Lecture 14: Classification

COMP90049
Knowledge Technologies

Sarah Erfani and Karin Verspoor, CIS

Semester 2, 2017

THE UNIVERSITY OF MELBOURNE

What is Classification?

Classification involves predicting a discrete class or classes.
Those classes are defined in advance.

Binary (yes/no)

- Deciding whether a lone application is risky or not
- Predict whether a dwelling is an apartment or house based on its characteristics
- Predict whether a child will play or not, given weather.
- Will a student skip class on Friday?

Multi-class

- Categorise a document into newspaper sections (news, sports, entertainment, health)
- Recognise images of digits (0-9)
- Discriminating between different species of e.g. a kind of plant or an insect.
- Predicting type of cancer from gene expression data.

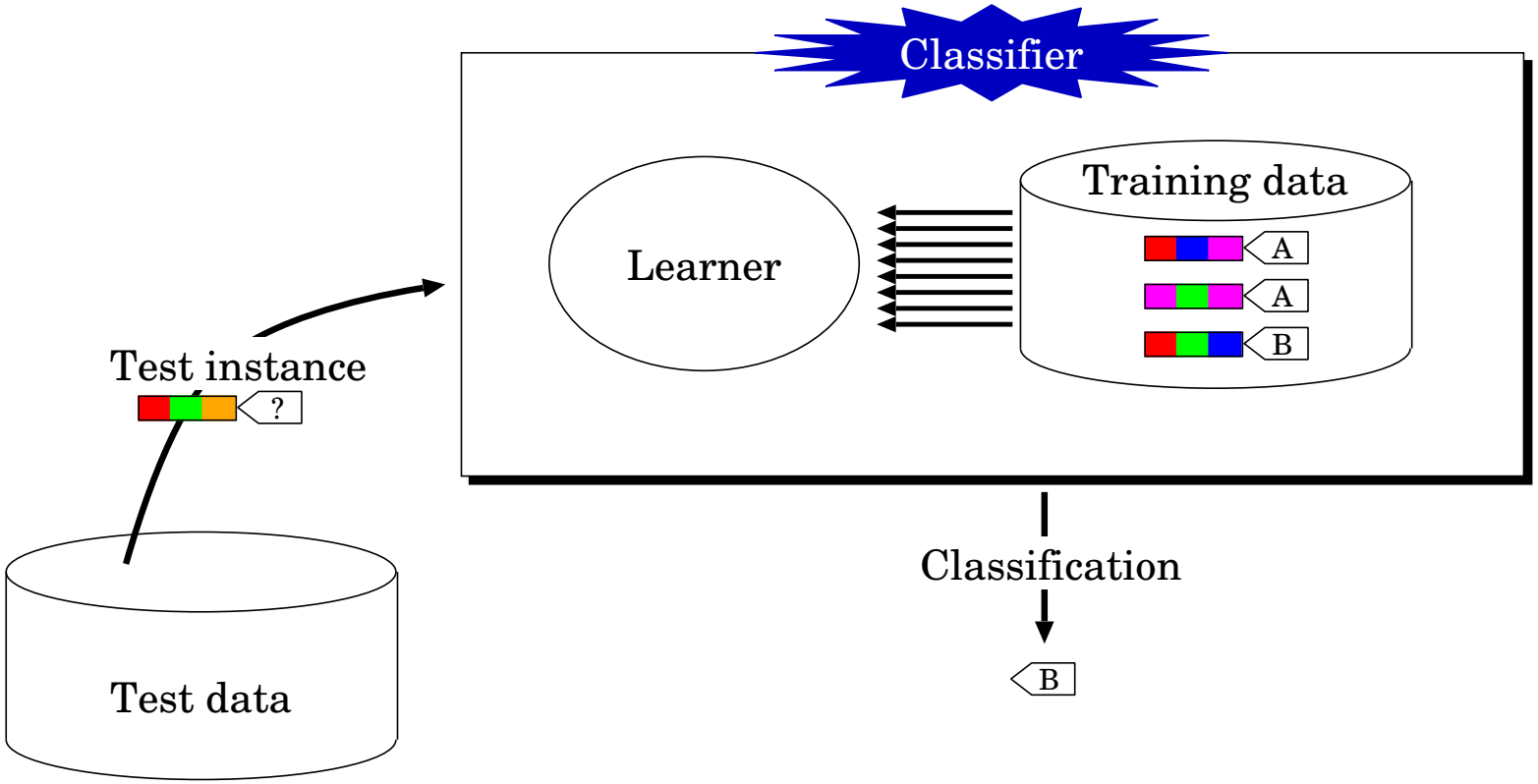
What are (Supervised) Classifiers?

