COMP6237 Data Mining
Lecture 8: Nearest Neighbours

Zhiwu Huang

Zhiwu.Huang@soton.ac.uk
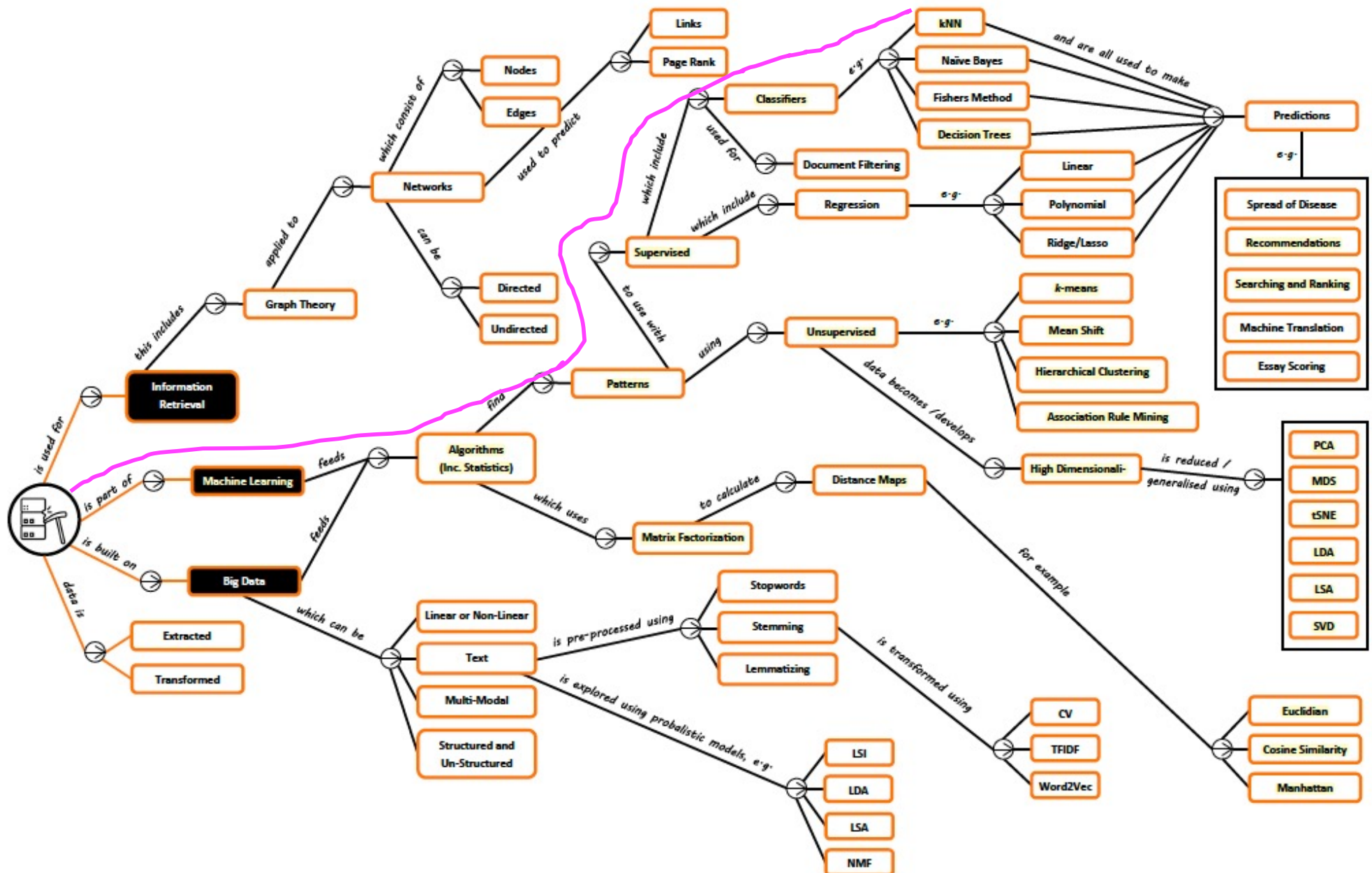
Lecturer (Assistant Professor) @ VLC of ECS

University of Southampton

Lecture slides available here:

http://comp6237.ecs.soton.ac.uk/zh.html

(Thanks to Prof. Jonathon Hare and Dr. Jo Grundy for providing the lecture materials used to develop the slides.)

Nearest Neighbours – **Roadmap**

University of **Southampton**

Nearest Neighbours – **Textbook**

PART FOUR **CLASSIFICATION**

The classification task is to predict the label or class for a given unlabeled point. Formally, a classifier is a model or function M that predicts the class label $\hat{y}$ for a given input example $\mathbf{x}$, that is, $\hat{y} = M(\mathbf{x})$, where $\hat{y} \in \{c_1, c_2, \ldots, c_k\}$ and each c_i is a class label (a categorical attribute value). Classification is a *supervised learning* approach,
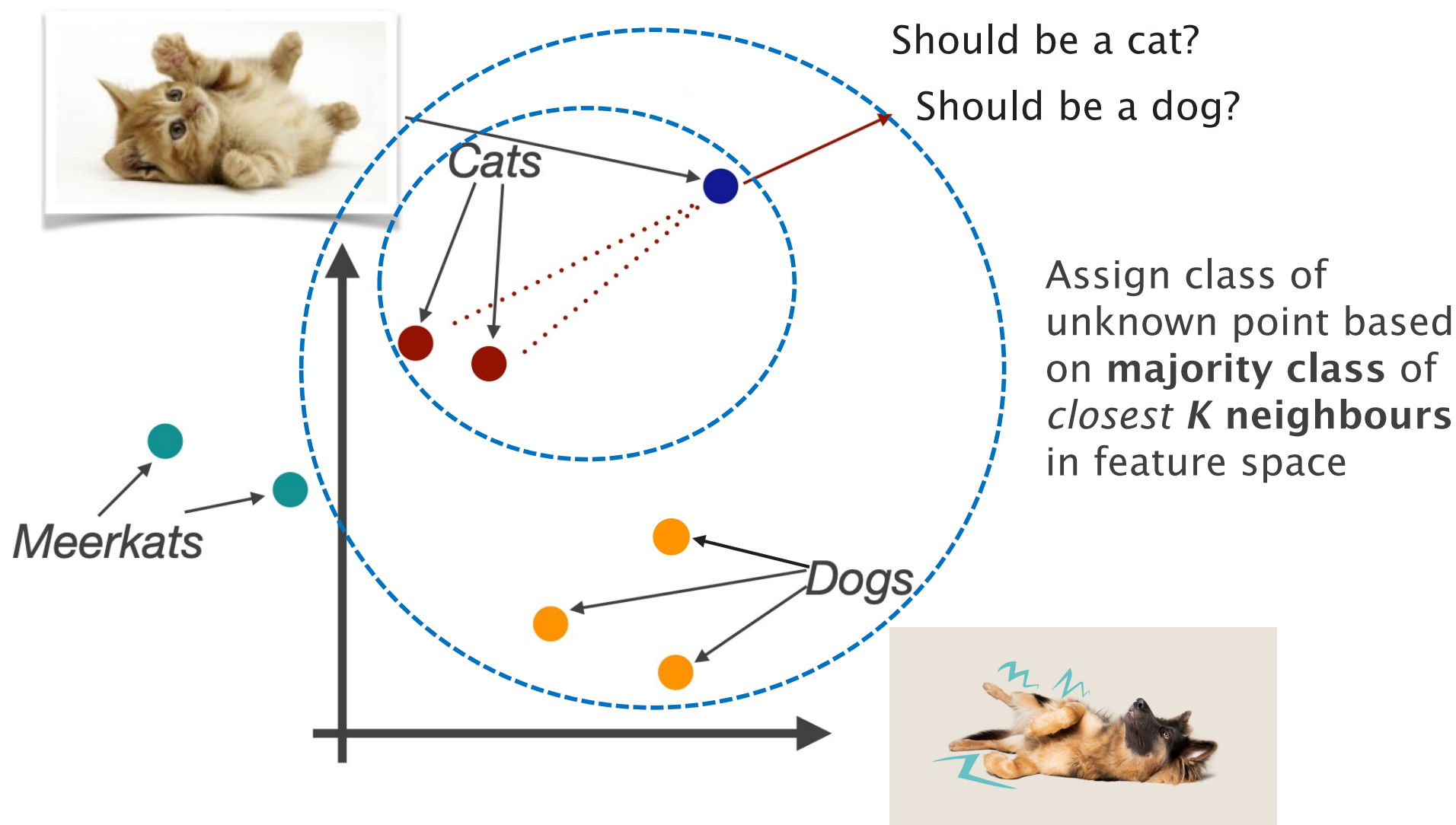
▶ Data Mining and Machine Learning: Fundamental Concepts and Algorithms M. L. Zaki and W. Meira

