POLITECNICO DI MILANO

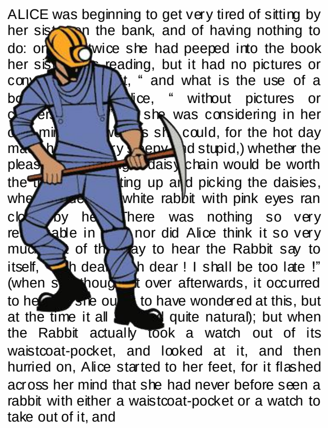
Natural Language Processing

Clustering Text

Natural Language Processing

Some slide content based on textbooks:
Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition *by Daniel Jurafsky and James H. Martin*

Mark Carman



ALICE was beginning to get very tired of sitting by
her sis[illegible]n the bank, and of having nothing to
do: on[illegible]twice she had peeped into the book
her sis[illegible]reading, but it had no pictures or
con[illegible]t, “ and what is the use of a
bo[illegible]ice, “ without pictures or
c[illegible]she was considering in her
[illegible]s sh[illegible] could, for the hot day
ma[illegible]d stupid,) whether the
pleas[illegible]daisy chain would be worth
the [illegible]ting up and picking the daisies,
whe[illegible]white rabbit with pink eyes ran
cl[illegible]by he[illegible]here was nothing so very
re[illegible]able in [illegible] nor did Alice think it so very
muc[illegible] of th[illegible]ay to hear the Rabbit say to
itself, [illegible]h dea[illegible]h dear ! I shall be too late !”
(when s[illegible]thoug[illegible]t over afterwards, it occurred
to he[illegible]she ou[illegible] to have wondered at this, but
at the time it all [illegible] quite natural); but when
the Rabbit actually took a watch out of its
waistcoat-pocket, and looked at it, and then
hurried on, Alice started to her feet, for it flashed
across her mind that she had never before seen a
rabbit with either a waistcoat-pocket or a watch to
take out of it, and

Miner Image source: https://freesvg.org/miner-1574424884

Lecture content:

- Why cluster text?
- Similarities between documents
- Basic clustering algorithms
- Evaluating clustering
- Topic Modelling
- Visualisation via dimensionality reduction

Why cluster text?

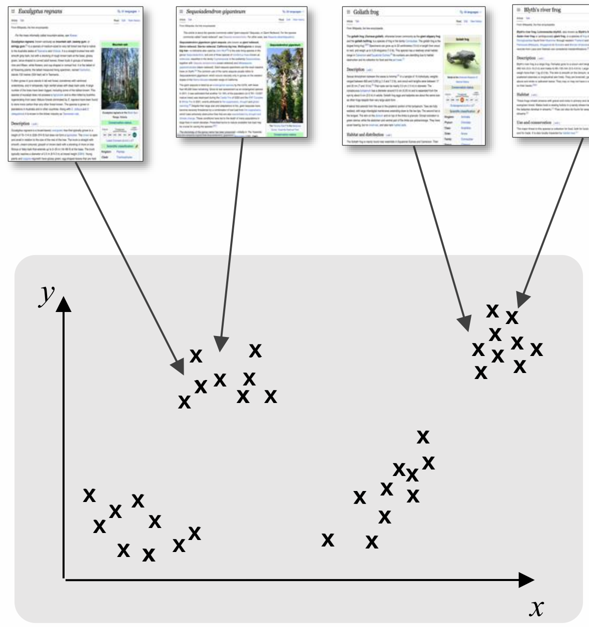
What is text clustering?

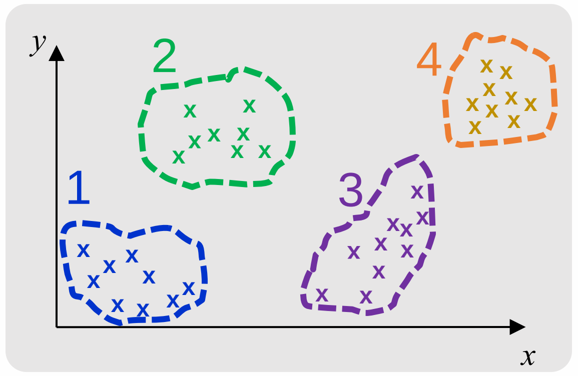
Process of **grouping documents** into **coherent subgroups**

- based on some **distance function** or **similarity measure**

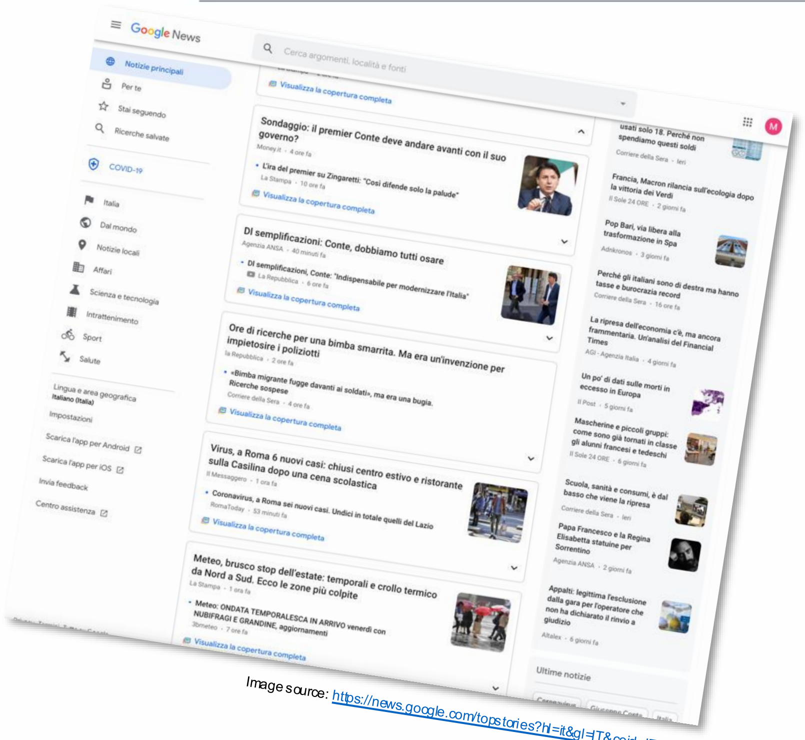
For **text documents**, usually means grouping documents into:

- same **topic**: e.g. *sports news vs politics vs technology vs …*
- same **emotion**: e.g. *angry, disappointed, elated, hopeful, …*
- same **problem** to be fixed: e.g. *no heating (faulty thermostat), unable to print (missing drivers), no hot water, etc.*
- same **complaint/praise**: e.g. *hotel has beautiful view, motel on noisy road, breakfast was excellent, …*
- same **author**: e.g. *Shakespeare, Dylan, Eminem, Trump, Satoshi Nakamoto*
- same **text generator**: e.g. *used to generate spam emails*
- same **source document**: e.g. *for de-duplicating content during a web-crawl*





Example Clustering Applications



Visualizza la copertura completa

Sondaggio: il premier Conte deve andare avanti con il suo governo?
Money.it · 4 ore fa
- L'ira del premier su Zingaretti: "Così difende solo la palude"
  La Stampa · 10 ore fa

Visualizza la copertura completa

Dl semplificazioni: Conte, dobbiamo tutti osare
Agenzia ANSA · 40 minuti fa
- Dl semplificazioni, Conte: "Indispensabile per modernizzare l'Italia"
  La Repubblica · 6 ore fa

Visualizza la copertura completa

Ore di ricerche per una bimba smarrita. Ma era un'invenzione per impietosire i poliziotti
la Repubblica · 2 ore fa
- «Bimba migrante fugge davanti ai soldati», ma era una bugia. Ricerche sospese
  Corriere della Sera · 4 ore fa

Visualizza la copertura completa

Virus, a Roma 6 nuovi casi: chiusi centro estivo e ristorante sulla Casilina dopo una cena scolastica
Il Messaggero · 1 ora fa
- Coronavirus, a Roma sei nuovi casi. Undici in totale quelli del Lazio
  RomaToday · 53 minuti fa

