COMP4650/6490 Document Analysis

Clustering

ANU School of Computing

Administrative Matters

- Assignment 2
  - Due: 5pm Tuesday 19 September
- First feedback survey
  - Closes: 5pm Friday 1 September

Recap

So far, we have built document classifiers given labeled data, which is an instance of supervised learning

Contents

- Clustering overview
- K-Means
- Evaluating clustering

Unsupervised Learning

What if we don't have labeled data?

Then we are doing unsupervised learning

Can we still assign reasonable labels?

- We could try to cluster documents into groups (cluster = class)
- Clustering is a type of unsupervised learning

Other types of unsupervised learning (not covered in this course):

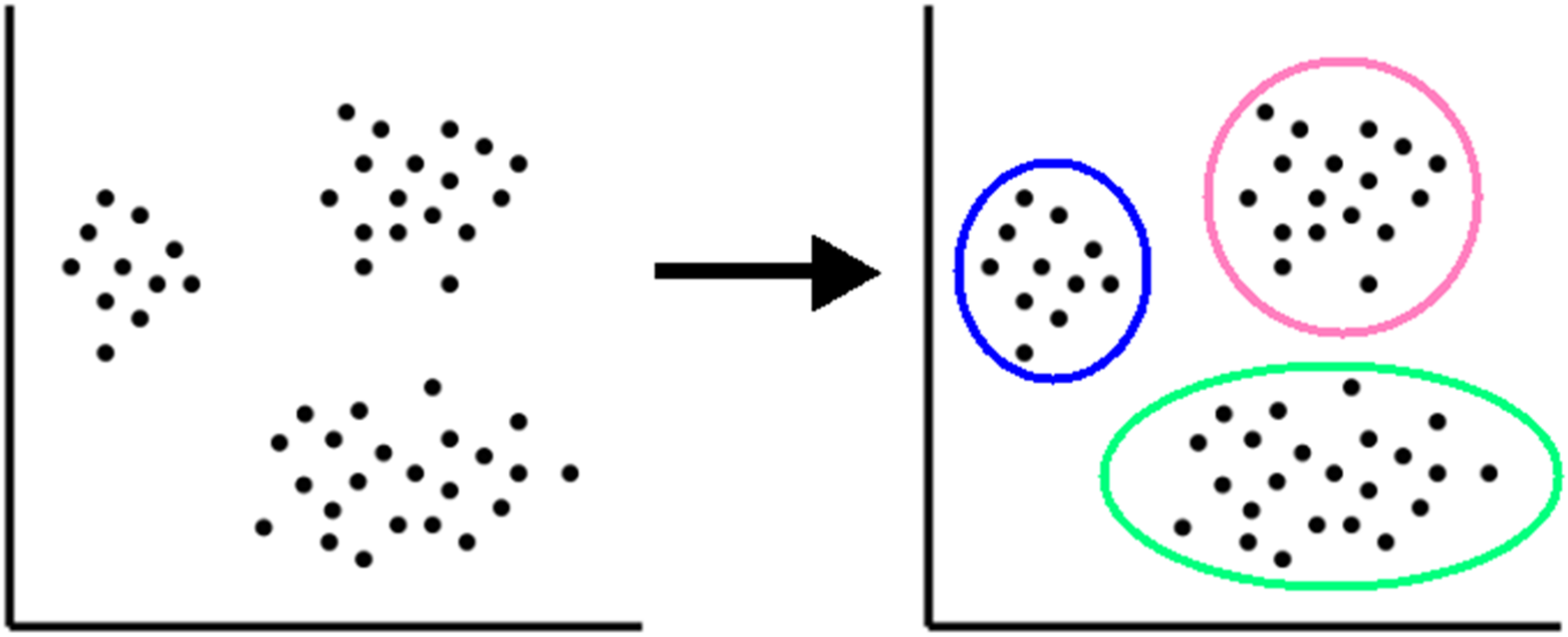
- Auto-encoders, Rule Mining, PCA

Clustering: Definition

Document clustering is the process of grouping a set of documents into clusters of similar documents.

- Documents within a cluster should be *similar*.
- Documents from different clusters should be *dissimilar*.
- Clustering is a very common form of unsupervised learning.

Data set with clear cluster structure



Finding the cluster structure in this example.

