Classification Overview

Supervised Learning

- Given a set of data points with an outcome, create a model to describe them

- Classification
  - outcome is a discrete variable (typically <10 outcomes)

- Regression
  - outcome is continuous

Training Data

$$\left.\begin{array}{cc} \underbrace{\begin{array}{c} x_{11}x_{12}x_{13} \\ x_{21}x_{22}x_{23} \\ \dots \\ x_{m1}x_{m2}x_{m3} \end{array}}_{\text{Input}} & \underbrace{\begin{array}{c} y_1 \\ y_2 \\ \\ y_m \end{array}}_{\text{Output/Label}} \end{array}\right\} \text{Data set.}$$

Each y_i was generated by equation $f(x_i) = y_i$, and more generally $f(x) = y$.

Since $f(x)$ is unknown, machine learning aims to discover a function $h(x)$ that approximates it

Inductive Learning

- Given a set of observations come up with a model, h, that describes them

- What does “describes” mean?
  - h is the same as the function f that generated them

Inductive Learning

- Given a set of observations come up with a model, h, that describes them

- What does “describes” mean?
  - ~~h is the same as the function f that generated them~~
  - h models the observations well, and is *likely to predict future observations well*

Avoiding Overfitting the Model

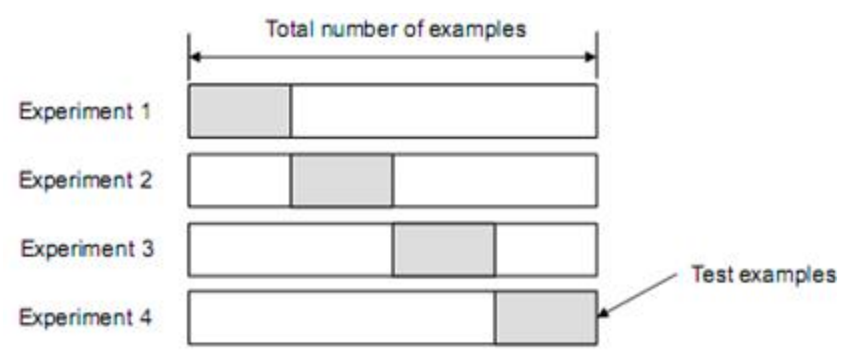
1. Divide the data that you have into a distinct training set and test set.
2. Use only the training set to train your model.
3. Verify performance using the test set.
    - Measure error rate

- Drawback of this method: the data withheld for the test set is not used for training
    - 50-50 split of data means we didn't train on half the data
    - 90-10 split means we might not get a good idea of the accuracy

K-fold Cross-Validation

1. Divide the data into k equal subsets.
2. Run learning k times, each time leave out $\frac{1}{k}$ of the data (1 set) for testing and use the rest for training
3. The average error rate of all k rounds is a better estimate of the model accuracy.

- k is usually 5 or 10.
- $k = n$ (number of samples) is "leave-one-out cross-validation"

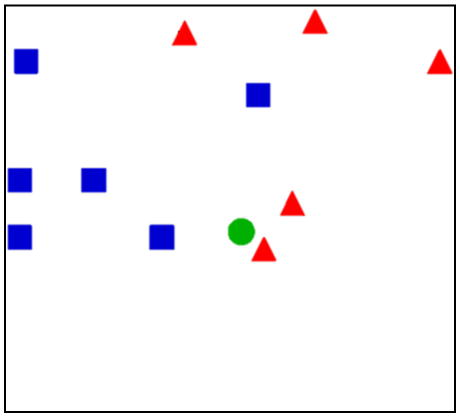


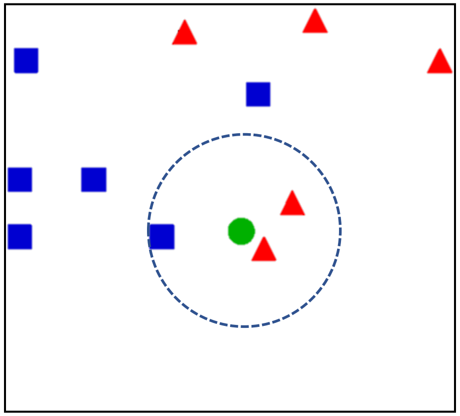
Classification Algorithms

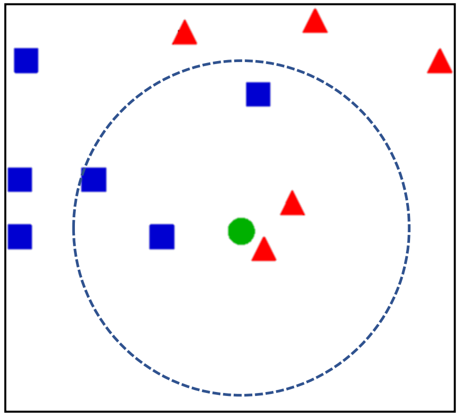
- Nearest neighbors
- Decision trees
- Neural networks
- Support Vector Machines
- Random Forest
- ADA Boost
- …

