



# CS 6474 Social Computing: Quantitative Methods Review II

*Munmun De Choudhury*

[munmund@gatech.edu](mailto:munmund@gatech.edu)

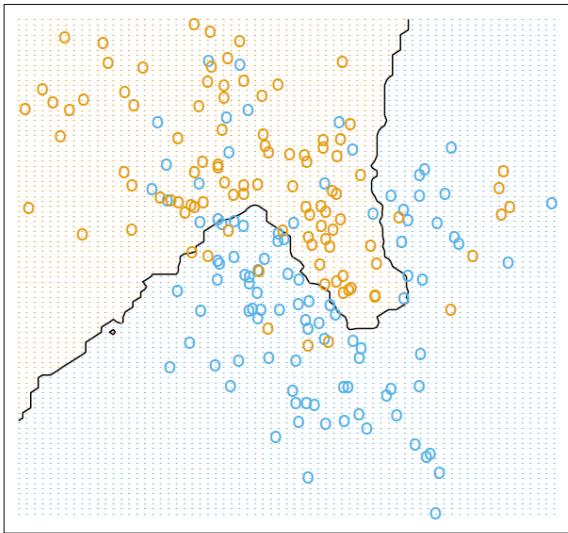
Week 7 | October 4, 2017

# Different tasks

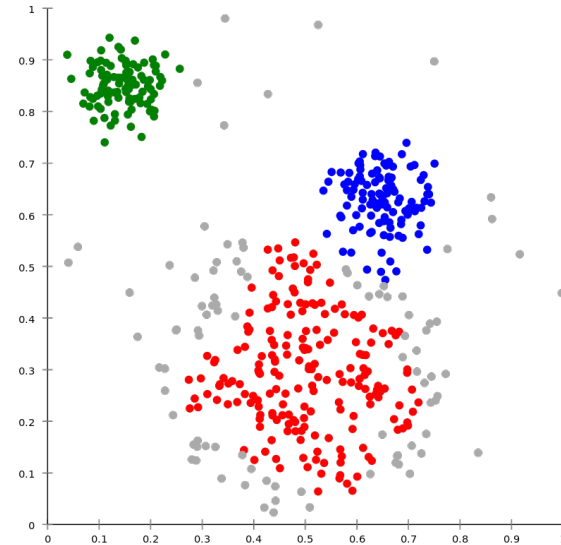
- *Supervised learning:* We're predicting a target variable for which we get to see examples. (regression, classification)
- *Unsupervised learning:* We're predicting a target variable for which we never get to see examples. (clustering)

# Class Exercise I

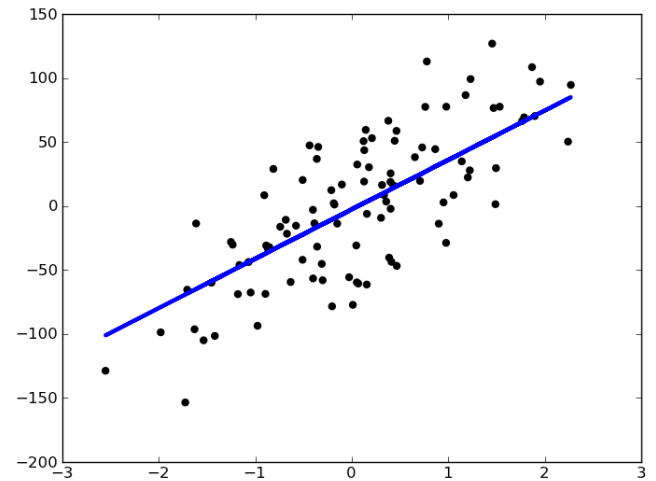
# What we will cover...