4 Classification: Alternative Techniques

The previous chapter introduced the classification problem and presented a technique known as the decision tree classifier. Issues such as model overfitting and model evaluation were also discussed. This chapter presents alternative techniques for building classification models—from simple techniques such as rule-based and nearest neighbor classifiers to more sophisticated techniques such as artificial neural networks, deep learning, support vector machines, and ensemble methods. Other practical issues such as the class imbalance and multiclass problems are also discussed at the end of the chapter.

▶ Introduction to Data Mining, *P. Tan et al*

Nearest Neighbours – **Overview**

Should be a cat?

Should be a dog?

Cats

Meerkats

Dogs

Assign class of unknown point based on **majority class** of *closest K neighbours* in feature space

Credit: Jon Hare

Nearest Neighbours – Learning Outcomes

- **LO1**: Demonstrate an understanding of the fundamentals of nearest neighbor classifiers, such as: (exam)

  - ❖ Understanding the k-NN algorithm, including key steps like distance calculations, voting, etc

  - ❖ Applying weighted k-NN and its use of weighted distance calculations

  - ❖ Employing K-D trees for efficient nearest neighbor search

  - ❖ Discussing Advantages and disadvantages of k-NN models

- **LO2**: Implement the learned algorithms using K-NN methods (course work)

Assessment hints: Multi-choice Questions (single answer: concepts, calculation etc)

- *Textbook Exercises: textbooks (Programming + Mining)*
- *Other Exercises: https://www-users.cse.umn.edu/~kumar001/dmbook/sol.pdf*
- *ChatGPT or other AI-based techs*

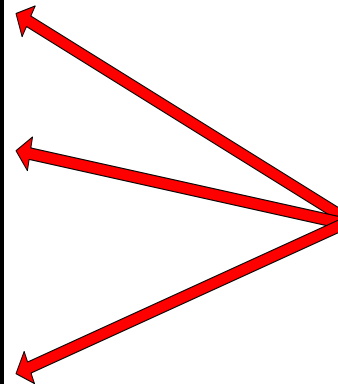# Nearest Neighbours – KNN Algorithm

- Key idea: the neighbors should share the same label
  - Use training records directly to predict the class label of test/unseen cases by considering their neighbor correlations
  - Store the training records without training explicit models

Set of Stored Cases

Atr1	………	AtrN	Class
			A
			B
			B
			C
			A
			C
			B

Unseen Case

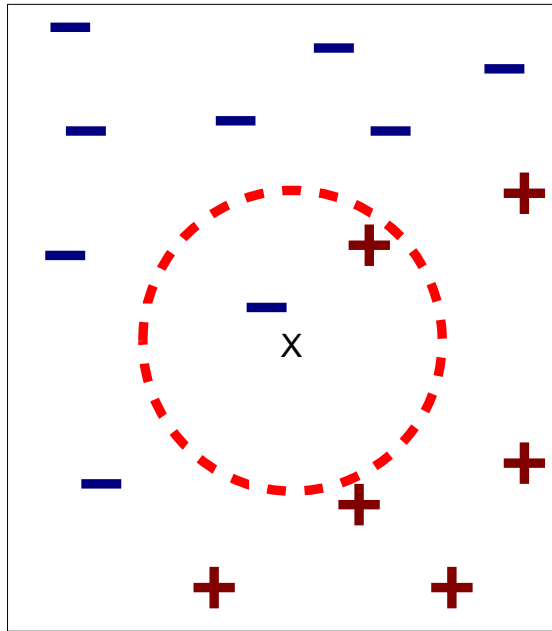| Atr1	………	AtrN

Nearest Neighbours – KNN Algorithm

- Uses k "closest" training points (nearest neighbors) for performing classification on test data
  - Properties include
    - Little or no prior knowledge about the distribution of the data
    - A simple algorithm that stores all available cases and classifies new cases based on a **similarity measure** (e.g., L2 distance and cosine similarity)
    - Computation merely happens on classification over test data
    - Lazy algorithm, as they don't use training data to do any generalization

- **k-Nearest Neighbors** of a record x are data points that have the k shortest distance to x
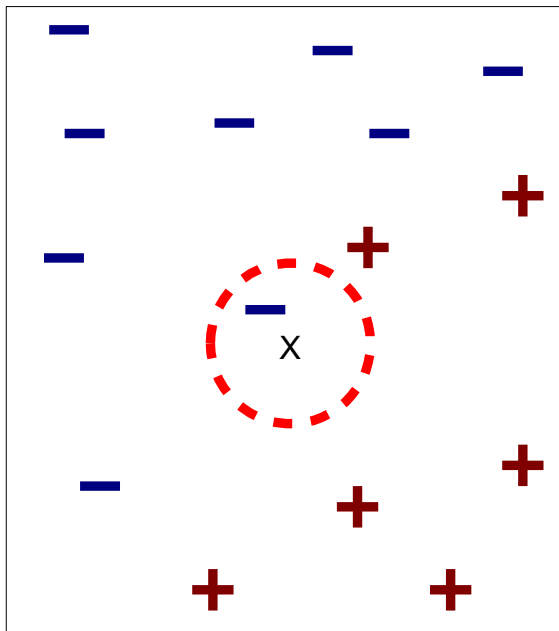
(a) 1-nearest neighbor   (b) 2-nearest neighbor   (c) 3-nearest neighbor
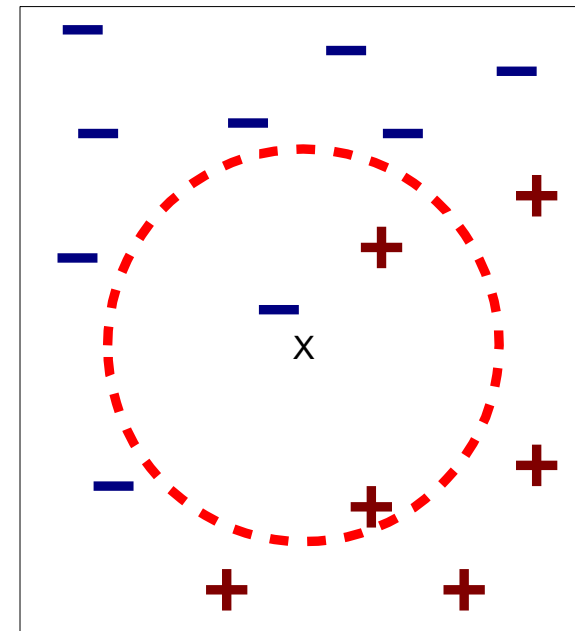
Nearest Neighbours – KNN Algorithm
Measuring distance

