**Nearest Neighbour Classifier:** Supervised learning method that classify objects based on the euclidean distance to the closest training data point.

The euclidean distance between two points in n dimensions:

To classify it, then simply compute the euclidean distance to all points and choose the smallest.

KNearest, is simply the nearest neighbours but taking into account multiple K neighbours, take into account the classes of K nearest neighbours

Testing point x

For each training datapoint x

measure distance(x,x’ )

End

Sort distances

Select K nearest

Assign most common class

----

Perceptron:

classify(I) = 1 if threshold <= for the sum of all (inputs \* weights)

0 Otherwise

To improve the accuracy of the algorithm we can update the weights:  
weight = weight \* learning \_rate \* (actual-output) \* input

The weight is how important a certain feature is   
learning rate: is user defined parameter that affects how fast the output converge to a stable solution.  
TrueLabel: known output  
output: the output label from the perceptron   
input: feature of training data

we update this for i number of iteration, the ore iteration the slower the training algorithm. O(input \* iteration \* weight)

Classification is however relatively cheap with a O(N) run time

Perceptron Convergence Theorem: if the data is linearly separable, then application of the perceptron learning rule will find a decision boundary within a finite number of iterations.

Learning Algorithm for perceptron:

Initialise weights to random numbers in range -1 to +1

for n = 1 to Num\_iteration

for each training example (x,y)

calculate activiation

for each weight

update weight by learning rule

end

end

end

Sigmoid activation without threshold:

Entropy: is the amount of information contained in a variable. let entropy be H

H(x) =

The best feature to use in a decision tree is the one where H(T) - H(T\F) is largest:

H(T) = the entropy before the split

H(T|F1) = the entropy on data of first branch

H(T|Fn) = the entropy on data on the nth branch

H(T\F) the weighted average of the entropy on all branches

Cross validation: split training data into n chunks train on n-1 and test on the chunk we cut out.

False positive: falsely predicting an event

False negative: missing an incoming event

*ROC stands for Receiver Operator Characteristics: Used to assess problems with a significant class imbalance, since accuracy alone will not take the imbalance into account. The two main measures are sensitivity and specificity*

Sensitivity (Class of 1) = true positive / (true positive + false negative)

Specificity (Class of 0) = true negative / (true negative + false positive)

cost of classifier = (false negative \* cost(false negative))+ (false positive \* cost(false positive)), useful if one is more important than the other

**Ensemble System:** fits multiple models ( base learners) to the training data, when classifying. it is uses the models as a committee to vote on testing data.

**Bootstrap:** way of generating numtiple datasets from original dataset. selecting N training example from total N with replacement.

**Bagging (independent)**: is bootstrap aggregating, ensemble method that generates m boostraps and trains m models on each. To classify new data use a simple majority vote.

**Boosting (dependent):** 1- take bootstrap of the dataset. 2- train one model on the bootstrap. 3- take a look at which examples the model gets wrong. 4- upweight the hard example and downweight the easy ones. 5- go bak to step 1 with weighted bootstrap until you run out of models to train.

**Random Forests:** For all the trees, take a Bootstrap sample T’, building decision tree using T’, at every split point: choose random fraction K of remaining feature, pick best feature from the subset. add the tree to the set.

For a test point: take a response from each tree, and then take majority votes

Random forests = Kinect

Boost = face recognition in phone cameras

-----

Three ways to establish a classifier:  
Discriminative: Model a classification rule directly. E.G K-NN, Decision trees, perceptron, SVM. Or Model the probability of class memberships given input data, E.G Perceptron

Generative: Make a probabilistic model of data within each class, example Naive bayes, Model based classifiers.