Classification vs. Clustering

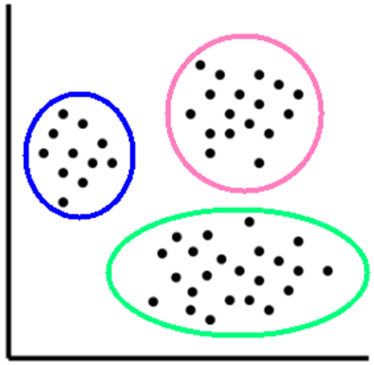
- Classification: supervised learning
- Clustering: unsupervised learning
- Classification: Classes are human-defined and part of the input to the learning algorithm.
- Clustering: Clusters are inferred from the data without human input.
  - There are many ways of influencing the outcome of clustering: number of clusters, similarity measure, representation of documents, other algorithm hyper-parameters

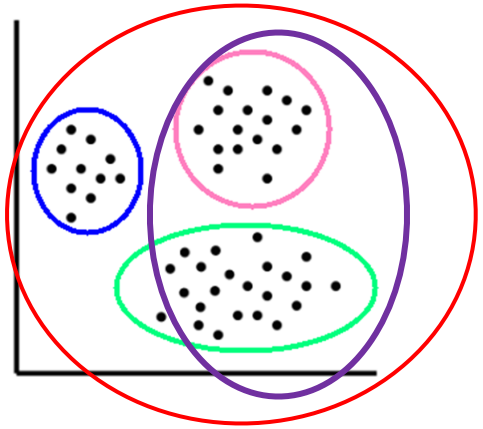
Desiderata for clustering

- General goal: put related docs in the same cluster, put unrelated docs in different clusters.
  - How do we formalise this?
- The number of clusters should be appropriate for the data set we are clustering.
  - Initially, we will assume the number of clusters *K* is given.
  - There are (semi-)automatic methods for determining *K*
- Secondary goals in clustering (not always necessary)
  - Avoid very small and very large clusters
  - Defined clusters that are easy to explain to the user

Flat vs. Hierarchical clustering

- **Flat algorithms**
  - Usually start with a random (partial) partitioning of docs into groups
  - Refine iteratively
  - Common algorithm: *K*-means
- **Hierarchical algorithms**
  - Create a hierarchy
    - Cluster the clusters
  - Bottom-up, agglomerative
  - Top-down, divisive





Hard vs. Soft clustering

- Hard clustering: Each document in exactly one cluster.
  - More common and generally easier to do
  - Example algorithm: K-means
- Soft clustering: A document is assigned a membership probability for each cluster.
  - A higher score means the document is more likely to belong to that cluster or exhibit characteristics of that cluster
  - Documents belong to multiple clusters
  - Example algorithm: Gaussian Mixture Models (via EM)

We will only consider flat, hard clustering in this course

Flat clustering algorithms

Flat algorithms compute a partition of *N* documents into a set of *K* clusters.

- Given: a set of documents and the number *K*
- Find: a partition into *K* clusters that optimises the chosen partitioning criterion
- Global optimisation: exhaustive search (enumerate all partitions, pick the optimal one)

- Effective heuristic method: K-Means algorithm

Document representation for clustering

- Represent documents as vectors
  - Sparse count vectors, LSA, aggregated word2vec, pre-trained language model (e.g. BERT)
- Measure similarity between vectors by Euclidean distance which is *almost* equivalent to cosine distance
  - cosine distance = 1 - cosine similarity
  - cosine distance is length-normalised
- Some clustering algorithms (e.g. Affinity propagation) only require similarities between documents (e.g. edit distance)

Contents

- Clustering overview
- K-Means
- Evaluating clustering

Centroids in K-Means

- Each cluster in K-Means is defined by a centroid
- Definition of centroid:

$$\vec{\mu}(\omega) = \frac{1}{|\omega|} \sum_{\vec{x} \in \omega} \vec{x}$$

where we use ω to denote a cluster

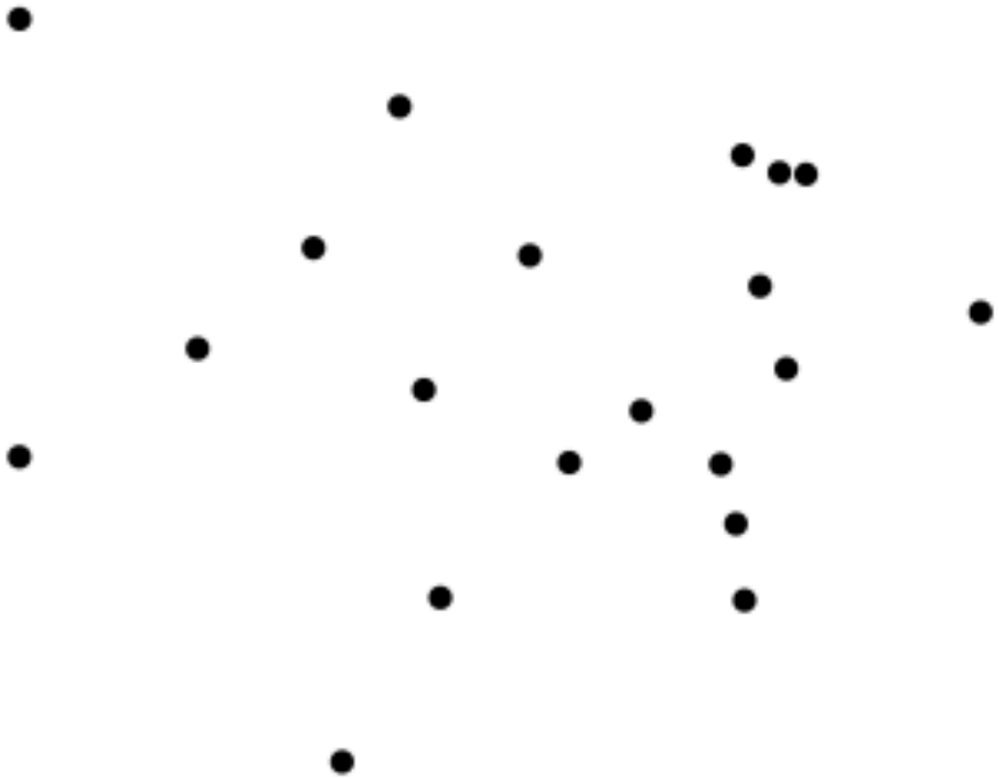
- Centroids act as centres of gravity, abstractions of the class.

K-Means

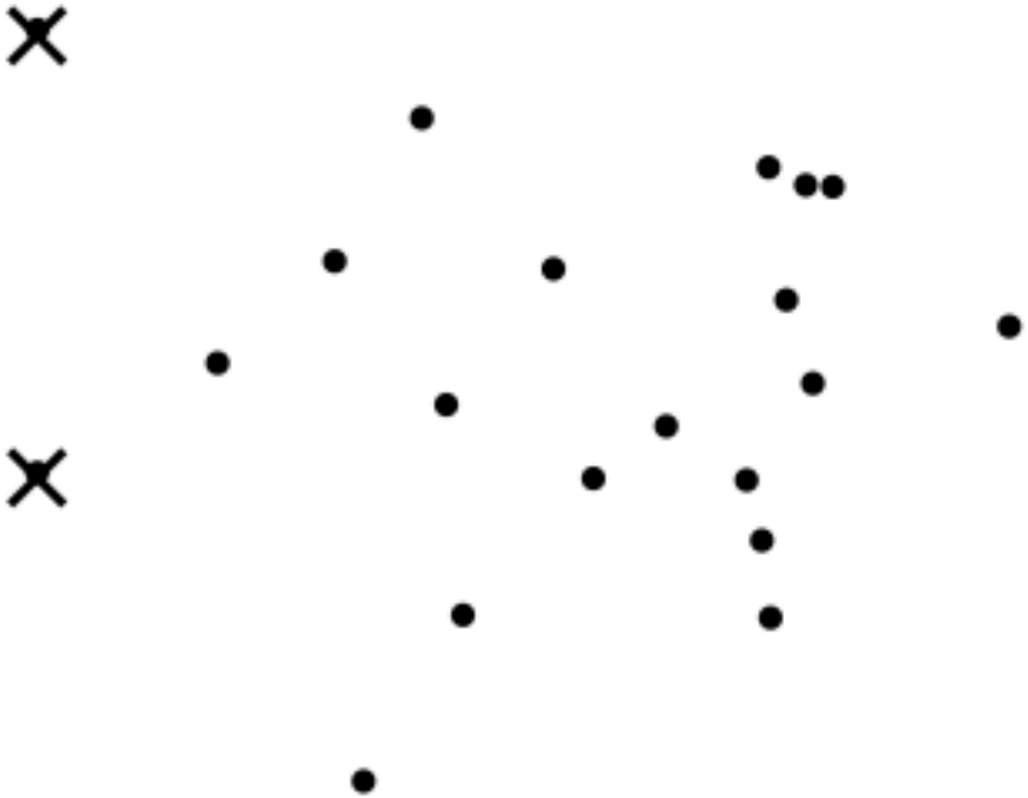
- Objective / partitioning criterion: minimise the average squared difference from the centroid