- Given:
  1. a fixed representation language of *attributes*
  2. a fixed set of pre-classified *training instances*
  3. a fixed set of classes C
  4. a "learner" algorithm which can identify patterns in the training instances
- Estimate:

  the category of a novel input x : $c(x) \in C$
- Model:

  discover the function that predicts the label $c(x)$ *given a previously unseen* x

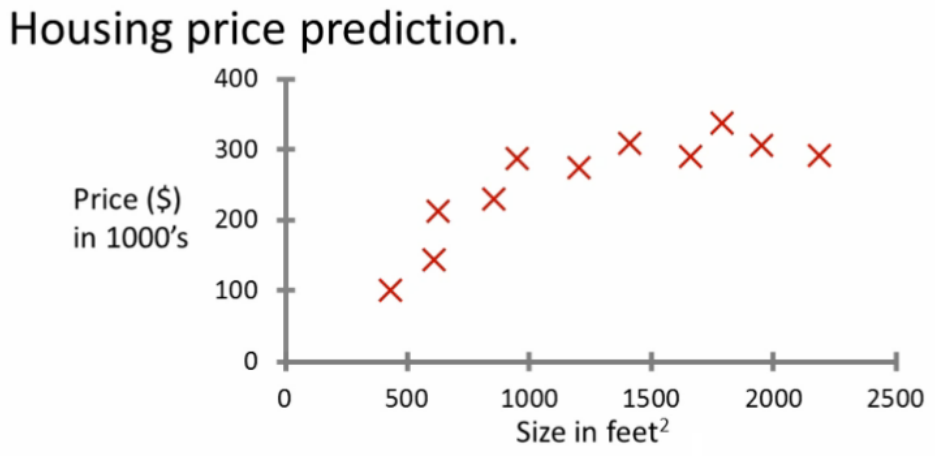
Supervised classification paradigm



The goal of learning from examples is not to **memorise** but rather to **generalise**, e.g., predict.

Example: Supervised Learning (Regression)

Can we predict housing prices?



A friend has a house which is 750 square feet – how much can he expect to get?

(draw a straight line vs. fit a curve)

Linear regression, mathematically

Linear regression captures a relationship between two variables or attributes.

It makes the assumption that there is a *linear* relationship between the two variables.

1. An outcome variable (aka response variable, dependent variable, or label)
2. A predictor (aka independent variable, explanatory variable, or feature)

At its most basic, the relationship can be expressed as a *line* (a deterministic function).

$$y = f(x)$$
$$y = \beta_0 + \beta_1 * x$$
$$y = \beta \cdot x \text{ (given } x_0 = 1)$$

A simple assumption!

Linear functions are more basic than non-linear functions (mathematically).

