Applied Natural Language Processing

Dr Jeff Mitchell, University of Sussex

Autumn 2025

Document similarity and clustering

Previously

- Binary classification scenarios e.g., sentiment and relevance
- Wordlist classifiers
- Machine learning approaches
  - Automatically deriving wordlists
  - Naïve Bayes
- Evaluation
  - accuracy and error rate
  - the confusion matrix
  - precision, recall and F1-score

This time

- Document Similarity
  - Vector Space Model
  - TF-IDF
  - Cosine similarity
  - Beyond words as dimensions
- Clustering
  - k-means

Document similarity

Part 2

Document similarity scenarios

- Navigating a document collection
  - Given a large document collection (e.g., the web)
  - Find documents which are similar to ones we already know about
- Clustering a document collection
  - Given a large document collection
  - Find natural groupings of documents which seem to be about similar topics
- Searching a document collection
  - Given a large document collection
  - Find documents which are relevant to a search query

Similarity vs classification

- Related problem to topic classification but not identical

Classification

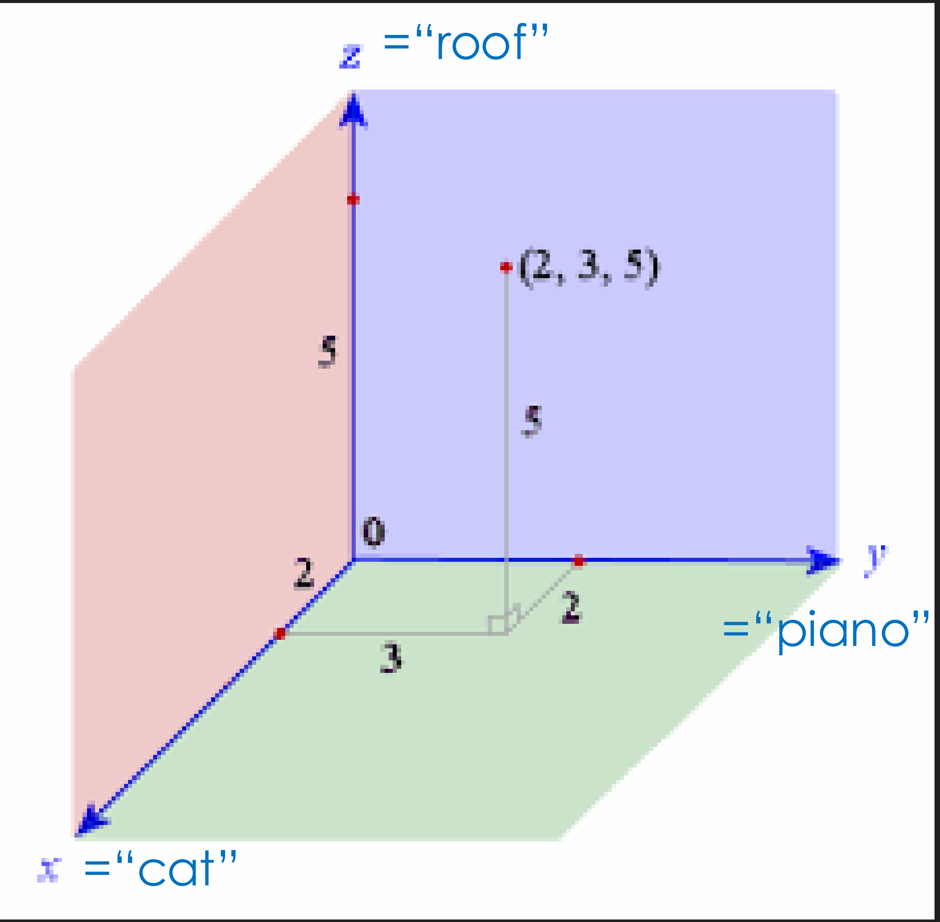
- Fixed number of classes
- Classes known in advance

Similarity

- May not know the classes or even the number of classes of interest
- Documents in different classes may actually be similar
- Similarity can be used for classification by assigning the class of the most similar document(s) in a training sample.

Vector space model for documents

- Identify a fixed set of topic terms
  - Maybe 100s of thousands of topic terms
  - Maybe all of the content words or lemmas in the vocabulary
- One dimension in space for each term
  - This is a really high-dimensional space
  - We will normally only draw 2 (or 3) but really there are 100s of thousands of dimensions
- Every document is associated with a point (or vector) in this space



Measuring value of a term

- Weight or value of a term should reflect importance in document
- Term Frequency
  - More frequent terms may be more important
  - But high frequency words are not always discriminating
    - E.g., stop-words
    - The term "*country*" in a collection of documents about travel
- Document Frequency
  - Terms which occur in less documents may be more important
  - The term "*Kenya*" in 3 documents in a collection of documents about travel

Term Frequency Inverse Document Frequency (TFIDF)

- Term frequency is the number of occurrences of a term in a document:

$$\mathrm{tf}(d, t)$$

- Document frequency is the number of documents in which a term occurs:

$$\mathrm{df}(t)$$

- If there are *N* documents in total, inverse document frequency is given by:

$$\mathrm{idf}(t) = \log\left({}^{N}\!/_{\mathrm{df}(t)}\right)$$

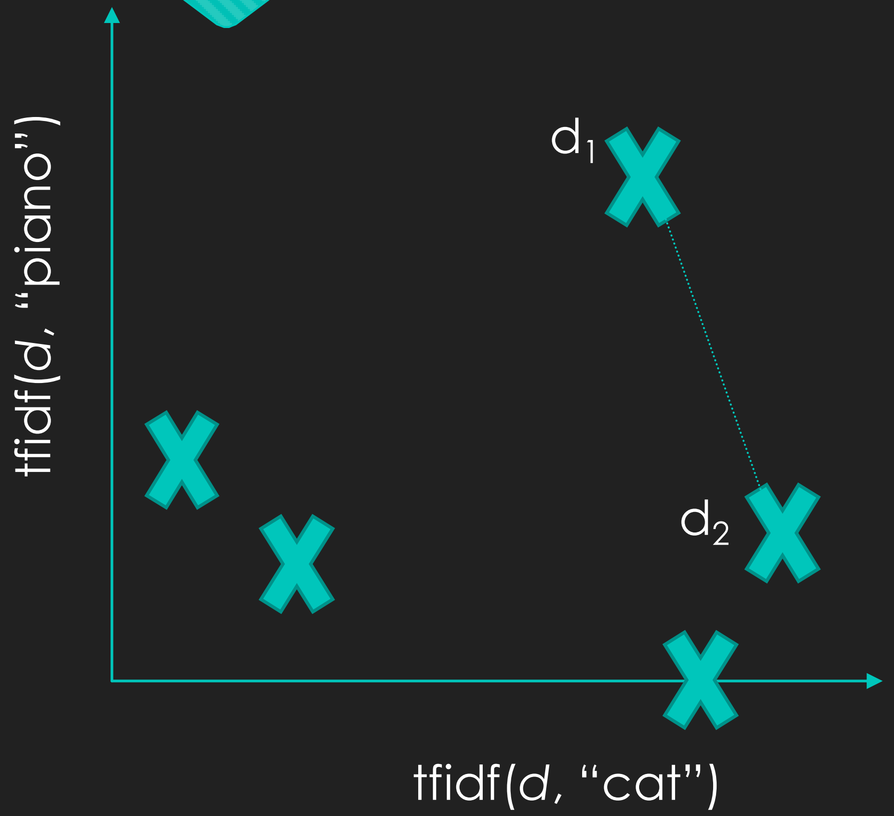
- Term frequency inverse document frequency:

$$\mathrm{tfidf}(d, t) = \mathrm{tf}(d, t) \times \mathrm{idf}(t)$$

Variants on TFIDF

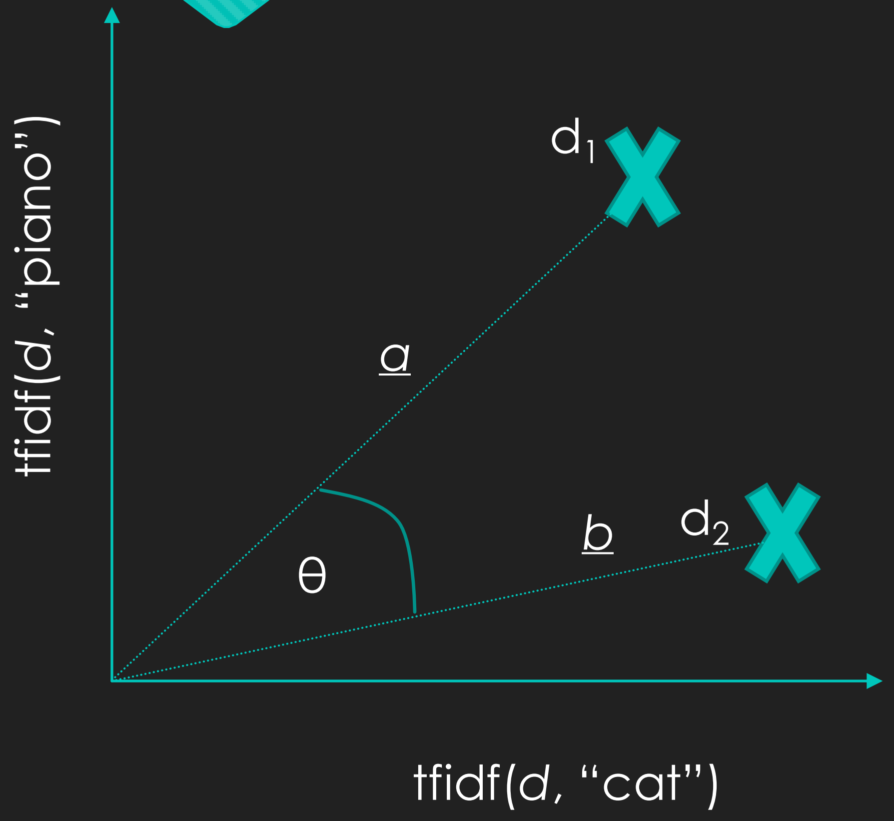
- Many variations on the original TFIDF measure e.g.,
  - Boolean "frequencies": $tf(d,t) \; = \; 1$ if t appears in d and 0 otherwise
  - term frequency adjusted for document length: $tf(d,t) = \frac{freq(d,t)}{\sum_{t' \in d} freq(d,t)}$
  - logarithmically scaled term frequency: $tf(d,t) \; = \; \log(1 + freq(d,t))$
- why do you think each of the above adjustments might be useful?

Vector similarity



- Similar items are close together in the vector space
- Dissimilar items are far apart
- How do we measure distance or closeness in a vector space?
  - Euclidean distance
  - Cosine similarity

Cosine similarity



- The more similar two documents are, the smaller the angle θ between their vectors will be.
- Recall from Maths:
  - $\cos(0) = 1$
  - $\cos(90) = 0$
- So:

$$sim(d_1, d_2) = \cos(\theta)$$

dot product

$$= \frac{\underline{a}.\underline{b}}{\sqrt{\underline{a}.\underline{a} \times \underline{b}.\underline{b}}}$$

Where:

m=number of dimensions

$$\underline{a}.\underline{b} = \sum_{i}^{m} a_i b_i$$

Cosine similarity intuitions

- Dot product of two vectors is high when they are in the same direction
- For example:

$$\underline{a} = \begin{pmatrix} 5 \\ 0 \\ 1 \end{pmatrix} \qquad \underline{b} = \begin{pmatrix} 10 \\ 1 \\ 0 \end{pmatrix} \qquad \underline{c} = \begin{pmatrix} 1 \\ 10 \\ 1 \end{pmatrix}$$

$$\underline{a}.\underline{b} = 5 \times 10 + 0 \times 1 + 1 \times 0 = 50 \qquad \underline{b}.\underline{c} = ?$$

$$\underline{a}.\underline{c} = 5 \times 1 + 0 \times 10 + 1 \times 1 = 6$$

Cosine Similarity denominator

- The denominator of the cosine calculation normalizes for the lengths of the vectors, so that we get an answer between 0 and 1

$$\underline{a} = \begin{pmatrix} 5 \\ 0 \\ 1 \end{pmatrix} \qquad \underline{b} = \begin{pmatrix} 10 \\ 1 \\ 0 \end{pmatrix} \qquad \underline{c} = \begin{pmatrix} 1 \\ 10 \\ 1 \end{pmatrix}$$

$$\underline{a}.\underline{b} = 50 \qquad \underline{a}.\underline{a} = 5 \times 5 + 1 \times 1 = 26 \qquad \cos(\underline{a}, \underline{b}) = \frac{50}{\sqrt{26 \times 101}}$$

$$\underline{a}.\underline{c} = 6 \qquad \underline{b}.\underline{b} = 10 \times 10 + 1 \times 1 = 101$$

$$= 0.976$$

$$\underline{b}.\underline{c} = ? \qquad \underline{c}.\underline{c} = 1 \times 1 + 10 \times 10 + 1 \times 1 = 102$$

Using cosine

Measure similarity of

- Two documents
- Document and a query
- Two queries
- Two terms
  - where terms have vectors with documents as dimensions
    - Latent Semantic Analysis
  - where terms have vectors with co-occurring terms as dimensions
    - Distributional semantics

Beyond words

Beyond words as dimensions

- So far have assumed that topic terms are **unigrams** (single words)
- Often a phrase or multiword term is characteristic of a topic
  - **n-grams**
  - e.g., “hedge fund”, “black hole”, “surface to air missile”
  - Individual words in phrase may be ambiguous or too general
- Same TFIDF weighting can be applied to phrases

Phrases as topic terms

- Lots of unigrams. Maybe 100 000
- Even more bigrams. Maybe $100\,000^2$
- Some useful phrases even longer. How many possible 5 word phrases are there in the English language?
- How do we find useful ones?

