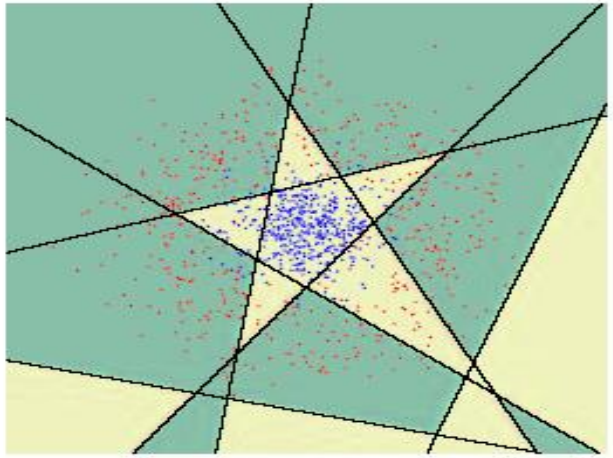


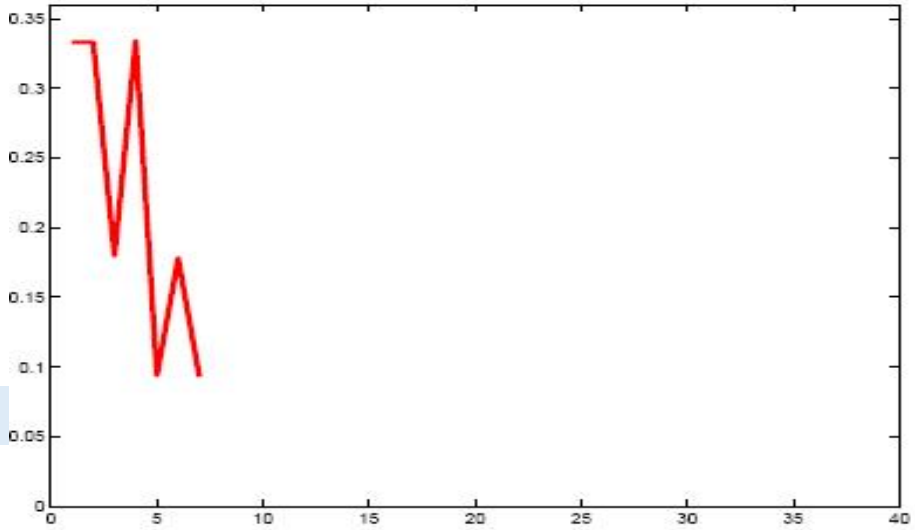



Sample plot of validation error

# Illustration

$t = 7$

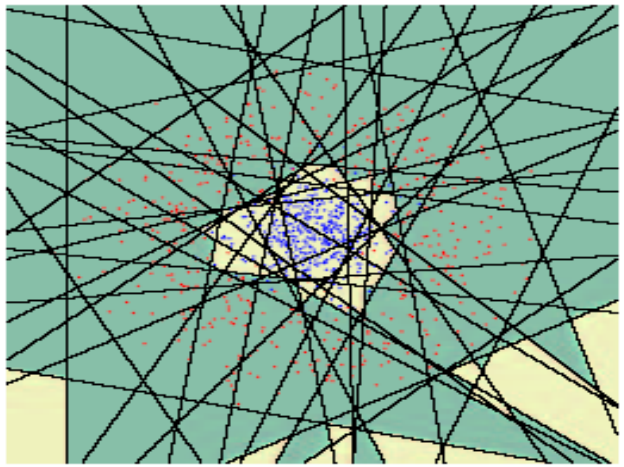


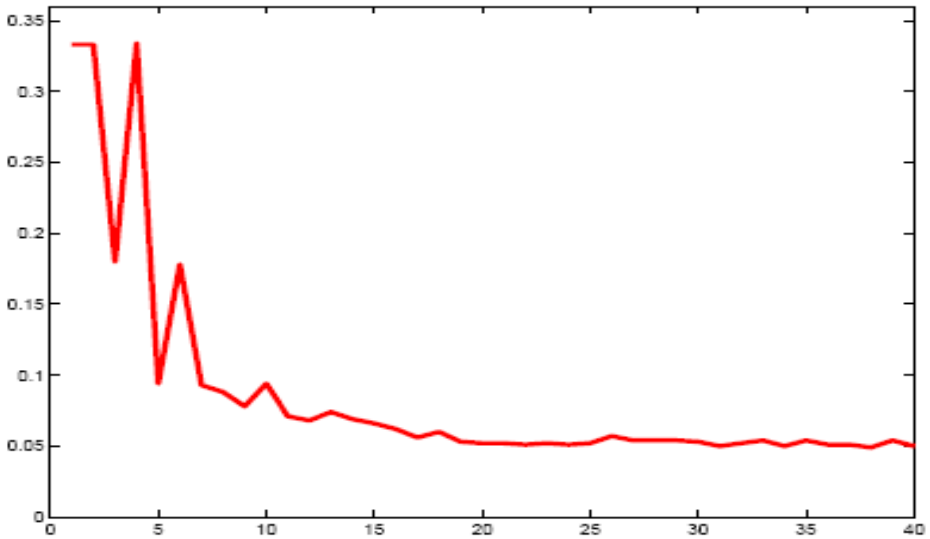



Sample plot of validation error

# Illustration

$t = 40$

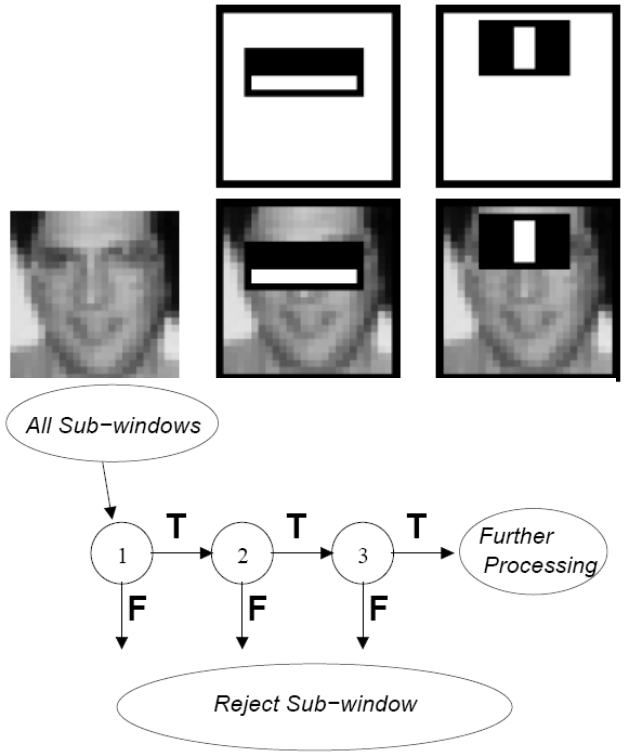




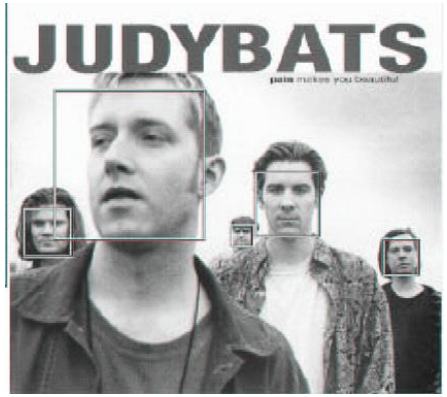

Sample plot of validation error

# Real-world Use of Boosting: Face Detection

Viola and Jones 2001




A landmark paper in vision!




More details: https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework

# Adaboost vs Random Forests

- Dietterich (1998) showed that
  - when a fraction of the output labels in the training set are randomly altered, the accuracy of Adaboost degenerates, while bagging is more immune to the noise.