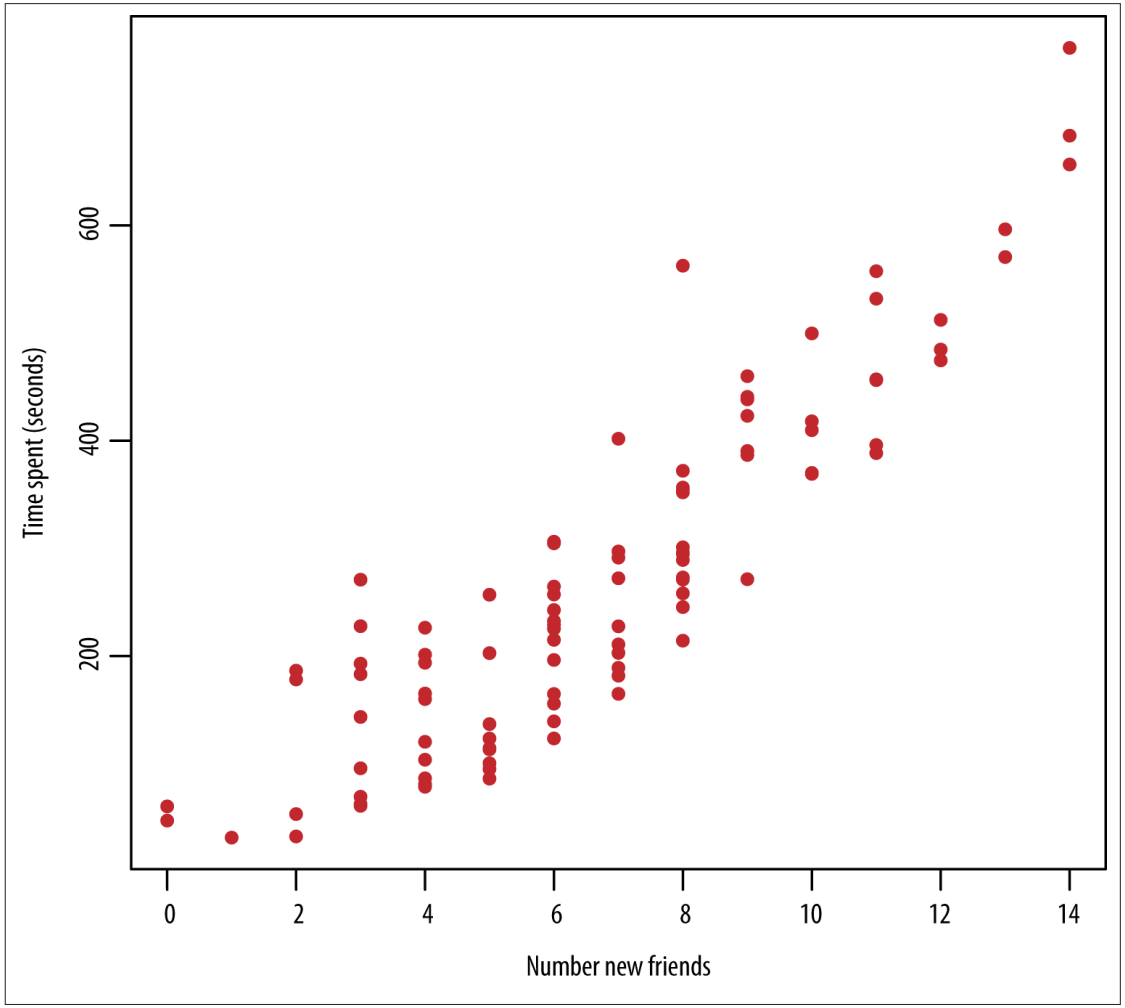
They capture that changes in one variable correlate linearly with changes in another variable.

For some variables, this makes sense.

[The more umbrellas you sell, the more money you make. How much money you make is directly proportional to how many umbrellas you sell.]

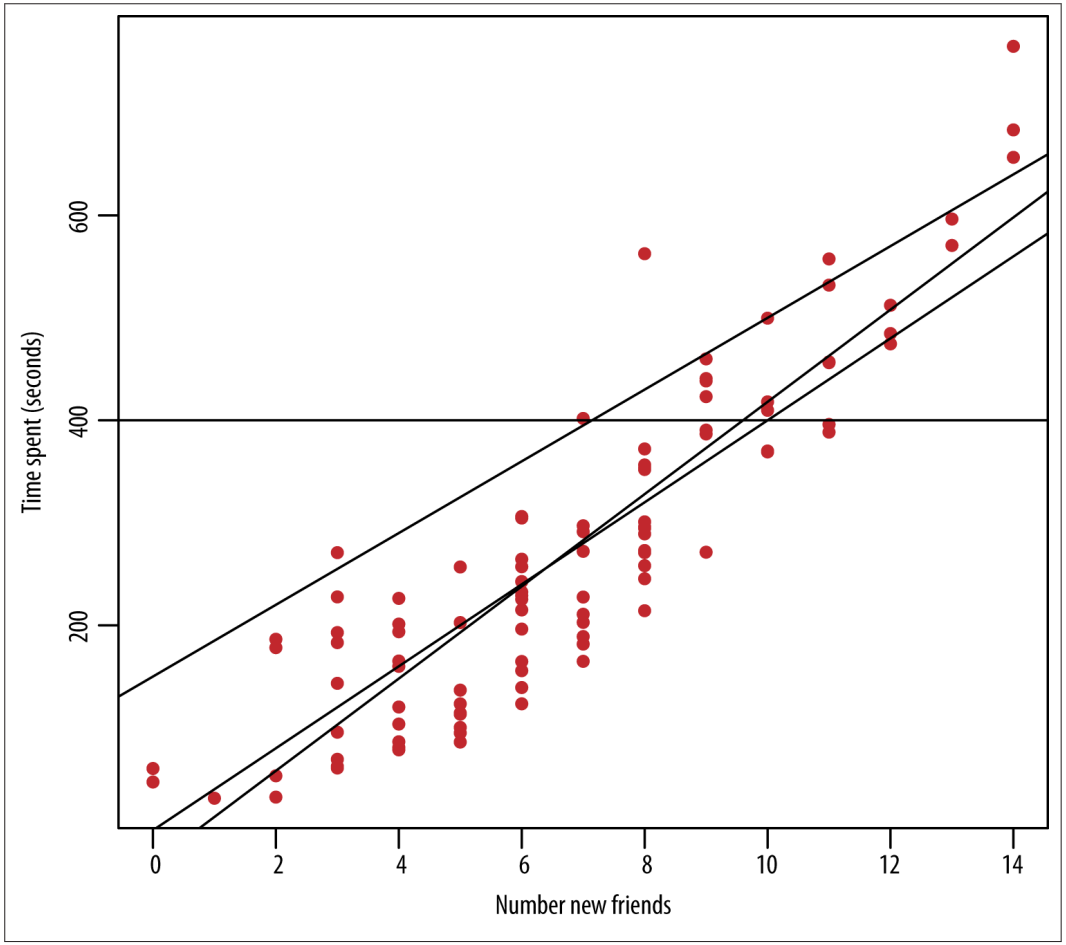
Applicability: Regression can be applied when all variables/attributes are real numbers.