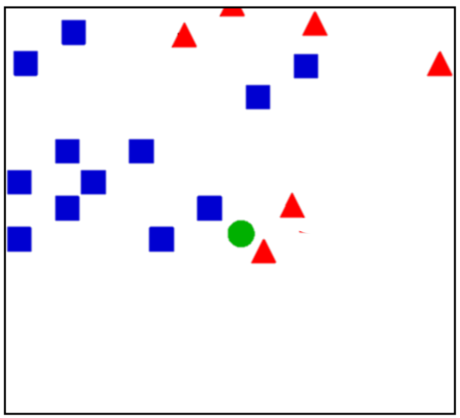
K-Nearest Neighbors

- What value do we assign to the green sample?





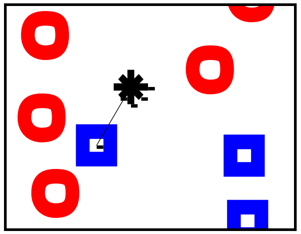




K-Nearest Neighbors

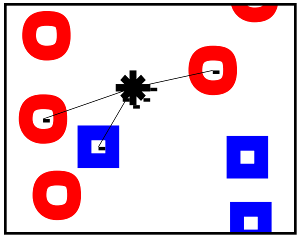
- 1-NN:
  - For a given query point q, assign the class of the nearest neighbour.

k = 1

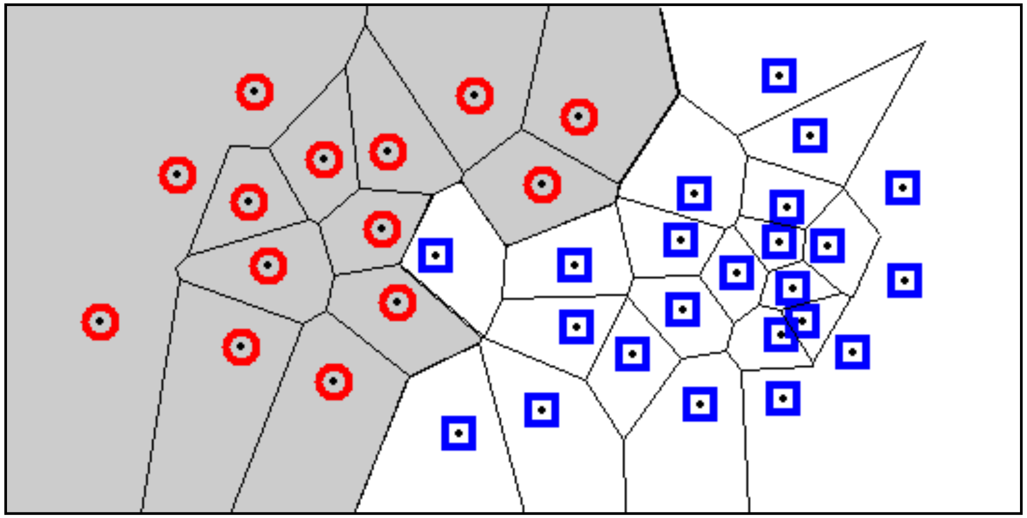


- K-NN
  - Compute the k nearest neighbours and assign the class by majority vote.

