COMP6237 Data Mining

Discovering Groups
Part 1: Clustering data

Jonathon Hare
jsh2@ecs.soton.ac.uk

Introduction

- Modelling text documents as vectors 101
- Clustering
    - Hierarchical
    - K-Means
    - Mean-shift

Problem statement

- Understanding large datasets is **hard**
  - especially if the data is represented by high dimensional features

- In order to explore a dataset we might:
  - want to be able to understand which *data items* are *similar* to each other
  - want to be able to understand which *features* are *similar* to each other

Aside: Representing text documents as vectors

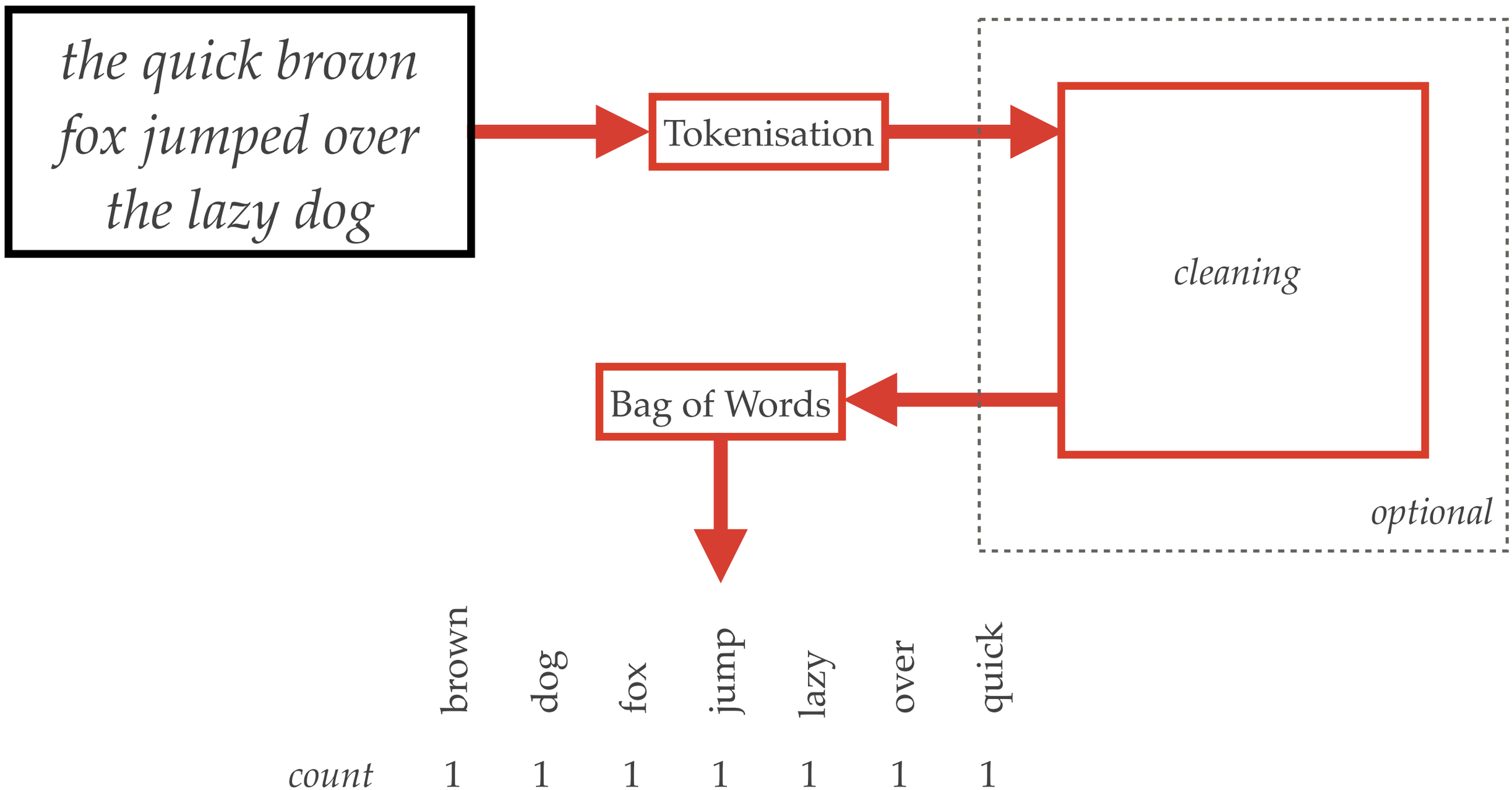
Bag of Words

A document

the quick brown fox jumped over the lazy dog

The bag of words describing the document

Text processing (feature extraction)

