Data Mining – aims to find patterns or a model embedded in a set of given training examples

Looking for a model M so the prob(M|D) is maximized where D is the given dataset

**Reasons for DM**

* Domain knowledge (hard to get)
* Processes are too complicated
* Data are easy to collect
* Data are not accurate (missing data, misclassified)

Building a model model (global) vs patterns (local)

**Components of a DM system**

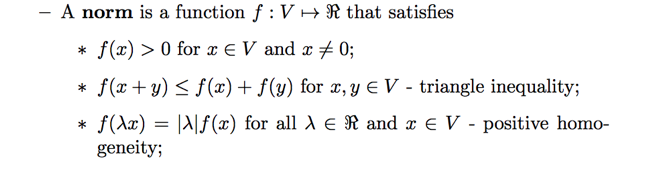
* Models (structures parameters, model fitting)
* Score function (similarity measure)
* Optimization methods
* Data management strategy

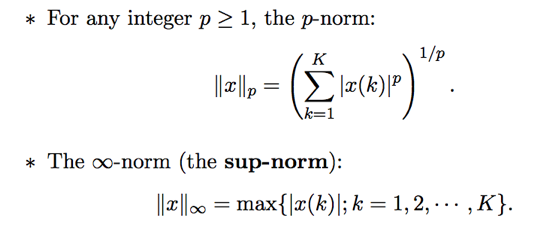
**Picture of Data Mining System**

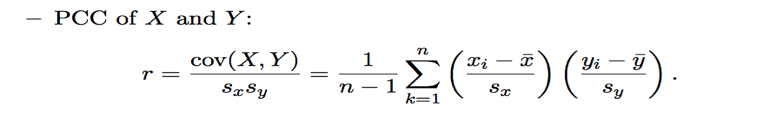
**Types of Data**

* Numerical (ordered and ranked)
* Categorical (not ordered or ranked)
* Hierarchical categorical (can be ranked but difficult to compute distance)
* Complex data (image/voice)

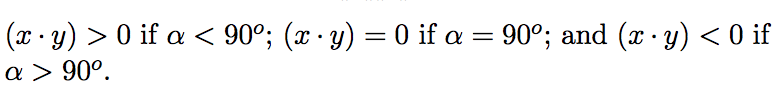
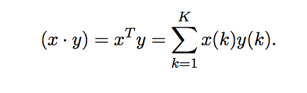
**Similarity Measure**





**PCC Coefficient** (measures how correlated two variables are – normalized value, +1 means positive linear correlation, 0 is no linear correlation, -1 is negative linear correlation)

**Dot Product** – returns a scalar value



**Entropy** – measures the uncertainty or information in a random variable 

More information gain is good

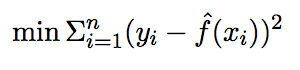
**Relative Entropy** – Information gain measure of information gain when one revises ones beliefs from a prior probability distribution Q to a posterior probability distribution P

**Regression –** learn an unknown function of y=f(x) given training data

Estimation of the model is the first step – estimation of the form of the function is hard often solved heuristically

Once the form of the model is determined then we do parameter estimation

Want to minimize E\_in which is error on in sample data



**Logistic Regression** – Predicts probabilities the function f() can be classified as a decision boundary f(.) is classified as + on one side and – on the next side

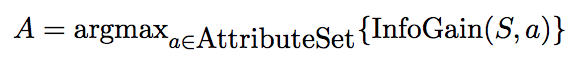
Use the nonlinear sigmod function 

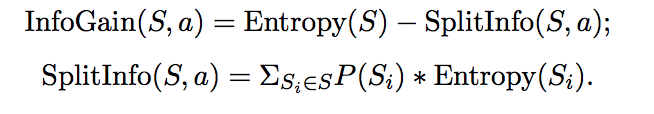
**k-NN** – classifies based on the labels of objects surrounding a data point (x, y) instead of using one NN use k (positive integer) NN.

If kNN have different labels use majority vote or random scheme

Two problems arise from k-NN (optimal parameter k) – no silver bullet here use heuristics and distance measure (use similarity measure)

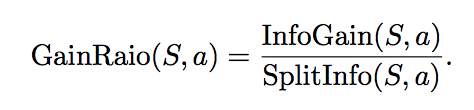
**Decision Trees –** iterative partitioning to build a decision tree start at the root with whole data set and chose feature or attribute a to partition the dataset S into subsets Si until all examples in leaf node are classified the same





Two main types of Decision Tree algorithms difference is what they split on. ID3 splits on InfoGain (favors attribute with large number of values)

C4.5 uses gain ratio to overcome this



Minimize overall entropy or maximize information gain gives us the smallest tree possible – maximum uncertainty = minimum information gain

One danger of this is overfitting (fitting the data perfectly to our test sample so we lose generalization.

Introduce pruning the tree (ie node splits stop when < than a certain number of node)

**Random Forest/Bagging** – Decision tree has **bias** towards certain features over others and can possibly **overfit** the model to the data.