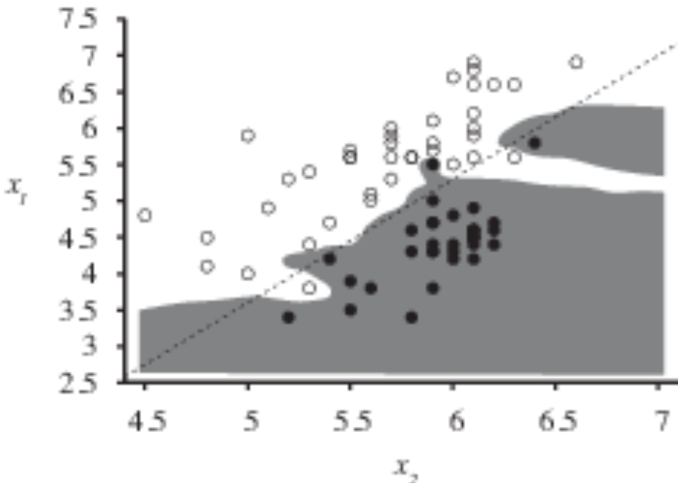
k = 3



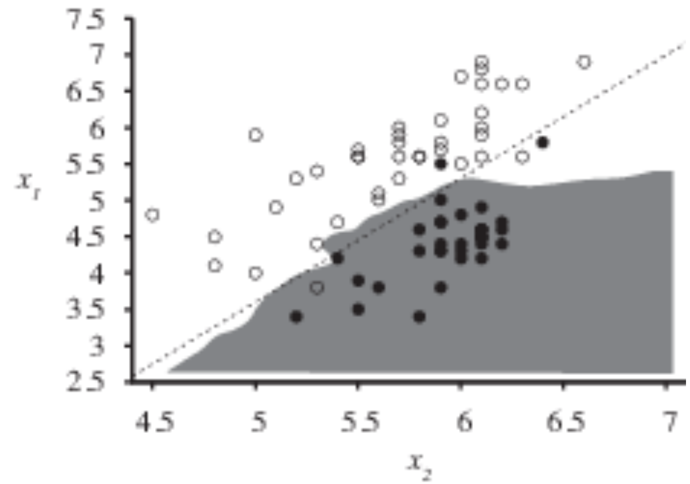
Decision Regions for 1-NN



Effect of k



$k = 1$



$k = 5$

K-Nearest Neighbors

- Euclidian Distance:

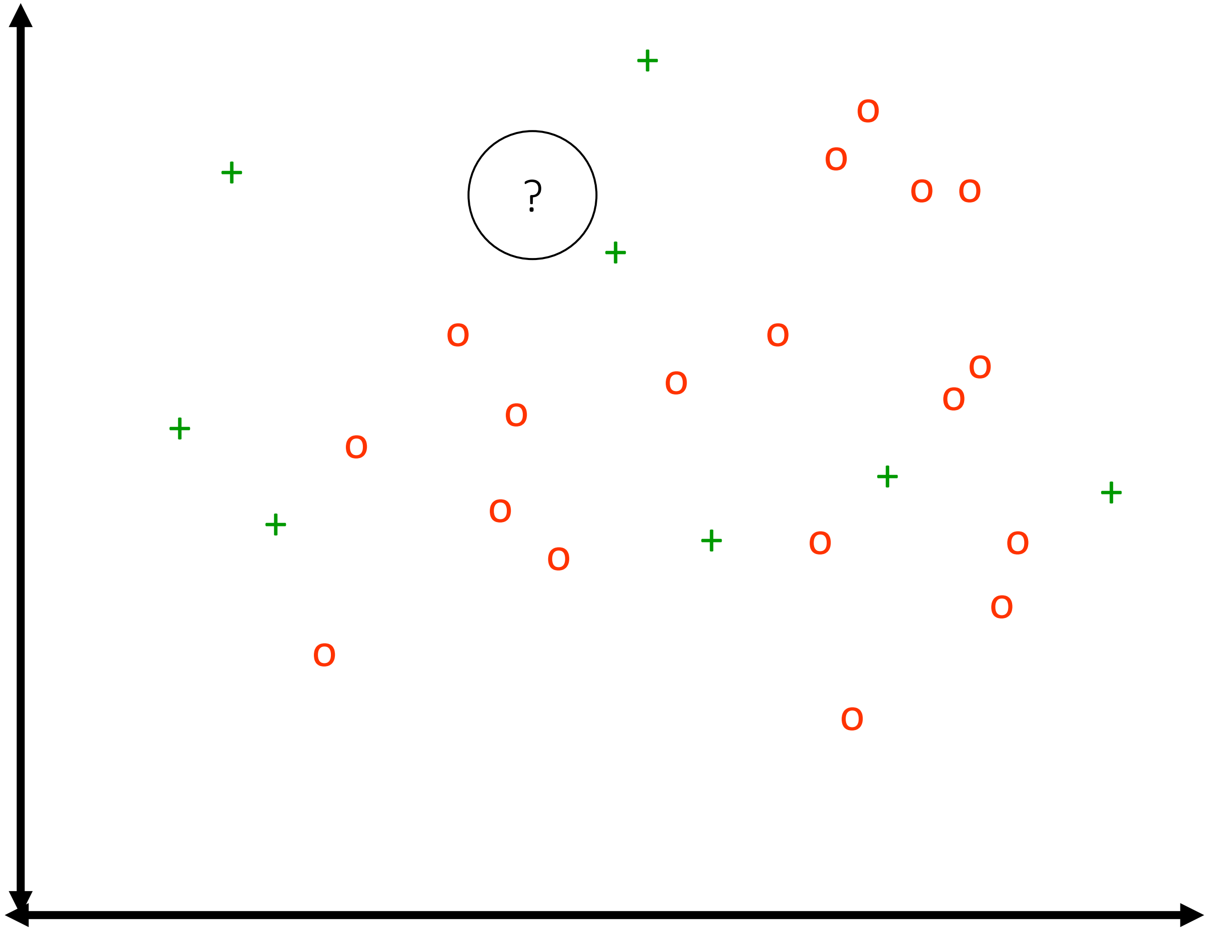
$$D(x_1, x_2) = \frac{\sum_i \left(x_{1,i} - x_{2,i}\right)^2}{2}$$