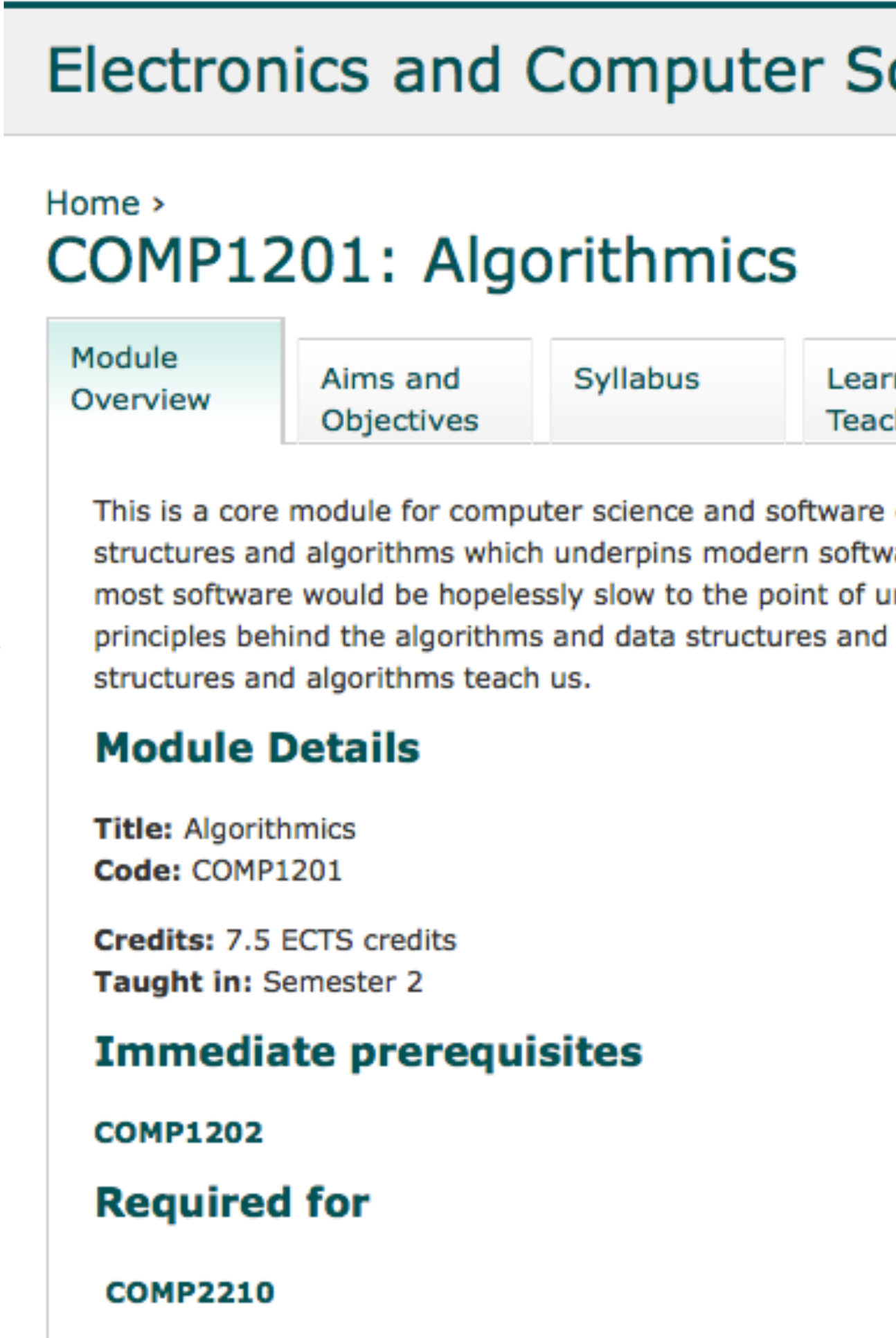


Sparsity

- The bag of words histogram representing an individual document is very sparse
    - i.e. not all words appear in all documents!
- This is subtly different to the sparse vectors we encountered when we were talking about recommender systems
    - there, vectors were sparse because we had unknown values; this time they are sparse because the value is zero
    - The tricks we applied to distance functions don’t apply here - we need to account for the zeros
    - **But** we can still modify our implementations of distance functions to be more efficient given sparse vectors

