Machine Learning, Machine Learning (extended)

6 – Supervised Learning: Instance-based Classification

Kashif Rajpoot

k.m.rajpoot@cs.bham.ac.uk

School of Computer Science

University of Birmingham

Outline

- Supervised learning
- Instance based learning
- k-nearest neighbour (kNN) classification
- Distance measure
- Difficulties with kNN
- How to choose k?
- Distance-weighted kNN

Supervised learning

- Regression
  - Minimised loss (e.g. least squares)
  - Maximum likelihood

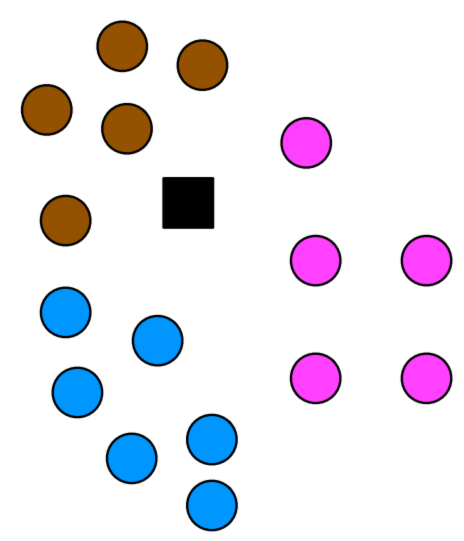
- Classification
  - Generative (e.g. Bayesian)
  - Instance-based (e.g. k-NN)
  - Discriminative (e.g. SVM)

Classification

- A set of N objects with attributes $\boldsymbol{x}_n$
- Each object has an associated target label t_n
- Binary classification

  $t_n \in \{0,1\}$ or $t_n \in \{-1,1\}$
- Multi-class classification

  $t_n \in \{1,2,\dots,C\}$



- Classifier learns from $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N$ and $t_1, t_2, \dots, t_N$ so that it can later classify $\boldsymbol{x}_{new}$

Instance-based supervised learning

- There is no ‘training’ involved
  - Stores all ‘training’ samples
- Non-probabilistic classification
  - Look at nearest instances from the past examples to determine target labels (discrete or real-valued) of new unseen samples
  - k-nearest neighbour (kNN) classification
- Suitable for binary or multi-class classification

kNN classification

- Store all N ‘training’ samples represented by attributes $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ and labels $t_1, t_2, \ldots, t_N$

- Choose parameter k
  - i.e. number of nearest neighbours to consider for assigning target label to a new test sample $\boldsymbol{x}_{new}$

- For discrete labels, classify a new test sample $\boldsymbol{x}_{new}$:
  - Find k nearest (closest) training samples/instances
  - Find the most common class (label) out of these k neighbours
  - Assign this class to $\boldsymbol{x}_{new}$
  - In case of a tie, assign randomly from tied classes

kNN classification

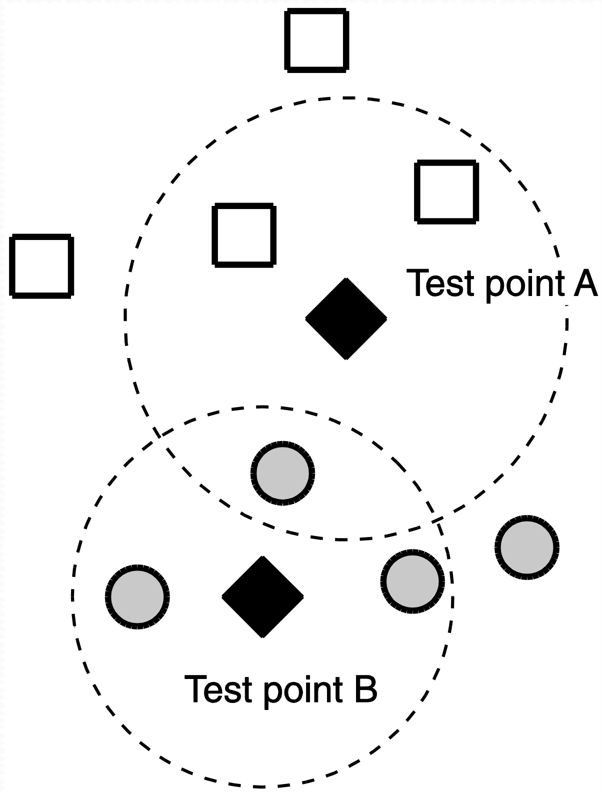
- Training algorithm
  - Store all training instances ($\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N$ and $t_1, t_2, \dots, t_N$) as *training_examples*
- Classification algorithm
  - Given a test instance $\boldsymbol{x}_{new}$ to be classified:
    - Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k$ denote the k nearest instances to $\boldsymbol{x}_{new}$ from the *training_examples*
    - Let $t_1, t_2, \dots, t_k$ denote the labels of k nearest instances to $\boldsymbol{x}_{new}$
    - Assign target label t_{new} to $\boldsymbol{x}_{new}$ as below:

$$t_{new} = \underset{c \in \{1,2,\dots,C\}}{argmax} \sum_{i=1}^{k} \delta(c, t_i)$$

where $\delta(c, t_i) = 1$ if $c = t_i$, or $\delta(c, t_i) = 0$ otherwise

kNN classification

- For discrete labels, classify a new test sample $\boldsymbol{x}_{new}$ as below:
  - Find k nearest (closest) training samples/instances
  - Find the most common class (label) out of these k neighbours
  - Assign this class to $\boldsymbol{x}_{new}$
  - In case of a tie, assign randomly from tied classes
- $k = 3$



kNN classification

- For real-valued target labels, classify a new test sample $\boldsymbol{x}_{new}$ as below:
  - Find k nearest (closest) training samples/instances
  - Find the average of k nearest target labels

$$t_{new} = \frac{1}{k}\sum_{i=1}^{k} t_i$$

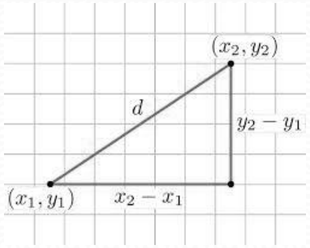
  - Assign this target label to $\boldsymbol{x}_{new}$

- Regression?

Distance measure

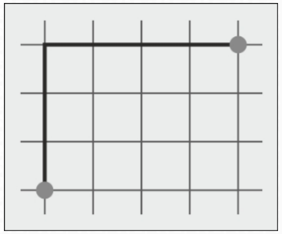
- To determine the nearest neighbours, a distance measure is required
  - kNN algorithm is flexible to use any distance measure
  - Thus, kNN can be used for any data type (images, text, audio, etc.) for which distance measure is available

Euclidean distance



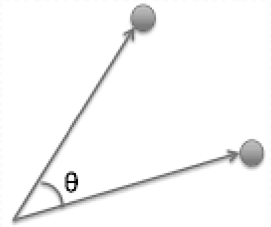
$$D_e(\boldsymbol{p},\boldsymbol{q}) = \sqrt{\sum_{i=1}^{d}(\boldsymbol{p}_i - \boldsymbol{q}_i)^2}$$

Manhattan distance



$$D_m(\boldsymbol{p},\boldsymbol{q}) = \sum_{i=1}^{d}|\boldsymbol{p}_i - \boldsymbol{q}_i|$$