- We try to find the minimum average squared difference by iterating two steps (initial centroids are randomly selected):
  - Re-assignment: assign each vector to its *closest* centroid
  - Re-computation: recompute each centroid as the *average* of the vectors that were assigned to it

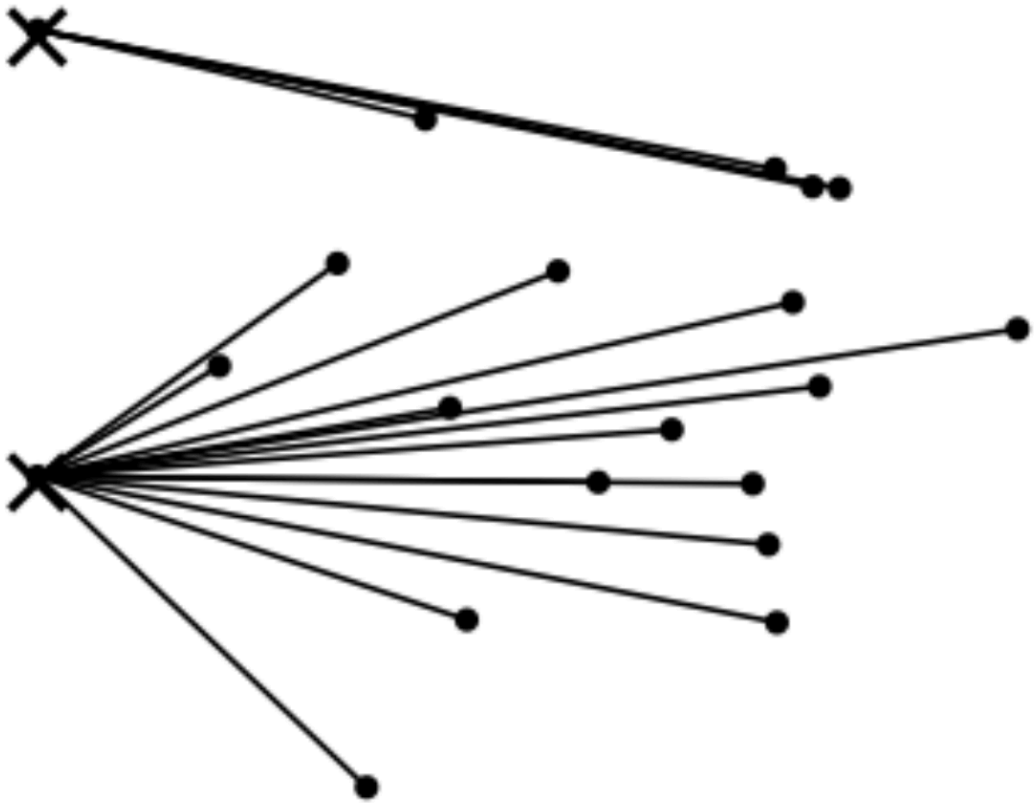
Working Example:
Data points to be clustered