A sample data set

- The syllabus description pages from all ECS COMPXXXX modules
- Relevant sections from the overview, aims and syllabus parts
- Tokenised by splitting on
  [^A-Z^a-z]+
- Terms with a **document frequency** of <10% and >70% removed
- Words with 2 or less characters removed



Clustering

Clustering

- Clustering aims to group data without any prior knowledge of what the groups should look like or contain.
- In terms of featurevectors, items with similar vectors should be grouped together by a clustering operation.
- Some clustering operations create overlapping groups; others assign an item to a single group.

Hierarchical clustering

- Important set of techniques for understanding the structure of data
    - Creates a binary tree that recursively groups pairs of similar items or clusters
- Two approaches:
    - Agglomerative clustering (bottom up)
    - Divisive clustering (top-down)

Hierarchical Agglomerative Clustering

- Simple algorithm:
  - `Initially every item is in a cluster of its own`
  - `While there is more than one single cluster:`
    - `The closest pair of clusters according to a` **`linkage criterion`** `are merged into a bigger cluster`
- By recording the merges at each step a binary tree structure linking the clusters can be formed
  - Often a useful way of utilising this is by drawing a diagram known as a **dendrogram** that shows the structure of the tree

Linkage criterion

- A measure of dissimilarity between clusters
  - **Centroid-based**
    - Dissimilarity between clusters is equal to the distance between their centroids (for an arbitrary distance measure)
    - Requires items being clustered are represented by numeric feature vectors
- **Distance-based**
  - Dissimilarity between clusters is a function of the distances between the items in those clusters (for an arbitrary distance measure)
  - Don’t need a feature vector for each item; only need a precomputed measure of similarity between items

Centroid-based linkage

- **Weighted Centroid Clustering** (**WPGMC** - *Weighted Pair Group Method with Centroids*)
  - When two clusters *s* and *t* are combined into a new cluster *u*, the average of centroids *s* and *t* give the new centroid *u*

- **Unweighted Centroid Clustering** (**UPGMC** - *Unweighted Pair Group Method with Centroids*)
  - When two clusters *s* and *t* are combined into a new cluster *u*, the average of the positions of all the items within *s* and *t* give the new centroid *u*

Distance-based linkage

- Common linkage criteria between two sets of items *A* and *B*:
  - **Minimum** or **single-linkage clustering**: min{ *d*(*a*,*b*) : *a* ∈ *A*, *b* ∈ *B* }
    - tends to produce long, thin, clusters
  - **Maximum** or **complete-linkage clustering**: max{ *d*(*a*,*b*) : *a* ∈ *A*, *b* ∈ *B* }
    - Avoids problems of single-linkage clustering; tends to find compact clusters of approximately equal diameter
- **Mean** or **average linkage clustering** (**UPGMA** - *Unweighted Pairwise Group Method with Arithmetic Mean*):

$$\frac{1}{|A||B|}\sum_{a\in A}\sum_{b\in B} d(a,b)$$

Demo showing 2d hierarchical agglomerative clustering

Demo showing hierarchical agglomerative clustering on real data

Aside: Hierarchical Divisive Clustering

- Divisive clustering algorithms do exist
  - e.g. DIANA (DIvisive ANAlysis)
- Not widely used in practice
- Computationally much harder problem to compute divisive clustering compared to agglomerative