Visualizza la copertura completa

Meteo, brusco stop dell'estate: temporali e crollo termico da Nord a Sud. Ecco le zone più colpite
La Stampa · 1 ora fa
- Meteo: ONDATA TEMPORALESCA IN ARRIVO venerdì con NUBIFRAGI E GRANDINE, aggiornamenti
  3bmeteo · 7 ore fa

Visualizza la copertura completa

usati solo 18. Perché non spendiamo questi soldi
Corriere della Sera · Ieri

Francia, Macron rilancia sull'ecologia dopo la vittoria dei Verdi
Il Sole 24 ORE · 2 giorni fa

Pop Bari, via libera alla trasformazione in Spa
Adnkronos · 3 giorni fa

Perché gli italiani sono di destra ma hanno tasse e burocrazia record
Corriere della Sera · 16 ore fa

La ripresa dell'economia c'è, ma ancora frammentaria. Un'analisi del Financial Times
AGI - Agenzia Italia · 4 giorni fa

Un po' di dati sulle morti in eccesso in Europa
Il Post · 5 giorni fa

Mascherine e piccoli gruppi: come sono già tornati in classe gli alunni francesi e tedeschi
Il Sole 24 ORE · 6 giorni fa

Scuola, sanità e consumi, è dal basso che viene la ripresa
Corriere della Sera · Ieri

Papa Francesco e la Regina Elisabetta statuine per Sorrentino
Agenzia ANSA · 2 giorni fa

Appalti: legittima l'esclusione dalla gara per l'operatore che non ha dichiarato il rinvio a giudizio
Altalex · 6 giorni fa

Image source: https://news.google.com/topstories?hl=it&gl=IT&ceid=IT:it

News aggregators like Google News (https://news.google.com/)
use clustering to **summarise/characterise** a collection:

- what **topics** are being discussed?
- how **prevalent** is each topic?
- what topics are **trending**?
- which articles are most **exemplary** (central to topic)?

In security applications, such as phishing and malware detection,

- determine which groups might be malicious (based on a few identified instances)
- use clusters to discover useful **features** for describing documents
- what aspects distinguish the groups?

Representing documents & measuring similarity

Document similarity

In Section 6.3.1 we normalized each document vector by the Euclidean length of the vector, so that all document vectors turned into unit vectors. In doing so, we eliminated all information on the length of the original document; this masks some subtleties about longer documents. First, longer documents will – as a result of containing more terms – have higher tf values. Second, longer documents contain more distinct terms. These factors can conspire to raise the scores of longer documents, which (at least for some information needs) is unnatural. Longer documents can broadly be lumped into two categories: (1) *verbose* documents that essentially repeat the same content – in these, the length of the document does not alter the relative weights of different terms; (2) documents covering multiple different topics, in which the search terms probably match small segments of the document but not all of it – in this case, the relative weights of terms are quite different from a single short document that matches the query terms. Compensating for this phenomenon is a form of document length normalization that is independent of term and document frequencies. To this end, we introduce a form of normalizing the vector representations of documents in the collection, so that the resulting "normalized" documents are not necessarily of unit length. Then, when we compute the dot product score between a (unit) query vector and such a normalized document, the score is skewed to account for the effect of document length on relevance. This form of compensation for document length is known as *pivoted document length normalization.*

similar?

In Section 6.3.1 we normalized each document vector by the Euclidean length of the vector, so that all document vectors turned into unit vectors. In doing so, we eliminated all information on the length of the original document; this masks some subtleties about longer documents. First, longer documents will – as a result of containing more terms – have higher tf values. Second, longer documents contain more distinct terms. These factors can conspire to raise the scores of longer documents, which (at least for some information needs) is unnatural. Longer documents can broadly be lumped into two categories: (1) *verbose* documents that essentially repeat the same content – in these, the length of the document does not alter the relative weights of different terms; (2) documents covering multiple different topics, in which the search terms probably match small segments of the document but not all of it – in this case, the relative weights of terms are quite different from a single short document that matches the query terms. Compensating for this phenomenon is a form of document length normalization that is independent of term and document frequencies. To this end, we introduce a form of normalizing the vector representations of documents in the collection, so that the resulting "normalized" documents are not necessarily of unit length. Then, when we compute the dot product score between a (unit) query vector and such a normalized document, the score is skewed to account for the effect of document length on relevance. This form of compensation for document length is known as *pivoted document length normalization.*

Consider a document collection together with an ensemble of queries for that collection. Suppose that we were given, for each query q and for each document d, a Boolean judgment of whether or not d is relevant to the query q; in Chapter 8 we will see how to procure such a set of relevance judgments for a query ensemble and a document collection. Given this set of relevance judgments, we may compute a *probability of relevance* as a function of document length, averaged over all queries in the ensemble. The resulting plot may look like the curve drawn in thick lines in Figure 6.16. To compute this curve, we bucket documents by length and compute the fraction of relevant documents in each bucket, then plot this fraction against the median document length of each bucket. (Thus even though the "curve" in Figure 6.16 appears to be continuous, it is in fact a histogram of discrete buckets of document length.)

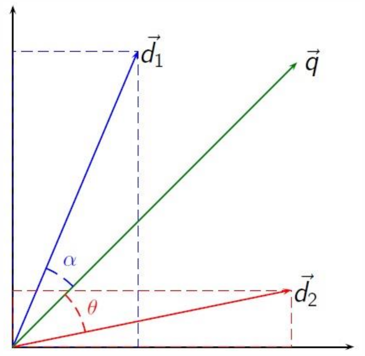
On the other hand, the curve in thin lines shows what might happen with the same documents and query ensemble if we were to use relevance as prescribed by cosine normalization Equation (6.12) – thus, cosine normalization has a tendency to distort the computed relevance vis-à-vis the true relevance, at the expense of longer documents. The thin and thick curves crossover at a point p corresponding to document length ℓ_p, which we refer to as the *pivot*

To cluster documents need to **measure similarity** between them

- based on the similarity of their content
- which is subjective of course: *is Stephen King closer to Agatha Christie or J K Rowling?*

Traditionally represent documents as **tf-idf vectors** (scaled **bag-of-words**)

- sparse representation: *most values in vector are zero*
- so measure similarity based on **number of shared words**
- **scaled by the importance** of the word: *how rare is it?*



Source: https://en.wikipedia.org/wiki/Vector_space_model#/media/File:Vector_space_model.jpg

Usually assume **similarity doesn't depend on length** of documents

- so compute **cosine similarity** between vectors
- = dot product between length normalised vectors

$$sim(\mathbf{d}_1, \mathbf{d}_2) = \frac{\mathbf{d}_1 \cdot \mathbf{d}_2}{||\mathbf{d}_1|| \, ||\mathbf{d}_2||} \qquad ||\mathbf{x}|| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Clustering techniques

Types of clustering algorithms

Many different clustering algorithms in common use

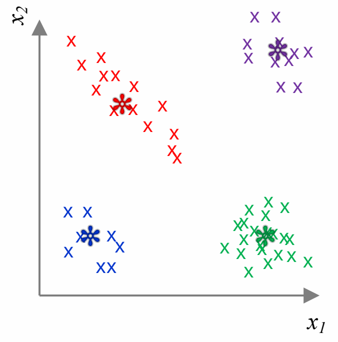
- k-Means/k-Medoids
- Agglomerative hierarchical clustering
- Density-based clustering
- Spectral clustering
- Topic Models

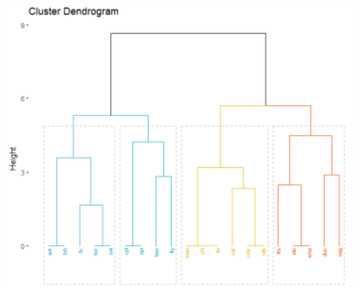
All these models are implemented in sklearn:

- https://scikit-learn.org/stable/modules/clustering.html

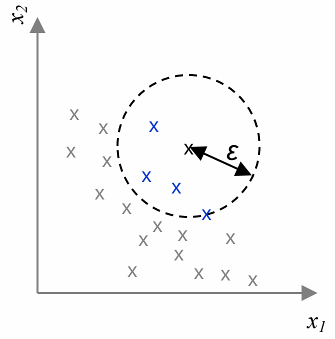
We'll concentrate on those **most used with text**:

- primarily ***k*-Means** & **Topic Models**





Source : https://commons.wikimedia.org/wiki/File:Dendrogram 3.png



Clustering Algorithms: *k*-Means

k-Means clustering

"Goto" clustering method

- simple, fast and relatively robust

How does it work?