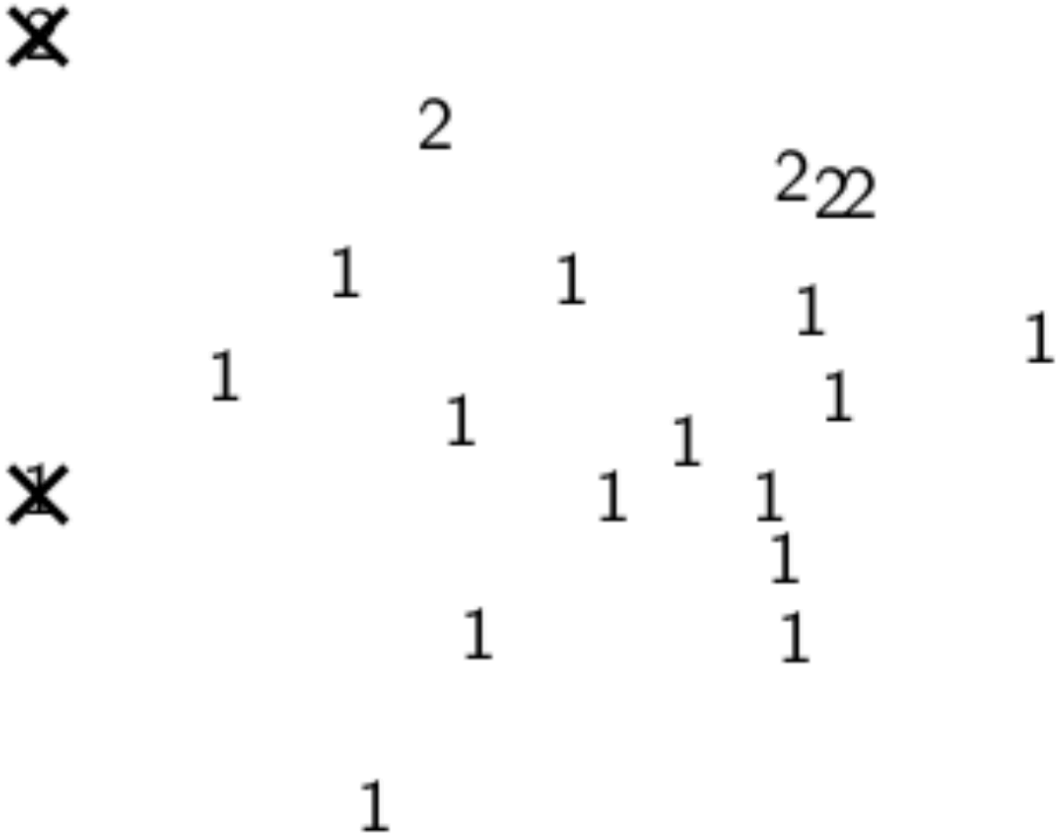
Working Example:
Random selection of initial centroids



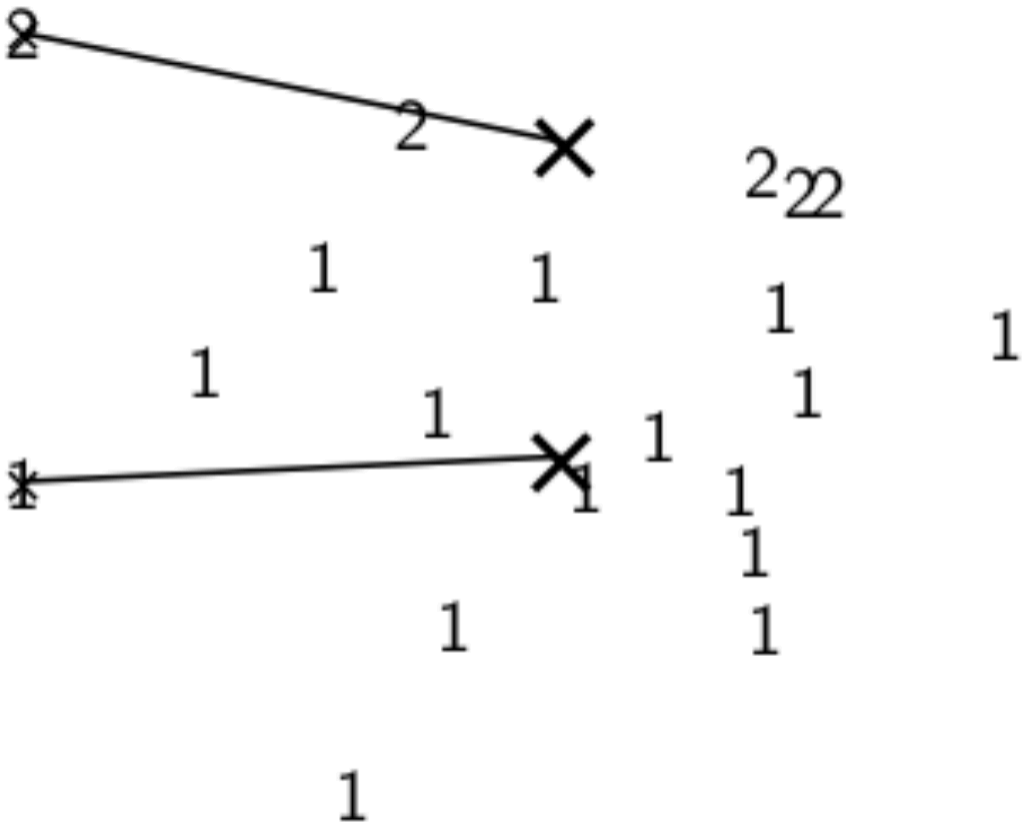
Working Example:
Assign data points to closest centroid