Classification



Clustering



Regression



# Representing data

- *Data/points/instances/examples/samples/records*: rows
- *Features/attributes/dimensions/independent variables/covariates/predictors/regressors*: columns
- *Target/outcome/response/label/dependent variable*: special column to be predicted

	$X_1$	$X_2$	$X_3$	$Y$
$x_1$				
$x_2$				
$x_3$				
$x_4$				
$x_5$				
$x_6$				

# Representing data

*Continuous*: a number, like #followers, #tweets

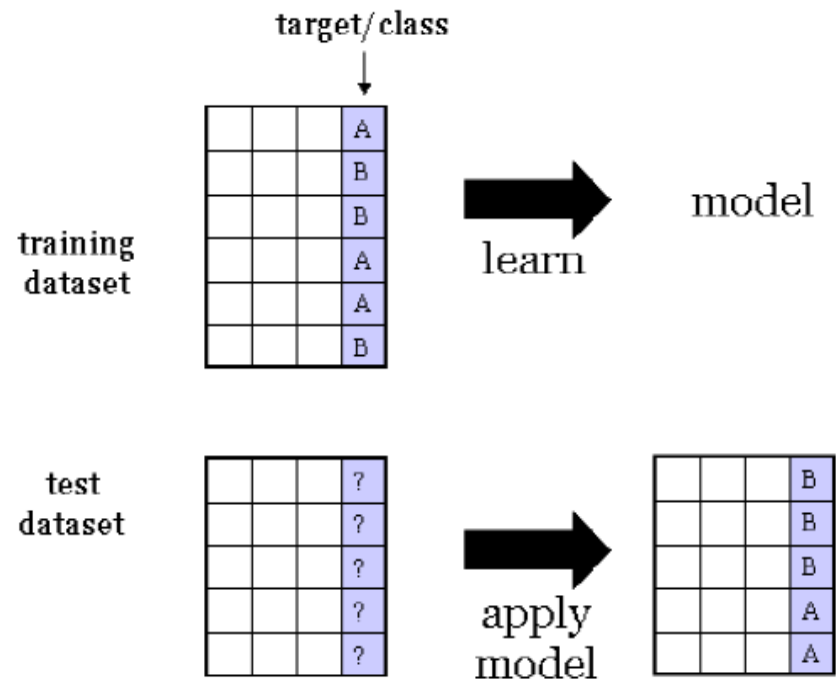
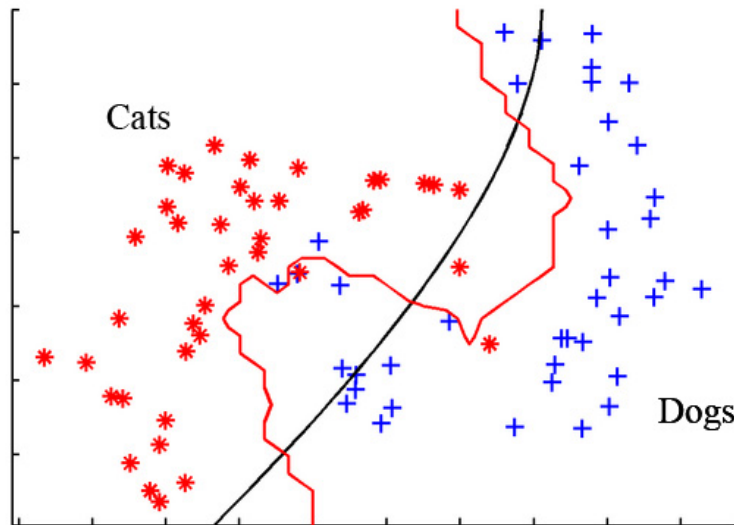
*Discrete*: a symbol, like gender, ethnicity

	$X_1$	$X_2$	$X_3$	$Y$
$x_1$				
$x_2$				
$x_3$				
$x_4$				
$x_5$				
$x_6$				

Classification (supervised  
learning)

