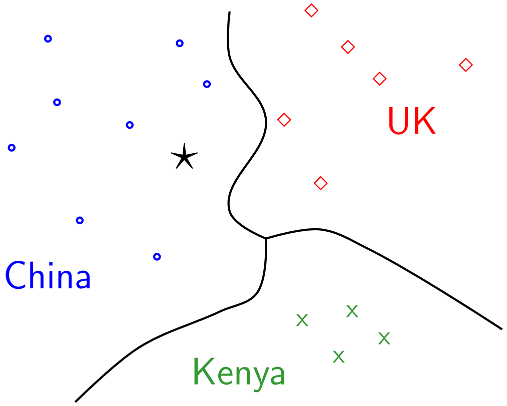
Recall vector space representation

- Each document is a vector, one component for each term.
- Terms are axes.
- High dimensionality: 100,000s of dimensions
- Normalize vectors (documents) to unit length
- How can we do classification in this space?

Vector space classification

- As before, the training set is a set of documents, each labeled with its class.
- In vector space classification, this set corresponds to a labeled set of points or vectors in the vector space.
- Premise 1: Documents in the same class form a contiguous region.
- Premise 2: Documents from different classes don't overlap.
- We define lines, surfaces, hypersurfaces to divide regions.

Classes in the vector space



Should the document $\star$ be assigned to *China*, *UK* or *Kenya*?
Find separators between the classes
Based on these separators: $\star$ should be assigned to *China*
How do we find separators that do a good job at classifying new documents like $\star$?

kNN classification

- kNN classification is another vector space classification method.
- It also is very simple and easy to implement.
- kNN is more accurate (in most cases) than Naive Bayes and Rocchio.
- If you need to get a pretty accurate classifier up and running in a short time . . .
- . . . and you don't care about efficiency that much . . .
- . . . use kNN.

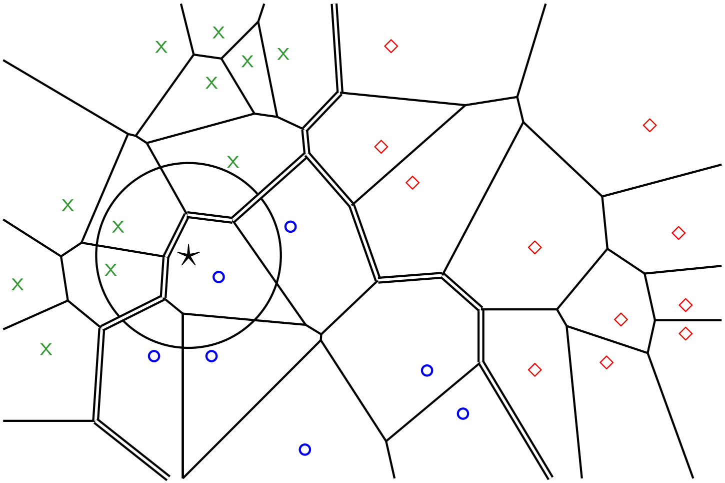
kNN classification

- kNN = k nearest neighbors
- kNN classification rule for $k = 1$ (1NN): Assign each test document to the class of its nearest neighbor in the training set.
- 1NN is not very robust – one document can be mislabeled or atypical.
- kNN classification rule for $k > 1$ (kNN): Assign each test document to the majority class of its k nearest neighbors in the training set.
- Rationale of kNN: contiguity hypothesis
    - We expect a test document d to have the same label as the training documents located in the local region surrounding d.

Probabilistic kNN

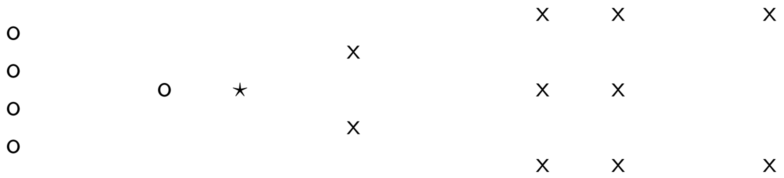
- Probabilistic version of kNN: $P(c|d) =$ fraction of k neighbors of d that are in c
- kNN classification rule for probabilistic kNN: Assign d to class c with highest $P(c|d)$

kNN is based on Voronoi tessellation



1NN, 3NN classifica- tion decision for star?