Explore the relationship



From Schutt & O'Neil, *Doing Data Science*

Explore the relationship



From Schutt & O'Neil, *Doing Data Science*

Fitting the model

Want to choose the *best* line.

Operationally, the line that minimises the *distance* between all points and the line.

Recall Euclidean distance: $d(A, B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$

Least squares estimation: find the line that minimises the sum of the squares of the vertical distances between approximated/predicted $\hat{y}_i$s and observed y_is. Put another way, we want to find the β that produces $\hat{y}_i$ for each x_i that is closest to the known y_i.

Minimise the Residual Sum of Squares (RSS)
(aka Sum of Squares Due to Error (SSE)):

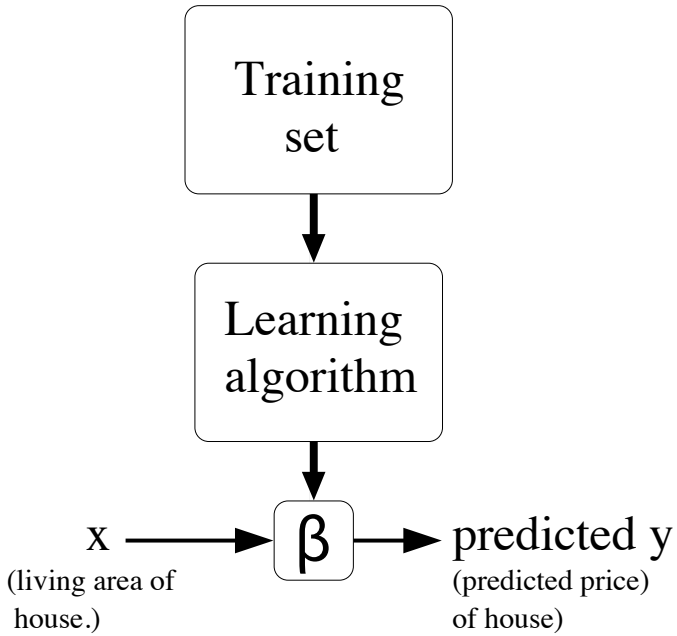
$$RSS(\beta) = \sum_i (y_i - \beta x_i)^2$$

Prediction

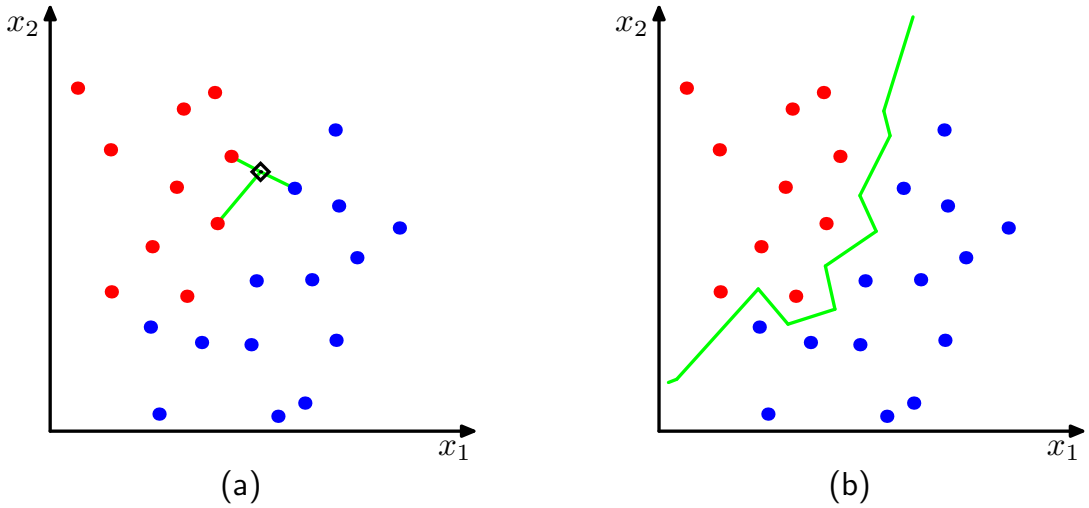
Armed with a linear model $y = \beta_0 + \beta_1 * x$, we can straightforwardly predict a continuous valued output for y given a value of x.