- Weighted Euclidian Distance:

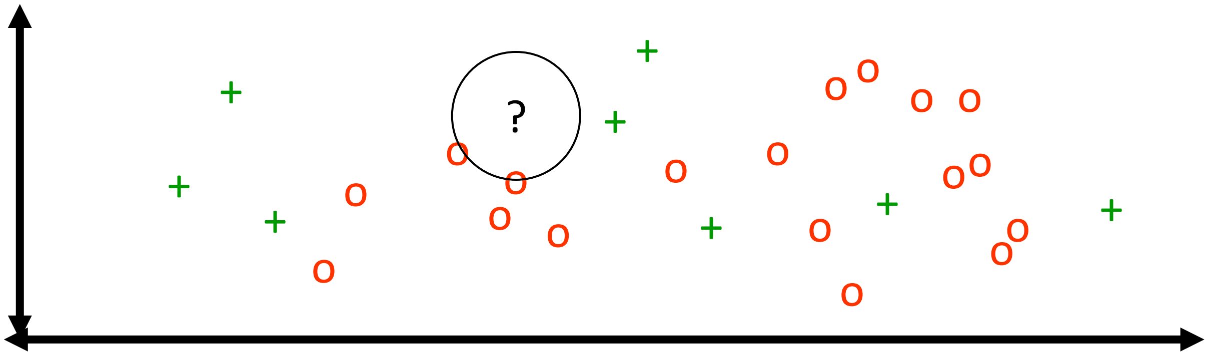
$$D(x_1, x_2) = \frac{\sum_i w_i \left(x_{1,i} - x_{2,i}\right)^2}{2}$$

Where i is the dimensionality of the data.

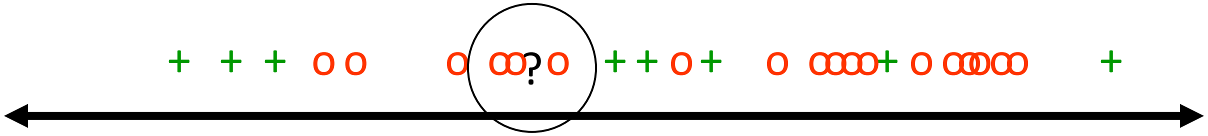
Weighting the Distance to Remove Irrelevant Features