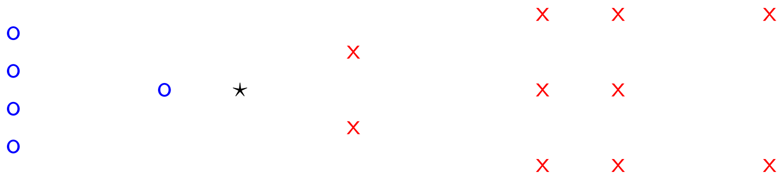
Exercise



How is star classified by:

(i) 1-NN (ii) 3-NN (iii) 9-NN (iv) 15-NN

Exercise



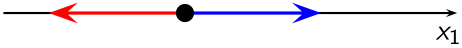
How is star classified by:

(i) 1-NN (ii) 3-NN (iii) 9-NN (iv) 15-NN

Linear classifiers

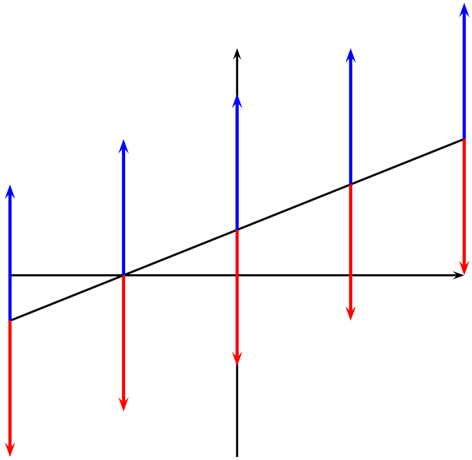
- Linear classifiers compute a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
- Classification decision: $\sum_i w_i x_i > \theta$?
- ...where θ (the threshold) is a parameter.
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities)
- Assumption: The classes are linearly separable.
- Can find hyperplane (=separator) based on training set
- Methods for finding separator: Perceptron, Rocchio, Naive Bayes – as we will explain on the next slides

A linear classifier in 1D



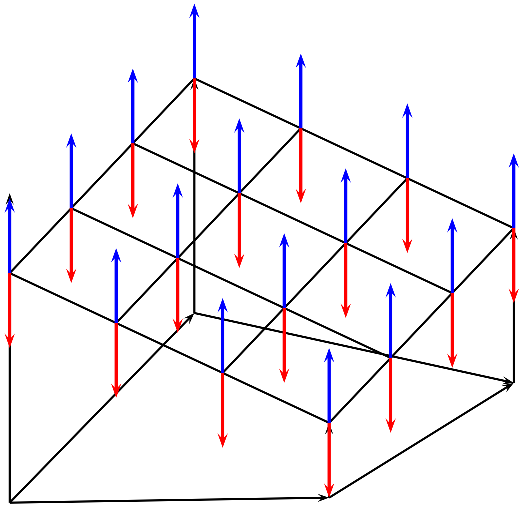
- A linear classifier in 1D is a point described by the equation $w_1 x_1 = \theta$
- The point at θ / w_1
- Points (x_1) with $w_1 x_1 \geq \theta$ are in the class c.
- Points (x_1) with $w_1 x_1 < \theta$ are in the complement class $\overline{c}$.

A linear classifier in 2D



- A linear classifier in 2D is a line described by the equation $w_1 x_1 + w_2 x_2 = \theta$
- Example for a 2D linear classifier
- Points $(x_1\ x_2)$ with $w_1 x_1 + w_2 x_2 \geq \theta$ are in the class c.
- Points $(x_1\ x_2)$ with $w_1 x_1 + w_2 x_2 < \theta$ are in the complement class $\overline{c}$.