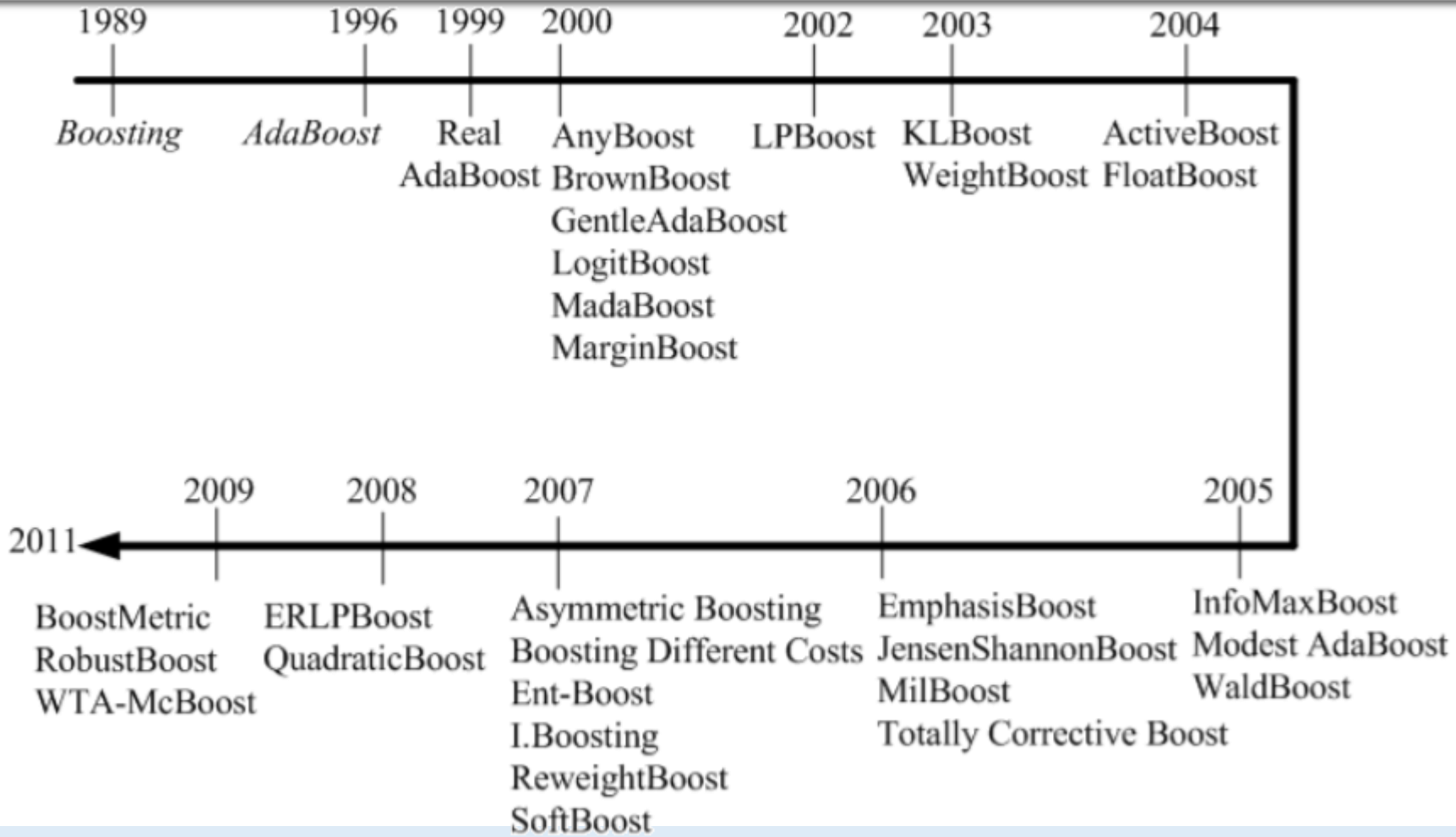
Increases in error rates due to noise

| Data set | Adaboost | Forest-RI |
|---|---|---|
| Glass | 1.6 | .4 |
| Breast cancer | 43.2 | 1.8 |
| Diabetes | 6.8 | 1.7 |
| Sonar | 15.1 | −6.6 |
| Ionosphere | 27.7 | 3.8 |
| Soybean | 26.9 | 3.2 |
| Ecoli | 7.5 | 7.9 |
| Votes | 48.9 | 6.3 |
| Liver | 10.3 | −.2 |