Working Example:
Assign data points to closest centroid

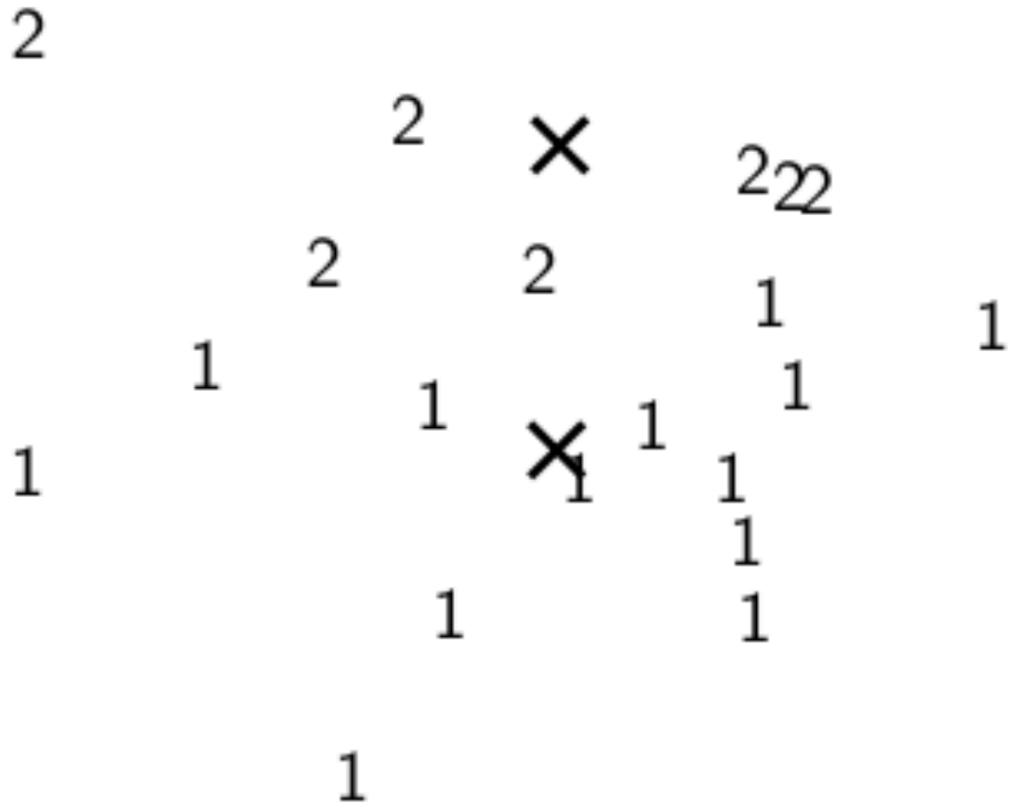


Working Example:
Re-compute cluster centroids

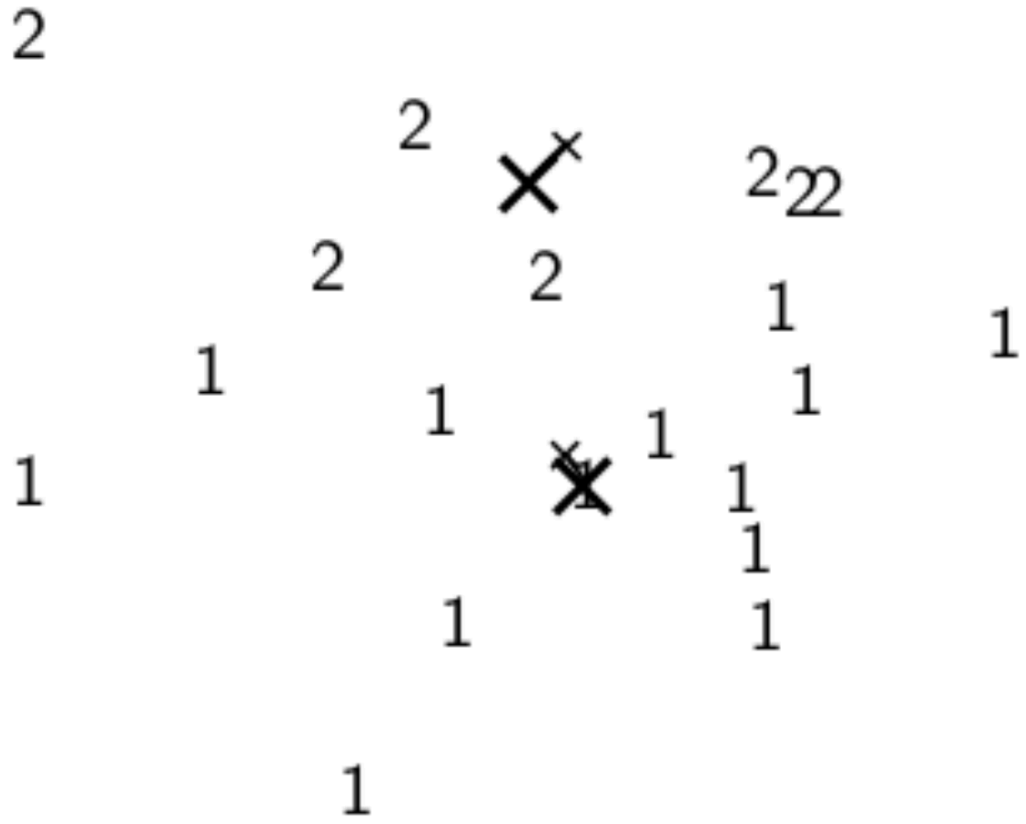


Working Example:
Assign data points to closest centroid

Working Example:
Assign data points to closest centroid



Working Example:
Re-compute cluster centroids



K-Means algorithm

K-MEANS$(\{\vec{x}_1, \ldots, \vec{x}_N\}, K)$

1 $(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_K) \leftarrow$ SELECTRANDOMSEEDS$(\{\vec{x}_1, \ldots, \vec{x}_N\}, K)$
2 **for** $k \leftarrow 1$ **to** K
3 **do** $\vec{\mu}_k \leftarrow \vec{s}_k$
4 **while** stopping criterion has not been met
5 **do for** $k \leftarrow 1$ **to** K
6 **do** $\omega_k \leftarrow \{\}$
7 **for** $n \leftarrow 1$ **to** N
8 **do** $j \leftarrow \arg\min_{j'} |\vec{\mu}_{j'} - \vec{x}_n|$
9 $\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$ *(reassignment of vectors)*
10 **for** $k \leftarrow 1$ **to** K
11 **do** $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$ *(recomputation of centroids)*
12 **return** $\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$

K-Means is guaranteed to converge

- RSS (residual sum of squares) = sum of all squared distances between document vector and the closest centroid
- RSS decreases during each reassignment step.
  - because each point is assigned to a closer centroid