K-Means Clustering

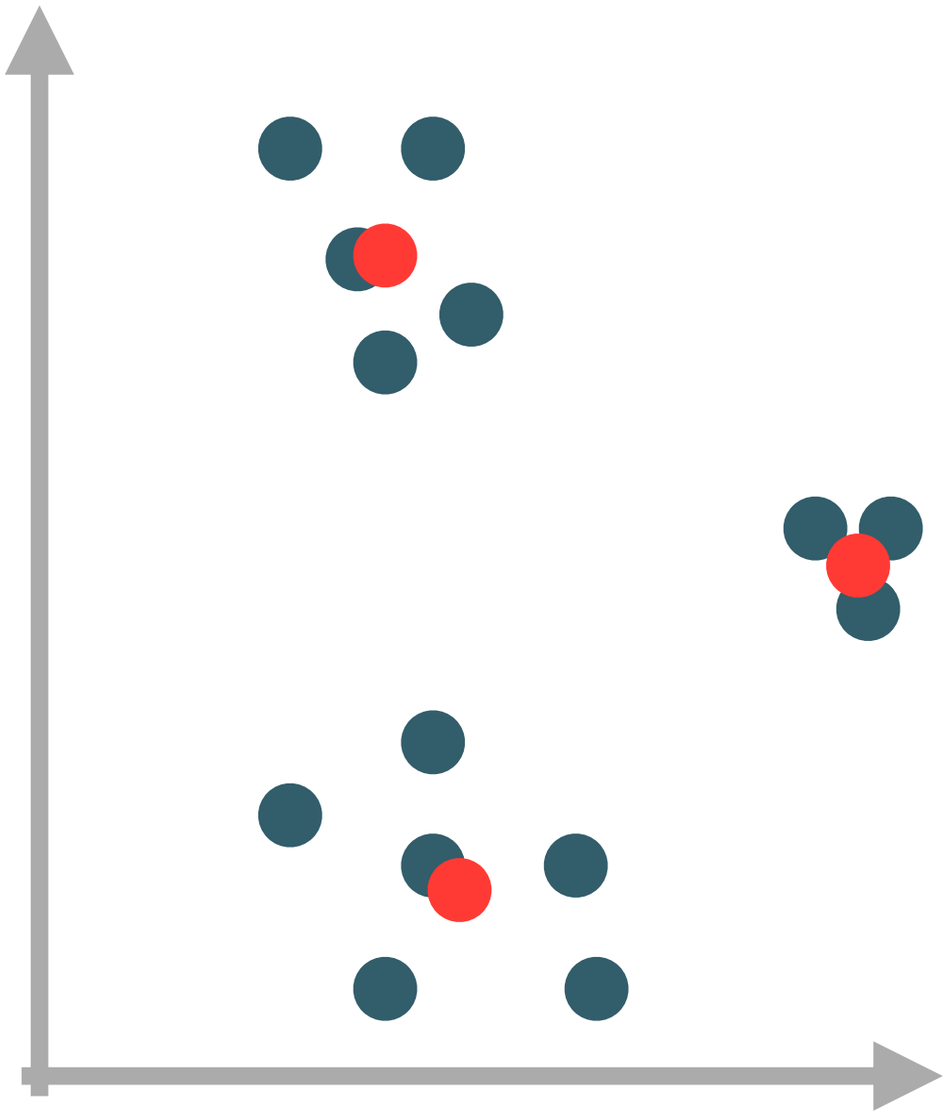
- K-Means is a classic featurespace clustering algorithm for grouping data into K groups with each group represented by a *centroid*:
  - The value of K is chosen
  - K initial cluster centres are chosen
  - Then the following process is performed iteratively until the centroids don’t move between iterations:
    - Each point is assigned to its closest centroid
    - The centroid is recomputed as the mean of all the points assigned to it. If the centroid has no points assigned it is randomly re-initialised to a new point.
  - The final clusters are created by assigning all points to their nearest centroid.

2D K-Means Demo

Demo showing K-Means of Real Data

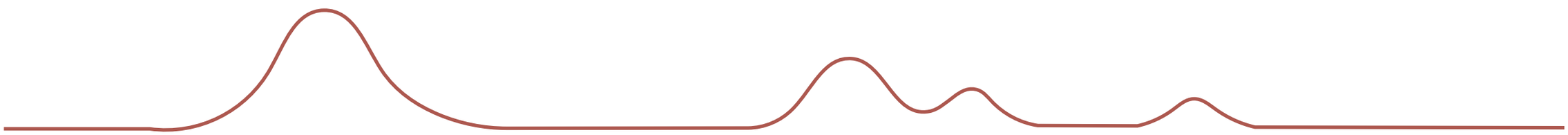
Mean shift clustering

- Mean shift is an algorithm that finds the modes of a probability density function
    - Quite literally this means it finds the points in a feature space with the highest feature density
        - Points in the feature space that are most likely given the dataset
- Doesn't need to know the number of clusters *a priori*
    - But does need a *kernel* and a *kernel bandwidth*



Probability Density Functions

PDF of the features



unbiased sample of 1D features from a dataset

PDF Estimation

- How can we estimate the PDF?
  - Histogram?