# What is a good weak learner?

- The set of weak rules (features) should be flexible enough to be (weakly) correlated with most conceivable relations between feature vector and label.
- Small enough to allow exhaustive search for the minimal weighted training error.
- Small enough to avoid over-fitting.
- Should be able to calculate predicted label very efficiently.
- Rules can be "specialists" – predict only on a small subset of the input space and abstain from predicting on the rest (output 0).

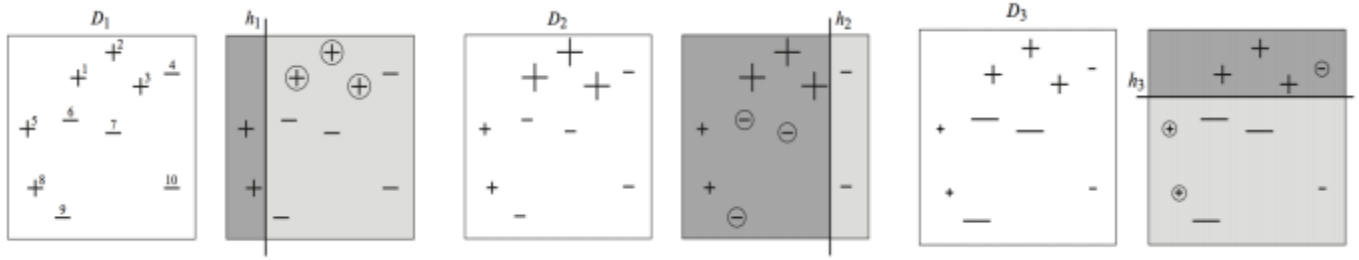
# Boosting Variants




Source: Boosting Algorithms: A Review of Methods, Theory, and Applications, Ferreira and Figueiredo

# Gradient Boosting

- **Gradient Boosting = Gradient Descent + Boosting**




- Fit an additive model (ensemble) in a forward stage-wise manner.
- In each stage, introduce a weak learner to compensate the shortcomings of existing weak learners.
- In Gradient Boosting, "shortcomings" are identified by gradients.
- Recall that, in Adaboost, "shortcomings" are identified by high-weight data points.
- Both high-weight data points and gradients tell us how to improve our model.

# Gradient Boosting

Input: training set $\{(x_i, y_i)\}_{i=1}^n$, a differentiable loss function $L(y, F(x))$, number of iterations $M$.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For $m$ = 1 to $M$:

   1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

   2. Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^n$

   3. Compute multiplier $\gamma_m$ by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

   4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

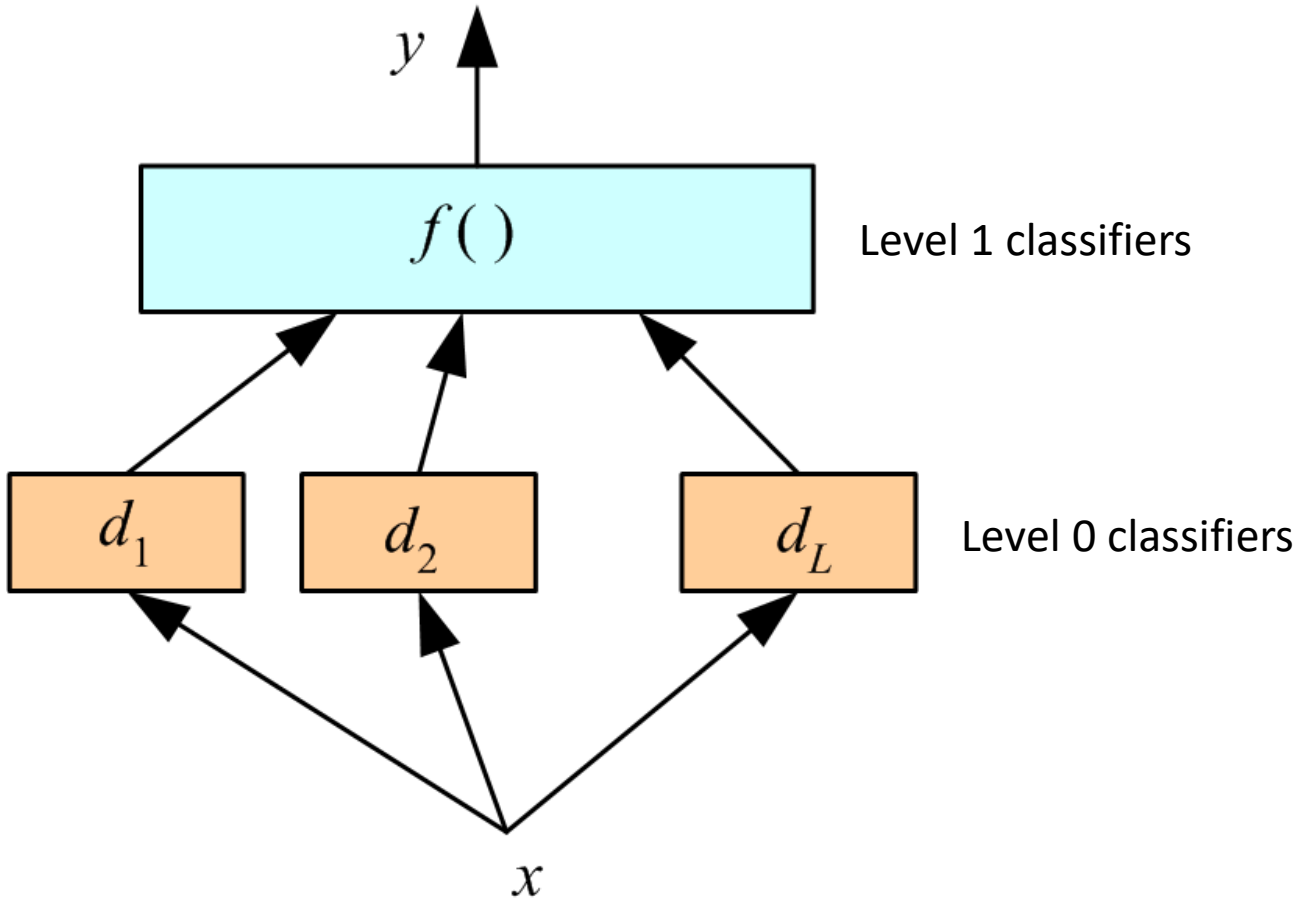
3. Output $F_M(x)$.

# Types of Ensemble Classifiers

- Bagging (bootstrap aggregating)
  - Train several models using bootstrapped datasets
  - The majority classification is selected
- Boosting
  - Use several weak classifiers to create a strong classifier
  - Resample previously misclassified points
- Stacking (stacked generalization)
  - Train multiple tiers of classifiers
  - Higher tiers can correct lower tiers

# Stacked Ensembles

- Combiner $f$ () is another learner (Wolpert, 1992)
- Idea:
  - Generate component (level 0) classifiers with part of the data (half, three quarters)
  - Train combiner (level 1) classifier to combine predictions of components using remaining data
  - Retrain component classifiers with all of training data
- In practice, often equivalent to voting



# Classification Methods

- k-Nearest Neighbors
- Decision Trees
- Naïve Bayes
- Support Vector Machines
- Logistic Regression (we will cover this during regression methods)
- Neural Networks
- Ensemble Methods (Boosting, Random Forests)

# Readings

- "Introduction to Machine Learning" by Ethem Alpaydin, Chapter 17
- Bishop, PRML (2006 edn), Sections 14.2-14.4