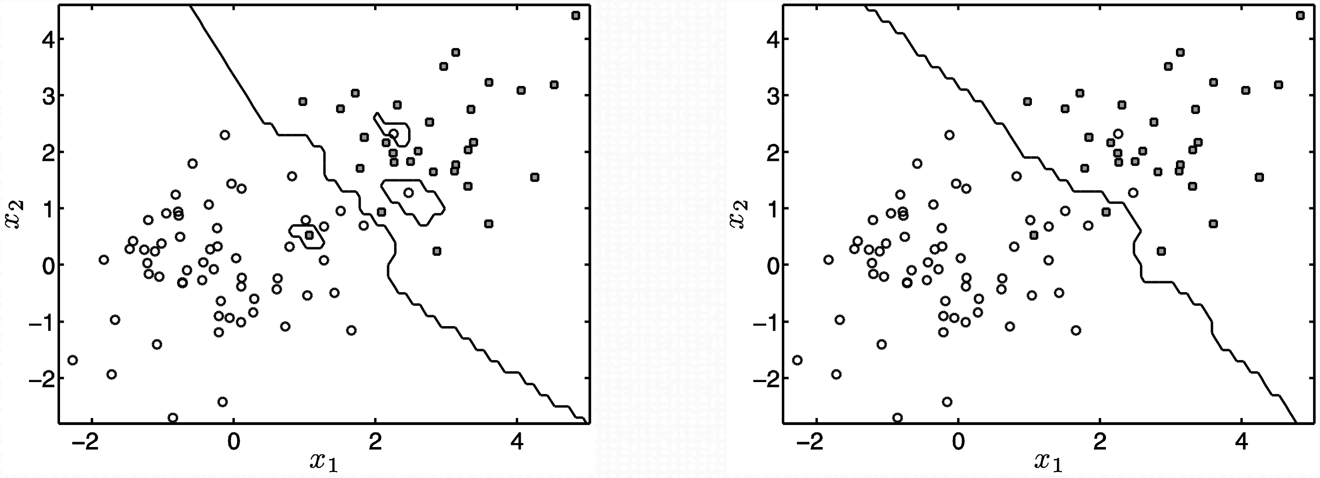
Cosine angle



$$D_c(\boldsymbol{p},\boldsymbol{q}) = cos^{-1}\left(\frac{\sum_{i=1}^{d}\boldsymbol{p}_i\boldsymbol{q}_i}{\sqrt{\sum_{i=1}^{d}\boldsymbol{p}_i^2}\sqrt{\sum_{i=1}^{d}\boldsymbol{q}_i^2}}\right)/\pi$$

Role of k

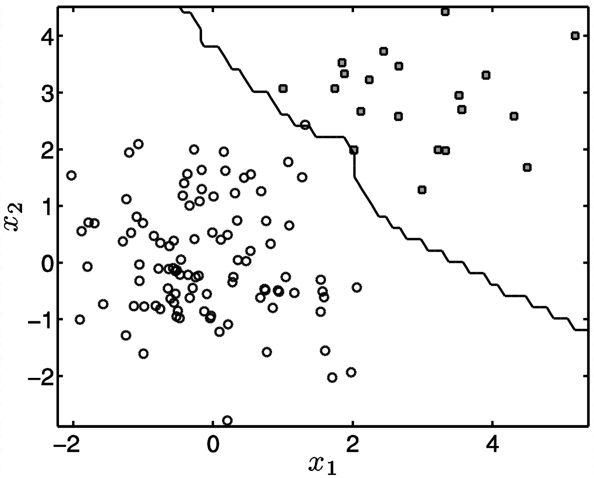
- Line indicates decision boundary
- Noisy data



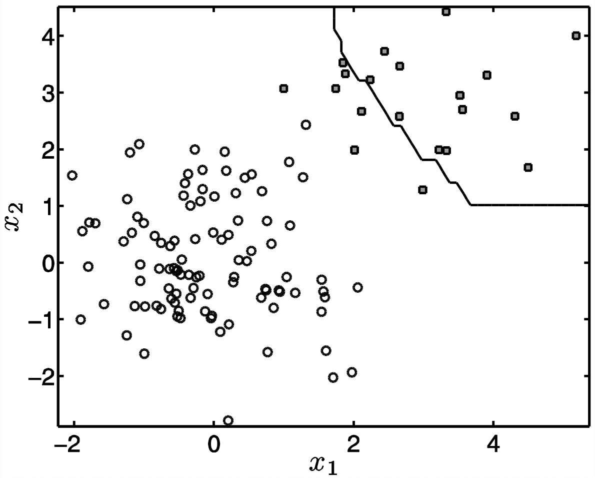
(a) Decision boundary when $K = 1$ (b) Decision boundary when $K = 5$

Role of k

- Binary classification
  - 50 training samples vs 20 training samples
- Suitable value of k regularizes boundary and thus avoids over-fitting