Probabilistic: Use the probability of data. E.G perceptron, naive bayes, model based classifiers

Prior probability: P(X)

Conditional probability: P(X1|X2)

Joint probability: X = (X1, X2). P(x) = P(X1, X2)

Relationship: P(X1, X2) = P(X2|X1)P(X1) = P(X1|X2)P(X2)

Independance: P(X2|X1) = P(X2), P(X1|X2) = P(X1), P(X1, X2) = P(X1)P(X2)

Bayesian Rule

P(C|X) = P(X|C) \*P(C) / P(X)

-> Posterior = Likehood \* prior / evidence

Establishing a probabilistic model for classification:

* Discriminative model: P(C|X) C =c1,...,cL, X=(X1,...,Xn)  
  P(C1|X) P(c2|X) … P(CL|X)  
  x = (x1, x2, …, xn)
* Generative Model: P(X|C)C=c1,...,cL, X =(X1,...,Xn)  
  P(X|c1) P(X|c2) … P(X|cL)  
  X = (x1,x2,....,xn)

Maximum a posterior (MAP): Assign x to c\* if   
 P(C=c\* | X = x) > P(C = c | X = x) c != c\*, c=c1,...,cL  
basically assign it to the class with highest probability given a set of probabilities X and classes c   
  
Generative classification with MAP rule:  
- Apply bayesian rule to convert them into posterior probabilities:  
P(C = ci|X = x) = P(X = x | C = ci) P(C = ci) / p(X = x)   
 p(X=x|C=i) P(C= ci). for i = 1,2,...L  
then choose according to MAP

Naive Bayes:   
Bayes classification P(C|X) = P(X|C)P(C) = P(X1,...,Xn|C)P(C)  
  
  
Naive bayes classification: Called naive because it assume all inputs features are conditionally independent.   
P(X1, X2, …, Xn | C) = P(X1 | X2, …, Xn, C)P(X2, …, Xn | C)  
 = P(X1 | C)P(X2, …, Xn | C)  
 = P(X1 | C)P(X2 | C) … P(Xn | C)  
MAP classification rule: for x = (x1, x2, …, xn)   
[P(x1|C\*)…p(Xn|C\*)]P(c\*) > [P(X1|C) … P(Xn |c)]P(C), c != c\*, c= c1,...,cl

Naive bayes, algorithm: discrete valued features

* Learning phase: given a training set S of F features and L claases,  
  For each target value of Ci(Ci = c1, ... , cL) // for each class

P(C = ci) ← estimate p(C = ci) from examples from S  
 for every feature value xjk of each feature Xj(j= 1, … F; k = 1,...,Nj)  
 P(Xj = xjk | C = ci) ← estimate p(Xj = xjk | C = ci) with examples in S;

output F \* L conditional probabilistic (generative) models

* Test phase: Given an unknown instances X’ = (a1,..., an)   
  look up tables to assign the label c\* to X’ if   
  [p(a1|c\*)...p(an|c\*)]P(c\*) > [p(a1|c)...P(an|c)p(c)] p(c), c != c’, c= c1,...,cl

Algorithm: continuous valued features:

* numberless value taken by a continuous valued feature
* conditional probability often modeled with normal distribution   
  P(Xj | C = ci) = )  
  μ ji: the average of feature values Xj of example for which C = ci  
  σ ji: the standard deviation of feature values Xj of example C = ci
* Learning phase: for X = (X1, …, Xn), C = c1, …, CL  
  output: n \* L Normal distributions and P(C= ci) i = 1, … ,L
* Test phase: given an unknown instance X’ = (a1, … , an)  
  calculate conditional probabilities with all the normal distributions achieved in the learning phrase.  
  apply MAP rule to make a decision  
    
  The mean: μ = 1/N   
  The standard Deviation: σ =

-----

**Cluster analysis:** involves finding similarities between data and grouping similar data points into clusters. It is type of unsupervised learning and used both as a stand alone tool to gain insight into data but also as a pre-processing step for other algorithms.

**Distance matrix**: N data point with their distance only. triangular matrix,

3 ways to measure the distances:

1- Euclidean distance: (2 dimension)   
d(x, y) =

2- Manhattan distance (moving in straight line): (basically one dimension)

d(x, y) =

3- Minkowski Distance:

d(x, y) = , p > 0, p is the point   
Minkowski with p = 1 this is Manhattan, if p = 2 that is Euclidean.