A linear classifier in 3D



- A linear classifier in 3D is a plane described by the equation $w_1x_1 + w_2x_2 + w_3x_3 = \theta$
- Example for a 3D linear classifier
- Points $(x_1\ x_2\ x_3)$ with $w_1x_1 + w_2x_2 + w_3x_3 \geq \theta$ are in the class c.
- Points $(x_1\ x_2\ x_3)$ with $w_1x_1 + w_2x_2 + w_3x_3 < \theta$ are in the complement class $\overline{c}$.

Naive Bayes classifier

$\vec{x}$ represents document, what is $p(c|\vec{x})$ that document is in class c?

$$p(c|\vec{x}) = \frac{p(\vec{x}|c)p(c)}{p(\vec{x})} \qquad p(\bar{c}|\vec{x}) = \frac{p(\vec{x}|\bar{c})p(\bar{c})}{p(\vec{x})}$$

$$\text{odds}: \quad \frac{p(c|\vec{x})}{p(\bar{c}|\vec{x})} = \frac{p(\vec{x}|c)p(c)}{p(\vec{x}|\bar{c})p(\bar{c})} \approx \frac{p(c)}{p(\bar{c})} \frac{\prod_{1\le k\le n_d} p(t_k|c)}{\prod_{1\le k\le n_d} p(t_k|\bar{c})}$$

$$\text{log odds}: \quad \log \frac{p(c|\vec{x})}{p(\bar{c}|\vec{x})} = \log \frac{p(c)}{p(\bar{c})} + \sum_{1\le k\le n_d} \log \frac{p(t_k|c)}{p(t_k|\bar{c})}$$

Naive Bayes as a linear classifier

Naive Bayes is a linear classifier defined by:

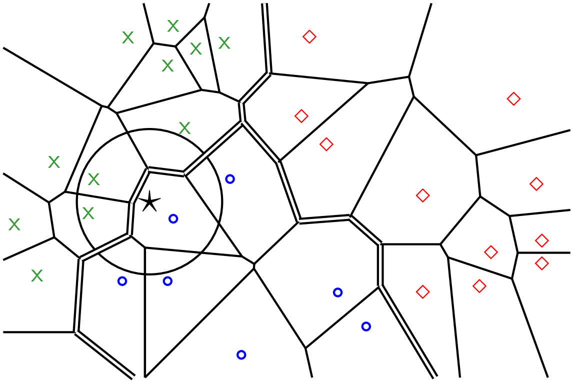
$$\sum_{i=1}^{M} w_i x_i = \theta$$

where $w_i = \log\big(p(t_i|c)/p(t_i|\bar{c})\big)$,
$x_i =$ number of occurrences of t_i in d,
and
$\theta = -\log\big(p(c)/p(\bar{c})\big)$.

(the index i, $1 \leq i \leq M$, refers to terms of the vocabulary)

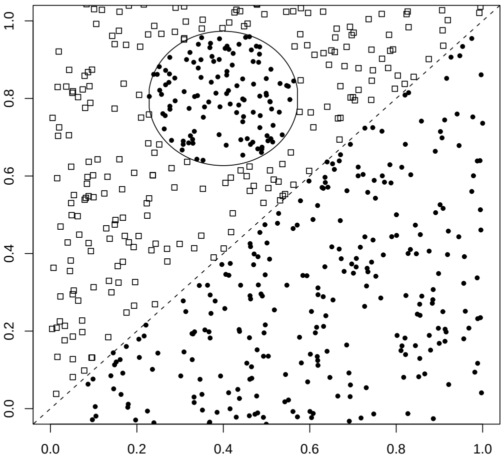
Linear in log space

kNN is not a linear classifier



- Classification decision based on majority of k nearest neighbors.
- The decision boundaries between classes are piecewise linear . . .
- . . . but they are not linear classifiers that can be described as $\sum_{i=1}^{M} w_i x_i = \theta$.