We derive that linear model by estimating it from training examples.

Given examples $(x_0, y_0), (x_1, y_1), ...(x_n, y_n)$, we determine β through least squares estimation.



$k-$Nearest Neighbour methods in Classification



Given class assignments for existing data points, classify a new point (black).

(a) According to the class membership of the K closest data points.
(b) For $k = 1$, the induced decision boundary.

See: Charles Elkan, UCSD, 2011 lecture notes (posted on LMS)

$k-$Nearest Neighbour classification strategies

[1-NN]: Classify the test input according to the class of the closest training instance.

[k-NN]: Classify the test input according to the majority class of the k nearest training instances.

[weighted k-NN]: Classify the test input according to the weighted accumulative class of the k nearest training instances, where weights are based on similarity of the input to each of the k neighbours.

[offset-weighted k-NN]: Classify the test input according to the weighted accumulative class of the k nearest training instances, where weights are based on similarity of the input to each of the k neighbours, factoring in an offset to indicate the prior expectation of a test input being classified as being a member of that class.

k—Nearest Neighbour classification implementation

The most naive neighbour search implementation involves the brute-force computation of distances between all pairs of points in the dataset.

For N samples in D dimensions, this approach scales as $O(DN^2)$.

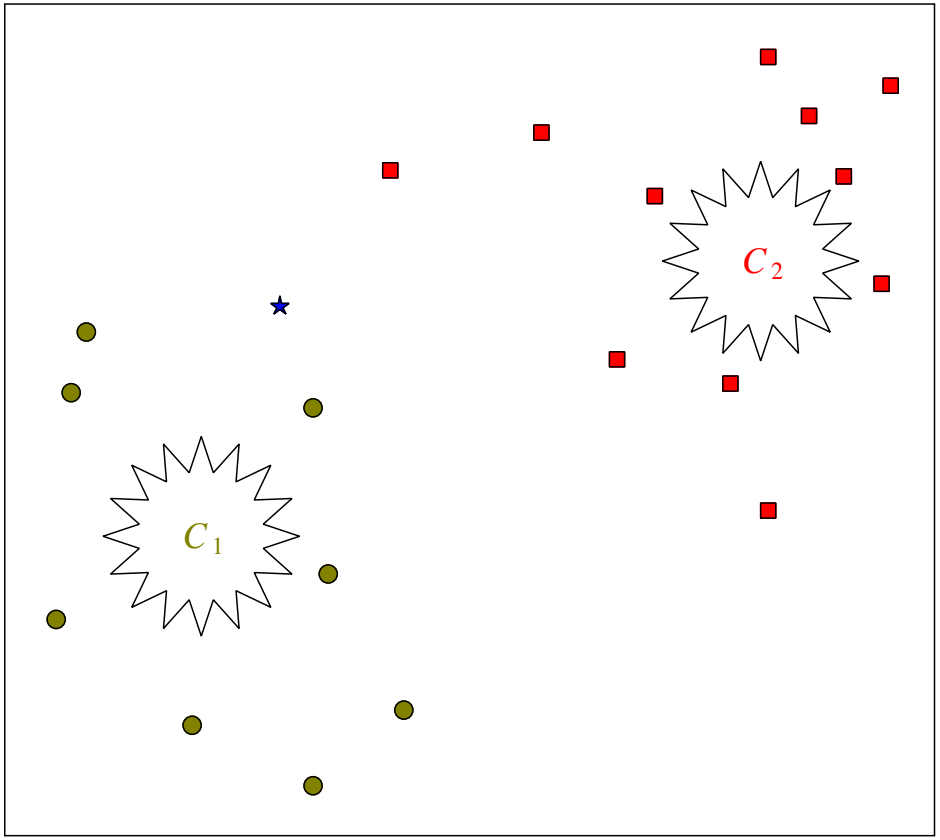
- Efficient brute-force neighbours searches can be very competitive for small data samples.
- However, as the number of samples N grows, the brute-force approach quickly becomes infeasible.