  - Crude/coarse/discrete
  - How to choose bin size?

Kernel Density Estimation (aka Parzen Window)

- Really want a technique that will give us a smooth, continuous estimate

  - use a “kernel density estimator” that can compute the value of the PDF at an arbitrary position **x**:

$$f(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K(\frac{\mathbf{x} - \mathbf{x}_i}{h})$$

featurevector of the *i*-th item

number of items

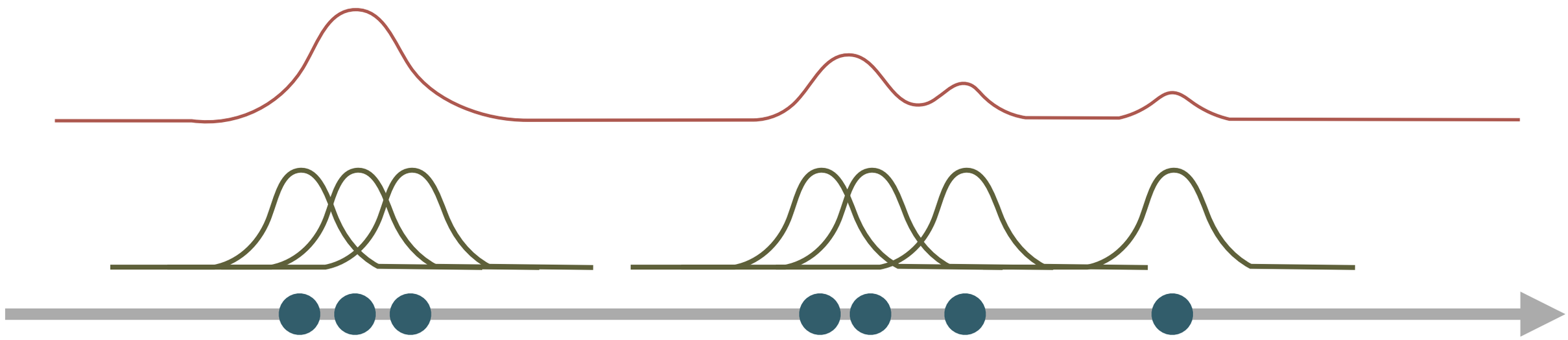
dimensionality of feature space

kernel function

kernel bandwidth

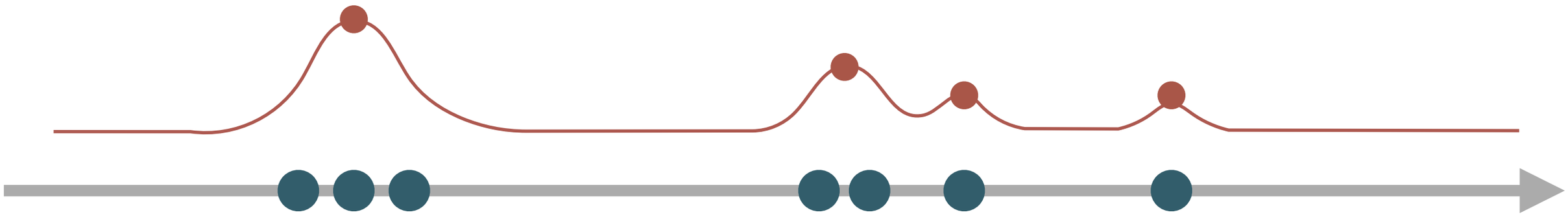
Kernel Density Estimation (aka Parzen Window)