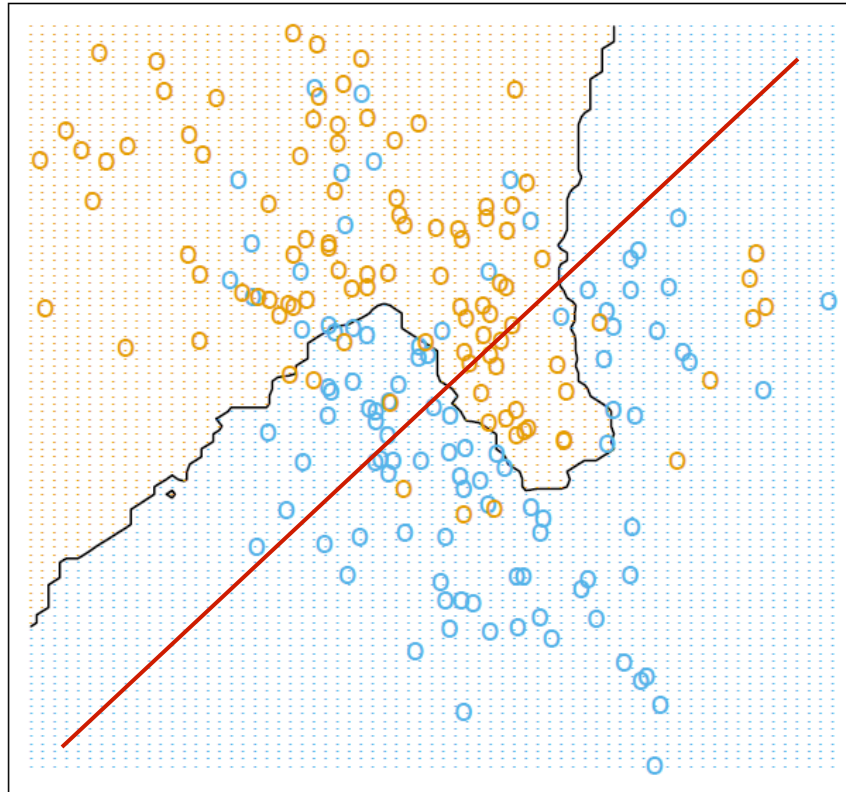
# Classification

- The setup:
  - You obtain some kind of model based on some examples, or *training data*, through a process called *learning* (also *estimation*).
  - Then you use that model to *predict* something about data you haven't seen before, but that comes from the same distribution as the training data, called *test data*.



# Classification

- However your prediction of classification may not always be correct—each data point is different!
- Typical error measurement—accuracy



# Classification Accuracy: Estimating Error Rates

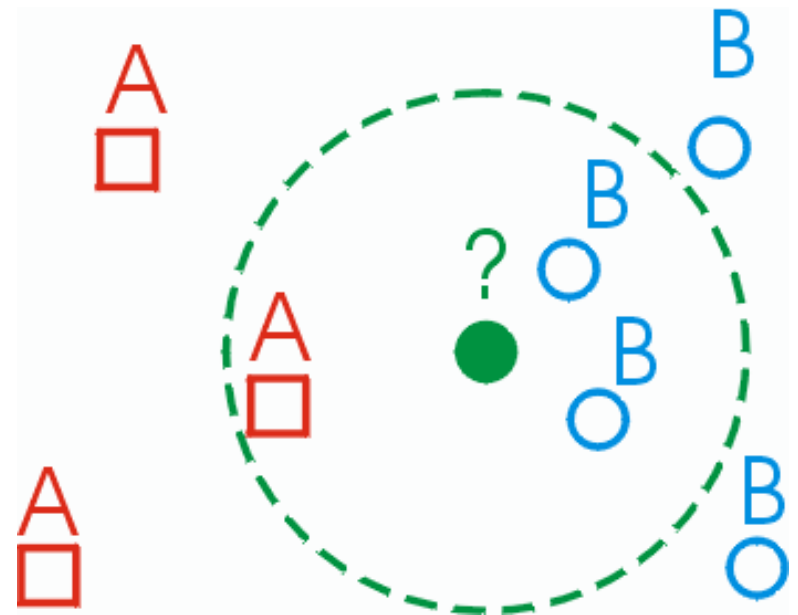
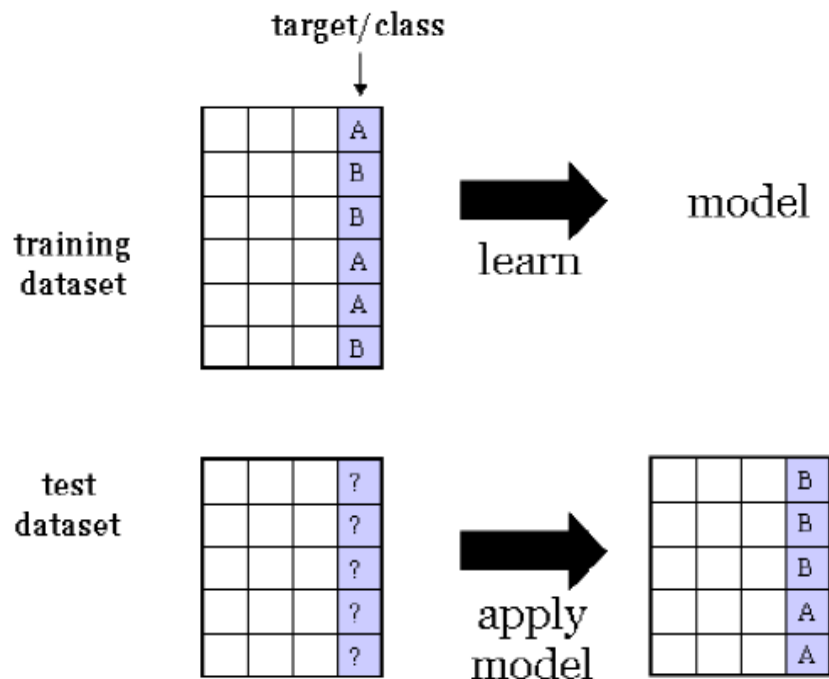
- Partition: Training-and-testing
  - use two independent data sets, e.g., training set ( $2/3$ ), test set ( $1/3$ )
  - used for data set with large number of samples
- Cross-validation
  - divide the data set into  $k$  subsamples
  - use  $k-1$  subsamples as training data and one sub-sample as test data— $k$ -fold cross-validation
  - for data set with moderate size
- Bootstrapping (leave-one-out)
  - for small size data