Alternative: tree-based data structures

- These structures attempt to reduce the required number of distance calculations by efficiently encoding aggregate distance information for the sample.
- The basic idea is that if point A is very distant from point B, and point B is very close to point C, then we know that points A and C are very distant, without having to explicitly calculate their distance.
- In this way, the computational cost of a nearest neighbours search can be reduced to $O(DN\log(N))$ or better.

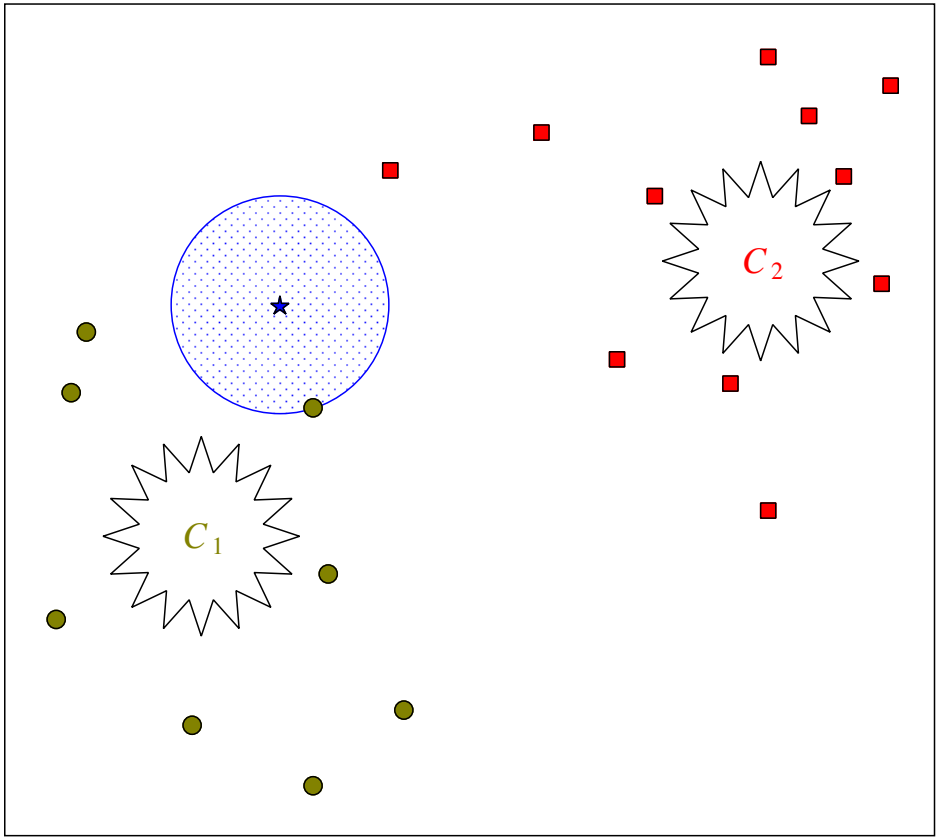
Visualisation of $k-$Nearest Neighbour classification

The nearest neighbour approach corresponds to classification by "hyper-spheres" (or "hyper-ellipsoids")



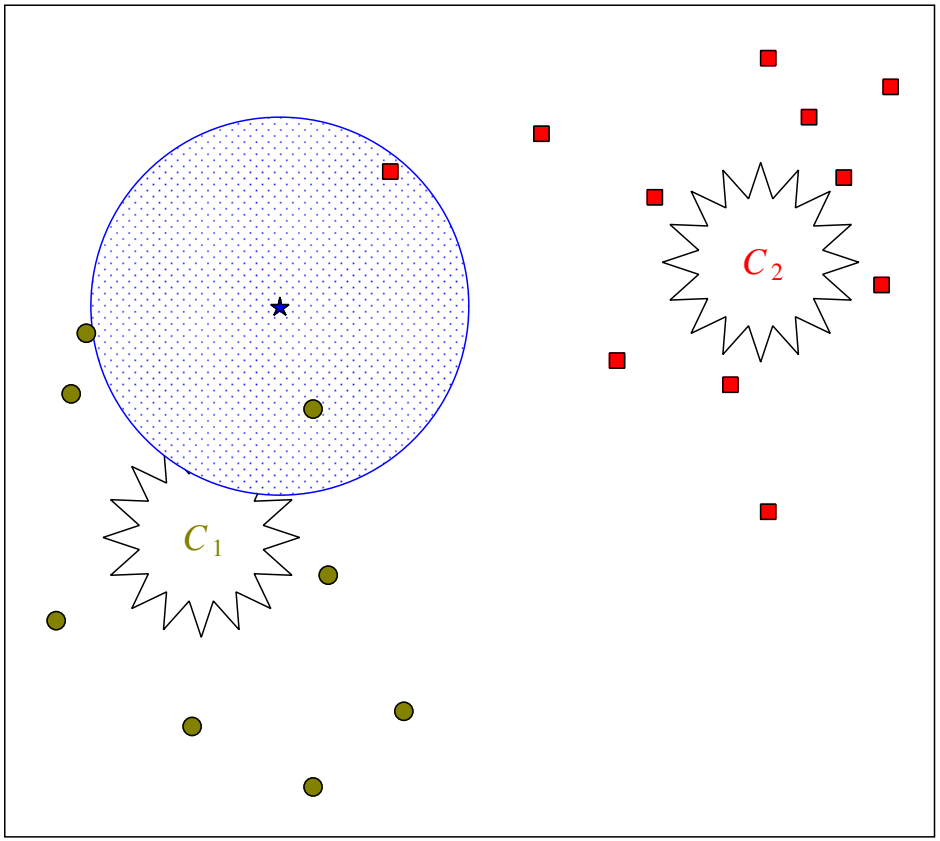
Visualisation of $k-$Nearest Neighbour classification