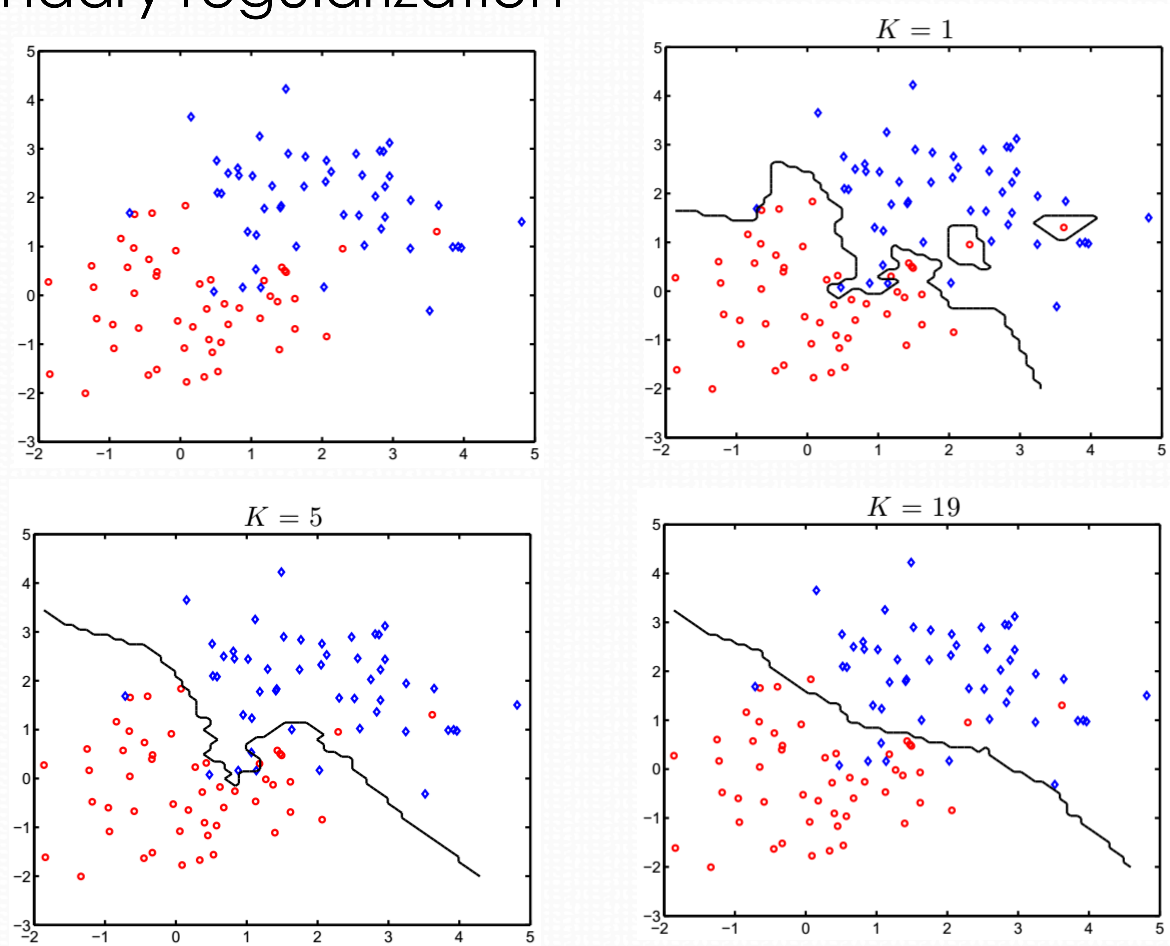
(a) Decision boundary when $K = 5$



(b) Decision boundary when $K = 39$

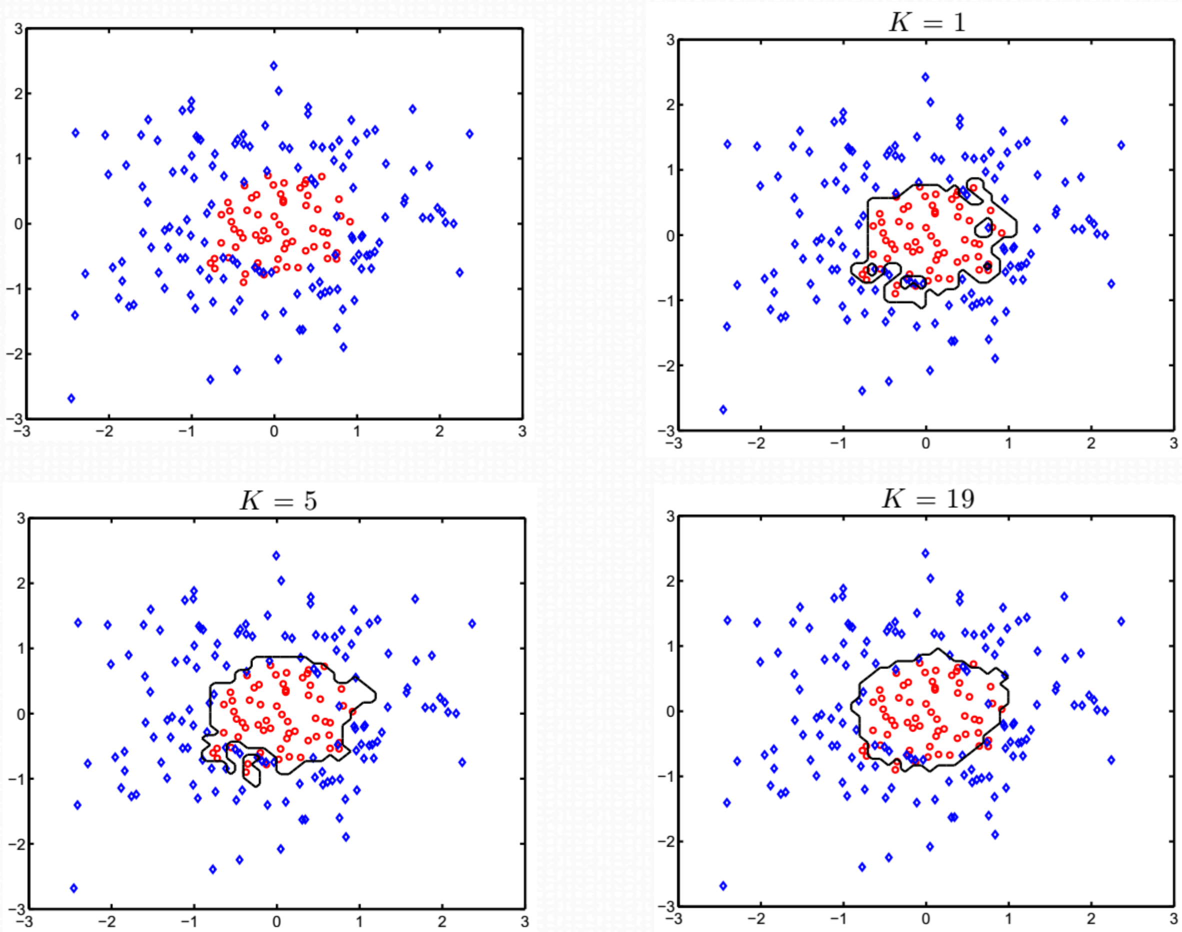
Role of k

- Boundary regularization



Role of k

- Boundary regularization



Difficulties with kNN

- Class imbalance : uneven number of objects in each class