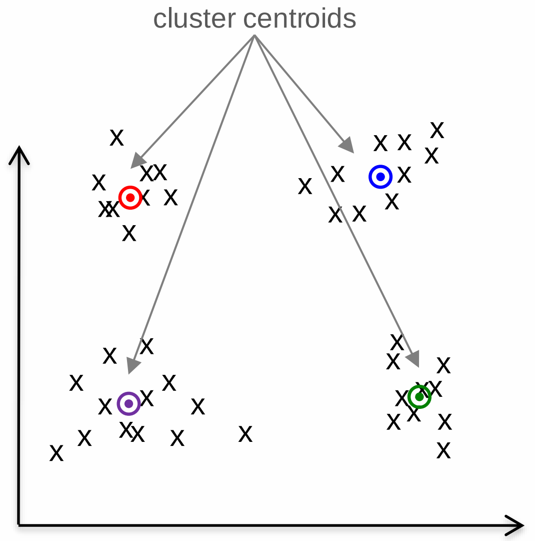
- searches for exactly *k* **clusters**
- represents each cluster by its centroid

Advantages:

- scales well to large collections
- (requires no pairwise distance calculations)

Disadvantages:

- searches for globular clusters
- implicitly assumes Euclidean distance metric (not ideal for text)
- number of clusters must be specified in advance



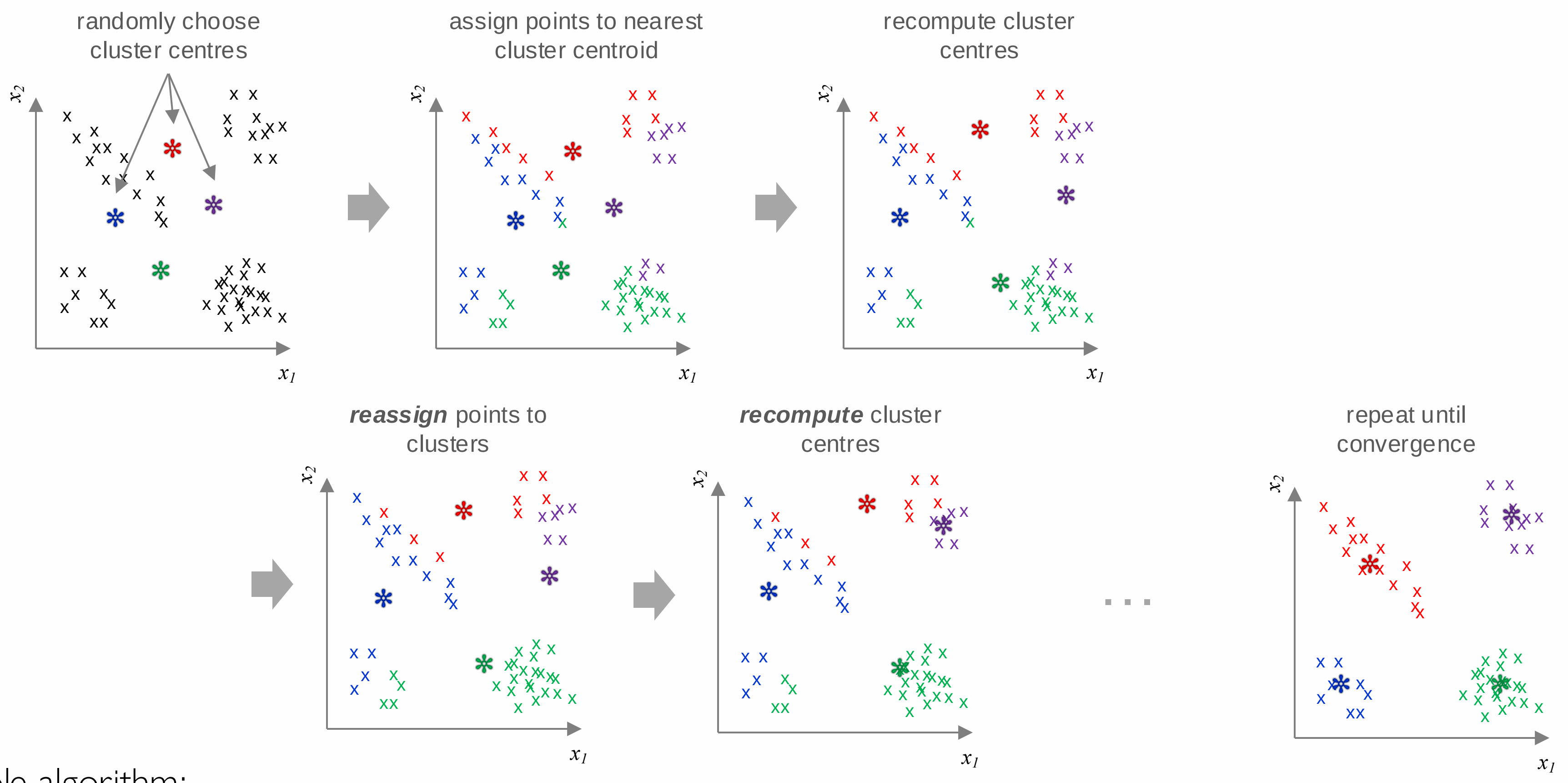
Centroid:

$$\vec{\mu}_C = \frac{1}{|C|}\sum_{\vec{x}\in C}\vec{x}$$

Euclidean distance:

$$d(x,y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$$

k-Means clustering: the process


randomly choose cluster centres
assign points to nearest cluster centroid
recompute cluster centres
reassign points to clusters
recompute cluster centres
repeat until convergence
x_1
x_2
. . .

Simple algorithm:

1. **initialize** k centroids randomly
2. **(re)assign each data point** to closest centroid
3. **(re)compute centroids** by averaging data points in cluster
4. **repeat** steps (2) & (3) until convergence, when centroids stop moving

algorithm **minimises variance of clusters** (squared distances between points & cluster centres)

$$\min \sum_{i=1}^{k} \sum_{\vec{x}\in C_i} d(\vec{x},\vec{\mu}_{C_i})^2$$

k-Means clustering – potential problems

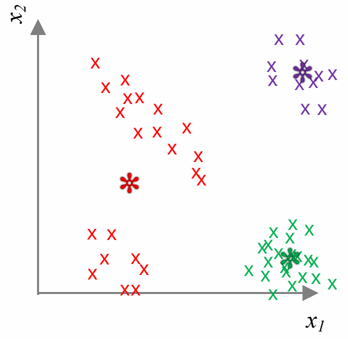
Potential problems with k-means:

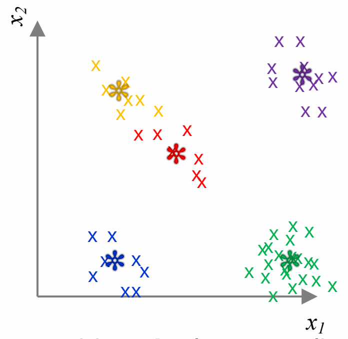
- choosing “right” value of ***k*** is critical
  - can use elbow method to choose *k*
- algorithm can converge on **local minimum**
  - run algorithm multiple times algorithm

As for all clustering algorithms, **scaling effects** similarity measure

- and thus clusters found
- but for text have both tf-idf weighting and document length normalisaton

