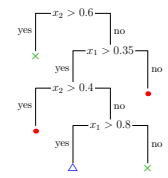
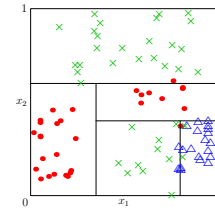


Decision Trees, Averaging, Bagging

1. Decision Trees
2. Bagging



## Removing Variance By Averaging

- We can reduce the variance and hence improve our generalisation error by averaging over different learning machines
- There are a number of different techniques for doing this that go by the name of **ensemble methods** or **ensemble learning**
- This trick can be used with many different learning machines, but is clearly most practical for machine that can be trained quickly
- (nevertheless, even for deep learning taking the average response of many machines is usually done to win competitions)

## Ensembling of Decision Trees

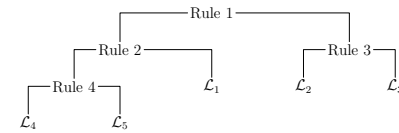
- One set of algorithms where ensembling are common place are decision trees
- These are particularly good for handling messy data
  - ★ categorical data
  - ★ mixture of data types
  - ★ missing data
  - ★ large data sets
  - ★ multiclass
- In many competitions ensembled trees, particularly *random forests* and *gradient boosting* beat all other techniques

## Decision Trees

- A decision trees builds a binary tree to partition the data,  $\mathcal{D} = \{(x_i, y_i) | i = 1, \dots, m\}$ , into the leaves of the tree
- Each decision rule depends on a single feature
- At each step the rule is chosen that maximise the “purity” of the leaf nodes
- Decisions can be made on numerical values or categories

## Partitioning

- Consider a classification problems with examples  $(x, y)$  belonging to some classes  $y \in \mathcal{C}$
- The data is partitioned by the tree into leaves



- The proportion of data points in leaf  $\mathcal{L}$  belonging to class  $c$  is

$$p_c(\mathcal{L}) = \frac{1}{|\mathcal{L}|} \sum_{(x,y) \in \mathcal{L}} \mathbb{I}[y = c]$$

where  $\mathbb{I}[y = c] = 1$  if  $y = c$  and 0 otherwise

## Leaf Purity

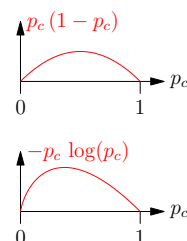
- Two different purity measures,  $Q_m(\mathcal{L})$ , for a leaf node  $\mathcal{L}$  are commonly used

★ Gini index

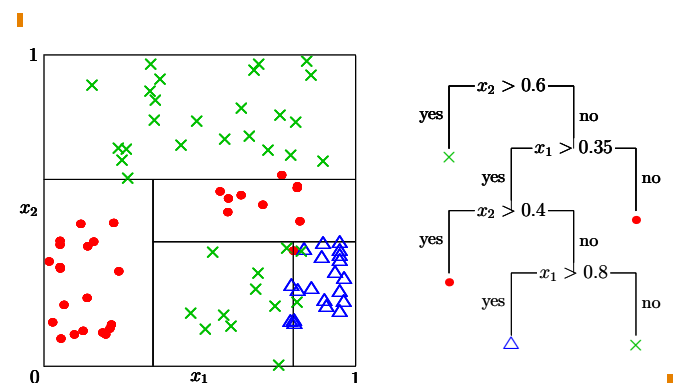
$$Q_m^g(\mathcal{L}) = \sum_{c \in \mathcal{C}} p_c(\mathcal{L}) (1 - p_c(\mathcal{L}))$$

★ Cross-entropy

$$Q_m^e(\mathcal{L}) = - \sum_{c \in \mathcal{C}} p_c(\mathcal{L}) \log(p_c(\mathcal{L}))$$



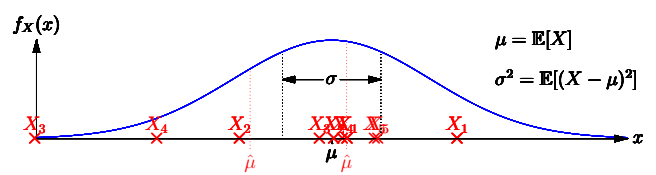
## Building Decision Trees



- Decision trees are very useful for exploring new data sets—the tree shows what features are most important
- Decision trees can also be used for regression problems
  - ★ Approximate function by a series of rules
  - ★ Reduce variance between data points assigned to leaf nodes
- CART is a classic implementation that builds **C**lassification **A**nd **R**egression **T**rees
- Decision trees depend strongly on the early decisions and so vary a lot for slightly different data sets—high variance