  - Small classes will 'disappear' with increase in k value
  - Example: 5 training samples from class 1 and 100 training samples from class 2
  - For $k >= 11$, class 1 'disappears'

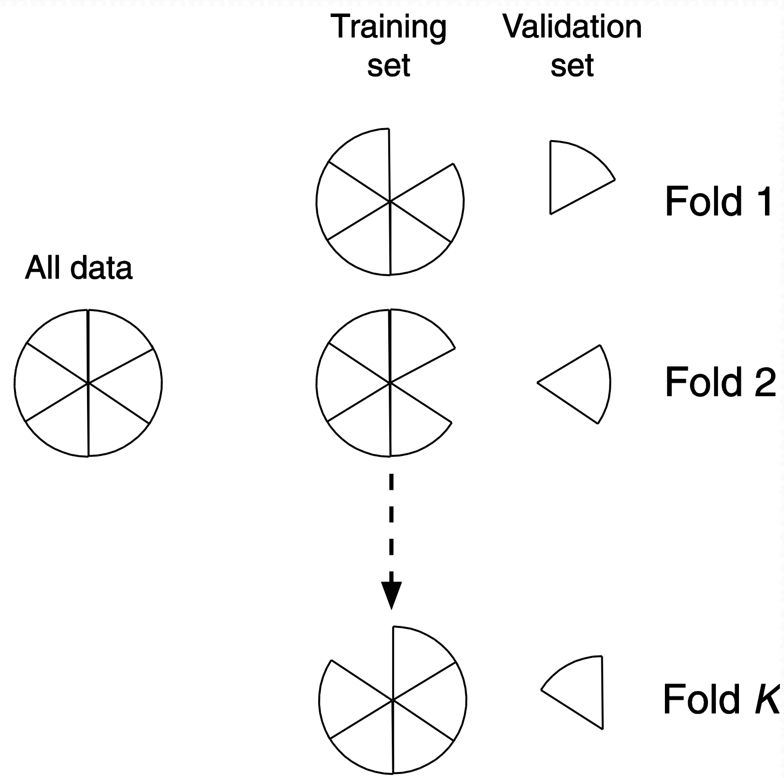
- How do we choose k?
  - Depends on data
  - Cross-validation