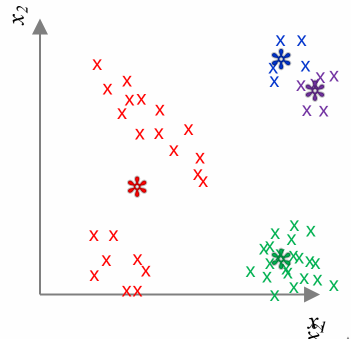
too small k → clusters merge



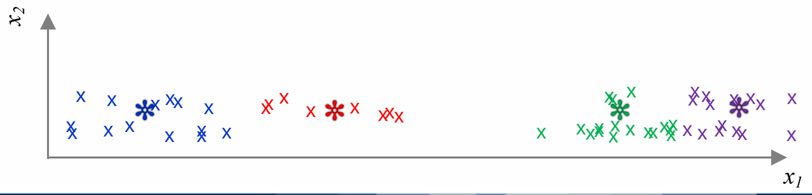


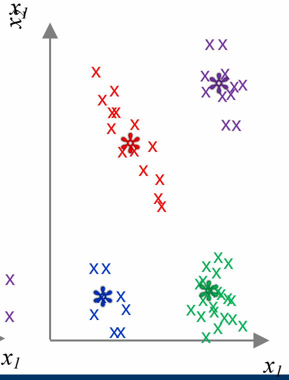
too big k → clusters split

local minimum → suboptimal groups



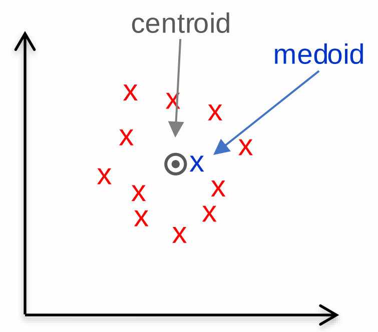
different scaling → different clusters





Clustering Algorithms: *k*-medoids

k-medoids



Represents each cluster by its **medoid** rather than its centroid

- **medoid** is one of the datapoints from dataset
  - point that is closest (***smallest average distance***) to all other points
- at each iteration, algorithm must:
  - reassign datapoints to cluster with closest medoid, and then recompute medoids

👍 Advantage:

- clustering algorithm can be used with other metrics, not just Euclidean
- medoid is a real document so provides realistic representation of cluster

👎 Disadvantage:

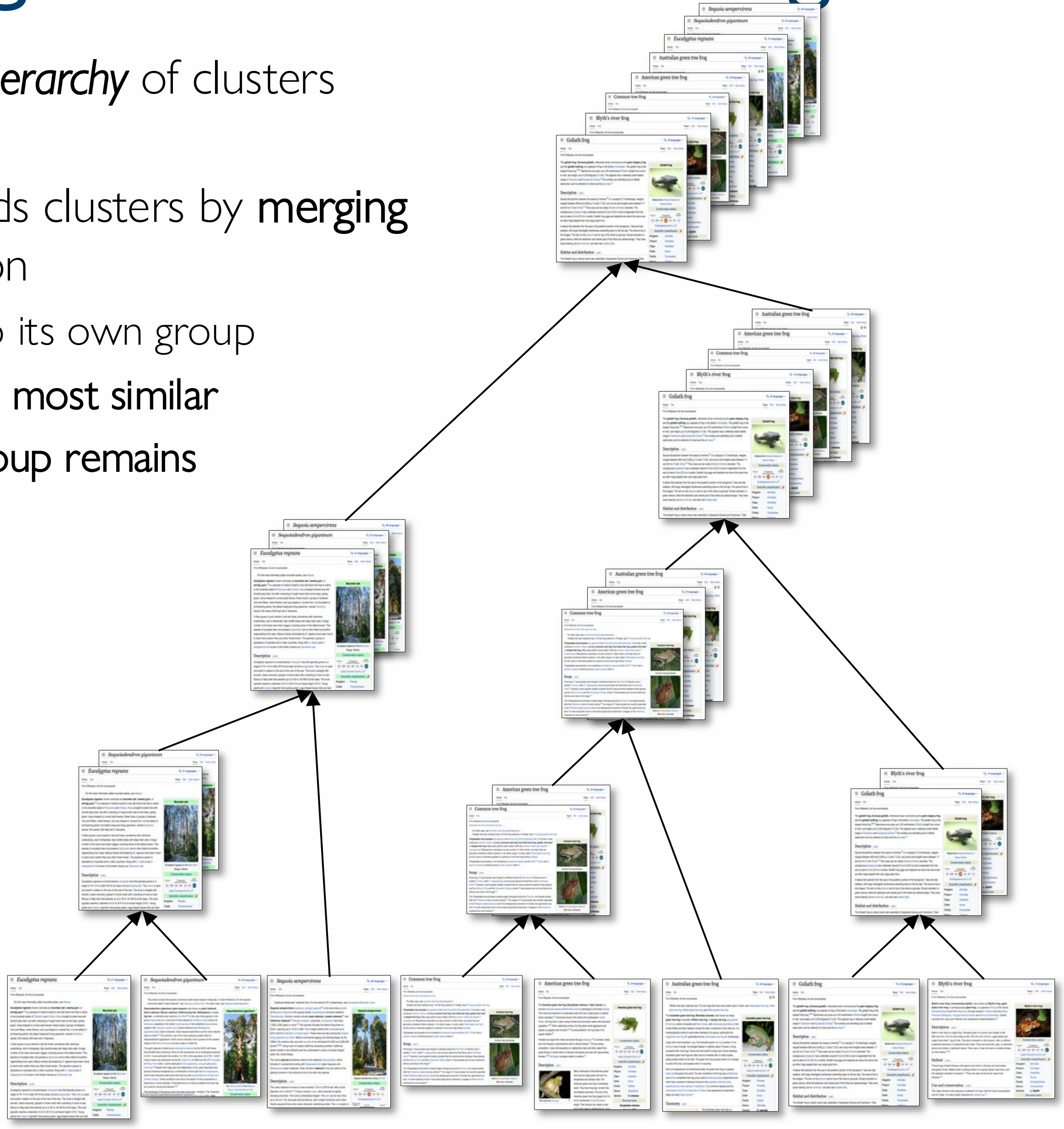
- algorithm has much **higher computational complexity** with respect to k-means
- needs to calculate distances for pairs of datapoints, which is an O(n^2) operation

Clustering Algorithms:
Agglomerative Hierarchical Clustering

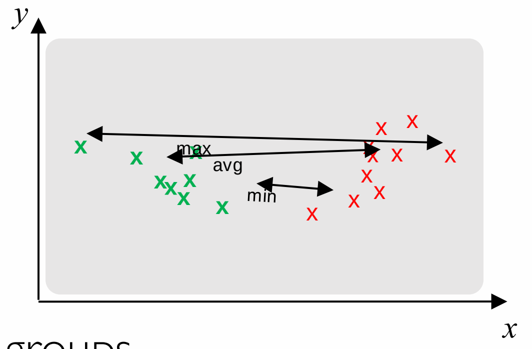
Hierarchical Agglomerative Clustering

Hierarchical clustering builds ***hierarchy*** of clusters called a ***dendrogram***

- *agglomerative clustering* builds clusters by **merging groups** in bottom-up fashion
  1. assign each document to its own group
  2. **merge** 2 groups that are **most similar**
  3. repeat **until** only **one group remains**



Agglomerative clustering: linking



Hierarchical clustering needs to compute **distances between groups** of documents

- uses a *linkage criteria*:
  - complete-linkage: **maximum** distance between points in 2 groups
  - single-linkage: **minimum** distance across groups
  - average-linkage: **average** distance across groups
- note: linking criteria used influences shape of clusters found
  - complete- or average-linkage will restrict to tight, globular clusters
  - single-linkage clustering will allow for finding long, thin clusters

👍 Advantage:

- works with *any distance metric / similarity function*
- *dendrogram* provides information about underlying structure of data

👎 Disadvantage:

- high time complexity makes it unsuitable for large datasets

Clustering Algorithms: DBScan

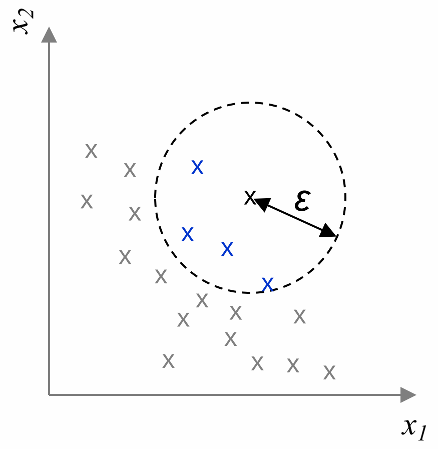
DBScan: what is it?