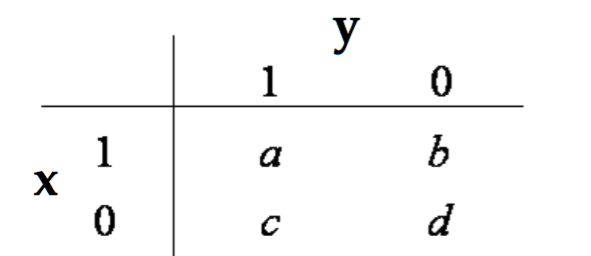
**Cosine Measure (Similarity vs distance):** the measure of similarity between two vectors in n dimensional space,

for x = x1x2...xn, and y = y1y2...yn

cos(x, y) =

distance between two vectors: d(x, y) = 1- cos(x, y)

**Distance for binary features**



a: number of features that equals to 1 for both x and y

b: number of features that equals to 1 for x but 0 to y

c: number of features that equal to 0 for x but 1 to y

d: number of features that equal to 0 for both x and y

* Symmetric binary features: If they are both of equal weight (e.g. gender):

d(x,y) = b + c / (a+b+c+d)

* Asymmetric binary features: both aren’t of equal weights(one is rare disease) - the rare one is set to 1, the other is 0

d(x, y) = b+c / (a+b+c)

**Distance for nominal features:**

two methods to handle variables of more than binary features

* simple mis-match:   
  d(x, y) = number of mismatch features between x and y/ total number of features
* convert it into binary variables: for example colours = {red, blue, orange} → {100, 010, 001}. then do a binary count on each BIT

**Major clustering approaches**

**Partitioning approach:**

* construct various partitions and then evaluate them by some criterion, e.g. minimising the sum of square distance cost.
* typical methods: k-means, k-medoids, CLARNS

**Hierarchical approach:**

* create a hierarchical decomposition of the set of data, using some criterion.
* typical methods: agglomerative, Diana, Agnes, Birch, Rock

**Density- based approach:**

* based on connectivity and density functions
* Typical methods: DBSCN, OPTICS, DenClue

**Model-based approach**

* Generative model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
* Typical methods: Gaussian Mixture Model, COBWEB

**Spectral clustering approach**

* convert data set into weighted graph, then cut graph into sub graphs corresponding to clusters.
* Typical methods: normalised-cuts

**Clustering ensemble approach**

* Combine multiple clustering results
* Typical methods: evidence-accumulation based, graph-based

----------

**K-mean clustering:**

Optimal partition achieved via minimising the sum of squared distance to its representative object in each cluster.

E =

k is the number of partitions to find, Sk is the kth partition of the data, and mk is the representative object (the mean of points) in the set Si

algorithm:

1- Pick k random seed points from the data.

2- assign each data object to the cluster with nearest seed point using a distance measure algorithm

3- compute the mean points of the clusters (centroids)

4- back to step 1, but the seed are the the means calculated in step 3. Do it until sets no longer change

runtime of O(tKn): where t is the iteration, K is number of clusters and n is number of objects.

problems:

* local optimum: sensitive to initial seed points, converge to local optimum.

Other problems:

* need to specify K the number of clusters in advance.
* Unable to handle noisy data and outliers (use K-medoids algo instead)
* not suitable for discovering clusters with non convex shapes
* applicable only when mean is defined. For categorical data use k-mode algo
* hard to evaluate the performance

K-medoids: resistance to noise and outliers

K-modes: extension to categorical data clustering analysis

CLARA: extension to deal with large data sets

Mixture models (EM algorithm): handling uncertainty of clusters

**Hierarchical clustering**

Partition data sequentially, constructing partitions layer by layer by grouping objects together into a tree. Using a distance matrix.