- Computational cost for classification can be high
  - Computation takes place at the time of classification
  - Calculates the distance of a test sample from all training samples

- There may be irrelevant attributes amongst the attributes

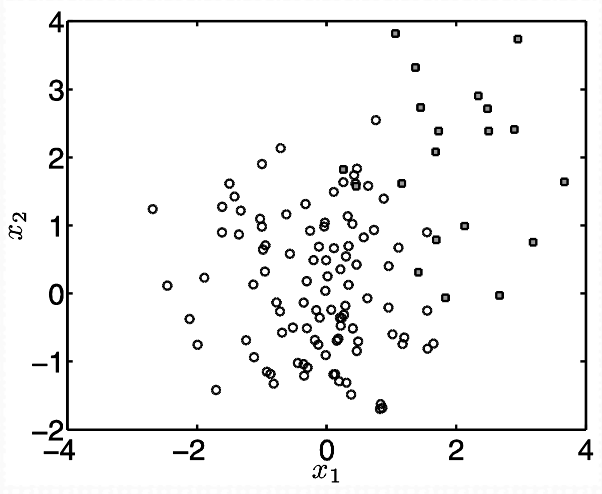
How to choose k?

- Cross-validation
  - Split the data
  - Use some for training, some for testing
  - Need a measure of ‘goodness’ or ‘accuracy’ or ‘error’
  - Determine percentage misclassification error
  - Find k that minimises misclassification error

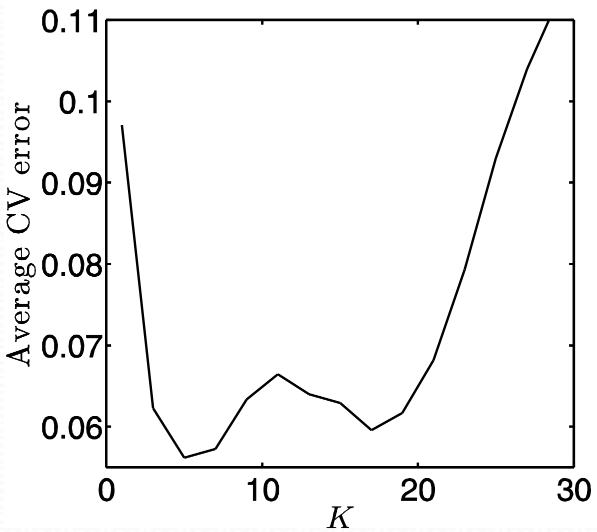


How to choose k?

- Cross-validation
  - 10 fold CV repeated 100 times to remove the effect of cross-validation partitioning bias



(a) Binary classification dataset. Note the class inbalance: the grey squares class has fewer members than the white circles



(b) Average cross-validation error as K is increased

Characteristics of instance-based learning

- Instance-based learner is a *lazy learner*
  - It does all the work when the test sample is presented

- In contrast, *eager learner* builds a parameterized model from training samples, in advance of being presented with a test sample

- Instance-based learner produces *local* approximation to the target function
  - Finds a different approximation with each test instance

When to consider kNN?

- Not too many attributes (about 20)
- Lots of training data
- Advantages
  - Training is super fast
  - It can learn complex target functions
  - Doesn’t loose information

Distance-weighted kNN

- How about weighting nearest neighbours more strongly than the farthest neighbours in determining the target label t_{new}?
- Classification algorithm
  - Given a test instance $\boldsymbol{x}_{new}$ to be classified:
    - Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ denote the k nearest instances to $\boldsymbol{x}_{new}$ from the *training_examples*
    - Let $t_1, t_2, \ldots, t_k$ denote the labels of k nearest instances to $\boldsymbol{x}_{new}$
    - Assign target label t_{new} to $\boldsymbol{x}_{new}$ as below:

$$t_{new} = \underset{c \in C}{argmax} \sum_{i=1}^{k} w_i \delta(c, t_i)$$

where $w_i = \frac{1}{D(\boldsymbol{x}_i, \boldsymbol{x}_{new})}$

Distance-weighted kNN

- This opens up the possibility to use all training samples as neighbours rather than only k neighbours