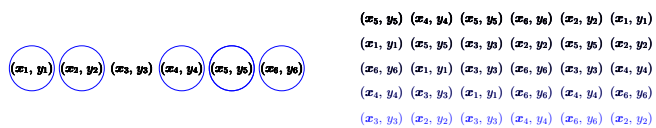
## Error In The Means

- By taking the mean over many samples we can reduce the variance and thus improve our generalisation performance
- To get a feel for this consider estimating the mean of a random variable,  $X$ , from a number of samples ( $n = 5$  in the example below)



## Bootstrap Aggregation (Bagging)

- To reduce the variance in a learning machine (such as a decision tree) we can average over many machines
- To average many machines they must learn something different
- We only have one data set, but we can resample from the data set to make them look a bit different—this is known as **bootstrapping**



## Variance of Positive Correlated Variables

- If we calculate the variance of the mean of positively correlated variables with correlation  $\rho$  we find

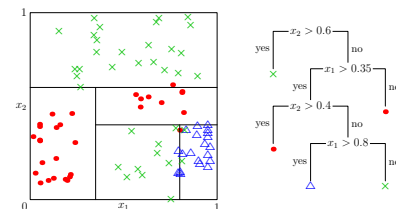
$$\frac{1}{n^2} \mathbb{E} \left[ \left( \sum_{i=1}^n X_i - n\mu \right)^2 \right] = \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2$$

$$(\rho = \mathbb{E}[(X_i - \mu)(X_j - \mu)]/\sigma^2)$$

- As  $n \rightarrow \infty$  the second term vanishes, but we are left with the first term
- If we want to do well we need our learning machines to be unbiased and decorrelated

## 1. Decision Trees

## 2. Bagging



## Mean and Variance

- The expected value of the mean,  $\hat{\mu}_n$ , of  $n$  random **independent** variables,  $X_i$ , is the expected value  $\mu = \mathbb{E}[X_i]$

$$\mathbb{E}[\hat{\mu}_n] = \mathbb{E} \left[ \frac{1}{n} \sum_{i=1}^n X_i \right] = \frac{1}{n} \sum_{i=1}^n \mathbb{E}[X_i] = \frac{1}{n} \sum_{i=1}^n \mu = \mu$$

- The variance is  $\mathbb{E}[(\hat{\mu}_n - \mu)^2]$  or equivalently

$$\begin{aligned} \frac{1}{n^2} \mathbb{E} \left[ \left( \sum_{i=1}^n (X_i - \mu) \right)^2 \right] &= \frac{1}{n^2} \mathbb{E} \left[ \sum_{i=1}^n (X_i - \mu)^2 + \sum_{i=1}^n \sum_{j \neq i}^n (X_i - \mu)(X_j - \mu) \right] \\ &= \frac{1}{n^2} \sum_{i=1}^n \left( \mathbb{E}[(X_i - \mu)^2] + \sum_{j=1, j \neq i}^n \mathbb{E}[X_i - \mu] \mathbb{E}[X_j - \mu] \right) \\ &= \frac{1}{n^2} \sum_{i=1}^n \sigma^2 = \frac{1}{n} \sigma^2 \end{aligned}$$

## Performance of Bagging

- Bootstrapping is an early form of data augmentation
- For classification we get our different machines to vote
- For regression we can average the prediction of different machines
- Bagging improves the performance of decision trees
- However, we can usually do better using Boosting
- This is because our decision trees are correlated

## Random Forest

- In random forests we average much less correlated trees
- To do this for each tree we choose a subset of  $p' \ll p$  of the features on which to split the tree
- Typically  $p'$  can range from 1 to  $\sqrt{p}$
- The trees aren't that good, but are very decorrelated
- By averaging over a huge number of trees (order of 1000) we typically get good results
- Random Forest won (wins?) many competitions

## Lessons

- Ensemble methods have proved themselves to be very powerful
- They work by averaging over different machines, trying to reduce their variance
- Here the variance comes from forcing the machines to learn different functions using Bootstrap Aggregation
- Tend to work best with very simple models (true of random forest and boosting)—seems to reduce over-fitting
- Random forest is very powerful, but gradient boosting is competitive