Fix these problems by introducing randomness into the algorithm (both of the below cause information loss)

Can *not chose whole set of attributes*may help overcome attribute bias issues or we can *use whole set of examples* may avoid overfitting issues

Ensemble of models construct a random set of features/examples (from train data) build classifier for this data sample, produce final outcome based on the average or majority vote of the individual models

This is called bootstrap aggregating or bagging

RF is one such method

**Support Vector Machines and Kernel Method**

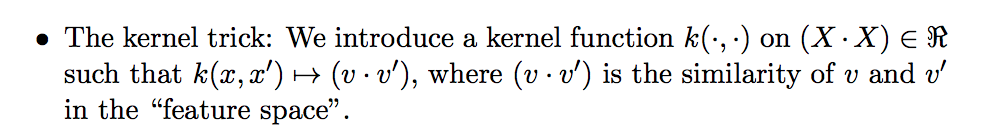
Consider a binary classification prob (multi class can be subdivided into multiple binary class probs)

Traditional classification problems face two main issues

* The curse of dimensionality
* Attempt to learn complex model/decision boundary

SVM involves three steps

1. Map the given data points X in the input space to the points V in a feature space that has higher dimension
2. Find a linear decision boundary in the higher dimensional feature space that separates the given data
   1. The features space uses one similarity measure regardless of input space
3. Transform the data back to the input space

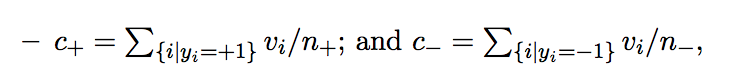


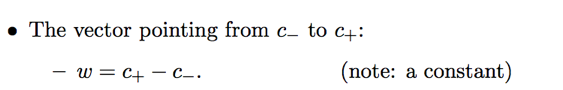
Benefits of using the features pace

* Dot product is a unified similarity measure regardless of input space (one size classifier fits all)
* Kernel trck applies k(x, x’) = (v\*v’) = sig(x)\*sig(x’)

To create a classifier we want to separate positive and negative values

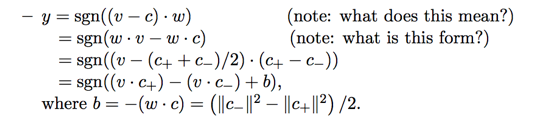
Use the centers of mass of the two classes as representative consider a vector pointing from one center to the other, the decision boundary goes through the middle point of this vector and is orthogonal



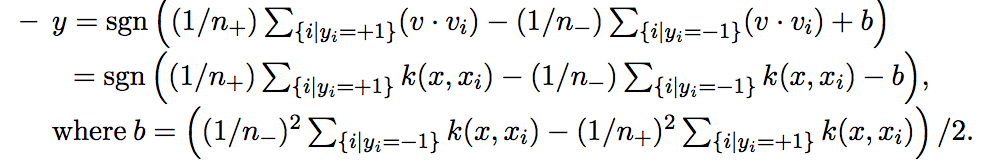


Using all of this we can determine how to classify new points and find a decision boundary

Given a new label v we label it as



This can be further reduced to

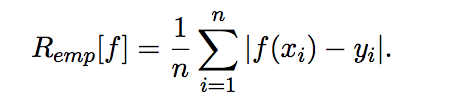


The above formula for y is the classifier

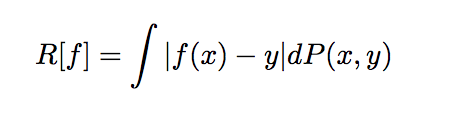
In feature space it is the linear classifier, in the input space it represents the kernel expansion involved through the mapping to feature space

We don’t necessarily know mapping but we use kernel trick

Want to learn predictive function f to minimize in sample error (empirical error)



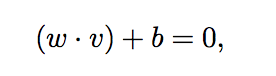
Out of sample error is described by the following but we don’t know P(x,y) so we cannot characterize



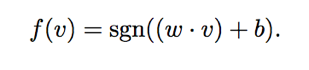
Minimizing Ein does not guarantee that Eout will be small

VC dimension is highest number of points for which a hypothesis set can generate the max number of dichotomies on (all classifications)

This bounds growth of infinite hypotheses sets

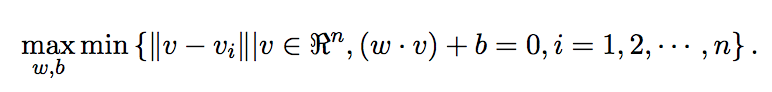
Focus on the class of the hyperplane  where w is our weight vector v is the input vector and b is the bias/threshold that we are classifying on

Corresponding decision function



Among all hyperplanes we want the one that yields the maximal margin in separating two classes

Smallest distance from any data point to classifier (max margin)



v-vi is the distance from vi to the classifier or decision boundary

Capacity increases with increasing the maximal margin

Margin di of each data point vi to f(v)

The length of vector vi-v which is perpendicular to f(v) is the margin di=||vi-v||

vi-v is in the same direction as w if it is in the + class

vi-v = diw/||w|| where w/||w|| is a vector of a unit length pointing in the direction of w (if vi is a positive example)