Weighting the Distance to Remove Irrelevant Features

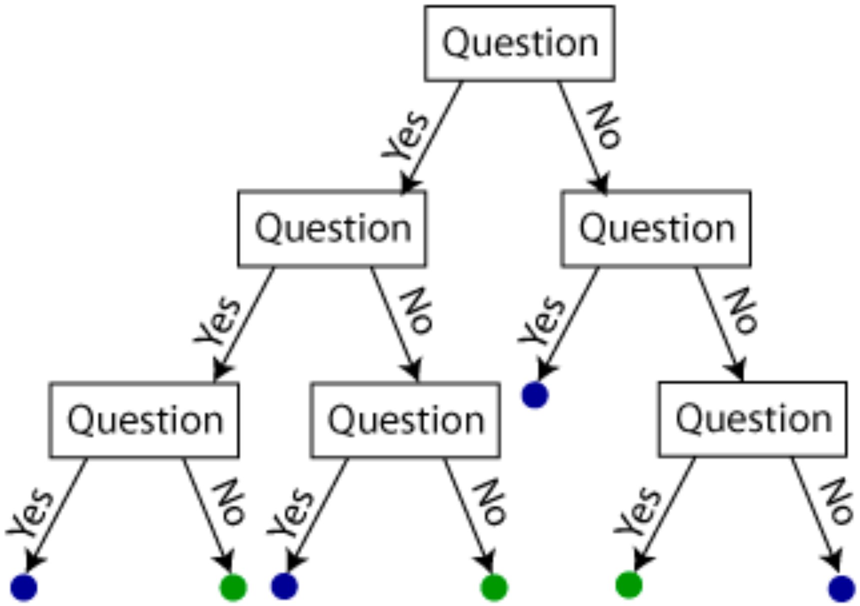


Weighting the Distance to Remove Irrelevant Features



Decision Trees

- …similar to a game of 20 questions

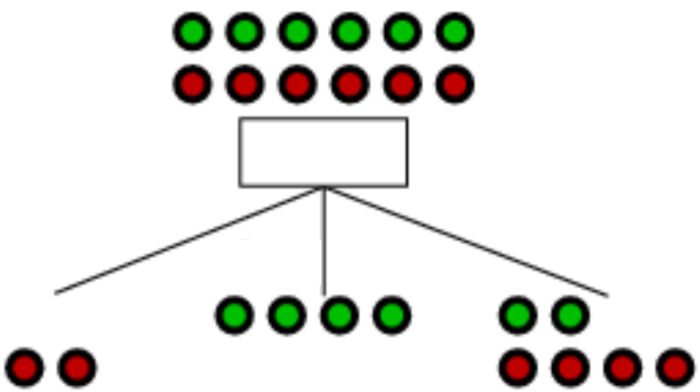


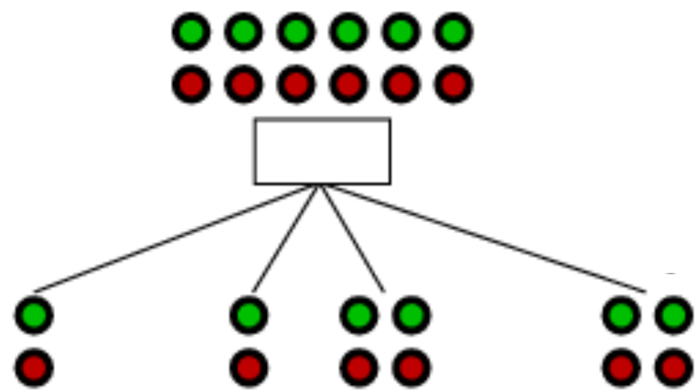
Example: road signs

- Determine feature/attribute vector used to represent signs
- Extract features for each sign
- Construct tree by splitting on features/attributes in order of importance

Choosing an attribute

- a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"





better choice

Using information theory

- Implementation of Choose-Attribute in the DTL algorithm based on information content – measured by Entropy

- Entropy is the measure of uncertainty of a random variable
  - More uncertainty leads to higher entropy
  - More knowledge leads to lower entropy

Entropy

- Entropy is the measure of uncertainty of a random variable
  - More uncertainty leads to higher entropy
  - More knowledge leads to lower entropy

$$H(V) = -\sum_{k} P(v_k) \log_2 P(v_k)$$

Fair coin flip:

$$H(tails) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$$

Biased coin flip:

$$H(tails) = -(0.99 \log_2 0.99 + 0.01 \log_2 0.01) = 0.08$$

Decision Trees and Information Gain

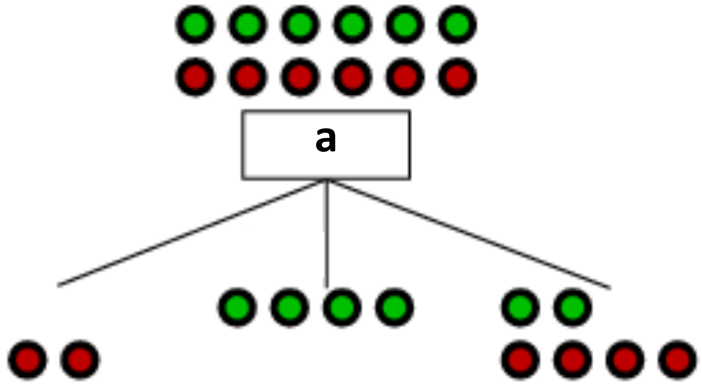
- Choose the attribute that leads to the greatest expected Information Gain