Collocations

- Collocations are n-grams which occur together more often than by chance
- Consider an observed bigram *"black hole"*
- How often does the bigram occur? How often do the individual unigrams "*black*" and "*hole*" occur?
- How frequently might we expect "*black hole*" to occur if words occurred independently of each other?
- Collocations are n-grams where the **observed joint probability** is greater than the **expected joint probability** for independent events

Point-wise Mutual Information (PMI)

$$PMI(w_1, w_2) = \log\left(\frac{P(w_1, w_2)}{P(w_1).P(w_2)}\right)$$

observed joint probability

expected joint probability
(assuming independence)

- PMI tells us how surprising it is that a phrase has occurred as frequently as it does
- If the **observed joint probability > expected joint probability**
  - Ratio greater than 1
  - PMI is positive
- If the **observed joint probability < expected joint probability**
  - Ratio less than 1
  - PMI is negative

COLLOCATION

Recap

TFIDF Questions

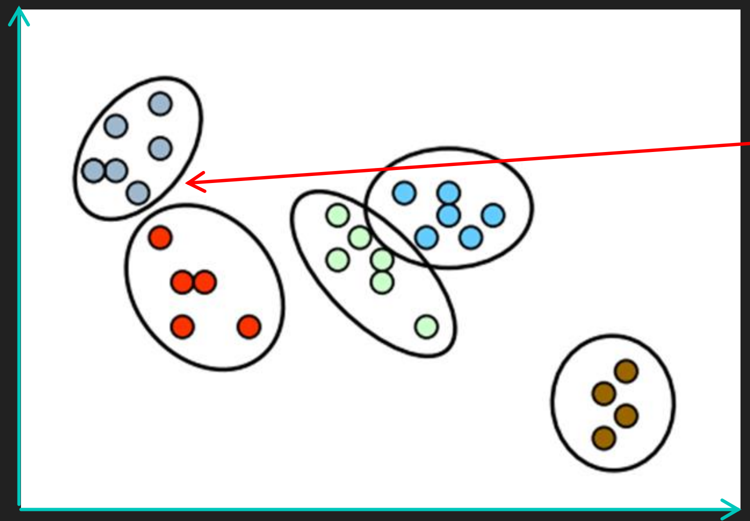
	Doc 1	Doc 2	Doc 3	Doc 4
word 1	10	15	8	12
word 2	20	0	0	0
word 3	5	7	0	0
word 4	0	3	7	8

1. What is the tfidf value of word 3 in Doc 2?
2. What kind of word is word 1 likely to be? Give an example.
3. What kind of word is word 2 likely to be? Give an example

Clustering

Clustering

Clustering (or cluster analysis) is the task of grouping objects in such a way that objects in the same group (called a **cluster**) are **more similar** to each other than to those in other clusters.

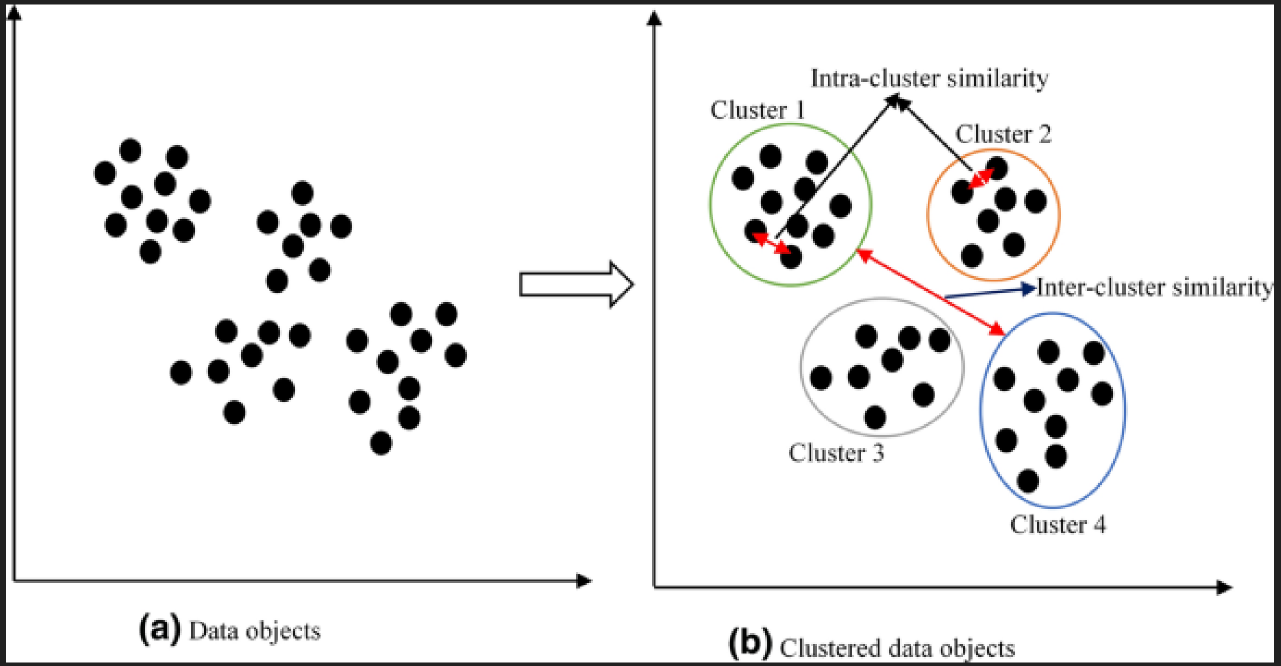


In general, there is no one "correct" clustering. For example, a different algorithm may assign the red point closest to the grey cluster to the grey cluster.

Clustering algorithms all affected by choice of similarity / distance measure and scaling of dimensions

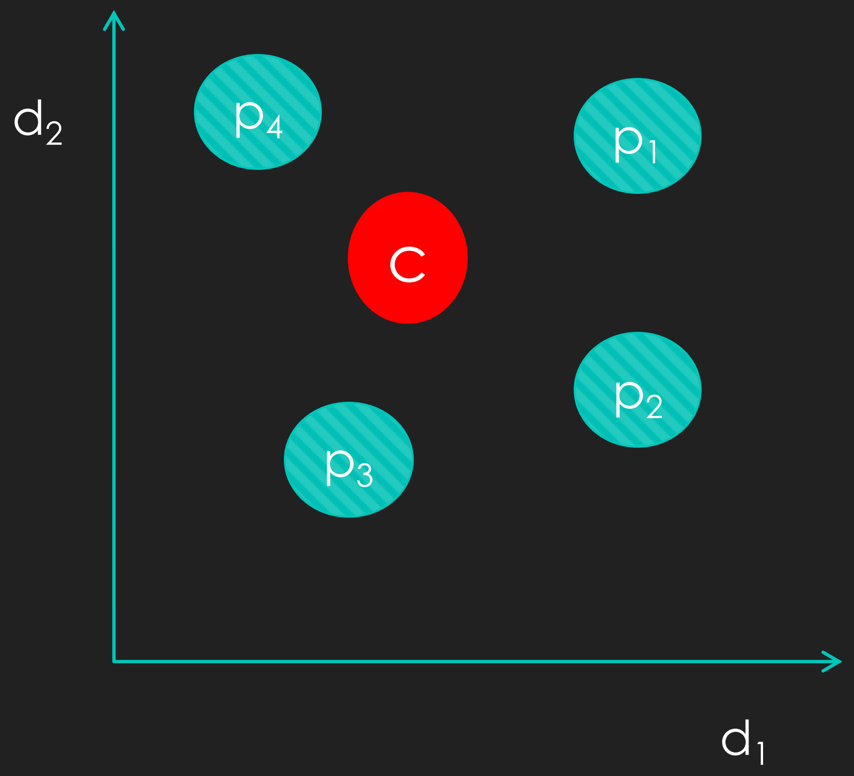
Clustering

- Clusters are groups of objects which are mutually similar to each other
  - => intra-cluster similarity is high
- and are dissimilar to objects outside of the cluster
  - => inter-cluster similarity is low
- objects can be documents in vector space
- how do we find the clusters?