The nearest neighbour approach corresponds to classification by "hyper-spheres" (or "hyper-ellipsoids")



Visualisation of $k-$Nearest Neighbour classification

The nearest neighbour approach corresponds to classification by "hyper-spheres" (or "hyper-ellipsoids")



Strengths and Weaknesses of Nearest Neighbour methods

Strengths

- Simple
- Can handle arbitrarily many classes (multi-class and multi-label)

Weaknesses

- We need a useful distance function, which may not be obvious to design for some sets.
- We need some sort of averaging or voting function for combining the labels of multiple training examples, which may also not be obvious to design.
- Expensive (in terms of index accesses)
- Everything is done at run time (lazy learner)
- Prone to bias
- Arbitrary k value

Bayesian Methods

- Learning and classification methods based on probability theory
- Build a *generative model* that approximates how data is produced
- Categorisation produces a posterior probability distribution over the possible categories given a description of an instance

Bayes' Rule

$$P(C, X) = P(C|X)P(X) = P(X|C)P(C)$$

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

Naive Bayes (NB) Classifiers

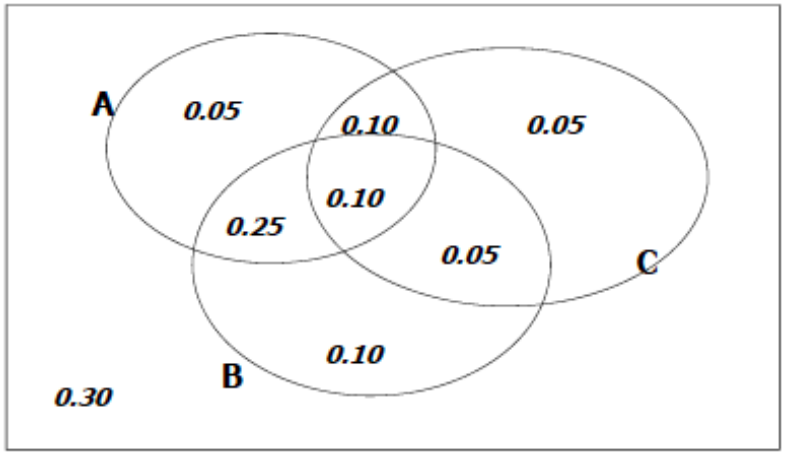
- Task: classify an instance $X = \langle x_1, x_2, ..., x_n \rangle$ according to one of the classes $c_j \in C$

$$\begin{aligned} c &= \text{argmax}_{c_j \in C} P(c_j | x_1, x_2, ..., x_n) \\ &= \text{argmax}_{c_j \in C} \frac{P(x_1, x_2, ..., x_n | c_j) P(c_j)}{P(x_1, x_2, ..., x_n)} \\ &= \text{argmax}_{c_j \in C} P(x_1, x_2, ..., x_n | c_j) P(c_j) \end{aligned}$$

posterior $P(c_j | x_1, x_2, ..., x_n) = \frac{likelihood * prior}{evidence}$

- Predicts X belongs to c_i iff the probability $P(c_i | X)$ is the highest among all the $P(c_k | X)$ for all the K classes
- Since $P(x_1, x_2, ..., x_n)$ is constant for all classes, only $P(x_1, x_2, ..., x_n | c_j) P(c_j)$ needs to be maximised.

Calculating the likelihood



Must determine the probability of *each combination of values* (given a class).