- Commonly we'll use a Gaussian kernel with unit s.d.
  - If the kernel is radially symmetric, then only need to consider the profile of the kernel, $k(\mathbf{x})$ that satisfies $K(\mathbf{x})=c_{k,d}k(||\mathbf{x}||^2)$



The Mean Shift Procedure

- Mean shift finds the modes of the PDF
  - points where gradient is zero: $\nabla f(\mathbf{x})=0$



Assume radially symmetric kernel:

$g(\bullet) = -f'(\bullet)$

$$\nabla f(\mathbf{x}) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (\mathbf{x} - \mathbf{x}_i) g\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right)$$

$$= \boxed{\frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right) \right]} \boxed{\left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} g\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right)} - \mathbf{x} \right]}$$

This is proportional to a density estimate with kernel $G(\mathbf{x}) = c_{g,d} g(\|\mathbf{x}\|^2)$

This is the **mean shift** …it's a vector that always points in the direction of maximum density

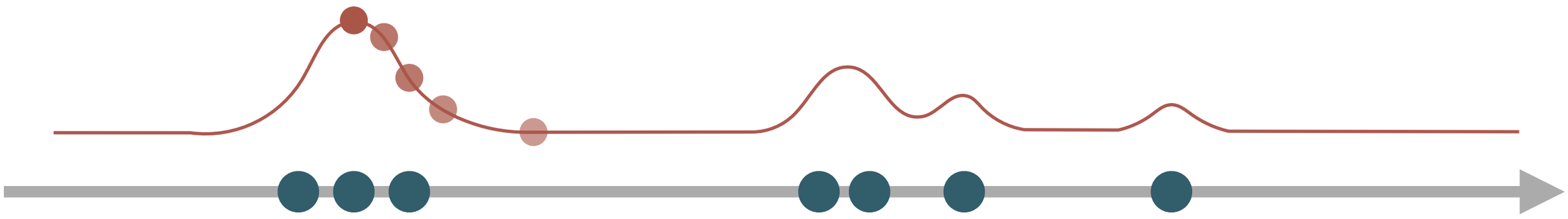
Mean shift procedure:

1. Start at a point $\mathbf{x}_t$

2. Compute the mean shift vector $\mathbf{m}_h(\mathbf{x}_t)$

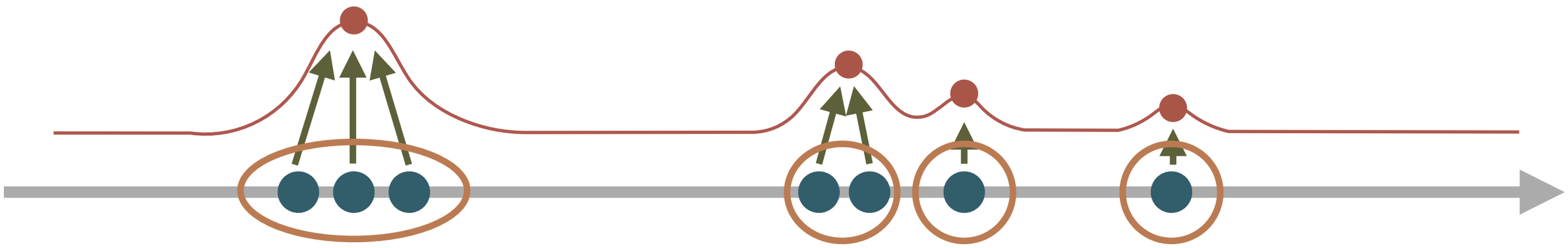
3. Move x towards the mode: $\mathbf{x}_{t+1} = \mathbf{m}_h(\mathbf{x}_t)$

4. Stop if $\mathbf{x}_{t+1} = \mathbf{x}_t$; otherwise goto step 2



Mean Shift Clustering

- Simple extension of the mean shift procedure:
- for each feature vector
  - apply the mean shift procedure until convergence and store the resultant mode
- the set of featurevectors that converge to the same mode define the basin of attraction of that mode; all features that converged to the same mode belong to the same cluster



Demo showing Mean-Shift in 2D

Summary

- Clustering is of key importance to understanding data
- Lots of different approaches - we've barely scratched the surface
- We have however looked at the clustering techniques you're most likely to start exploring a dataset with
    - …which leads on to the individual coursework…