K-means

- Simple partitioning approach based on the idea of centroids or prototypes

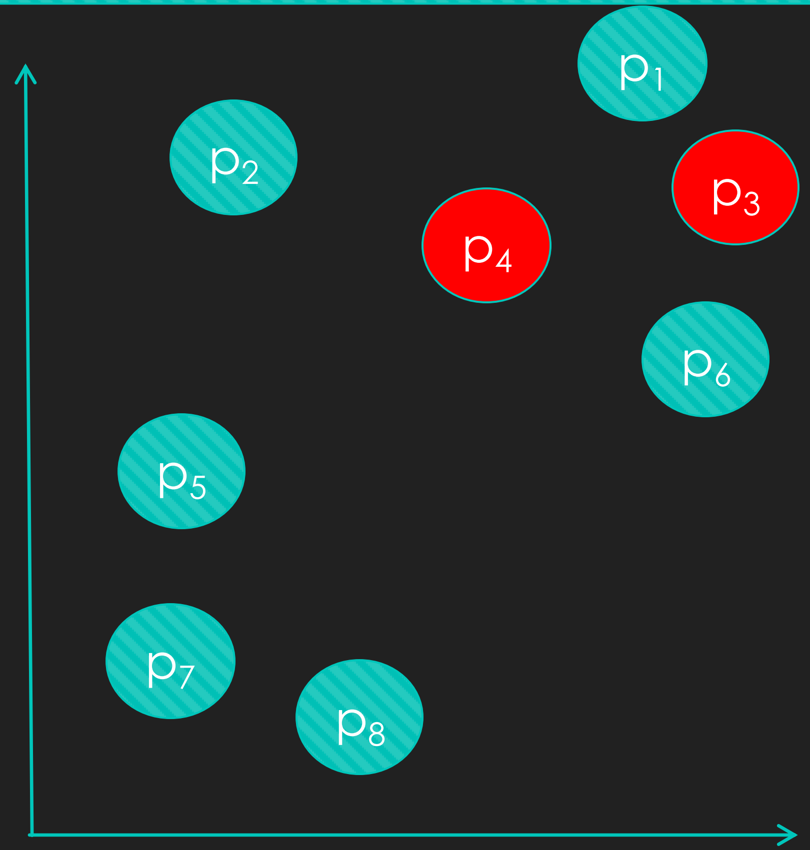


The centroid (middle) of any group of points in a Euclidean space can be found by taking the mean on every dimension.

$$c_{d_j} = \frac{\sum_{i=0}^{n} p_{d_j,i}}{n}$$

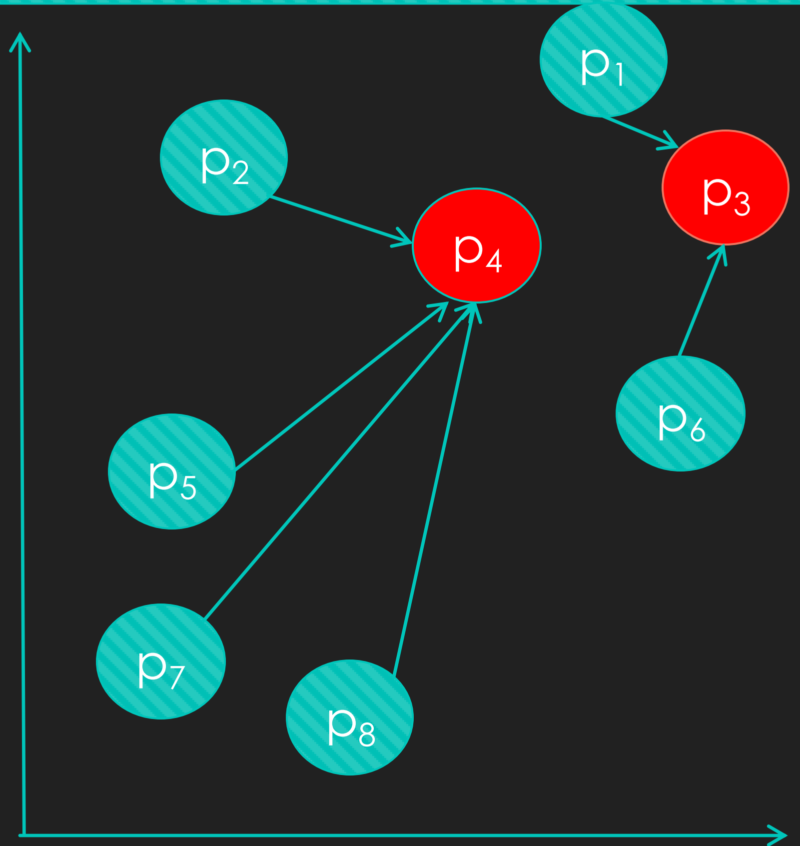
K-means

1. Select K points as initial centroids
2. while centroids changing:
    1. Form K clusters by assigning each point to its nearest centroid
    2. Recompute centroid of the cluster



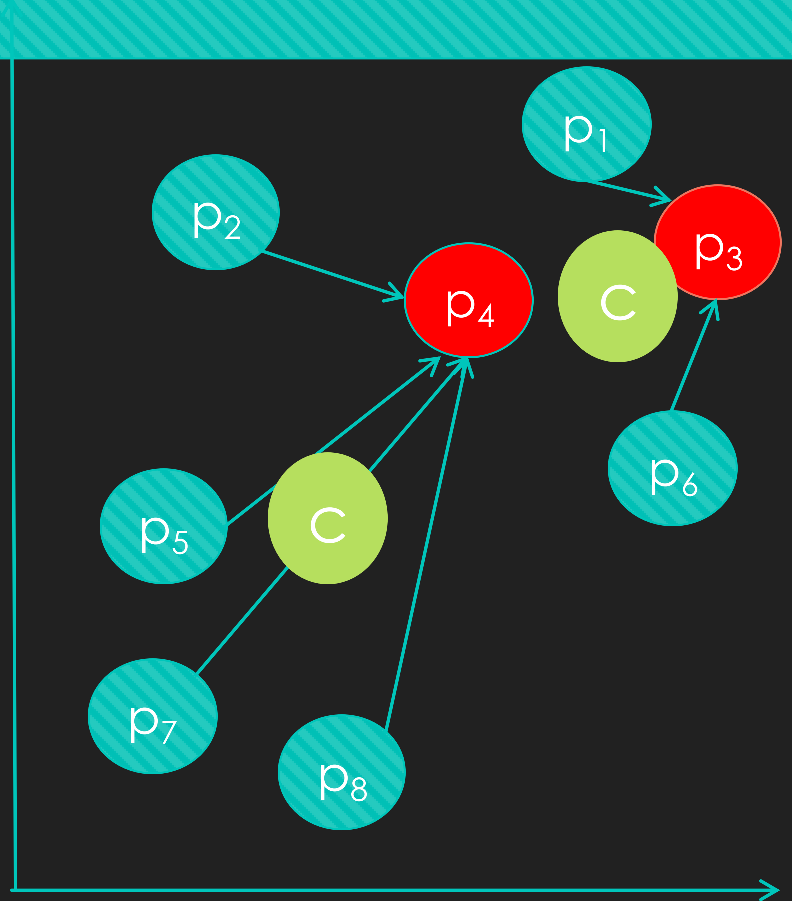
K-means

1. Select K points as initial centroids
2. while centroids changing:
   1. Form K clusters by assigning each point to its nearest centroid
   2. Recompute centroid of the cluster