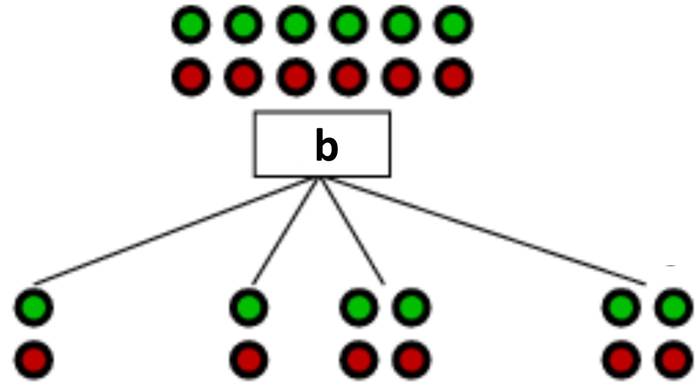
- The information gain from an attribute test is the expected reduction in entropy

$$IG(T,a) = H(T) = H(T|a)$$

- If attribute a is an attribute that can take on d distinct values:

$$IG(T,a) = H(T) - \sum_{k=1}^{d} \frac{|\{x \in T | x_a = k\}|}{|T|} H(\{x \in T | x_a = k\})$$





$$IG(T,a) = 1 - \left[\frac{2}{12}H\left(\frac{0}{2}\right) + \frac{4}{12}H\left(\frac{4}{4}\right) + \frac{6}{12}H\left(\frac{2}{6}\right)\right] = 0.0541$$

$$IG(T,b) = 1 - \left[\frac{2}{12}H\left(\frac{1}{2}\right) + \frac{2}{12}H\left(\frac{1}{2}\right) + \frac{4}{12}H\left(\frac{2}{4}\right) + \frac{4}{12}H\left(\frac{2}{4}\right)\right] = 0$$

$$IG(T,a) = H(T) - \sum_{k=1}^{d} \frac{|\{x \in T | x_a = k\}|}{|T|} H(\{x \in T | x_a = k\})$$

Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error does not provide a good estimate of how well the tree will perform on previously unseen records (need a test set)
- Solutions:
  - Pruning
  - Early stopping

Disadvantage of Decision Trees

- Can be sensitive to data, small changes causing significant changes in the model
- Balancing between overfitting and generalization can be tricky

Random Forest

(ensemble learning method)

- Create many (hundreds, thousands) of *simpler* trees, to avoid overfitting any one tree
  - Given training set X, select a random subset of *features*, and fit a tree to these samples using only those features
  - Repeat until desired number of trees is reached

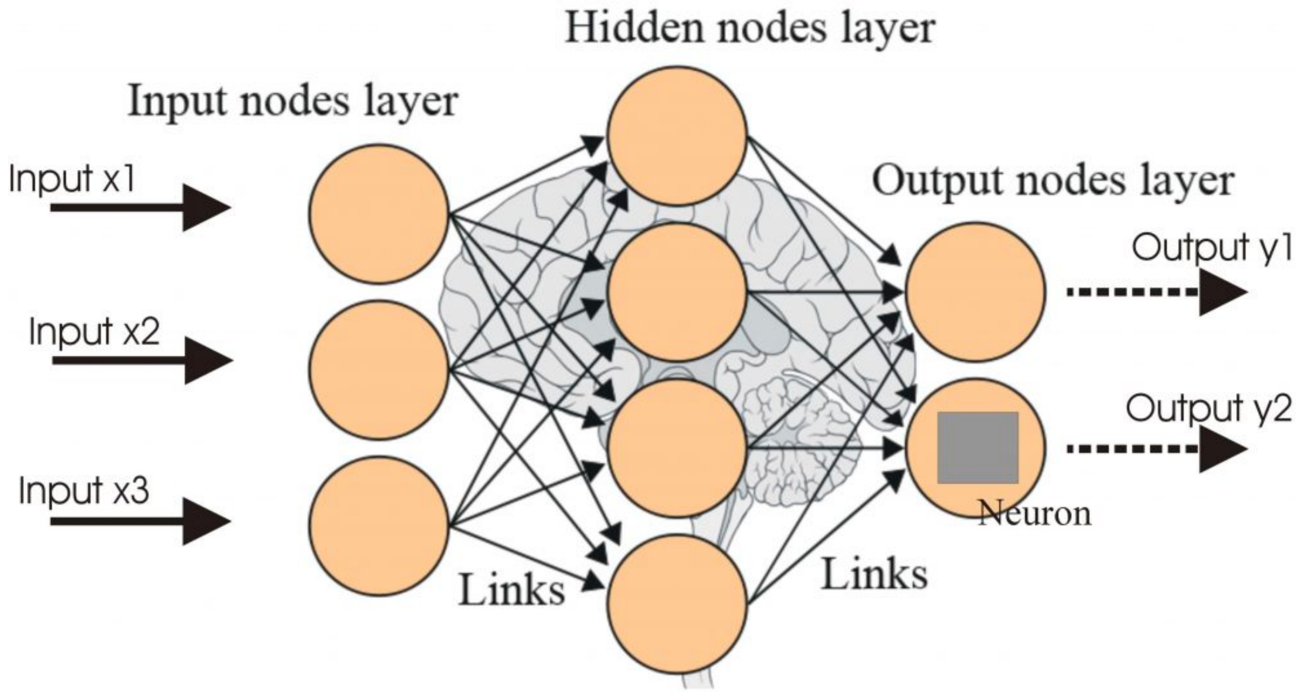
- Report the result obtained from majority vote by the collection of trees