- Measure
  - Euclidean Distance
  - Manhattan Distance
  - Hamming Distance
  - Cosine Similarity
  - Many more!
- Selection
  - The use of validation set could be one of the most popular options
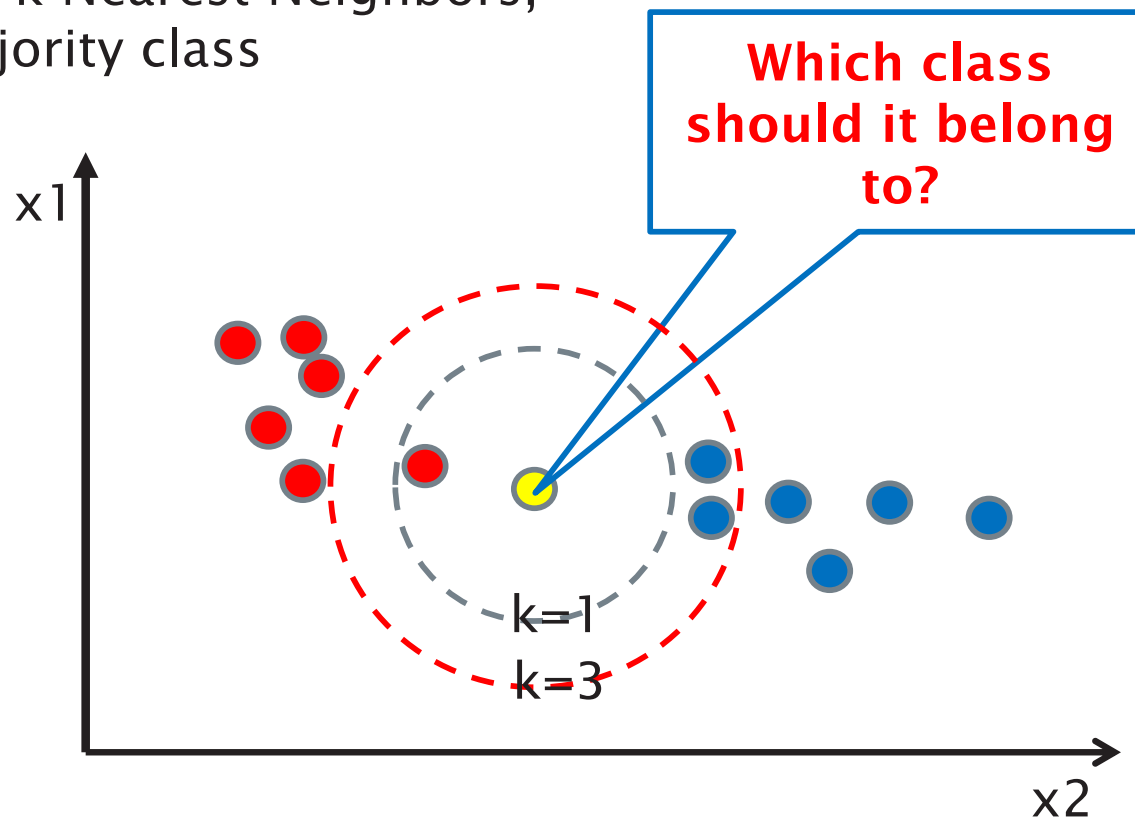
Nearest Neighbours – KNN Algorithm
Measuring distance

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 30kg to 100kg
    - income of a person may vary from $10K to $1M
  - Solution:
   - Data Normalization such as Z-score normalization

University of
Southampton

How k-NN works?

Key idea: 1) find k-Nearest Neighbors,
2) assign the majority class

Which class should it belong to?

Nearest Neighbours – K-NN Algorithm

- ## On choosing K:
  - *K* must not be multiples of the total class number when each vote is equally weighted
    - Odd numbers (1, 3, 5, 7, 9,…) for majority voting on 2-class classification
    - 1, ~~3 (not good),~~ 5, 7, ~~9 (not good)~~… for 3-class classification
  - If K is too small, sensitive to noise points
  - If K is too large, neighborhood may include points from other classes
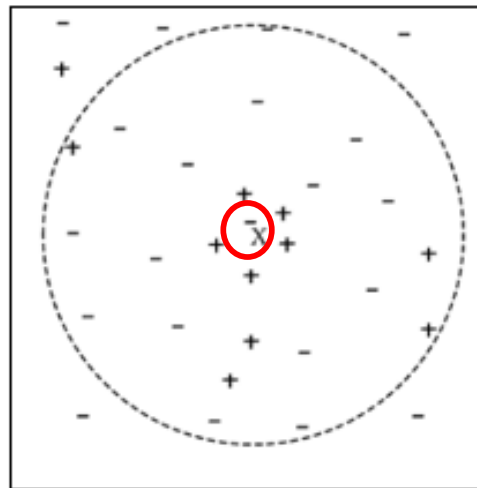
Nearest Neighbours – Example

- We want to train a model to decide an unknown goodie is a *Chocolate Cake* or *Pineapple Tart*

Chocolate Cake   Pineapple Tart

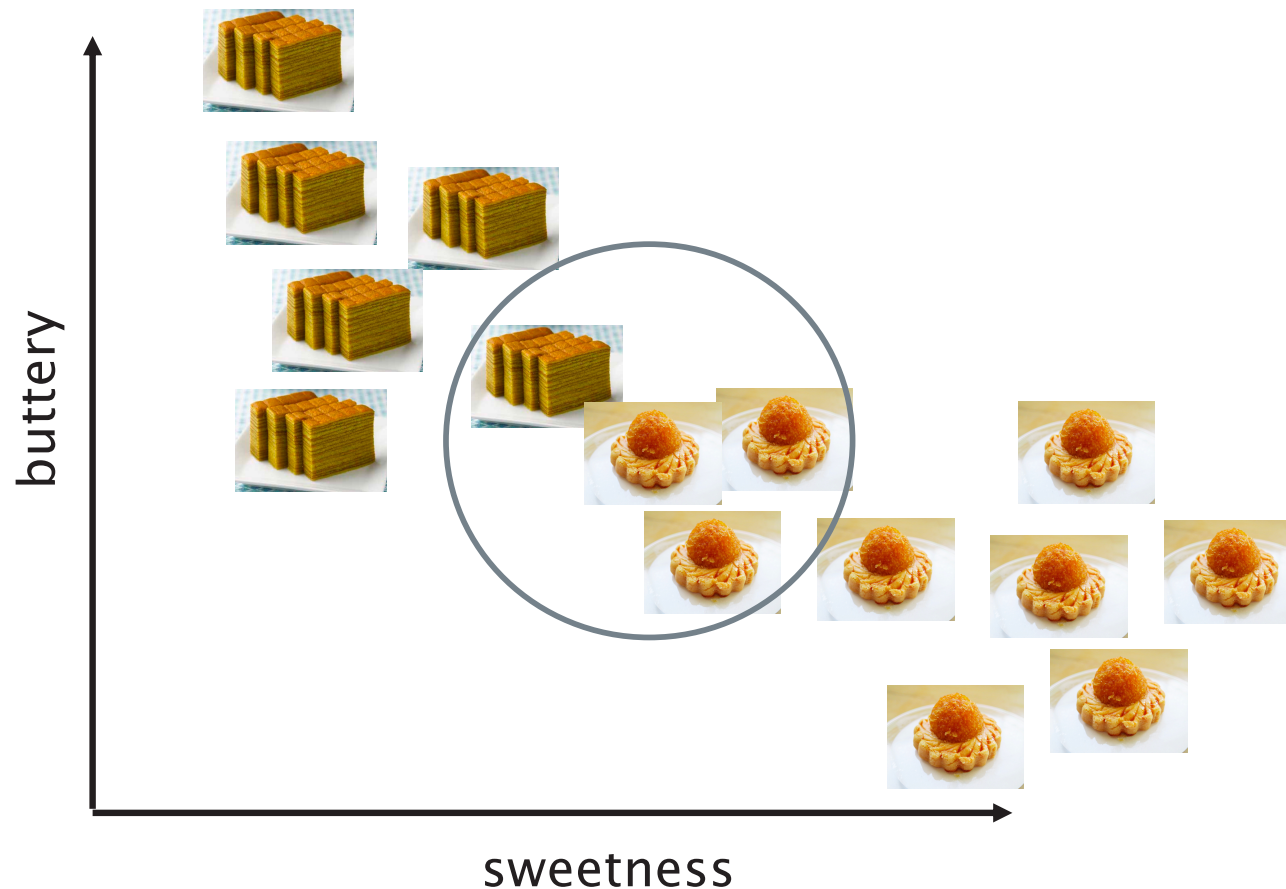Evaluate on 2 attributes: "sweetness" and "buttery" level