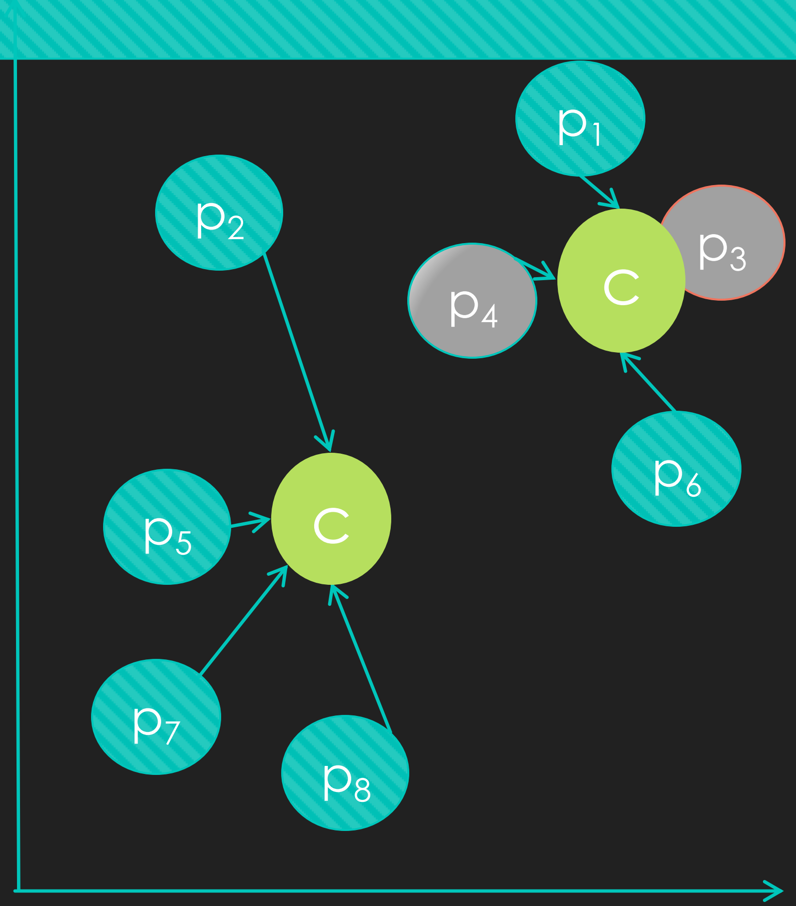
K-means

1. Select K points as initial centroids
2. while centroids changing:
    1. Form K clusters by assigning each point to its nearest centroid
    2. Recompute centroid of the cluster



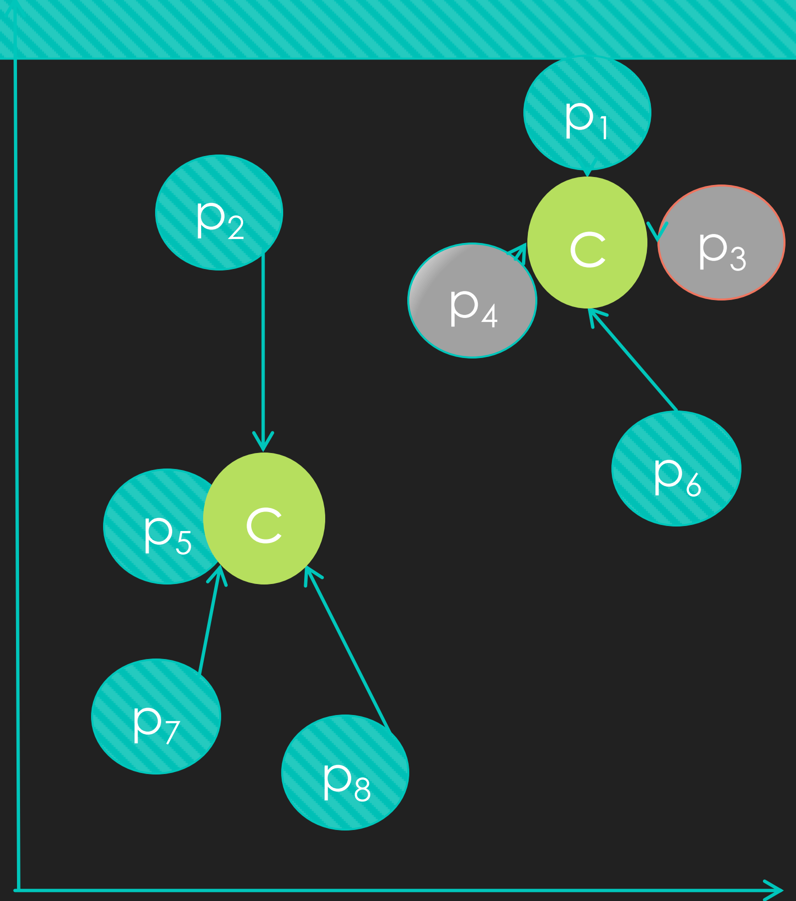
K-means

1. Select K points as initial centroids
2. while centroids changing:
    1. Form K clusters by assigning each point to its nearest centroid
    2. Recompute centroid of the cluster



K-means

1. Select K points as initial centroids
2. while centroids changing:
    1. Form K clusters by assigning each point to its nearest centroid
    2. Recompute centroid of the cluster



Disadvantages of K-Means

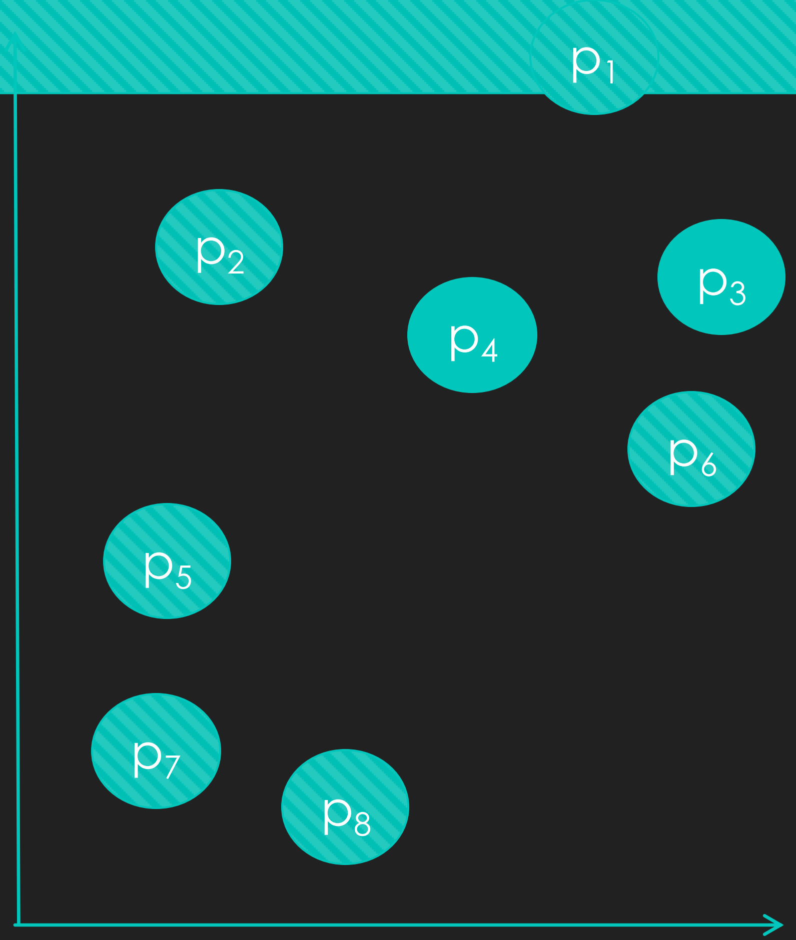
- Need to know the number of clusters in advance
- Each point is only assigned to a single cluster (hard clustering)
- Flat structure (no clusters within clusters)
- May converge on local mimimum i.e., suboptimal clustering (this can be overcome to some extent by repeated random initialisations)