- RSS decreases during each re-computation step.
  - see next slide
- There are only a finite number of clusterings (cluster assignments)
- Thus: We must reach a fixed point.
- *Question*: Does this conclusion still hold if using cosine distance (assuming all data points are normalised vectors)?

Re-computation decreases avg. distance

$$\mathrm{RSS}_k(\vec{v}) = \sum_{\vec{x}\in\omega_k} \|\vec{v}-\vec{x}\|^2 = \sum_{\vec{x}\in\omega_k} \sum_{m=1}^{M} (v_m - x_m)^2$$

$$\frac{\partial \mathrm{RSS}_k(\vec{v})}{\partial v_m} = \sum_{\vec{x}\in\omega_k} 2(v_m - x_m) = 0$$

$$v_m = \frac{1}{|\omega_k|} \sum_{\vec{x}\in\omega_k} x_m$$

i.e. choosing v_m that minimises the objective

Choosing the mean as the centroid minimises the average distance, i.e. during the re-computation step the distance always decreases or stays the same.

Question: What if we use cosine distance instead (for normalised vectors)? (Hint: $1-\cos(\alpha) = \frac{1}{2}\|\mathbf{a}-\mathbf{b}\|^2$ if $\|\mathbf{a}\| = \|\mathbf{b}\| = 1$, and $\|\frac{1}{|\omega|}\sum_{\mathbf{x}\in\omega}\mathbf{x}\| \leq 1$ if $\mathbf{x}$ is normalised.)

K-Means is guaranteed to converge

- But we don't know how long convergence will take!
- If we don't care about a few docs switching back and forth, then convergence is usually fast (less than10-20 iterations).
- However, complete convergence can take many more iterations.