A nonlinear problem

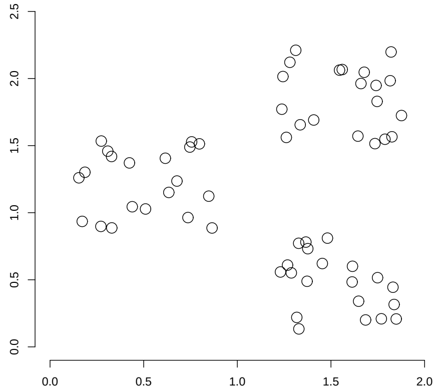


- Linear classifier does badly on this task.
- kNN will do well (assuming enough training data)

What is clustering?

- (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 the most common form of unsupervised learning.
- Unsupervised = there are no labeled or annotated data.

Data set with clear cluster structure



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.
  - However, there are many ways of influencing the outcome of clustering: number of clusters, similarity measure, representation of documents, . . .

The cluster hypothesis

Cluster hypothesis. Documents in the same cluster behave similarly with respect to relevance to information needs.

All applications in IR are based (directly or indirectly) on the cluster hypothesis.

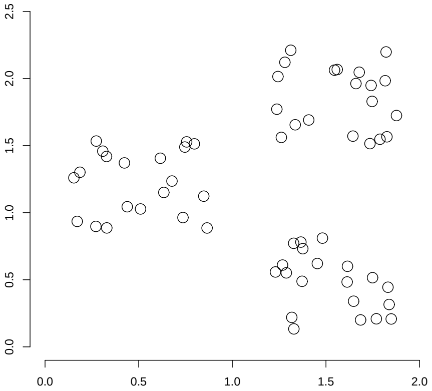
Global clustering for navigation: Google News

http://news.google.com

Clustering for improving recall

- To improve search recall:
  - Cluster docs in collection a priori
  - When a query matches a doc *d*, also return other docs in the cluster containing *d*
- Hope: if we do this: the query "car" will also return docs containing "automobile"
  - Because clustering groups together docs containing "car" with those containing "automobile".
  - Both types of documents contain words like "parts", "dealer", "mercedes", "road trip".

Data set with clear cluster structure



Exercise: Come up with an algorithm for finding the three clusters in this case

Document representations in clustering

- Vector space model
- As in vector space classification, we measure relatedness between vectors by Euclidean distance . . .
- . . . which is almost equivalent to cosine similarity.
- Almost: centroids are not length-normalized.
- For centroids, distance and cosine give different results.

Issues in clustering

- General goal: put related docs in the same cluster, put unrelated docs in different clusters.
  - But how do we formalize this?
- How many clusters?
  - Initially, we will assume the number of clusters K is given.
- Often: secondary goals in clustering
  - Example: avoid very small and very large clusters
- Flat vs. hierarchical clustering
- Hard vs. soft clustering

Flat vs. Hierarchical clustering

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

Hard vs. Soft clustering

- Hard clustering: Each document belongs to exactly one cluster.
  - More common and easier to do
- Soft clustering: A document can belong to more than one cluster.
  - Makes more sense for applications like creating browsable hierarchies
  - You may want to put a pair of sneakers in two clusters:
    - sports apparel
    - shoes
  - You can only do that with a soft clustering approach.

Flat 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 in K clusters that optimizes the chosen partitioning criterion
- Global optimization: exhaustively enumerate partitions, pick optimal one
  - Not tractable
- Effective heuristic method: K-means algorithm

K-means

- Perhaps the best known clustering algorithm
- Simple, works well in many cases
- Use as default / baseline for clustering documents

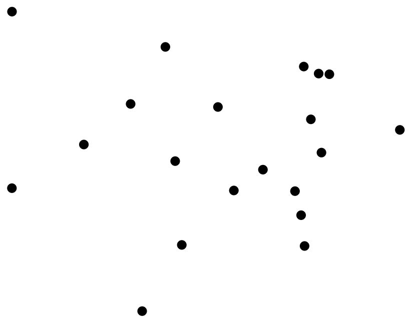
K-means

- Each cluster in K-means is defined by a centroid.
- Objective/partitioning criterion: minimize the average squared difference from the centroid
- Recall definition of centroid:

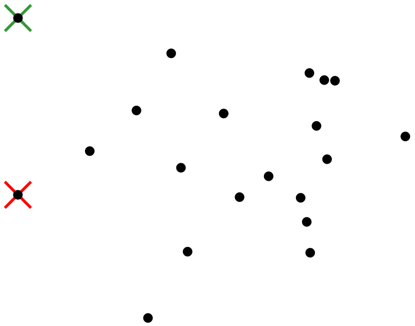
$$\vec{\mu}(\omega) = \frac{1}{|\omega|} \sum_{\vec{x} \in \omega} \vec{x}$$

  where we use ω to denote a cluster.
- We try to find the minimum average squared difference by iterating two steps:
    - reassignment: assign each vector to its closest centroid
    - recomputation: recompute each centroid as the average of the vectors that were assigned to it in reassignment

Set of points to be clustered

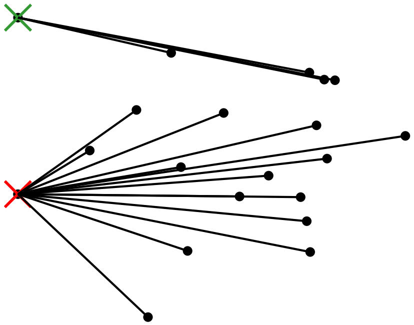


Random selection of initial cluster centers ($k = 2$ means)

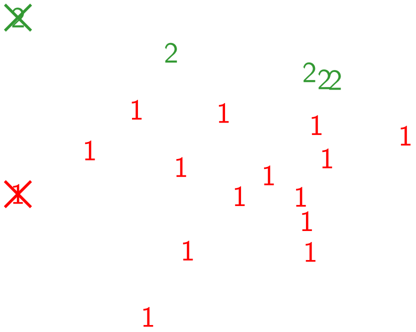


Centroids after convergence?