Agglomerative hierarchical clustering

- is an agglomerative technique which builds up clusters by repeatedly merging the closest pair of clusters
- Do not need to know the number of clusters in advance
- Hierarchical (clusters have internal structure)

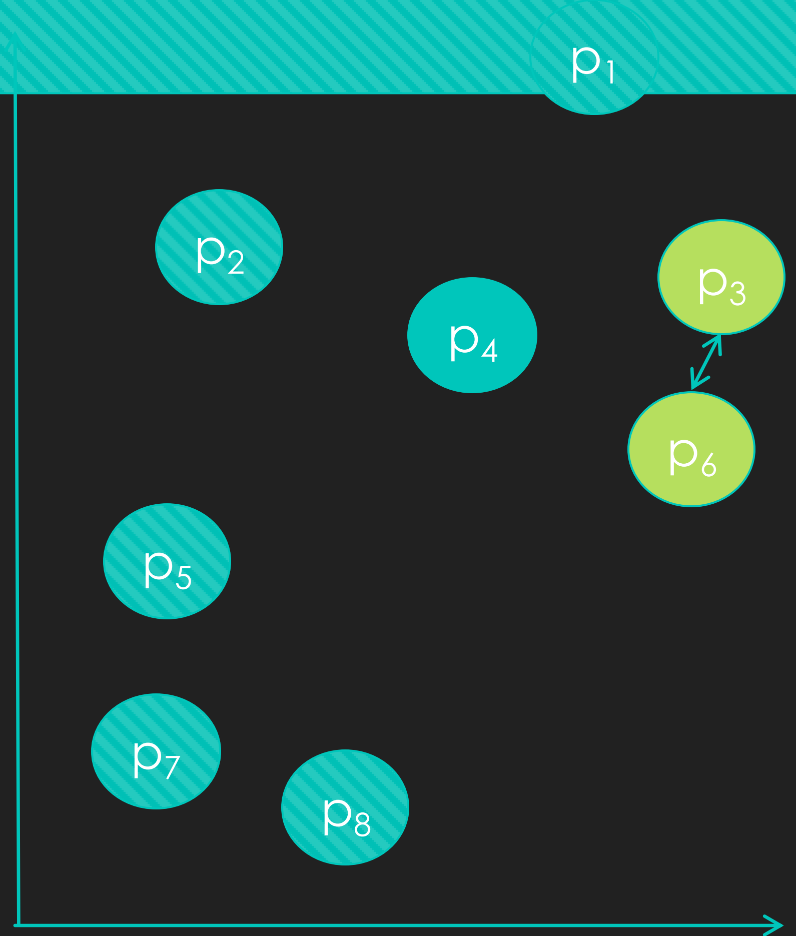
Agglomerative hierarchical clustering

1. Initialise n clusters as the n data points
2. Find closest pair <p_i,p_j> of clusters with distance d
3. while d < threshold:
    1. Merge clusters <p_i,p_j>
    2. Find closest pair <p_i, p_j> with distance d



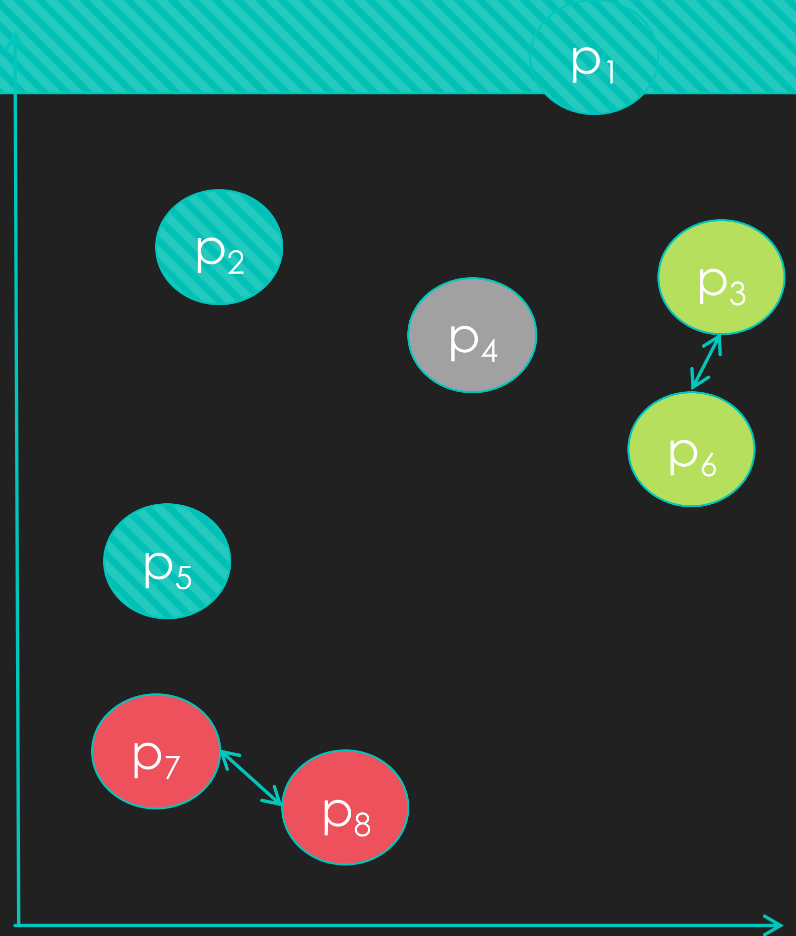
Agglomerative hierarchical clustering

1. Initialise n clusters as the n data points
2. Find closest pair $<p_i,p_j>$ of clusters with distance d
3. while d < threshold:
    1. Merge clusters $<p_i,p_j>$
    2. Find closest pair $<p_i, p_j>$ with distance d