# Classification

Simple example of a classification model (“classifier”):

- Use the label of the past training point which is most similar to the new test point, and return that as the prediction (“ $k$  nearest-neighbor”).



# Classification: $k$ -NN Algorithm

- The  $k$ -NN algorithm for *continuous-valued target functions*
- Calculate the mean values of the  $k$  nearest neighbors
- Distance-weighted nearest neighbor algorithm
- Weight the contribution of each of the  $k$  neighbors according to their distance to the query point  $x_q$ 
  - giving greater weight to closer neighbors
- Similarly, for real-valued target functions
- Robust to noisy data by averaging  $k$ -nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes.
- To overcome it, axes stretch or elimination of the least relevant attributes.

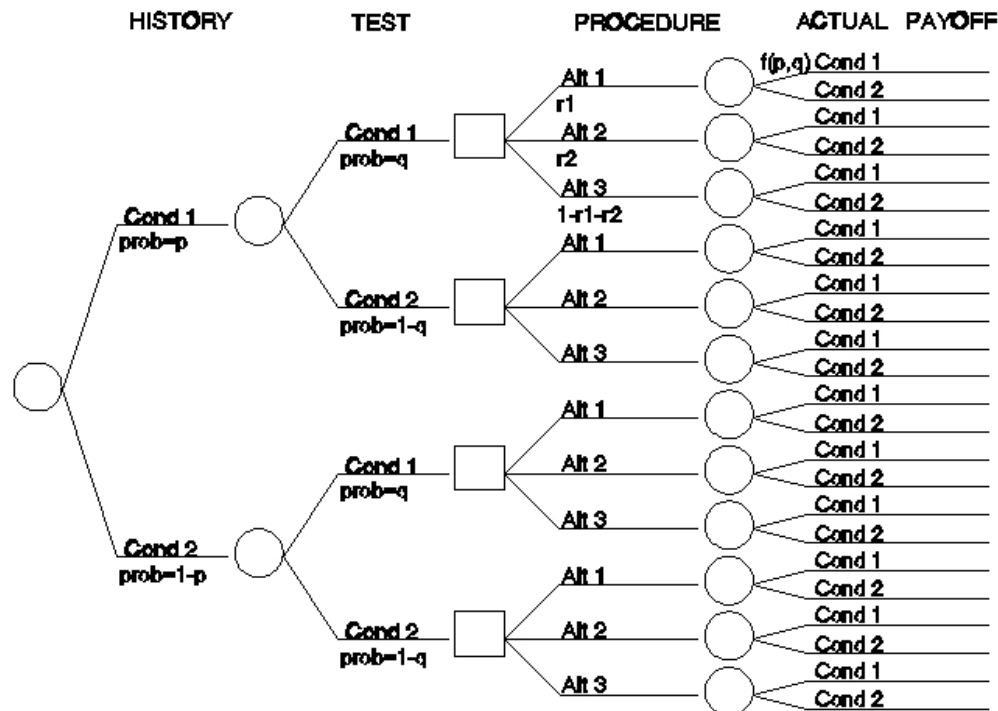
$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