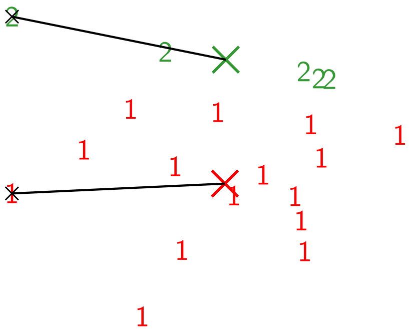
Assign points to closest centroid



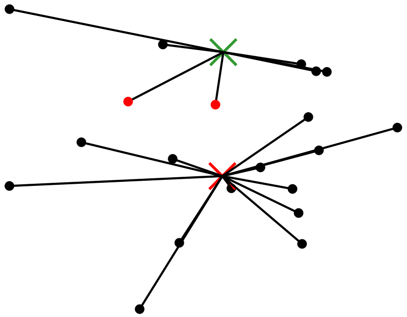
Assignment



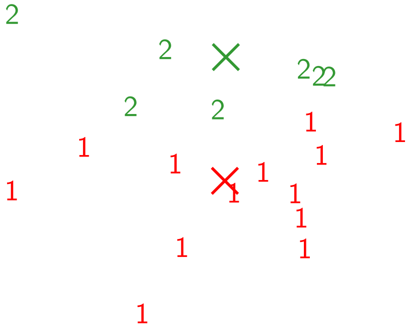
Recompute cluster centroids



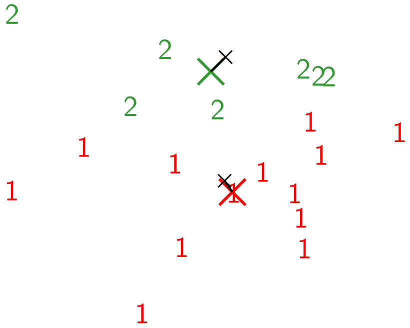
Assign points to closest centroid



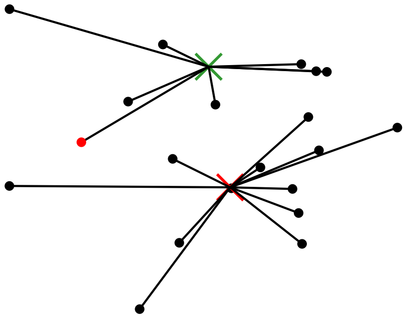
Assignment



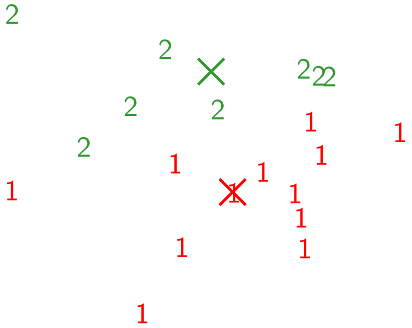
Recompute cluster centroids



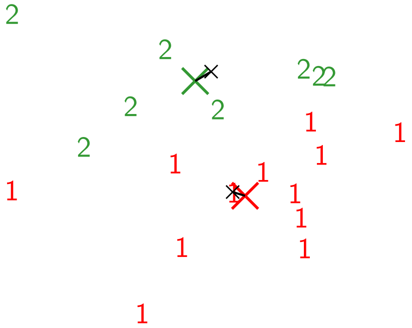
Assign points to closest centroid



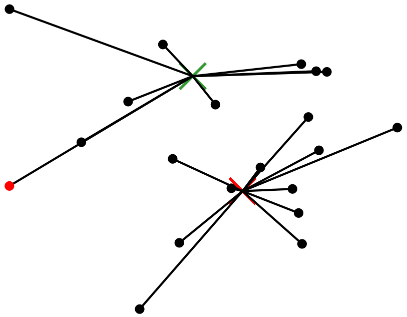
Assignment



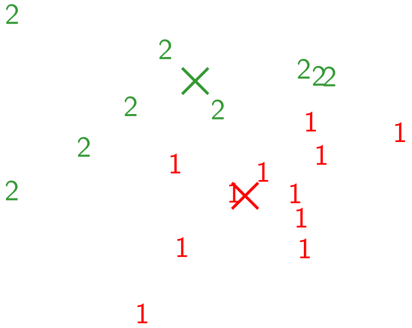
Recompute cluster centroids



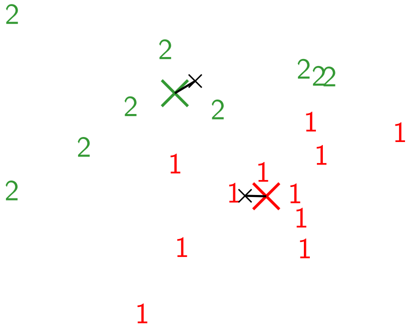
Assign points to closest centroid



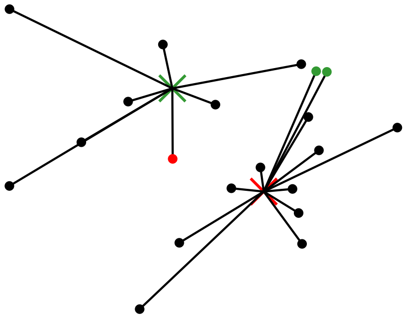
Assignment



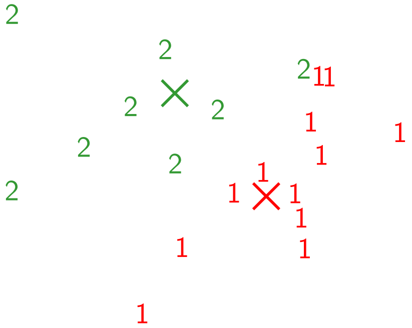
Recompute cluster centroids



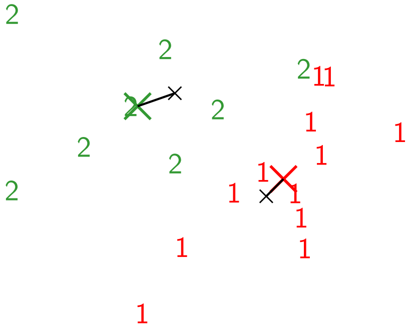
Assign points to closest centroid



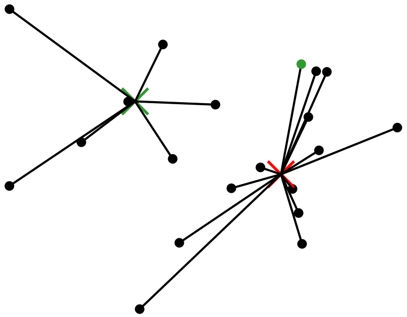
Assignment



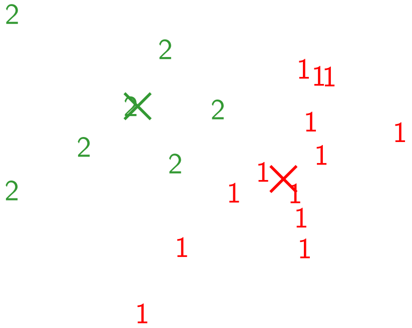
Recompute cluster centroids



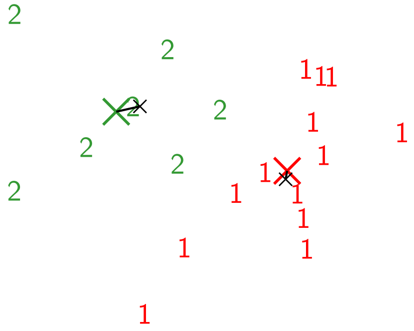
Assign points to closest centroid



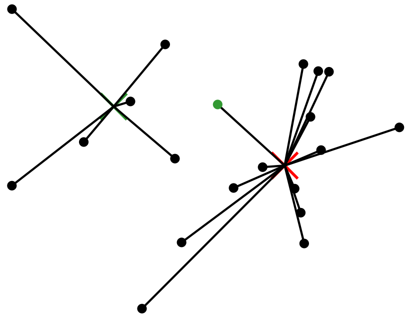
Assignment



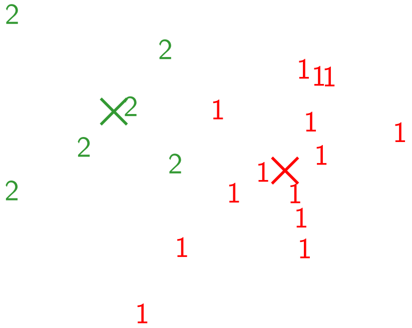
Recompute cluster centroids



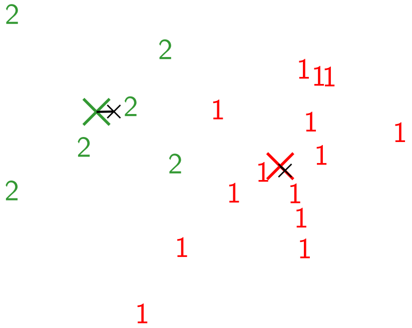
Assign points to closest centroid