Nearest Neighbours – Example

- Suppose we have k set to 3 (k = 3 nearest neighbors)
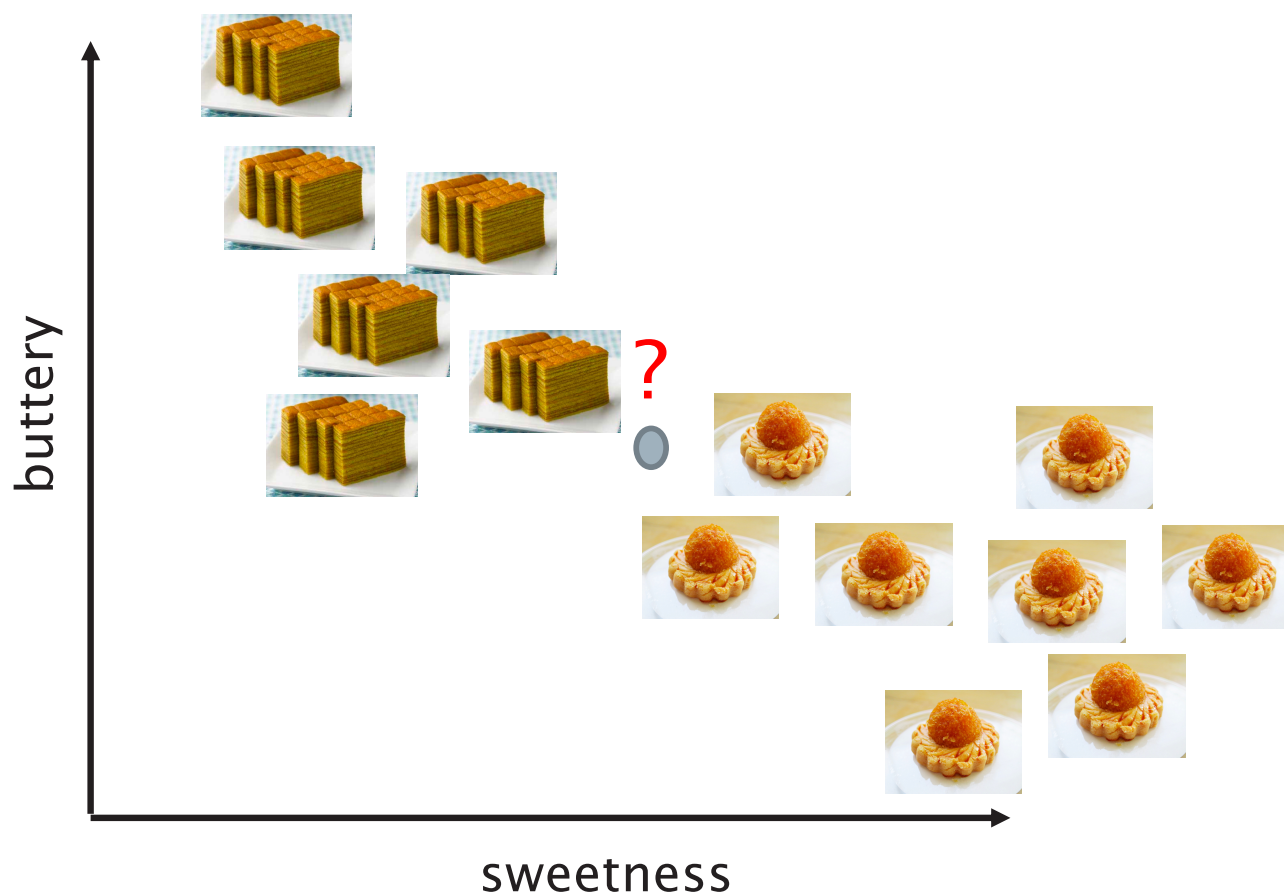 - 2 neighbors are *Pineapple Tarts* and 1 neighbor is *Chocolate Cake*

Credit: Zhaoxia Wang
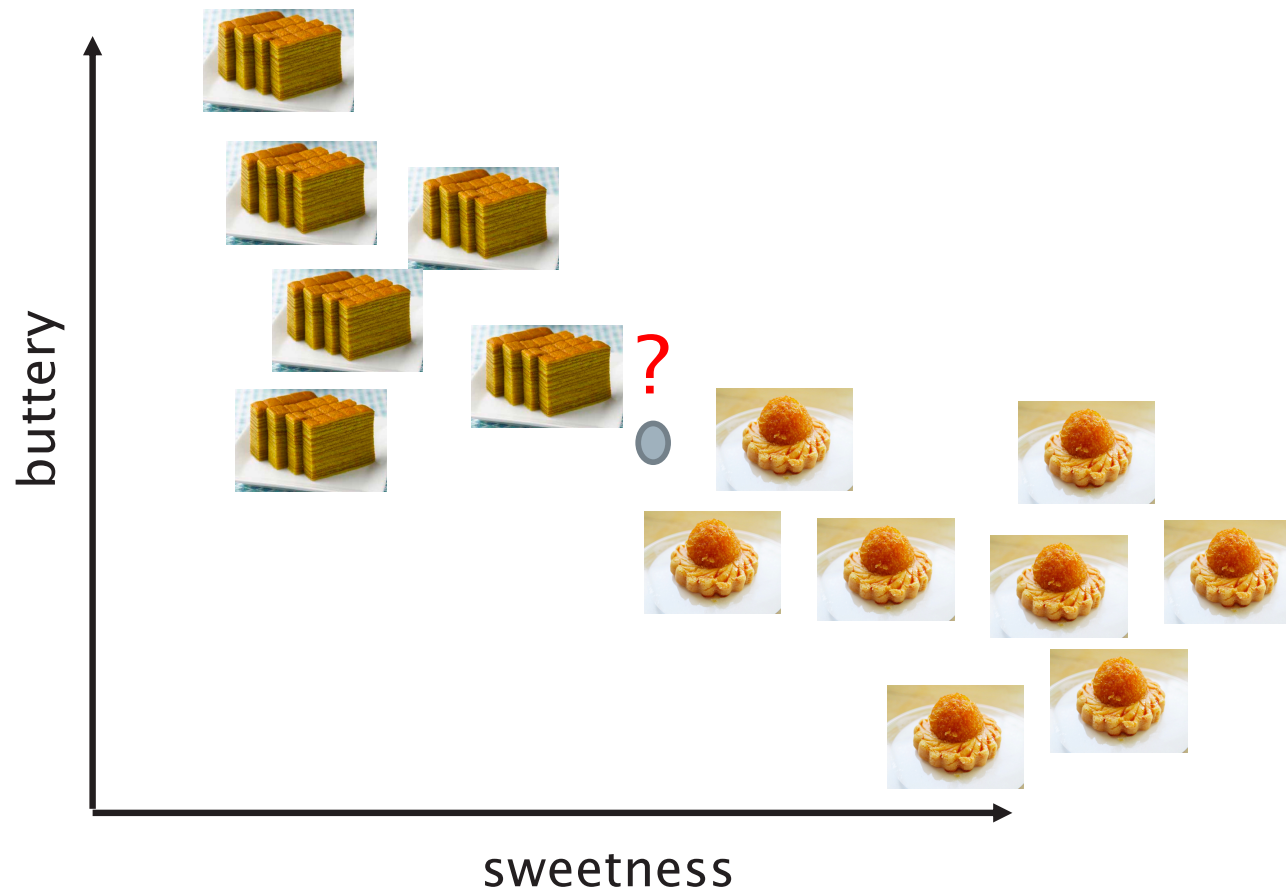
Nearest Neighbours – Example