Optimality of K-Means

- Convergence does not mean that we converge to the optimal clustering!
- This is the great weakness of K-Means
- If we start with a bad set of seeds (e.g. outliners), the resulting clustering can be terrible

Initialisation of K-Means

- Random seed selection is just one of many ways K-Means can be initialised.
- Random seed selection is not very robust: It's easy to get a suboptimal clustering.
- Better ways of computing initial centroids:
  - Select seeds using a heuristic (e.g., ignore outliers or find a set of seeds that has "good coverage" of the document space)
  - Use hierarchical clustering to find good seeds
  - Select *i* (e.g., *i* = 10) different random sets of seeds, do a K-Means clustering for each, select the clustering with lowest RSS

Contents

- Clustering overview
- K-Means
- Evaluating clustering

Evaluation

- Internal criteria
  - For example: RSS in K-Means
- But an internal criterion often does not evaluate the actual utility of a clustering to an application
- Alternative: External criteria
  - For example: Evaluate with respect to a human-defined classification

External criterion: Purity

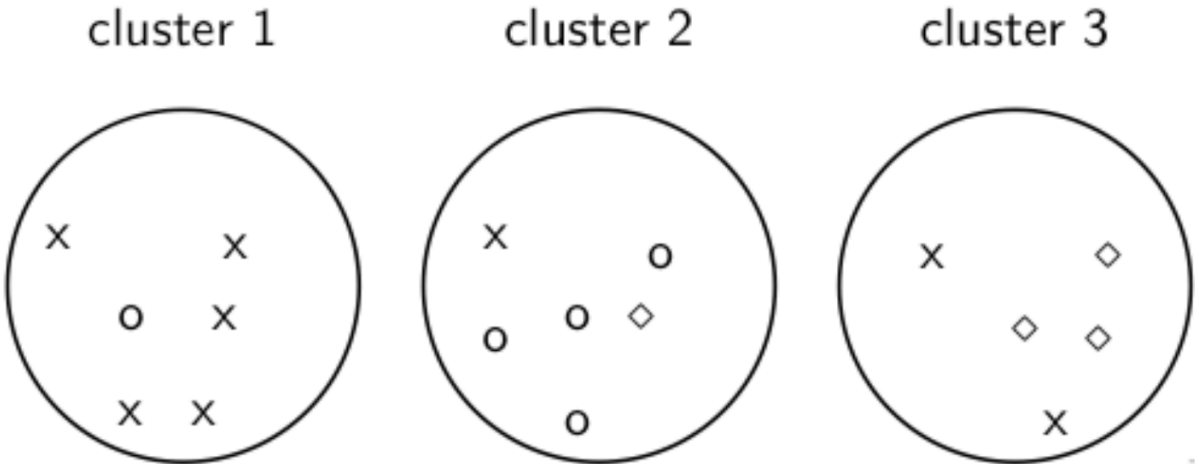
If we have a set of ground truth class labels we can use *purity*:

$$\text{purity}(\Omega, C) = \frac{1}{N} \sum_{k=1}^{K} \max_{j \in \{1,\ldots,K'\}} | \omega_k \cap c_j |$$

$\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ is the set of clusters
$\mathrm{C} = \{c_1, c_2, \ldots, c_{K'}\}$ is the set of classes
N is the number of data points

For each cluster ω_k: find class c_j with most members n_{kj} in ω_k
Sum all n_{kj} (the member counts) and divide by total number of data points

Example for computing purity



To compute purity:

$5 = \max_j |\omega_1 \cap c_j|$ (class x, cluster 1)

$4 = \max_j |\omega_2 \cap c_j|$ (class o, cluster 2)

$3 = \max_j |\omega_3 \cap c_j|$ (class ◇, cluster 3)

Purity is $(1/17) \times (5 + 4 + 3) \approx 0.71$.

How many clusters?

- Number of clusters *K* is given in many applications.
  - for example, there may be an external constraint on *K*.
    In the case of web search results, it might be optimal for the users to present more than 10 -15 topics.
- What if there is no external constraint? Is there a “right” number of clusters?
- One way: define an optimisation criterion that penalises a larger number of clusters
  - Given docs, find *K* for which the optimum is reached.
  - We can’t use RSS or average squared distance from centroid as the criterion because it would always choose *K* = *N* clusters.

References

Chapter 16, Introduction to Information Retrieval.