**Agglomerative vs Divisive**

* Two sequential clustering strategies for constructing tree of clusters.
* Agglomerative: bottom up strategy
  + initially each data object is in its own cluster
  + then merge these clusters into larger and larger cluster
* Divisive: a top down strategy
  + initiall y all objects are in one single cluster
  + then the cluster is subdivided into smaller and smaller clusters

**Cluster distance measures:**

* Single link: smallest distance between an element in one cluster and an element in the other, i.e d(Ci, Cj) = min{d(xip, xjq)}
* Complete link: largest distance between an element in one cluster and an element in the other, i.e d(Ci,Cj) = max{d(xip, xjq)}
* Average: avg distance between elements in one cluster and elements in the other, i.e d(Ci,Cj) = avg{d(xip, xjq)}

**Agglomerative algorithm**

1- Convert all object features into a distance matrix Set each object as a cluster (thus if we have N objects, we will have N clusters at the beginning)

2- merge the two nearest obejects

3- Repeat until number of cluster is one (or known # of clusters)

* merge two closest clusters
* update “distance matrix”

Problems:  
- less efficient than k-means with runtime of O(n^2 \* log n)  
- very sensitive to noise and outlying data points

Good:  
- doesn’t need to know K in advance, knowing K set the termination condition

Variants:

* Brich: scalable to a large data set
* ROCK: clustering categorical data
* CHAMELEON: hierarchical clustering using dynamic modelling

**Cluster Validation:** is the procedures that evaluate the results of clustering in a quantitative and objective fashion.

Internal Criteria:

* validate without external information
* with different number of clusters
* solve the number of clusters

External criteria:

* Validate against ground truth (using common sense or a priori knowledge)
* compare two partitions (how similar)

**Internal index:**   
Variances based methods:

* minimise within cluster variance (SSW)
* Maximise between cluster variance (SSB)
* Within cluster variance SSW

SSW(m) =

* Between cluster Variance SSB

SSB(m) =

where c is the mean (centroid) of whole data set, m is number of clusters, with d is the distance used in the algorithm

Variance based F-ratio index:

measures ratio of between-cluster against within-cluster variance

F-ratio index (W-B index) for a partition of m clusters:

F(m) =

where xij is the jth data point in the cluster ci

ni is the number of data points in cluster ci

• Application: finding the “proper” number of clusters, ...

**External index:**

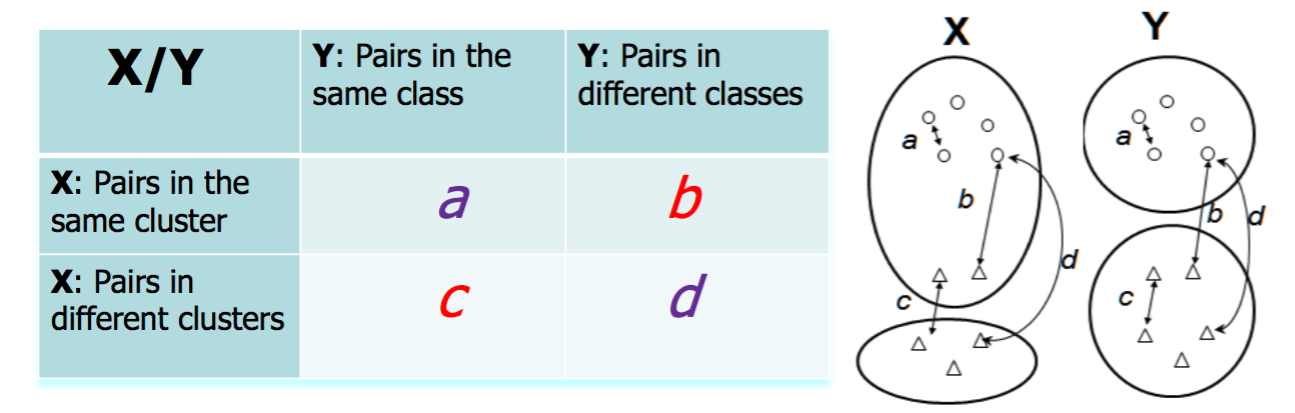
If ground truth is known, validity of clustering can be veerifide by comparing the class or clustering labels.

Problems:

* Permutation: Cluster IDs in a partition resulting from clustering have been assigned randomly due to unsupervised learning.
* Inconsistence: The number of clusters may be different from number of classes of the ground truth.
* Finding all possible correspondences between ground truth and a partition.

Rand Index:

To address the correspondence problem, consider all pairs in the data set by looking both into agreement and disagreement against the ground truth.



RI (X, Y) = (a + d) / (a + b + c +d)

Application: performance evaluation of clustering