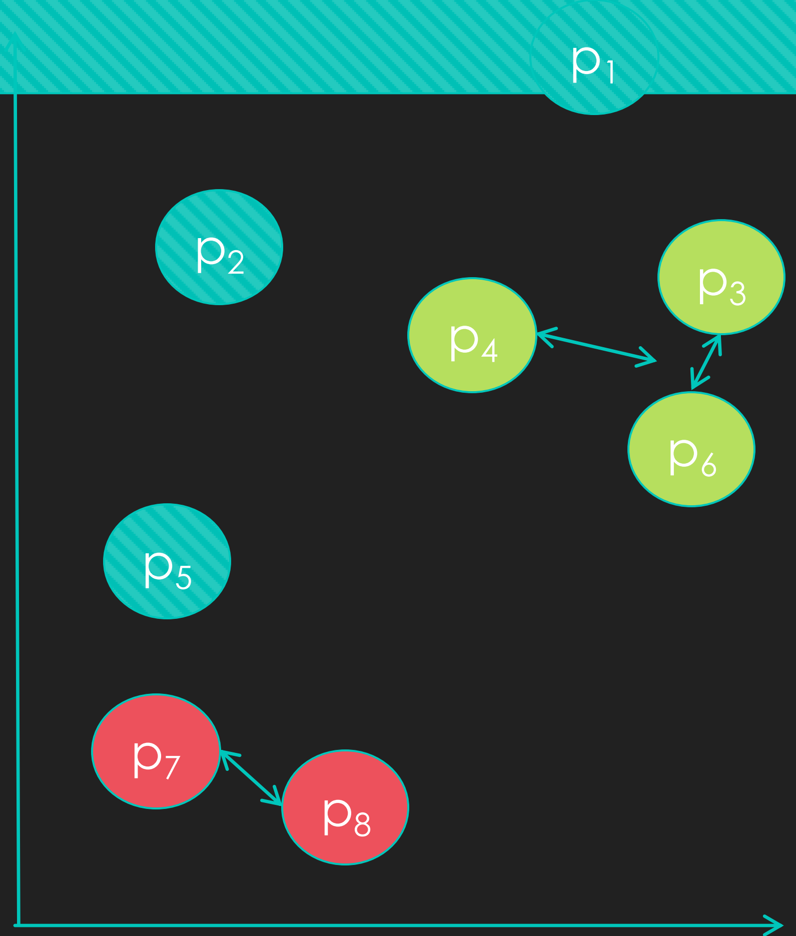
Agglomerative hierarchical clustering

1. Initialise n clusters as the n data points
2. Find closest pair $<p_i, p_j>$ of clusters with distance d
3. while d < threshold:
    1. Merge clusters $<p_i, p_j>$
    2. Find closest pair $<p_i, p_j>$ with distance d



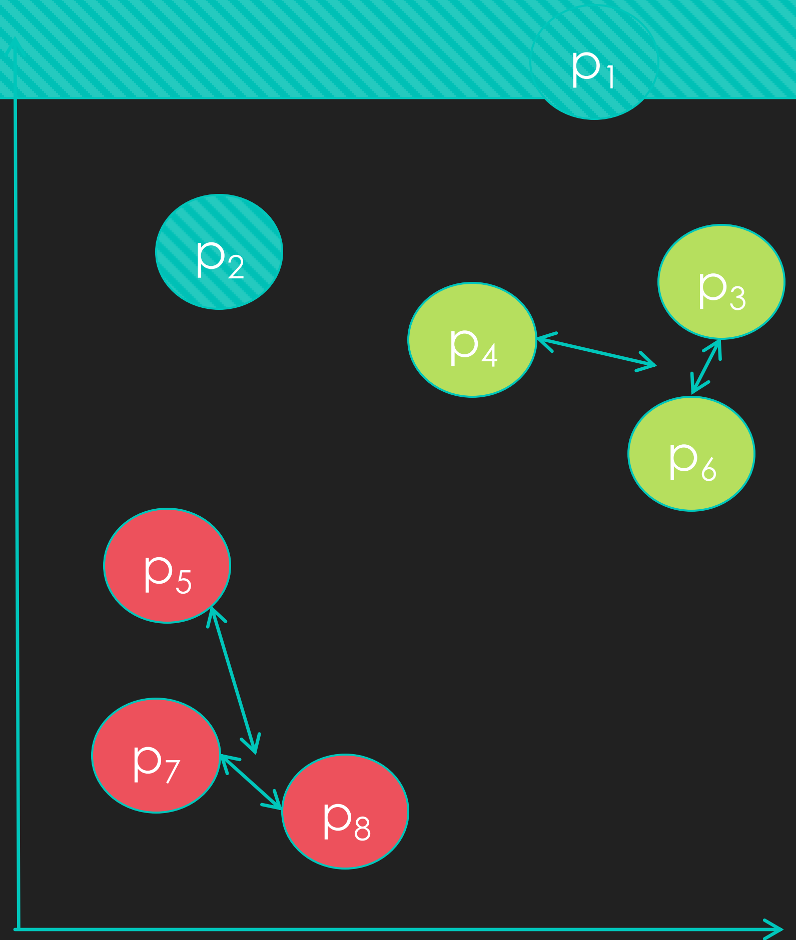
Agglomerative hierarchical clustering

1. Initialise n clusters as the n data points
2. Find closest pair <p_i,p_j> of clusters with distance d
3. while d < threshold:
    1. Merge clusters <p_i,p_j>
    2. Find closest pair <p_i, p_j> with distance d



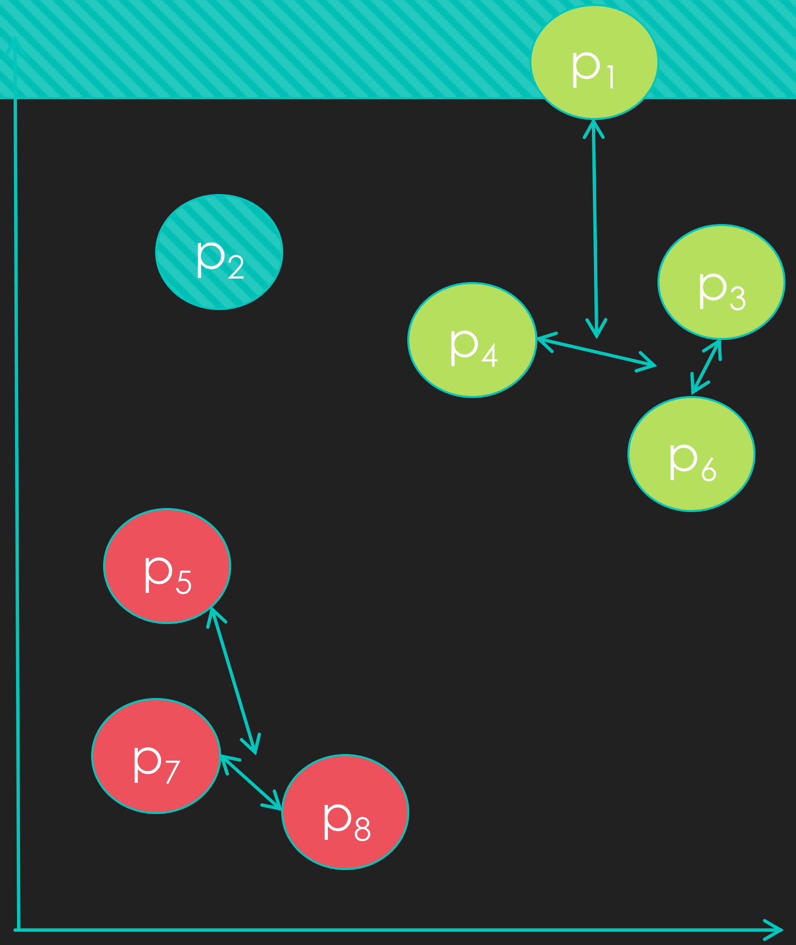
Agglomerative hierarchical clustering

1. Initialise n clusters as the n data points
2. Find closest pair $<p_i,p_j>$ of clusters with distance d
3. while d $<$ threshold:
    1. Merge clusters $<p_i,p_j>$
    2. Find closest pair $<p_i, p_j>$ with distance d