Density-Based Spatial Clustering of Applications with Noise

- **#clusters** not defined by user but inferred from data
- density-based algorithm: clusters found in high-density regions

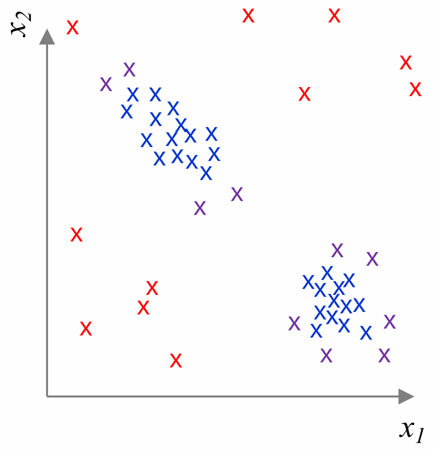
Algorithm takes 2 parameters:

- ε = radius of **neighbourhood** around each point
- *minPoints* = minimum # of points required to form cluster

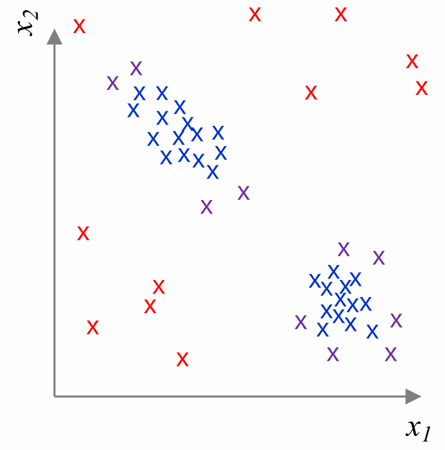


How does it work?

- # of datapoints in *ε-neighbourhood* is **density estimate**
- and *minPoints* is just a **threshold on density**
- algorithm classifies each point as either:
  - *core point*: has ≥*minPoints* neighbours inside radius ε
  - *border point*: not core, but within ε of core point
  - *noise point*: neither core nor border point



DBScan: pros and cons



👍 Advantages:

- no need to choose number of clusters in advance
- identifies and ignores noise/outlier instances
- find ***arbitrarily (oddly) shaped clusters***

👎 Disadvantages:

- performance depends critically on chosen parameters
  (usually set ε as a function of average distance to *MinPoints* nearest neighbour)
- may not work well if clusters have *different densities*

Evaluating Clustering

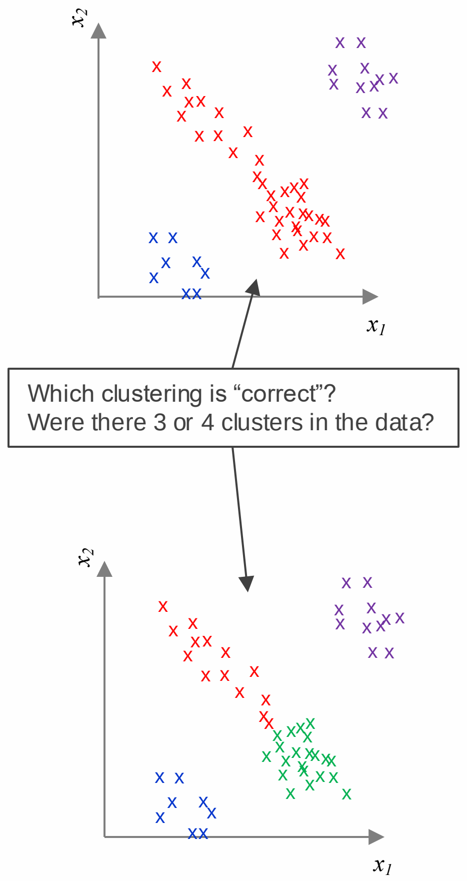
Evaluating clustering

Can be difficult to assess clustering results

- usually don't have access to ground-truth labelling
- without it, evaluation can be **subjective** (people disagree on what "right" clusters are)

Two ways to evaluate results of clustering algorithm:

1. **intrinsic** evaluation:
   - check **how coherent** and **well separated** clusters are
   - typical evaluation measures:
     - **Within-cluster Sum of Squares** (cluster variance)
     - **Silhouette Coefficient**: compares average distance to point in current cluster with points in next closest cluster
2. **extrinsic** evaluation:
   - check how well clusters **align with known document labels**
   - typical evaluation measures:
     - **Homogeneity, Completeness and V-Measure**: effectively Precision, Recall and F-measure applied to clustering problems



Which clustering is "correct"?
Were there 3 or 4 clusters in the data?

Evaluation: silhouette

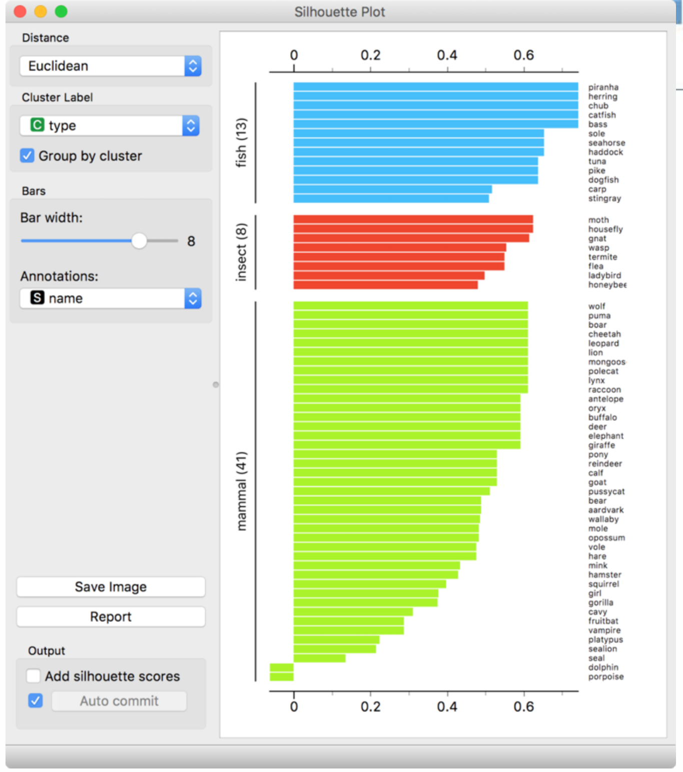
Silhouette coefficient

- score calculated **separately** for each sample
- uses distance metric (e.g., Euclidean)
- find average distance from sample x_i to all samples:
  - in *same cluster* (denote a_i)
  - in *next closest cluster* (denote b_i)
- compute coefficient as:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

  - ranges from −1 to +1 (really 0 to 1), higher = better
- average silhouette score over all datapoints to get single value representing quality of clustering

- often use silhouette plots to **visualise quality of clusters**
- issue: expensive to calculate, since done independently for each datapoint


Silhouette Plot

Distance: Euclidean

Cluster Label: type

☑ Group by cluster

Bars

Bar width: 8

Annotations: name

Save Image

Report

Output

☐ Add silhouette scores

☑ Auto commit

Silhouette-plot-orange:
https://commons.wikimedia.org/wiki/File:Silhouette-plot-orange.png

Evaluation: sum of squares

Very common to compute sum of squares values:

- *Within-cluster Sum of Squares*
  - sum over clusters the squared distances between cluster centroid and each point

$$\mathrm{WSS}(C) = \sum_{i=1}^{k} \sum_{x_j \in C_i} d(x_j, \mu_i)^2$$

- *Between-cluster Sum of Squares*
  - sum over clusters the squared distances between data centroid and each cluster centroid (weighted by data)

$$\mathrm{BSS}(C) = \sum_{i=1}^{k} |C_i| d(\mu, \mu_i)^2$$

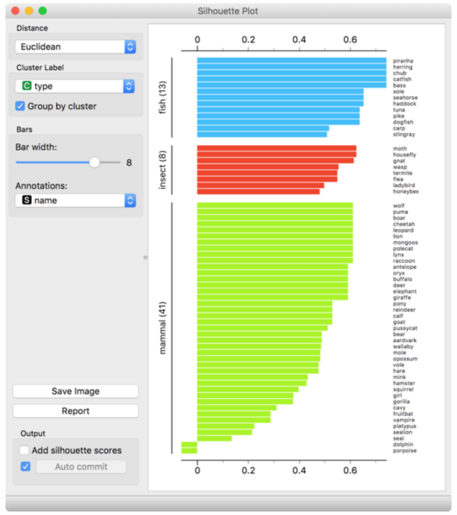
- Variance Ratio Criterion (aka Calinski-Harabasz index)
  - combines both measures into single metric
  - measures whether clusters are dense and well separated
  - faster to compute than Silhouette index, but really just objective function of *k*-Means

$$s = \frac{BSS(C)}{WSS(C)} \times \frac{N-k}{k-1}$$

Evaluating clustering – cont.