- Using majority voting, we will predict that the unknown goodie is likely to be a *Pineapple Tart*

Nearest Neighbours – Example

- However, for the situation shown below: suppose we have k set to 7 (k = 7 nearest neighbors)

  – 3 neighbors are *Pineapple Tarts* and 4 neighbors are *Chocolate Cakes*

Credit: Zhaoxia Wang

Nearest Neighbours – Example

- In such situation, using majority voting, we will predict that the unknown goodie is likely to be a *Chocolate Cake? Yes?*

How to select a good k?

Credit: Zhaoxia Wang
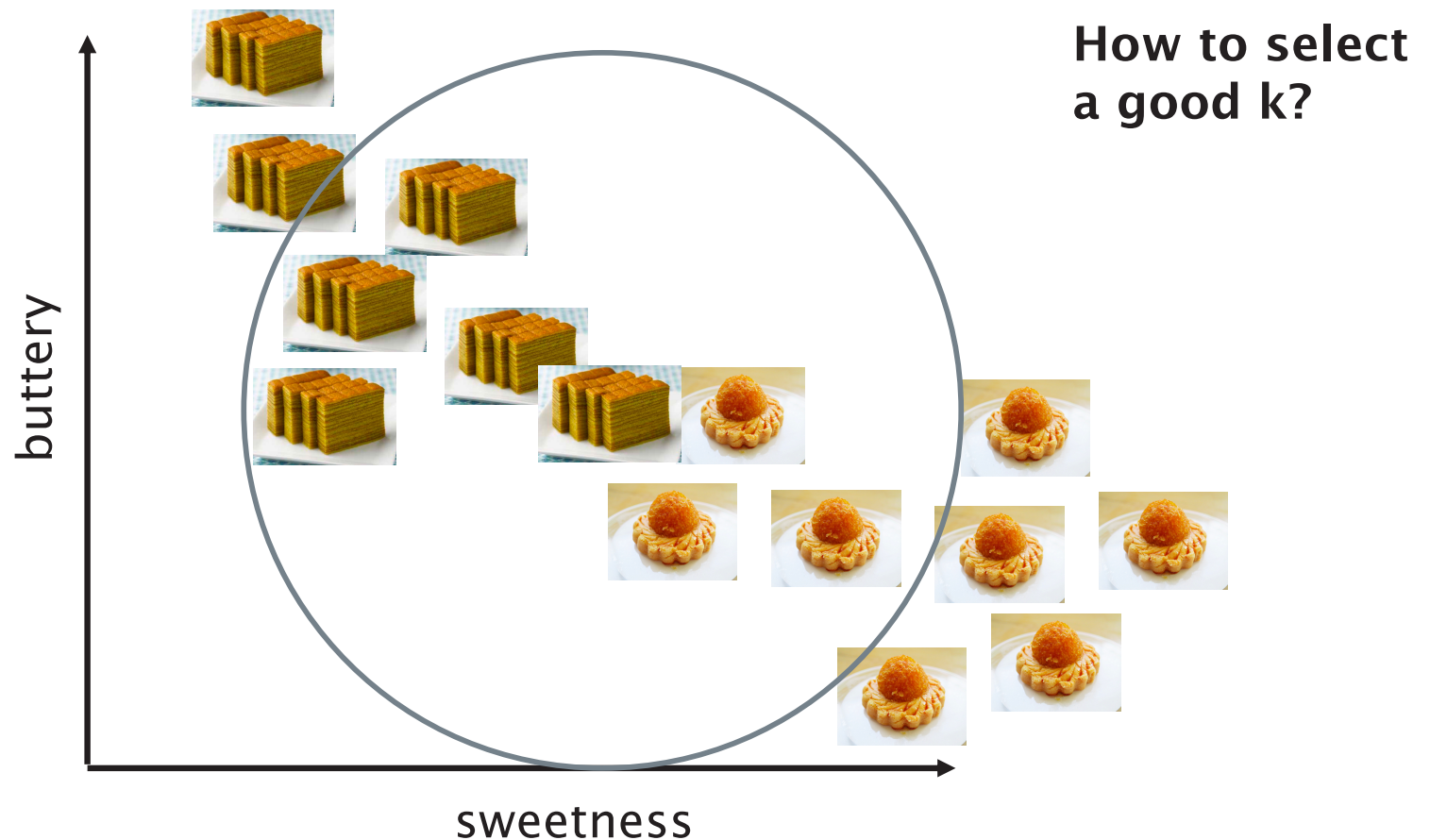
Nearest Neighbours – Weighted KNN

Up to now, each value in the k nearest neighbours has been treated equally.

Better: if closer neighbours are more important

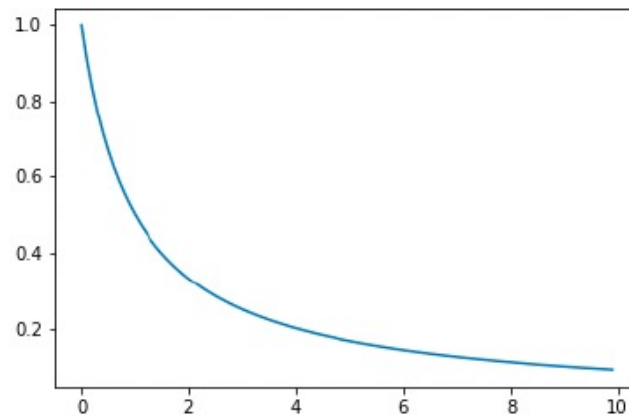We can use a range of weighting schemes to do this:

- ► Inverse Weighting
- ► Subtraction weighting
- ► Gaussian Weighting

Credit: Jo Grundy

Nearest Neighbours – Weighted KNN

Inverse Weighting:

$$w = \frac{1}{dist + c}$$

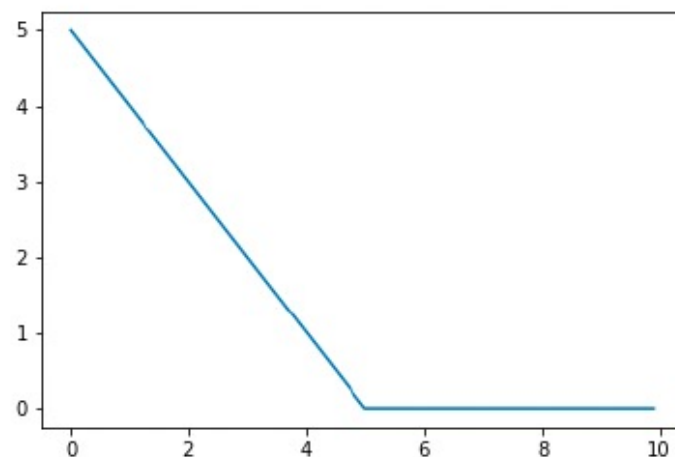Where c is a constant, avoiding division by zero error if $dist = 0$