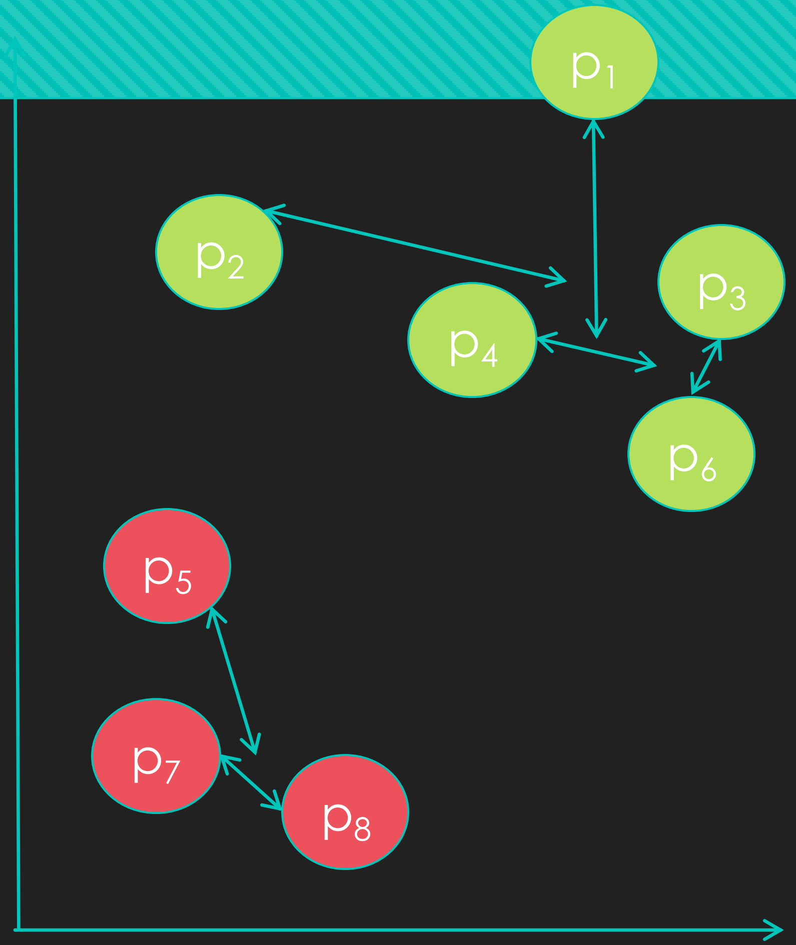
Agglomerative hierarchical clustering

1. Initialise n clusters as the n data points
2. Find closest pair $<p_i,p_j>$ of clusters with distance d
3. while d < threshold:
    1. Merge clusters $<p_i,p_j>$
    2. Find closest pair $<p_i, p_j>$ with distance d



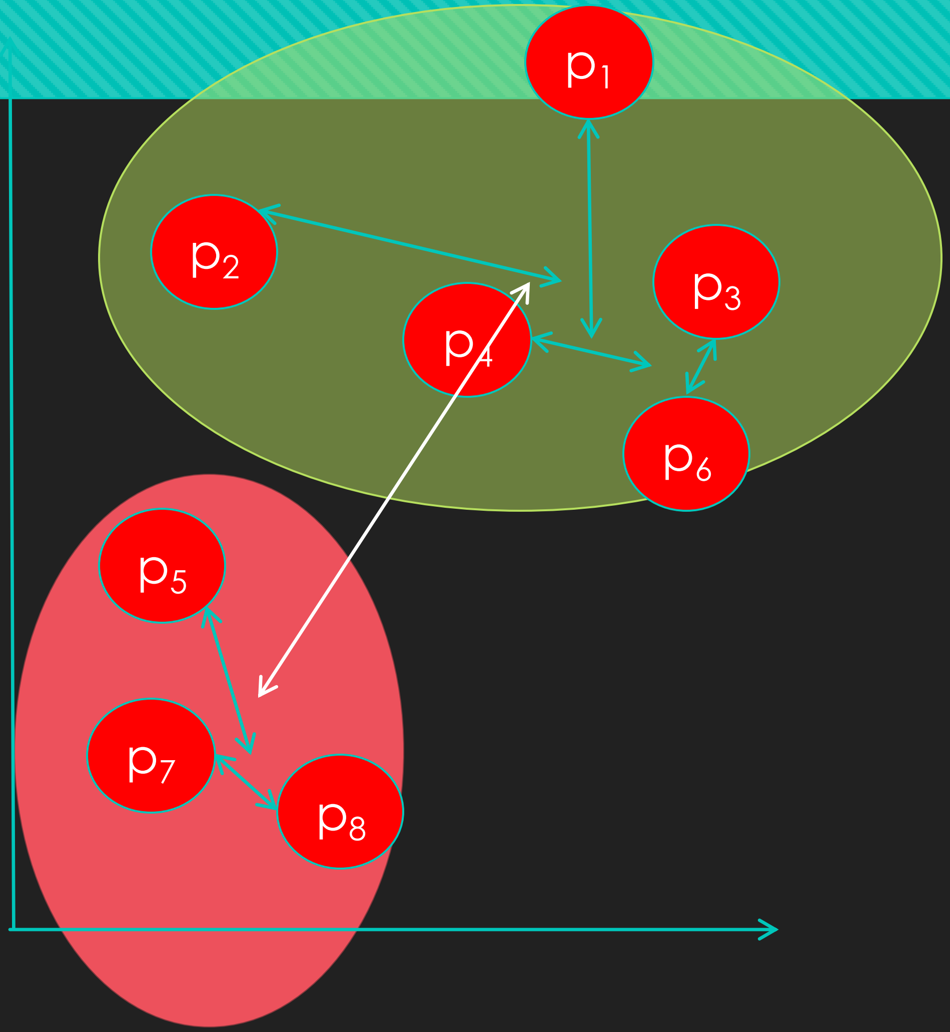
Agglomerative hierarchical clustering

1. Initialise n clusters as the n data points
2. Find closest pair $<p_i, p_j>$ of clusters with distance d
3. while d < threshold:
    1. Merge clusters $<p_i, p_j>$
    2. Find closest pair $<p_i, p_j>$ with distance d



Agglomerative hierarchical clustering

1. Initialise n clusters as the n data points
2. Find closest pair <p_i,p_j> of clusters with distance d
3. while d < threshold:
    1. Merge clusters <p_i,p_j>
    2. Find closest pair <p_i, p_j> with distance d



Disadvantages of agglomerative hierarchical clustering

- Computationally expensive to keep recomputing all nearest neighbours
- runtime = $O(n^2 \log n)$ where n is the number of data points
- and the constant is large -> multiple of d where d is the number of dimensions
- in comparison, k-means is O(n * d * k * I) where k is the number of clusters and I is the number of iterations

Making progress

- Next week you should complete **all** of the exercises in the single notebook for week 5 on Document Similarity

  - Part 1: Lab_5_1.ipynb

- And make progress with your coursework