Intrinsic evaluation measures can be useful to:

- visualise cluster quality in *Silhouette plots*
- select a value for *k* using the *Elbow method*
  - plot within cluster sum of squares versus number of clusters
  - as k gets larger, intrinsic measure always improves
  - so don't look for best value, but rather biggest jump:
    - point where performance increases substantially, but then slows is called elbow


Silhouette Plot

Distance
Euclidean

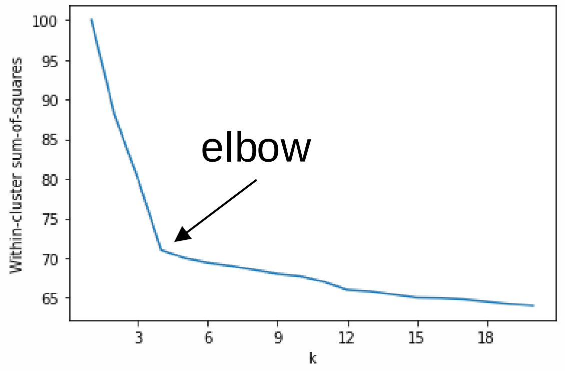
Cluster Label
C type
☑ Group by cluster

Bars
Bar width: 8

Annotations:
S name

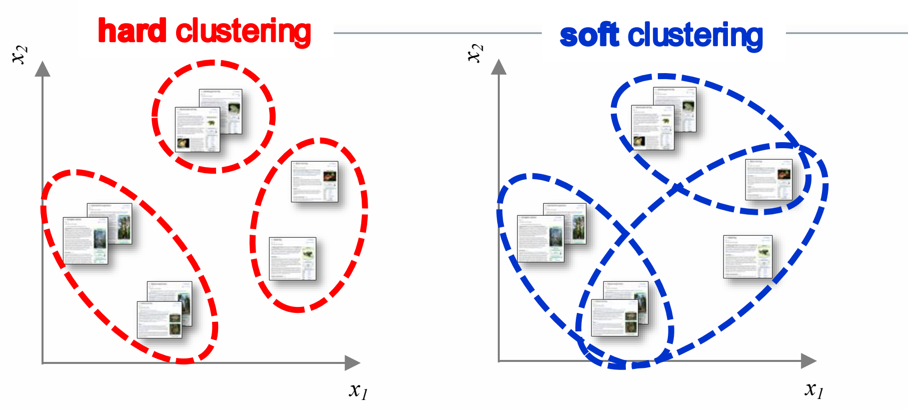
Output
☐ Add silhouette scores
☑ Auto commit

Silhouette-plot-orange: https://commons.wikimedia.org/wiki/File:Silhouette-plot-orange.png



Topic Modelling

What are topic models?



Soft clustering of documents

- allows documents to belong to multiple clusters

Provides **low-dimensional representation** of documents

- compared to high dimensional vocabulary space

Each cluster referred to as a "**topic**"

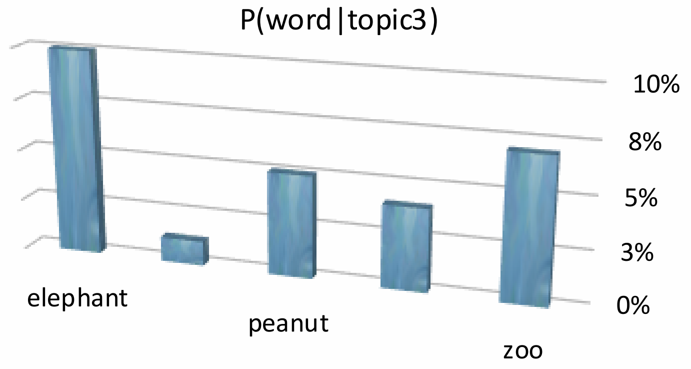
- described by distribution over words
- terms should be coherent (discuss single theme)

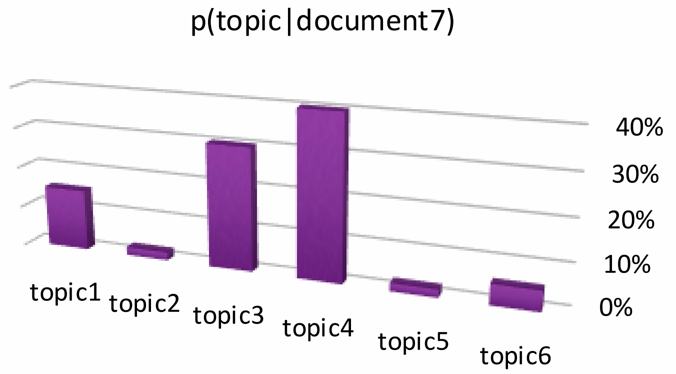
Topic vector = probability distribution over words

$$P(\texttt{elephant}|\texttt{topic3}) = 0.09$$

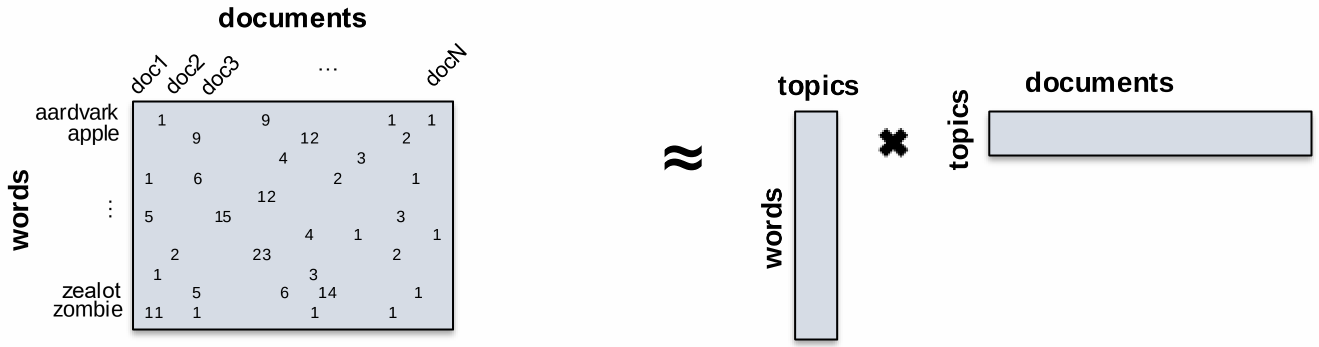
Document vector = probability distribution over topics

$$P(\texttt{topic3}|\texttt{doc7}) = 0.1$$





Matrix Decomposition



Topic Modeling is a form of matrix decomposition

- decomposes ***terms*documents*** count matrix
- into 2 smaller matrices: ***terms*topics*** and ***topics*documents***

Much smaller number of parameters need to be estimated: *V*T+T*D <<< V*D*

- for vocabulary size *V*, document count *D* and topic count *T*

Most famous technique is **Latent Dirichlet Allocation** (LDA)

- so named because it uses a Dirichlet prior when estimating the parameters
- related to Non-negative Matrix Factorization (NMF)
- also Latent Semantic Indexing (LSI), which applies Singular Value Decomposition (SVD) to TFIDF matrix

What is topic modelling useful for?