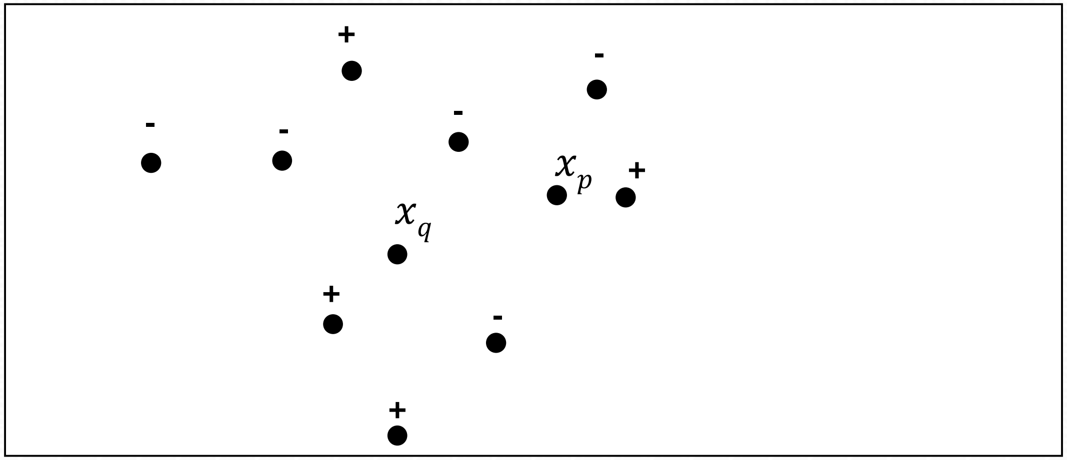
- As a result, the classifier becomes a global function approximation method
  - In contrast, typical kNN is a local function approximation method

Summary

- Instance-based classification
- Fast ‘learning’
- Simple setup
- Distance measure choice is flexible
- k is the only parameter that needs tuning
- Class imbalance in data needs consideration

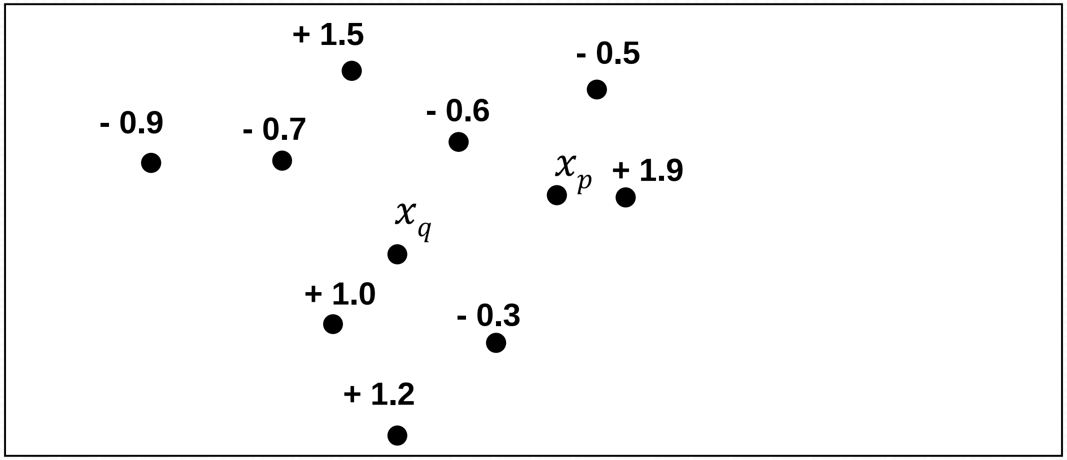
Exercise (ungraded)

- Assume a Boolean target function (i.e. binary classifier) and a two dimensional instance space. Determine how the kNN would classify the test instances x_p and x_q for $k = 1$, $k = 3$ and $k = 5$.