# Classification: Decision trees

- A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.
- A decision tree is a flowchart-like structure in which internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test and each leaf node represents a class label (decision taken after computing all attributes).
- The paths from root to leaf represents classification rules.

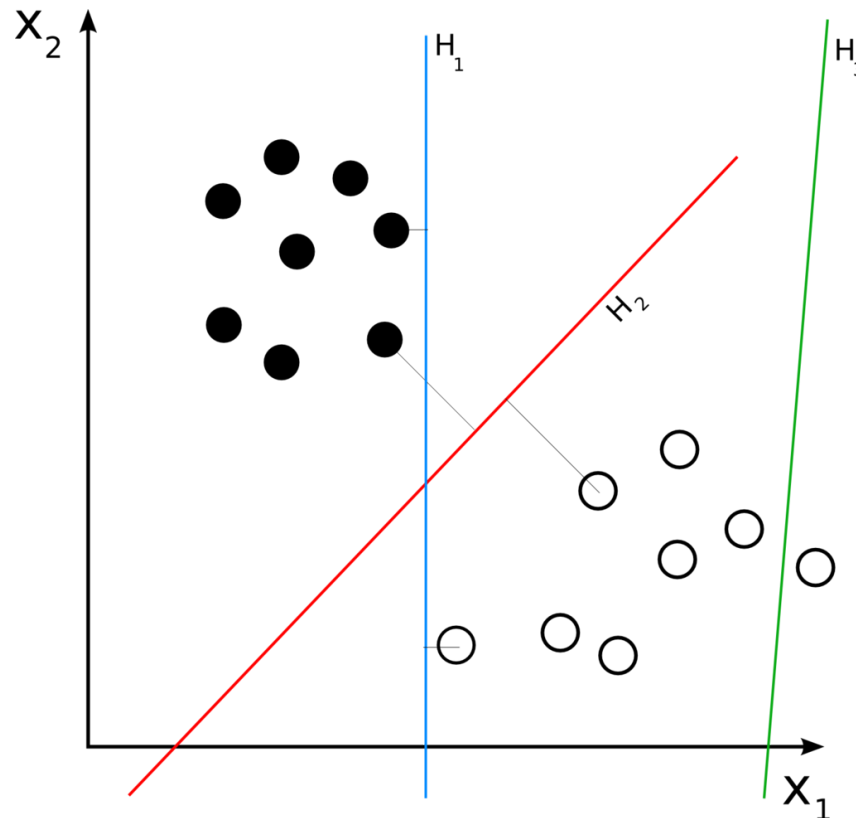
# Classification: Decision trees

- A decision tree consists of 3 types of nodes:
  - Decision nodes - commonly represented by squares
  - Chance nodes - represented by circles
  - End nodes - represented by triangles
- Example: Should you recommend smoking cessation content to B?



# Classification: linear classifier

- A linear classifier achieves this by making a classification decision based on the value of a linear combination of the characteristics.



# Classification: linear classifier

- **Naive Bayes classifiers** are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.
- A naive Bayes classifier assumes that the value of a particular feature is unrelated to the presence or absence of any other feature, given the class variable.
- Example: a fruit may be considered to be an apple if it is red, round, and about 3" in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of the presence or absence of the other features.