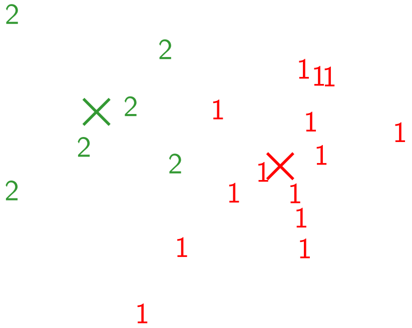
Assignment



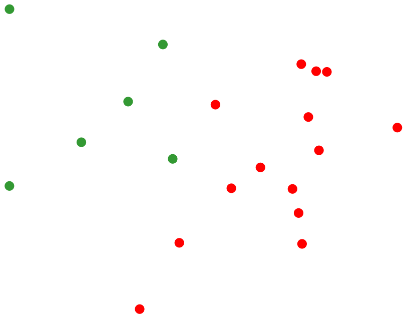
Recompute cluster centroids



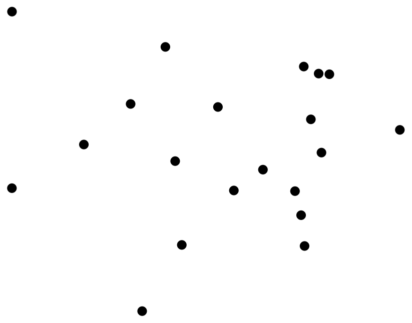
Centroids and assignments after convergence



Set of points clustered



Set of points to be clustered



K-means is guaranteed to converge

Proof:

- The sum of squared distances (RSS) decreases during reassignment, because each vector is moved to a closer centroid
(RSS = sum of all squared distances between document vectors and closest centroids)
- RSS decreases during recomputation (see next slide)
- There is only a finite number of clusterings.
- Thus: We must reach a fixed point.
(assume that ties are broken consistently)

Recomputation decreases average distance

$\text{RSS} = \sum_{k=1}^{K} \text{RSS}_k$ – the residual sum of squares (the "goodness" measure)

$$\begin{aligned} \text{RSS}_k(\vec{v}) &= \sum_{\vec{x} \in \omega_k} \|\vec{v} - \vec{x}\|^2 = \sum_{\vec{x} \in \omega_k} \sum_{m=1}^{M} (v_m - x_m)^2 \\ \frac{\partial \text{RSS}_k(\vec{v})}{\partial v_m} &= \sum_{\vec{x} \in \omega_k} 2(v_m - x_m) = 0 \end{aligned}$$

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

The last line is the componentwise definition of the centroid!
We minimize RSS_k when the old centroid is replaced with the new centroid.