Factorization/clustering **generalizes** observed term counts:

- making document representations more dense & useful
- allows calculating **more meaningful distances between documents**

Deals with problems of:

- **polysemy**: bank (financial institution or edge of river?)
- **synonymy**: cancer = oncology ?
- **short documents**: limited vocabulary

Sometimes useful for **visualizing** collections

- after further dimensionality reduction (using say t-SNE)

Image source: https://www.flickr.com/photos/wwarby/4915969081

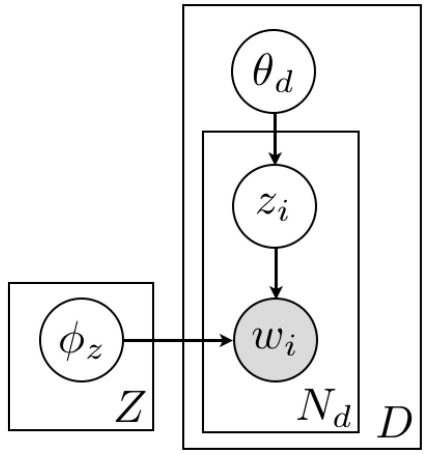
Details of LDA generative model

Generative model for latent Dirichlet Allocation (LDA)

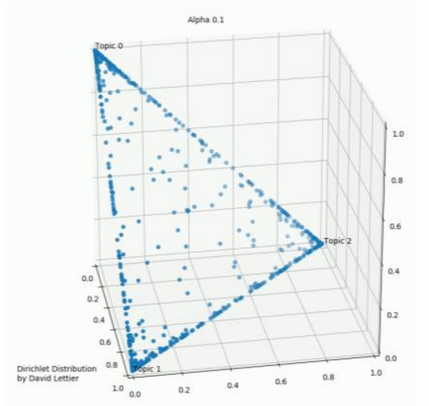
1. choose **word proportions** for each topic
2. choose **topic proportions** for each document
3. for each position in document
   1. choose **topic** based on document proportions
   2. choose **word** based on topic proportions

Estimating parameters for topic model:

- requires iteratively updating parameters
  1. topic probabilities for each document
  2. word probabilities for each topic
- place Bayesian priors on parameters to avoid over-fitting
- use (Gibbs) sampling techniques to avoid local maxima
- hyperparameters α and β determine the prior on the topic/document distributions (how concentrated they are)



$$\phi_z \sim Dirichlet(\beta)$$
$$\theta_d \sim Dirichlet(\alpha)$$
$$z_i \sim Discrete(\theta_d)$$
$$w_i \sim Discrete(\phi_{z_i})$$

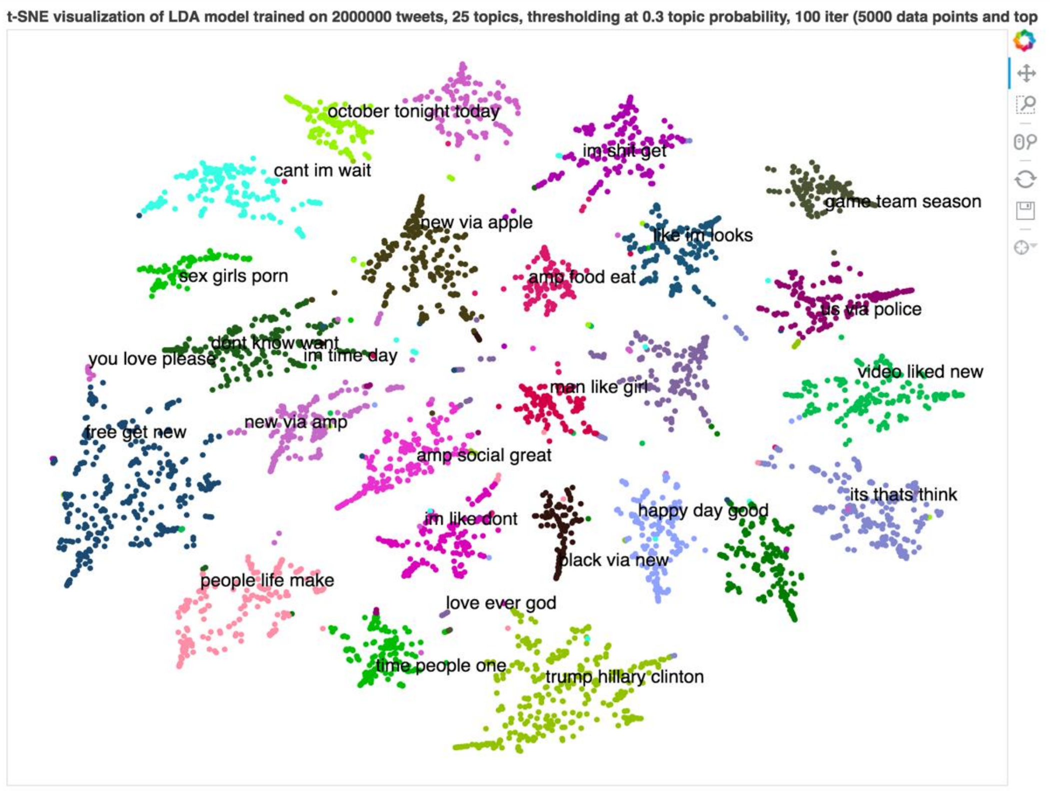


Source: David Lettier
https://medium.com/@lettier/how-does-lda-work-ill-explain-using-emoji-108abf40fa7d

Applications of Topic Models

Useful for visualising collections

- represent topics by most frequent terms
- common also to show changes to collection over time



Source: https://shuaiw.github.io/2016/12/22/topic-modeling-and-tsne-visualzation.html

Visualising document vectors: dimensionality reduction techniques

Can project high-dimensional data into lower dimensions using:

1. a **linear projection**

   with Principle Component Analysis

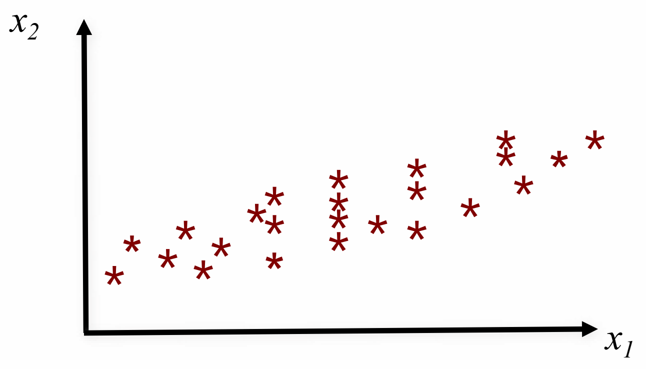
2. a **non-linear projection**

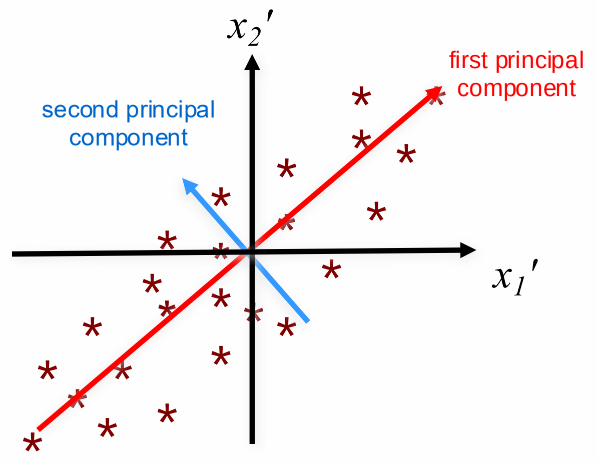
   e.g. using t-distributed Stochastic Neighbor Embeddings

Principal Component Analysis (PCA)