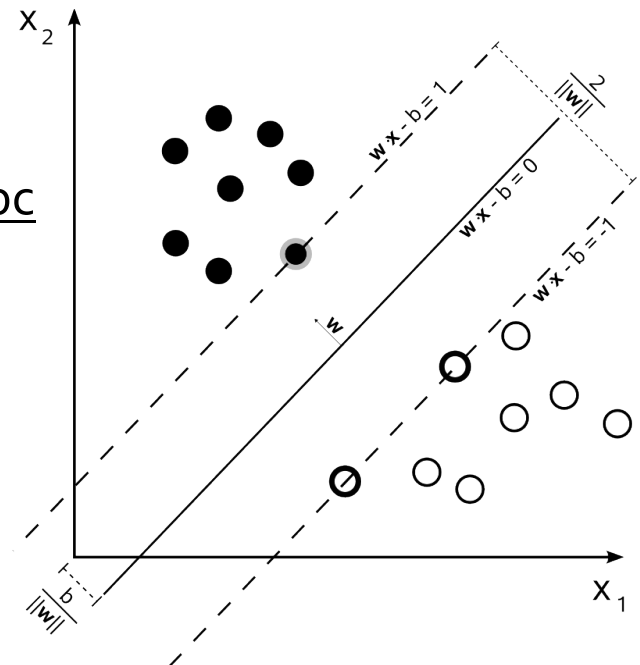
# Classification: linear classifier

- A **logistic regression** model predicts a binary response from a binary predictor
- It is used for predicting the outcome of a categorical dependent variable (i.e., a class label) based on one or more predictor variables (features). That is, **it is used in estimating the parameters of a qualitative response model.**
- The probabilities describing the possible outcomes of a single trial are modeled, as a function of the explanatory (predictor) variables, using a logistic function.

$$F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}.$$

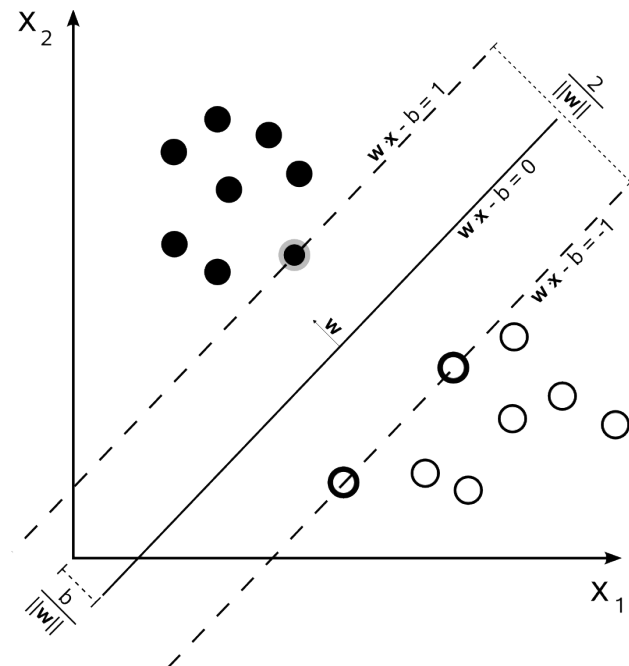
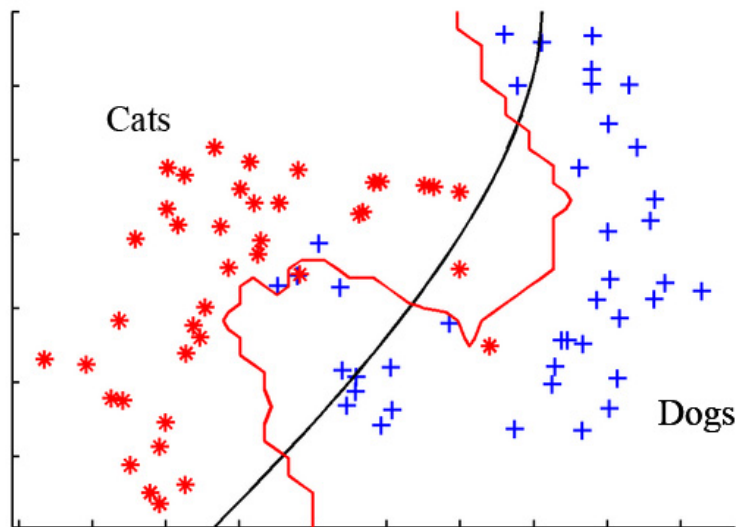
# Classification: Support Vector Machine