Credit: Jo Grundy

Nearest Neighbours – Weighted KNN

Subtraction Weighting:
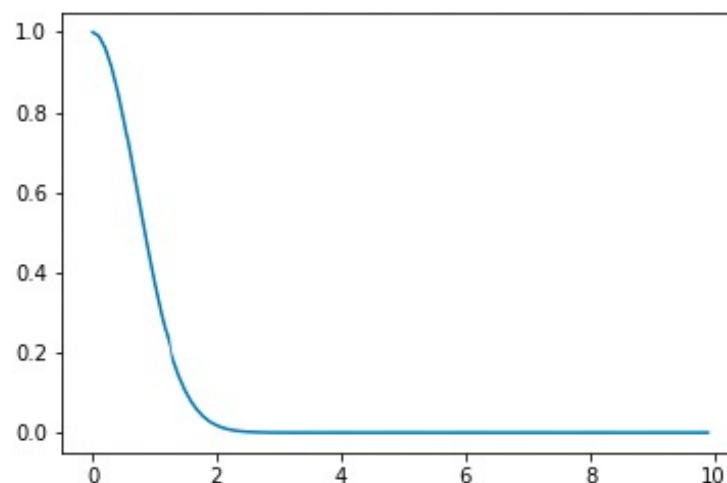
$$w = \max(0, c - dist)$$

Where c is a constant

Credit: Jo Grundy

Nearest Neighbours – Weighted KNN

Gaussian Weighting:

$$w = \exp \frac{-dist^2}{c^2}$$

Where c is a constant

Credit: Jo Grundy

Nearest Neighbours – KNN

Advantages?

- ► No assumptions made
- ► No training phase
- ► Simple and easy to implement

Problems?

- ► Doesn't scale well with lots of data
- ► Doesn't scale well with many dimensions

Credit: Jo Grundy

Nearest Neighbours – KNN

Again, to chose the best weighting scheme, measure performance using cross validation.

Problems?

▶ Heterogenous Data - features with larger ranges have greater effects

▶ Outliers affect data a good deal, especially for low k

▶ For larger k, less common classes can get ignored

▶ Distance metric determines similarity - usually Euclidean, works badly in high D

▶ Can use Hamming distance for categorical attributes

▶ Irrelevant data can force otherwise similar data samples to be far apart

▶ Computationally expensive if there are lots of data, or highly dimensional data

Credit: Jo Grundy
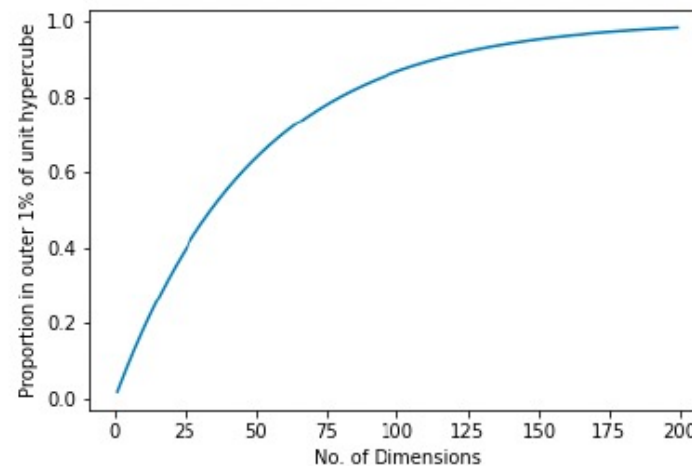
Nearest Neighbours – KNN

Curse of dimensionality:

For low dimensions, the number of points on the edge is very low

E.g. for a line, the outer 1% of a line is 2% of the line (values at $x > 0.99$, and $x < 0.1$)

For a square, the outer 1% is $1 - 0.98^2 = 0.0396 \approx 4\%$

For a cube, the outer 1% is $1 - 0.98^3 = 0.0588 \approx 6\%$

This means in higher dimensions, data is nearly always extrapolated
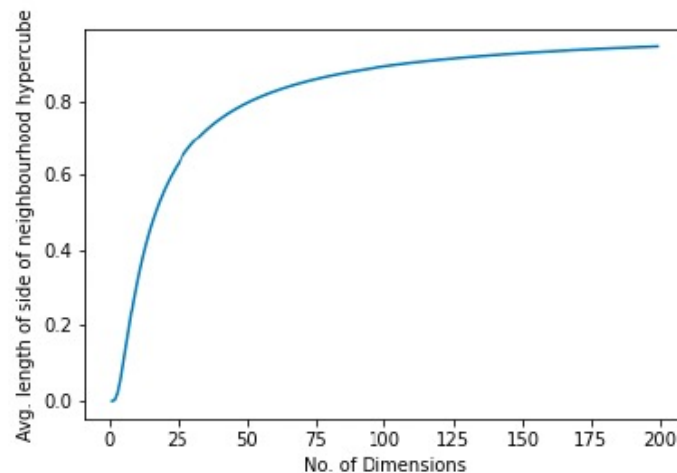
Credit: Jo Grundy

Nearest Neighbours – KNN

Curse of dimensionality; For low dimensions, the size of a neighbourhood is small.

e.g. for $k = 10$, number of points $N = 1,000,000$

In a unit line, the average neighbourhood is $\frac{10}{10^6} = 0.00001$ long

In a unit square, the average side length is $\sqrt{\frac{10}{10^6}} = 0.003$ long

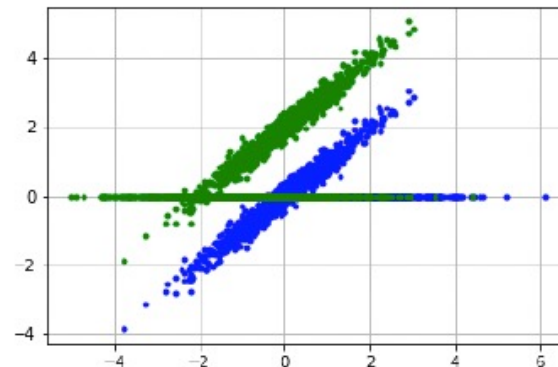In a unit cube, the average side length is $\sqrt[3]{\frac{10}{10^6}} = 0.02$ long

This can make it very difficult to work out which are closer, as the distances are nearly all the same

Credit: Jo Grundy

Nearest Neighbours – KNN

Solutions for high D?

- ▶ Dimensionality reduction.. Care! Some aren't suitable (e.g. MDS, SOM)
- ▶ Also.. PCA:

- ▶ A random direction could be better!
  Johnson Lindenstrauss lemma:
  if points in a vector space are of high enough dimensionality, they may be projected into a lower dimensional space in a way which approximately preserves the distances between the points, this basis can be generated randomly

Credit: Jo Grundy

Nearest Neighbours – KNN

Solutions: For heterogenous data?

► For heterogenous data, can *normalise*

► Better to scale factors for each feature to optimise performance

► Could use this to do feature selection
eg. if works best when scale factor $= 0$, then feature is useless!