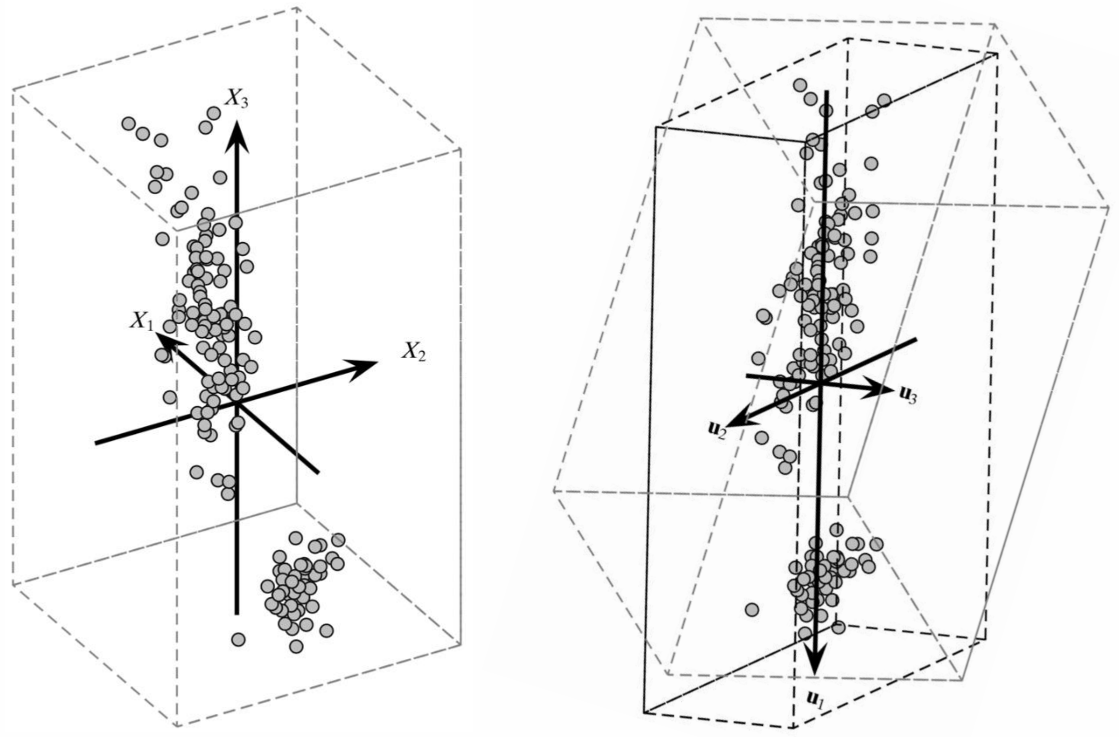
Investigates main sources of variation in dataset

- by finding **projection** that captures **largest amount of variation** in data
- method:
  1. scale features to have zero mean, unit variance
  2. find **direction** in **d**-dimensions that represent largest variation in data (*k*<*d*)
  3. remove variation from data
  4. repeat until *k* dimensions found
- use new dimensions to approximate original data using only largest components





PCA: example in 3d



For more information on PCA

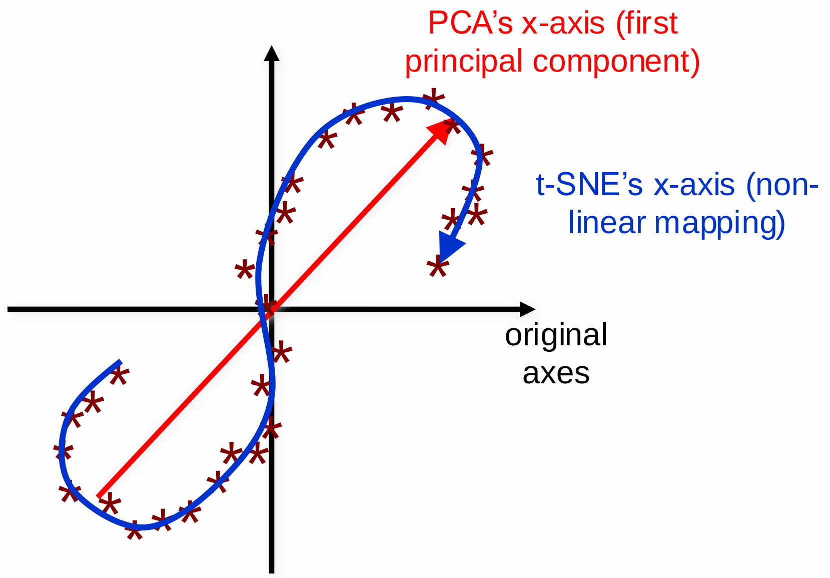
- see Chapter 7 of the textbook "Data Mining and Analysis"
- and: http://sebastianraschka.com/Articles/2015_pca_in_3_steps.html

t-distributed Stochastic Neighbor Embedding

Non-linear Dimensionality Reduction

Data in high dimensions never fills the entire space

- always lives on some lower-dimensional manifold

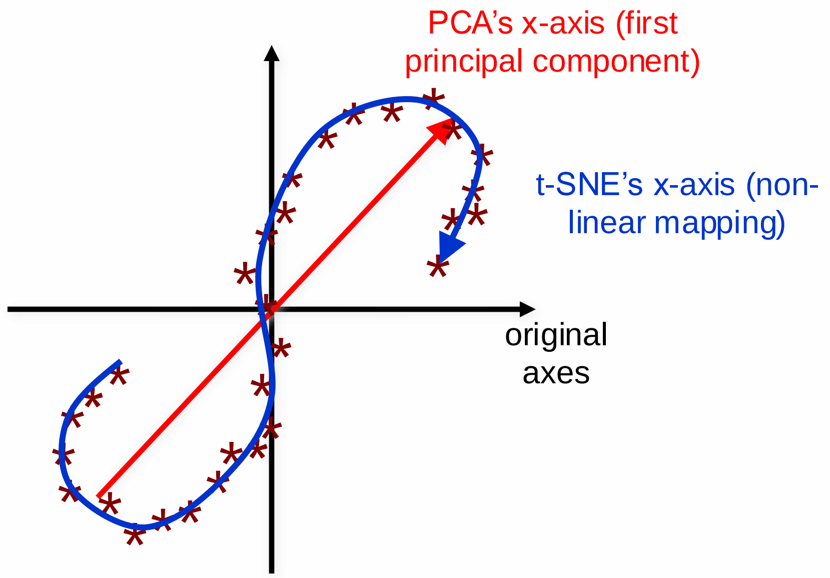

PCA's x-axis (first principal component)
t-SNE's x-axis (non-linear mapping)
original axes

t-SNE is a non-linear dimensionality reduction technique

- used to map high-dimensional data into 2 or 3 dimensions
  - points from original space mapped onto "map points" in 2D/3D
- unlike PCA:
  - mapped point is **not linear combination** of **original attribute values**
  - axis of mapped space is not linear combination (rotation) of original axes

41

t-distributed Stochastic Neighbor Embedding


PCA's x-axis (first principal component)
t-SNE's x-axis (non-linear mapping)
original axes

t-SNE:

- tries to preserve **local distances** to **nearby points**
  - unlike PCA which tries to preserve global (long range) distances between points
- converts distances between data points to joint probabilities
  - then maps original points to lower dimensional points such that position of mapped points **preserves structure of the data**
    - i.e. similar data points are assigned nearby map points
    - while dissimilar data points are assigned distant map points

t-SNE: algorithm

The t-SNE algorithm has two main steps:

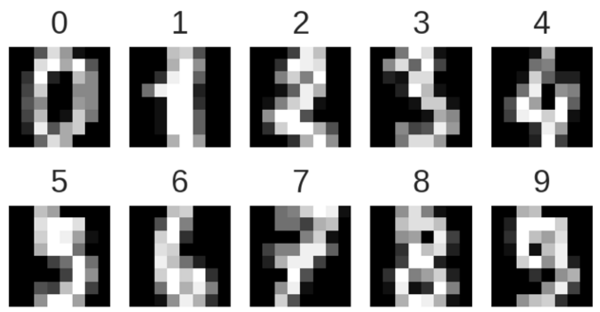
1. define **probability distribution over pairs** of high-dimensional points:
   - such that pairs of **similar data points have high probability** of being picked
   - and dissimilar points have extremely low probability of being picked
2. define a similar distribution over the points in the map space
   - and move around the mapped points to **minimize Kullback–Leibler divergence** between the two distributions
   - gradient descent is used to minimize the score

Caveats of t-SNE:

- **perplexity** parameter controls balance between local and global aspects of the data
- different initializations will lead to different results
- applies only to data with "reasonable" number of dimensions (e.g. 30-50)
  - if data has more dimensions, computational time will be prohibitive

t-SNE example: handwritten digit dataset

- Handwritten Digits Data Set from the UCI machine learning repository http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits
- Contains 1797 images with 8x8 pixels each



9

https://github.com/oreillymedia/t-SNE-tutorial

Conclusions

Conclusions

- Many applications of text clustering
- Many clustering techniques can be applied to tf-idf vectors
- Evaluating clustering can be subjective
- Topic modeling builds lower-dimensional representation of documents by performing soft clustering
- Dimensionality reduction techniques can be used to visualize document vectors