- A support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks.
- Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.
- Video explaining how SVM works (in detail):  
<https://www.youtube.com/watch?v=1NxnPkZM9bc>



# Classification: Support Vector Machine

- But data is not always linearly separable!
- SVMs can efficiently perform a non-linear classification using what is called the *kernel trick*, implicitly mapping their inputs into high-dimensional feature spaces.
- How it works: <https://www.youtube.com/watch?v=3liCbRZPrZA>



# Performance Evaluation

- $precision = \frac{tp}{tp+fp}$

- $recall = \frac{tp}{tp+fn}$

- $f1score = 2 \frac{precision \times recall}{precision + recall}$

- sklearn.metrics
  - precision\_score
  - recall\_score
  - f1\_score

	actual class (observation)	
	tp (true positive) Correct result	fp (false positive) Unexpected result
predicted class (expectation)	fn (false negative) Missing result	tn (true negative) Correct absence of result

# Cross Validation

- When tuning the parameters of model, let each article as training and testing data alternately to ensure the parameters are not dedicated to some specific articles.
  - from `sklearn.cross_validation` import `KFold`
  - for `train_index`, `test_index` in `KFold(10, 2)`:
    - `train_index = [5 6 7 8 9]`
    - `test_index = [0 1 2 3 4]`

# How to choose the right classifier?

- Predictive accuracy
- Speed and scalability
  - time to construct the model
  - time to use the model
- Robustness
  - handling noise and missing values
- Scalability
  - efficiency for high dimension, large-scale data
- Interpretability
  - understanding and insight provided by the model
- Goodness of rules
  - decision tree size
  - compactness of classification rules



# Class Exercise II and III

# Clustering (unsupervised learning)

# Clustering

- “Show me the sub-groups in the data.”
- Why show sub-groups in the data? Sometimes:
  - Computational reasons (e.g. use cluster centers instead of the dataset)
  - Statistical reasons (e.g. identify/remove outliers)
  - Mainly: Visualization/understanding reasons
- Cite examples where you’ll apply clustering to study a social computing problem?

# Clustering: $K$ means

- The  $K$ -means method is as follows:
  - First initialize the means  $\mu_k$  somehow, for example by choosing  $K$  different points randomly. Then:
  - Assign each point according to
$$C(i) = \arg \min_k \|x_i - \mu_k\|.$$
  - Recompute each  $\mu_k$  according to the new assignments.
  - Stop when no assignments change.
  - However, it does not necessarily obtain the global optimum. In practice, this is done, say, 10 times and the result with the lowest sum-of-squares is used.

# Clustering: how to choose $K$ ?

- A (heuristic) approach (out of many that have been proposed) uses the *gap statistic* – it chooses the  $K$  where the data look most clustered when compared to uniformly-distributed data.
  - For each value of  $K$ , compute the log of within-cluster scatter,  $\log W_K$  for the best clustering using that  $K$ .
  - For each value of  $K$ , also compute this quantity for  $m$  clusterings using uniformly-distributed data – call this  $\log W'_K$  and its standard deviation  $s_K$ . it the next cluster center.
  - Compute  $G(K) = |\log W_K - \log W'_K|$ .
  - Choose the  $K$  such that  $G(K) \geq G(K+1) - s_{K+1} \sqrt{1 + 1/m}$ , i.e. the smallest  $K$  producing a gap within one standard deviation of the gap at  $K+1$ .

# Clustering: hierarchical clustering

- Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:
  - Agglomerative: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
  - Divisive: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