For large n,

1. Typically not enough data to estimate this accurately.
2. Common to encounter the situation where there are no training examples for a particular combination.
3. This would likely lead to over-fitting (biased to combinations for which there are examples).

Simplifying Assumptions

- $P(c_j)$
  - can be estimated from the frequency of classes in the training examples *[maximum likelihood estimate]*
- $P(x_1, x_2, ..., x_n|c_j)$
  - $O(|X|^n|C|)$ parameters (cannot be estimated in practice)

- Naive Bayes Conditional Independence Assumption:
  - assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(x_i|c_j)$
    [hence "naive"]

The Final NB Formulation

- Applying the conditional independence assumption:

$$\begin{aligned} c &= \text{argmax}_{c_j \in C} P(x_1, x_2, ..., x_n | c_j) P(c_j) \\ &= \text{argmax}_{c_j \in C} P(c_j) \prod_i P(x_i | c_j) \end{aligned}$$

Naive Bayes Example

Given a training data set, what are the probabilities we need to estimate?

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Ann comes to the clinic with severe headache, no soreness, normal temperature and with cough. What does she have? Choose the case with highest probability.

$$P(Flu|Headache = severe, Sore = no, Temperature = normal, Cough = yes) \sim P(Flu) * P(Headache = severe|Flu) * P(Sore = no|Flu) * P(Temperature = normal|Flu) * P(Cough = yes|Flu)$$

$$P(Cold|Headache = severe, Sore = no, Temperature = normal, Cough = yes) \sim P(Cold) * P(Headache = severe|Cold) * P(Sore = no|Cold) * P(Temperature = normal|Cold) * P(Cough = yes|Cold)$$

Estimating probabilities

$$P(Flu) = 3/5$$
$$P(Headache = severe|Flu) = 2/3$$
$$P(Headache = mild|Flu) = 1/3$$
$$P(Headache = no|Flu) = 0/3\ (= e)$$
$$P(Sore = severe|Flu) = 1/3$$
$$P(Sore = mild|Flu) = 2/3$$
$$P(Sore = no|Flu) = 0/3\ (= e)$$
$$P(Temp = high|Flu) = 1/3$$
$$P(Temp = normal|Flu) = 2/3$$
$$P(Cough = yes|Flu) = 3/3$$
$$P(Cough = no|Flu) = 0/3\ (= e)$$

$$P(Cold) = 2/5$$
$$P(Headache = severe|Cold) = 0/2\ (= e)$$
$$P(Headache = mild|Cold) = 1/2$$
$$P(Headache = no|Cold) = 1/2$$
$$P(Sore = severe|Cold) = 1/2$$
$$P(Sore = mild|Cold) = 0/2\ (= e)$$
$$P(Sore = no|Cold) = 1/2$$
$$P(Temp = high|Cold) = 0/2\ (= e)$$
$$P(Temp = normal|Cold) = 2/2$$
$$P(Cough = yes|Cold) = 1/2$$
$$P(Cough = no|Cold) = 1/2$$

Set $0/y$ to e, a small value like 10^{-7} (or less than $\frac{1}{n}$ where n is the number of training instances)

$P(Flu|Headache = severe, Sore = no, Temperature = normal, Cough = yes)$
$\sim\ P(Flu) * P(Headache = severe|Flu) * P(Sore = no|Flu) * P(Temperature = normal|Flu) * P(Cough = yes|Flu) = 3/5 * 2/3 * e * 2/3 * 3/3 = 0.26e$

$P(Cold|Headache = severe, Sore = no, Temperature = normal, Cough = yes)$
$\sim\ P(Cold) * P(Headache = severe|Cold) * P(Sore = no|Cold) * P(Temperature = normal|Cold) * P(Cough = yes|Cold)$
$= 2/5 * e * 1/2 * 1 * 1/2 = 0.1e$ *Diagnosis is Flu*

Naive Bayes, analysis

Naive Bayes (NB) Classifier is very simple to build, extremely fast to make decisions, and easy to change the probabilities when the new data becomes available.

- Works well in many application areas.
- Scales easily for large number of dimensions (100s) and data sizes.
- Easy to explain the reason for the decision made.
- One should apply NB first before launching into more sophisticated classification techniques.

Summary

- How does the k-nearest neighbour method operate, and what are some of the variants on the original algorithm?
- How does the Naive Bayes algorithm work? What assumptions are required to make the computation tractable?

Resources

Charles Elkan, UCSD, lecture notes
`http://cseweb.ucsd.edu/~elkan/250Bwinter2010/nearestn.pdf`

Witten, Frank, Hall (2011) Data Mining. Chapter 4. (*kD – tree*, ball tree)