Boosting

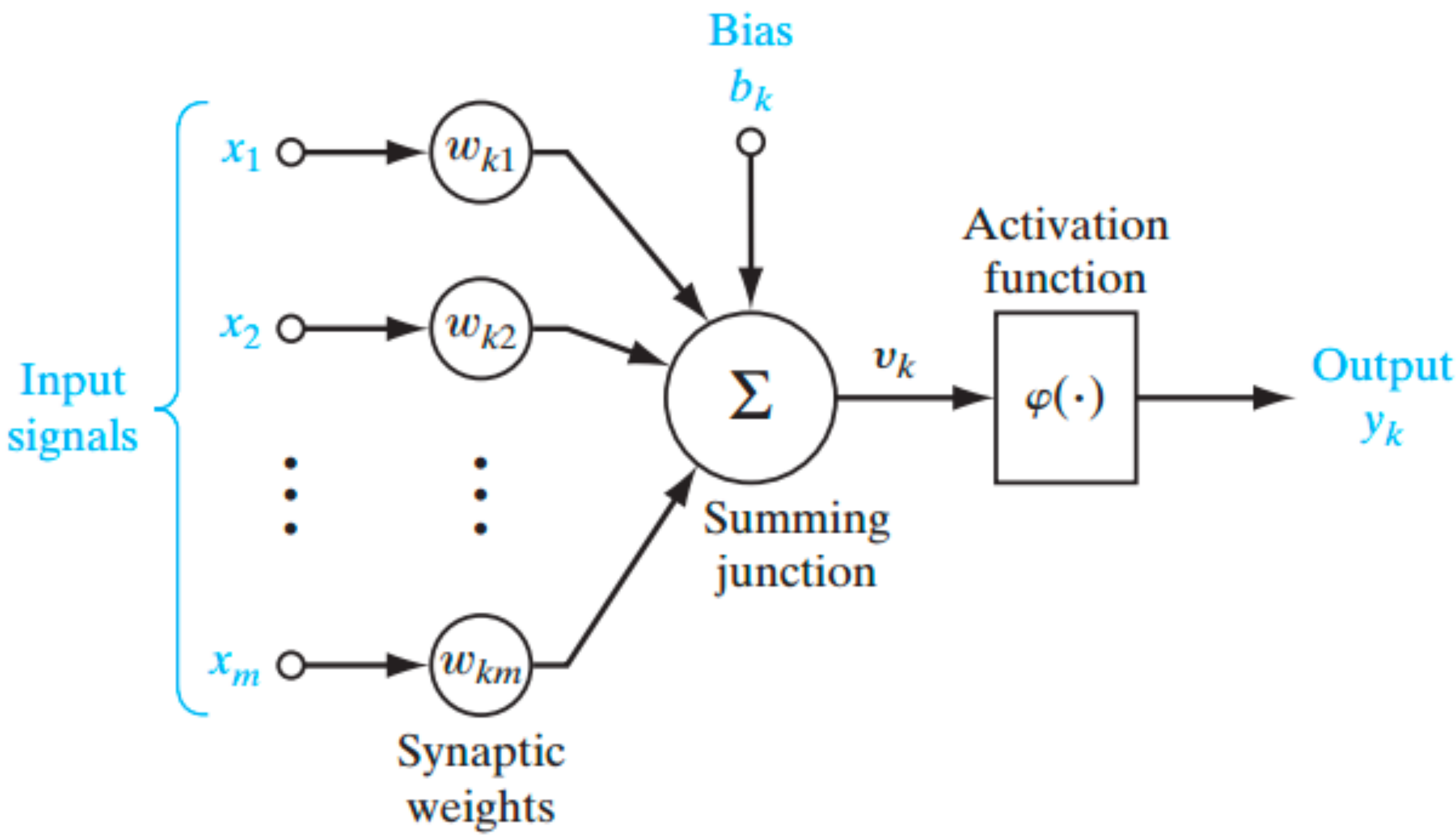
- Built many trees (hundreds, thousands), so that the weighted average over all trees is insensitive to fluctuations