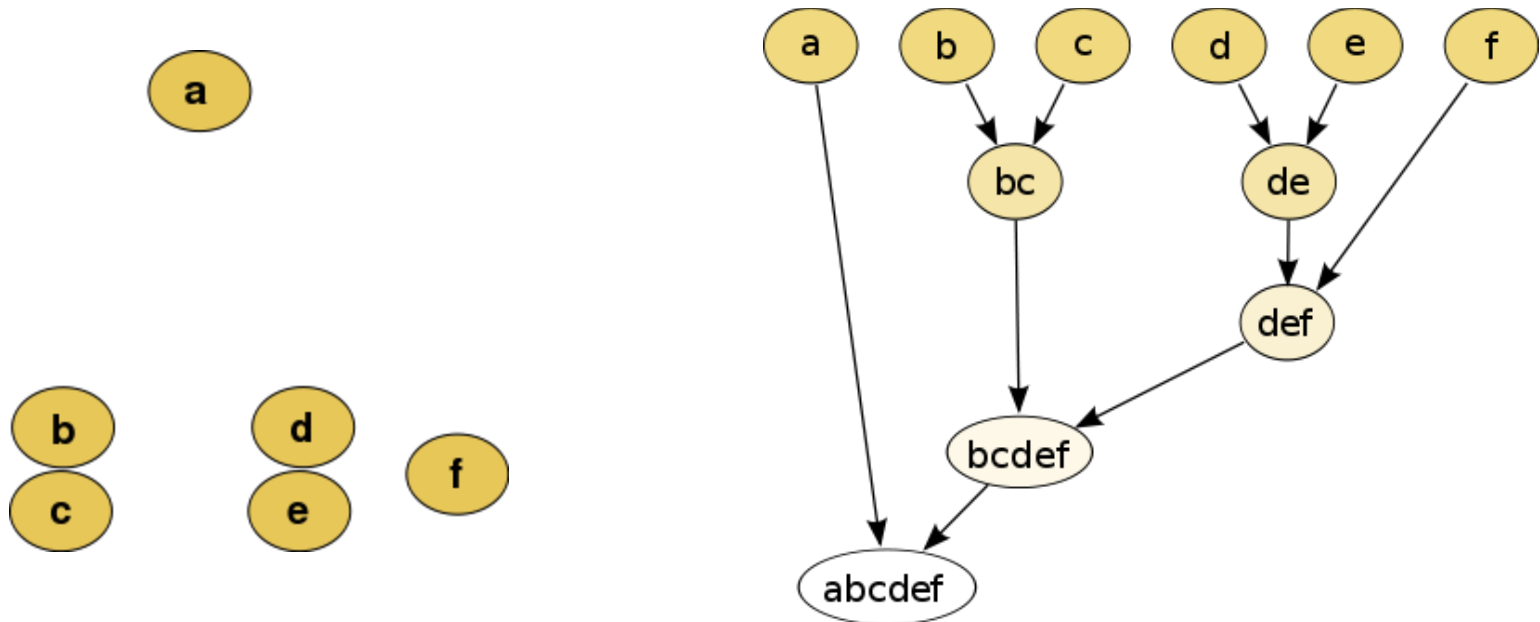


# Clustering: hierarchical clustering

In order to decide which clusters should be combined (for agglomerative), or where a cluster should be split (for divisive), a measure of dissimilarity between sets of observations is required.

Euclidean distance  $\|a - b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$

Cosine similarity  $\frac{a \cdot b}{\|a\| \|b\|}$



Feature extraction

# Practical Issues

- Tokenization
  - Convert document to word counts = “bag of words”
  - word token = “any nonempty sequence of characters”
  - for HTML (etc) need to remove formatting
- Canonical forms, Stopwords, Stemming
  - Remove capitalization
  - Stopwords
    - » remove very frequent words (a, the, and...) – can use standard list
    - » Can also remove very rare words, e.g., words that only occur in  $k$  or fewer documents, e.g.,  $k = 5$
- Stemming
- Data representation
  - e.g., sparse 3 column for bag of words: <docid termid count>
  - can use inverted indices, etc.

- Text Feature extraction in sklearn
  - `sklearn.feature_extraction.text`
  - `CountVectorizer`
    - Transform articles into token-count matrix
  - `TfidfVectorizer`
    - Transform articles into token-TFIDF matrix
  - Usage:
    - `fit()`: construct token dictionary given dataset
    - `transform()`: generate numerical matrix

# Feature Selection

- Decrease the number of features:
  - Reduce the resource usage for faster learning
  - Remove the most common tokens and the most rare tokens (words with less information):
    - Parameter for Vectorizer:
      - max\_df
      - min\_df
      - max\_features