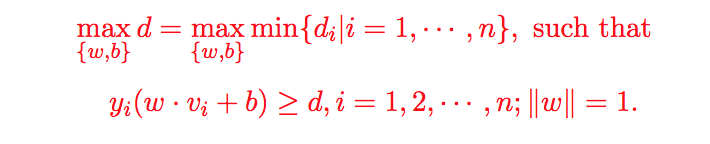
As v is on f(v) we have (w\*v )+ b = 0 which means w\*(vi-diw/||w||) + b = 0





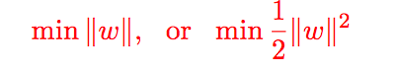
Where di is the margin from point v

We want to find a classifer that makes no mistakes on the training data



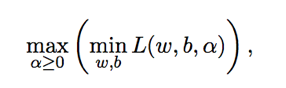
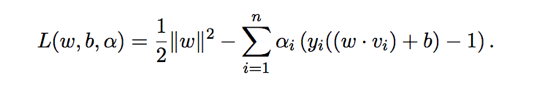
We introduce the ||w|| =1 to deal with issue that the same data may be represented by multiple scales // also makes this a nonconvex problem because of ||w||=1

Finding maximal margin is equivalent to

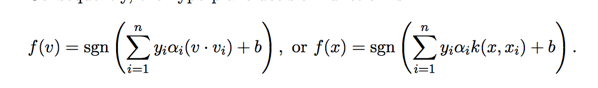


subject to the yi constraints above d> 1

One way to solve this is to use the Lagrangian dual

subject to 

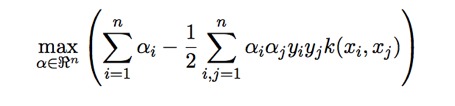
The hyperplane decision function is then



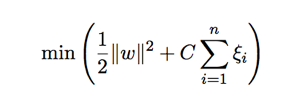
The solution vector w is an expansion of a subset of training examples that lie on the margin or xi whose ai are non zero

We look for the optimar margin support vector vector classifier by using the kernel

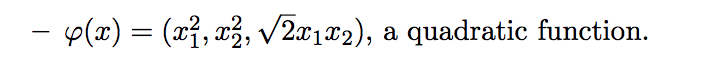
Solve the following optimization problem for a

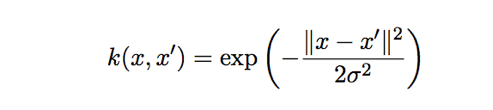


Can use a soft margin classifier if there is noise in the data

 squiggly > 0 C is a penalty for panelty

Kernels are nothing but a similarity measure in input space introduce kernel to avoid mapping x to v





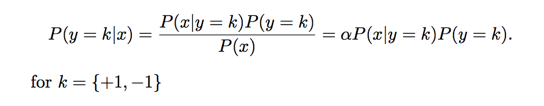
**Naïve Bayesian**

Discriminative analysis – given training examples of two classes drawn from probs P(x|y=+1)

Want to predict the label y of a new data point means we need to calculate P(y = k|x) where k = -1,+1

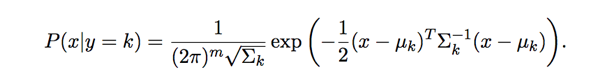
There are posterior probabilities of class label given new observation x

Pick a class k that has highest posterior probability then using the Bayesian formula we have



The above formula is equivalent to saying poster = prior\*likelihood/evidence

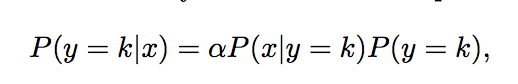
For linear and quadratic this is modelled as a multivariate Gaussian distribution



m is the number of features of x uk is the mean and Ek is the feature covariance matrix of x in class k

For linear discriminant assume each class as same covariance matrix (decision boundary is also linear)

Need to estimate parameters (uk Ek)from training set



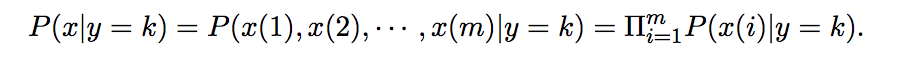
This is what we compute in discriminative analysis

P(y=k) is prior class distribution P(x|y = k) is conditional prior joint distribution of all the m features (like example with cloudy rainy P(R|y = 1)

Two issues

* Sampling complexity: When m is large we need a large number of samples in order to have sufficient num samples for each possible combination of the m features
* The complexity of computing covariant matrix increases with m

In naïve Bayesian assume that all features are conditionally independent of each other

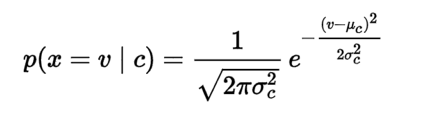


To assign a new label to x we do the following calculation



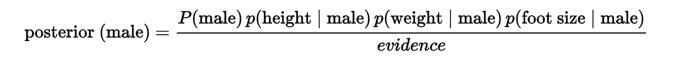
The independence argument is seldom satisfied in practice but this runs fast and gives us a good result

Example Naïve Bayes





P(female)… etc will be 0 for male probabilities



**Evaluation** – want to know which methods work better including strengths and weaknesses, understand the conditions under which the method works,

TruePositive = TP/P, FalsePositie = FP/N, sensitivity = 1-FP