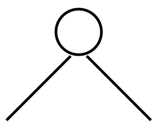
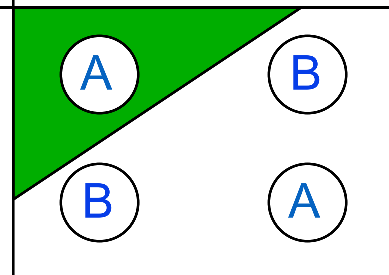
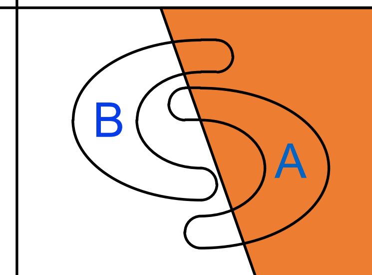
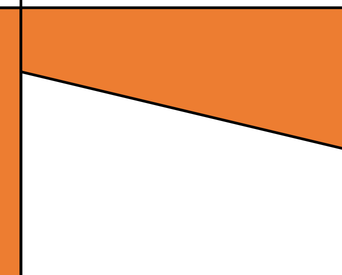
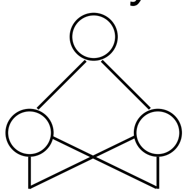
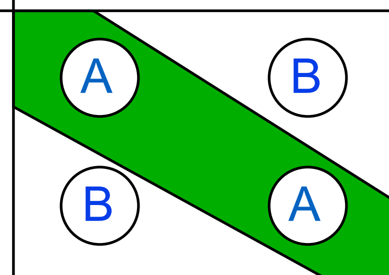
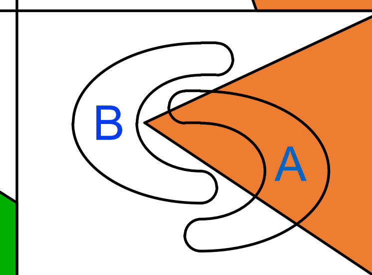
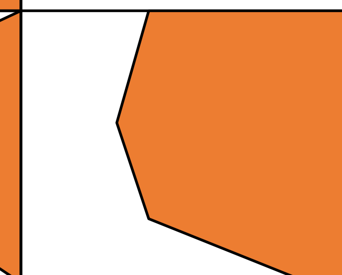
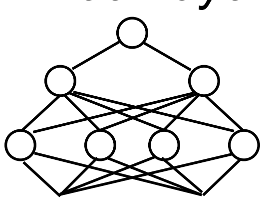
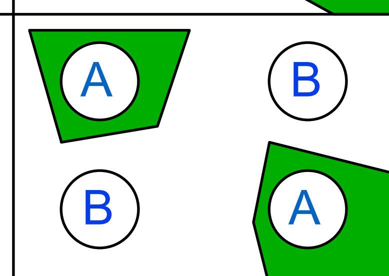
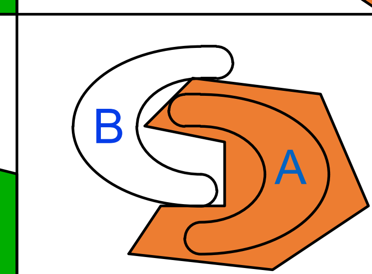
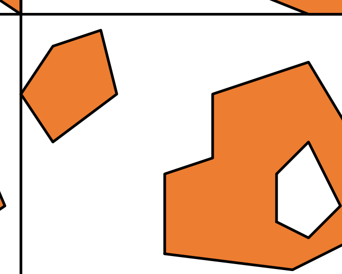
1. Create one tree
2. If there is misclassified data, create a second tree, giving more weight/importance to any misclassified examples
3. Score performance of each tree
4. Repeat to create more trees
5. Average over all trees, using the tree-scores as weights

Neural Networks



Single Layer Perceptron



Structure	*Types of Decision Regions*	*Exclusive-OR Problem*	*Classes with Meshed regions*	*Most General Region Shapes*
Single-Layer 	*Half Plane Bounded By Hyperplane*			
Two-Layer 	*Convex Open Or Closed Regions*			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

Example

- Handwriting character recognition
  - http://www.youtube.com/watch?v=ocB8uDYXtt0

Next time we will talk about how some of these techniques apply specifically in the context of (simple) object classification and OpenCV.