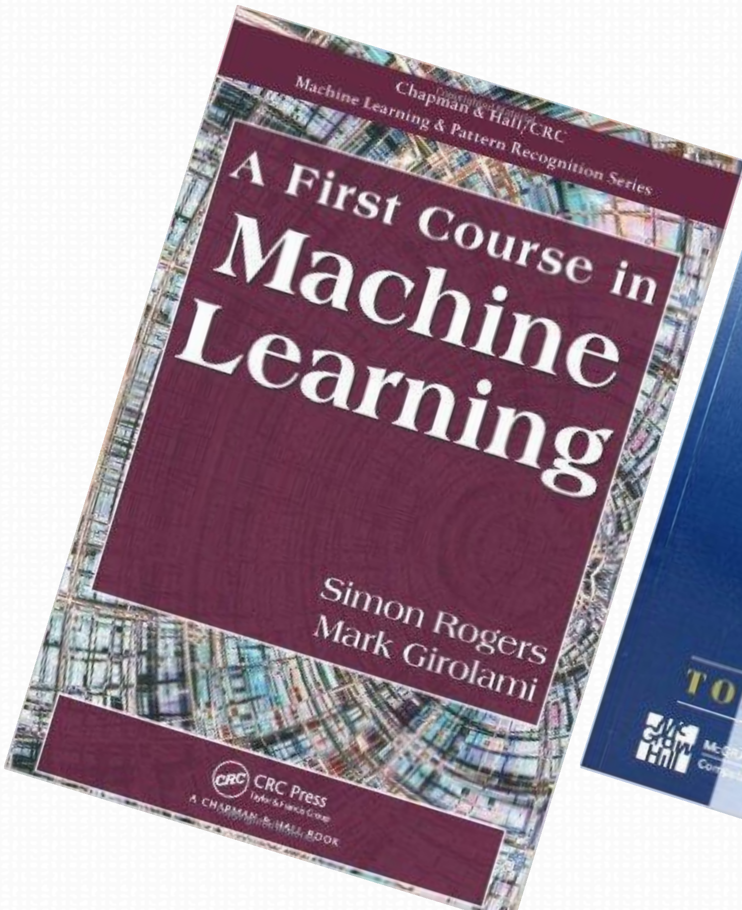
Exercise (ungraded)

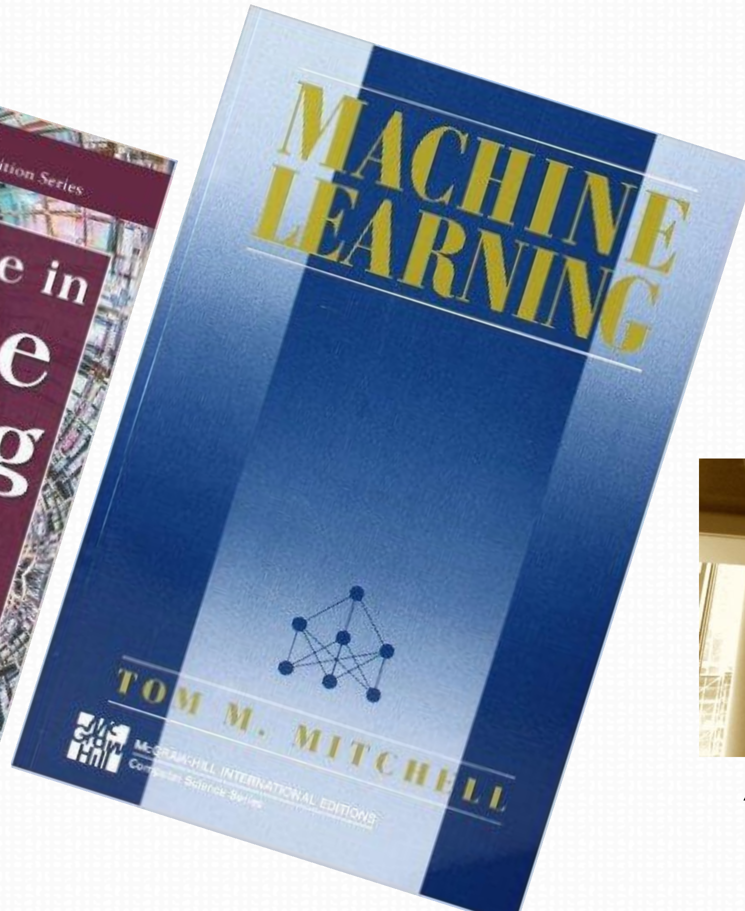
- In the diagram below, the numbers refer to the values taken by a real-valued target function. Calculate the values predicted for the target function at the test points x_p and x_q by kNN, with $k = 1$, $k = 3$, and $k = 5$.



Exercise (ungraded)

- Try MATLAB code – knnexample.m (from FCML book website)
- Try MATLAB code – knncv.m (from FCML book website)





Author’s material
(Simon Rogers)

That's All

Thank You