Credit: Jo Grundy

Nearest Neighbours – KNN

More solutions for high D?

▶ Use different metric:

    ▶ Hamming distance for categorical attributes

    ▶ BM25 or TF-IDF for text data

    ▶ Minkowski distance (p-norm) - generalisation of Euclidean distance

    ▶ Kullback - Liebler Divergence for histograms

Credit: Jo Grundy

Nearest Neighbours – KNN

Solutions for lots of data?

► Need to quickly find the nearest neighbour to a particular point in a highly dimensional space

  ► Could index points in a tree structure?
  ► Could hash the points?
  ► Could break up the space
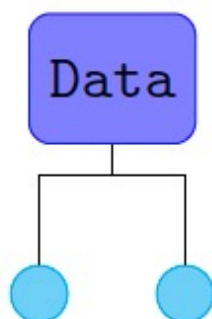
Credit: Jo Grundy

Nearest Neighbours – K-D trees

K-D trees are binary tree structures that partition the space along an axis-aligned hyperplane

- ▶ Chose random dimension
- ▶ Divide along median value
- ▶ Repeat until depth limit reached or certain number of items in each leaf

Credit: Jo Grundy

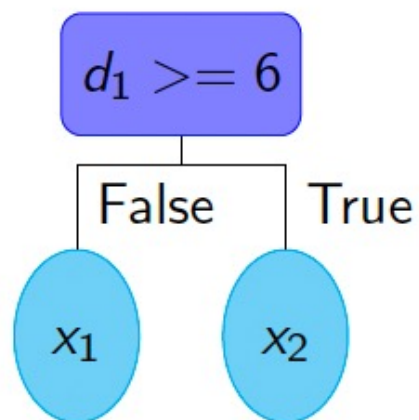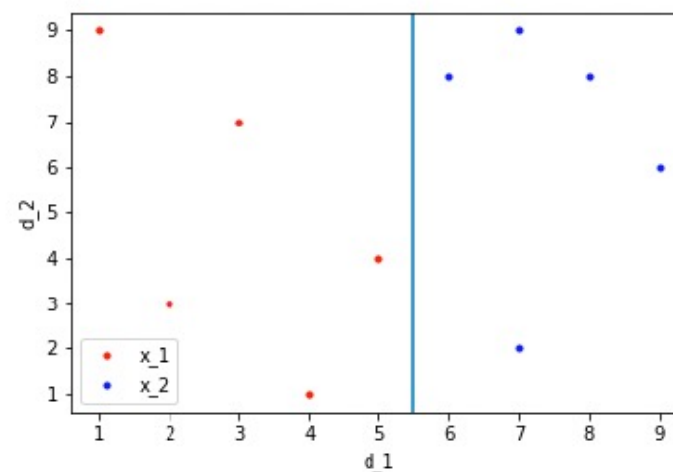# Nearest Neighbours – K-D trees

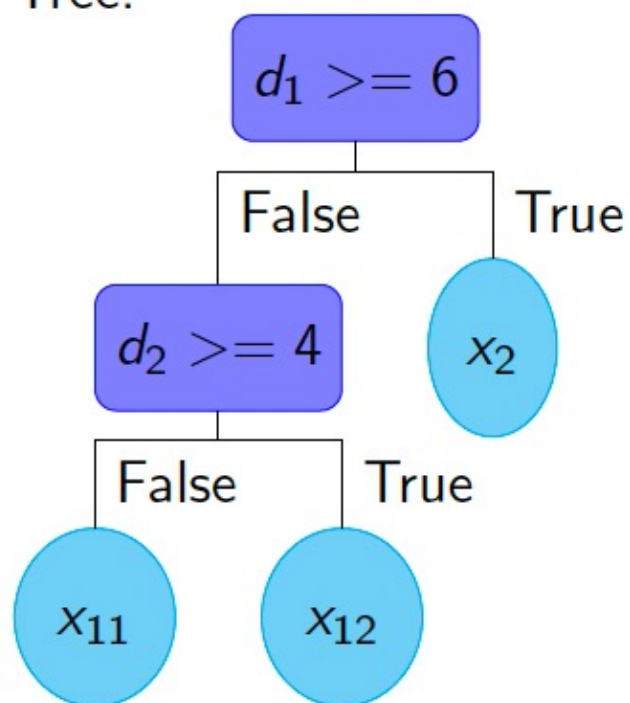For a simple dataset:

Tree:
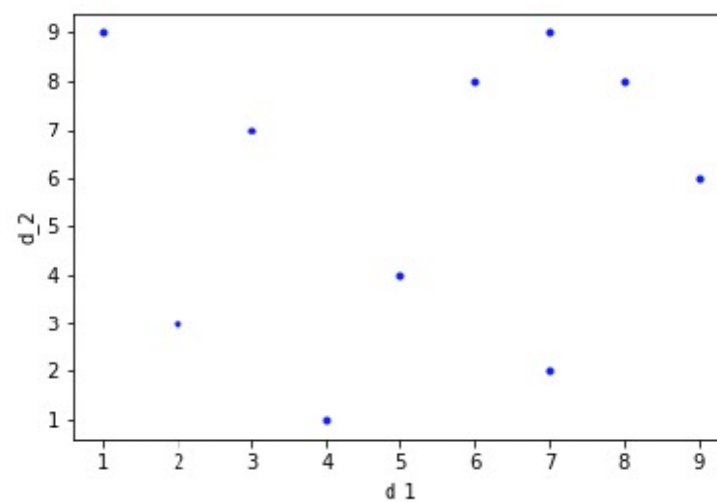
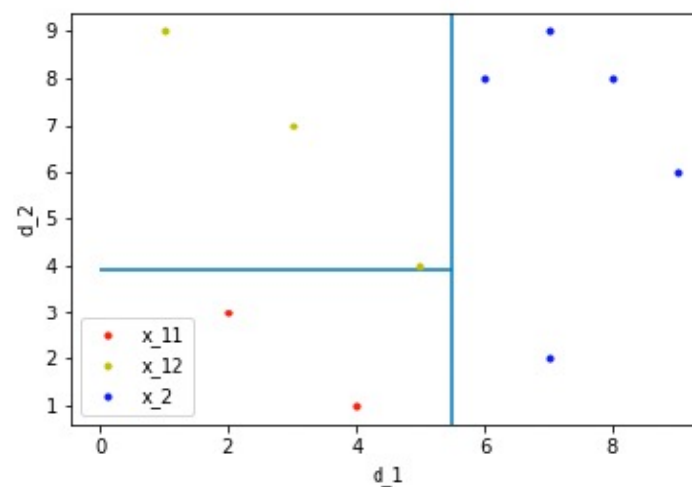# Nearest Neighbours – K-D trees

Tree:

Split:

Credit: Jo Grundy

Nearest Neighbours – K-D trees

Nearest Neighbours – K-D trees

Tree:

Split:

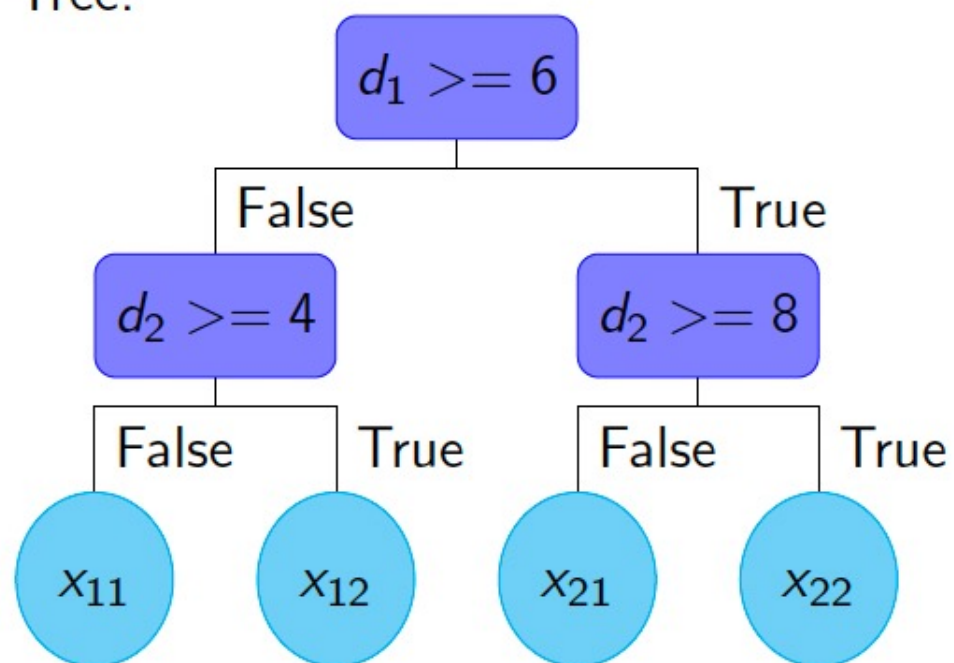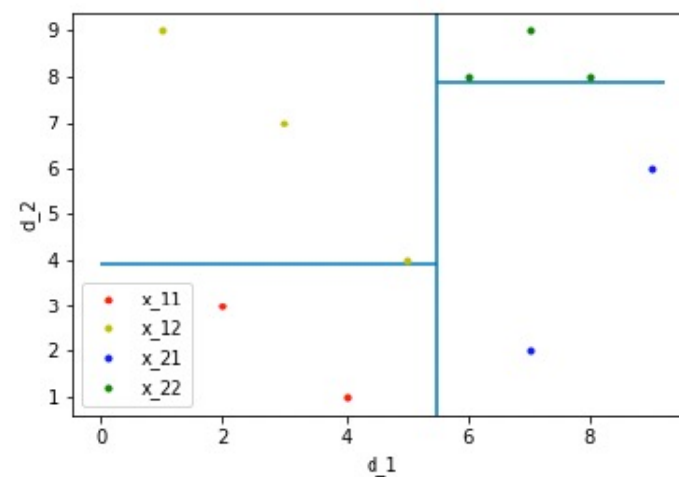Credit: Jo Grundy
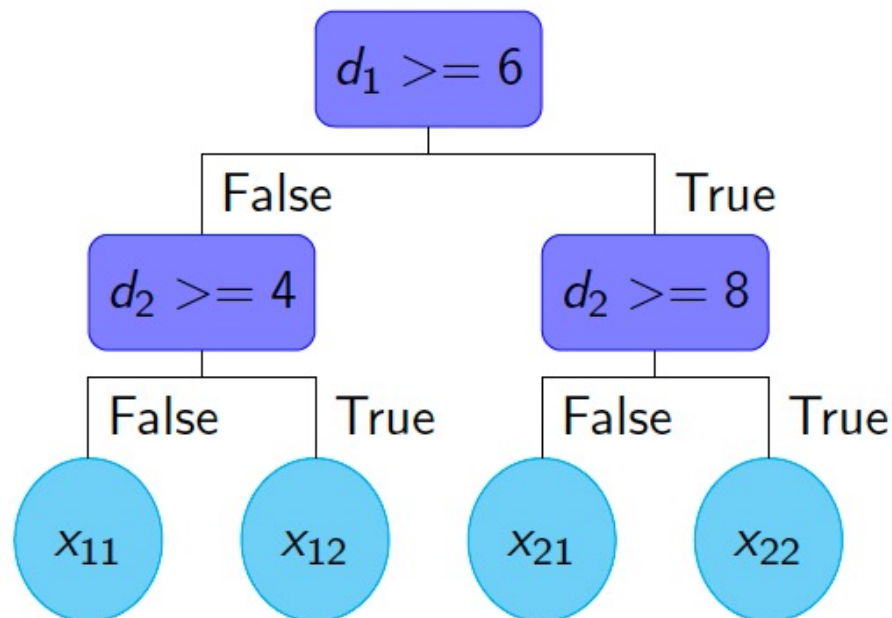
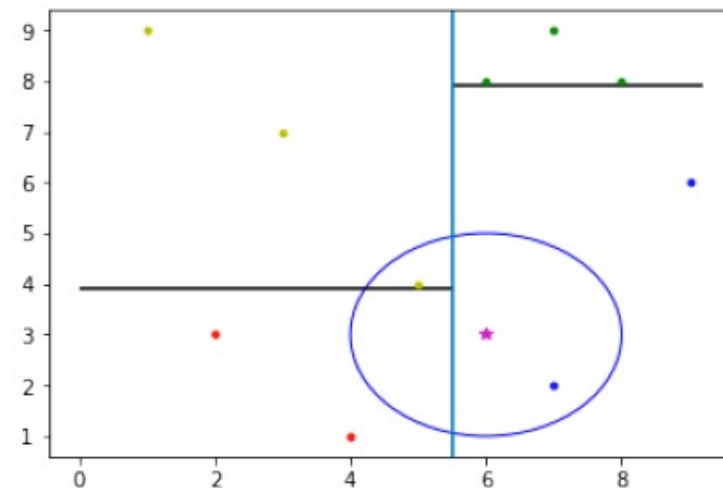# Nearest Neighbours – K-D trees

To Classify: (6, 3)

► Go to correct part of tree and search subspace
► If the border is closer than k-neighbours in the subspace:
   ► Go back up tree and search

Split:

Credit: Jo Grundy

Nearest Neighbours – K-D trees

Problems?

▶ Doesn't scale well to high dimensions

▶ Often need to search much of the tree

▶ Need *many* more examples than there are dimensions, at least 2^n

▶ There are approximate versions, not guaranteed exact answer but do scale

    ▶ Based on ensembles of trees with a randomised split dimension

Nearest Neighbours – LSH

Locality Sensitive Hashing

Makes hash codes that are similar for similar vectors

- ▶ Similar items map to the same buckets with high probability
- ▶ number of buckets much smaller than number of data samples
- ▶ Aims to maximise the probability of a collision for similar items

Credit: Jo Grundy

Nearest Neighbours – LSH

Accomplished by:

▶ Chose random hyperplanes $(h_1, h_2, \ldots, h_k)$

▶ Each hyperplane with split the space in to 2 regions

▶ ∴ the space will be sliced in to 2^k regions (buckets)

▶ Giving a simple code for each of the data points, depending on which side of each hyperplane the data points are

▶ The same code for each of the data points within the same region

▶ Compare new point only to training points in the same region

▶ Repeat with different random hyperplanes $(h_1, h_2, \ldots, h_k)$

on board

Gives low complexity , $\approx O(d \log n)$, as compare new data to only $\frac{n}{2^k}$

Credit: Jo Grundy

Nearest Neighbours – Summary

KNN can be used for regression as well as classification

- ▶ Using weighting can improve performance
- ▶ Poor performance with large data sets
- ▶ Can use K-D trees to help overcome these issues
  - ▶ Still can have issues with highly dimensional data
  - ▶ Often not much improvement in performance
- ▶ Curse of dimensionality
  - ▶ Affects neighbourhood size
  - ▶ Affects amount of extrapolation
- ▶ Can use dimensionality reduction to help (but be careful!)
- ▶ Fast approximate Nearest Neighbourhood methods - LSH

Also: *Final presentation does not need to be for full coursework, it is to show what you have done so far*