RSS, the sum of the RSS_k, must then also decrease during recomputation.

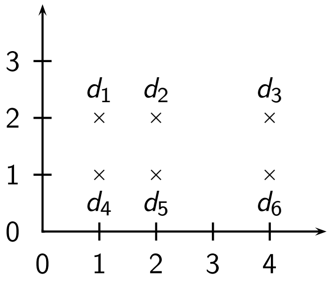
K-means is guaranteed to converge

- 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 ($<$ 10-20 iterations).
- However, complete convergence can take many more iterations.

Optimality of K-means

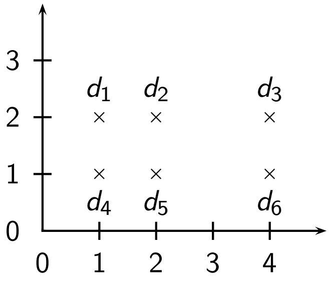
- Convergence does not mean that we converge to the optimal clustering!
- This is the great weakness of K-means.
- If we start with a bad set of seeds, the resulting clustering can be horrible.

Exercise: Suboptimal clustering



- What is the optimal clustering for $K = 2$?
- Do we converge on this clustering for arbitrary seeds d_{i_1}, d_{i_2}?

Exercise: Suboptimal clustering



- What is the optimal clustering for $K = 2$?
- Do we converge on this clustering for arbitrary seeds d_{i_1}, d_{i_2}?

For seeds d_2 and d_5, K-means converges to $\{\{d_1, d_2, d_3\}, \{d_4, d_5, d_6\}\}$ (suboptimal clustering).

For seeds d_2 and d_3, instead converges to $\{\{d_1, d_2, d_4, d_5\}, \{d_3, d_6\}\}$ (global optimum for $K = 2$).

k-means clustering, redux

Goal

- cluster similar data points

Approach:

given data points and distance function

- select k centroids $\vec{\mu}_a$
- assign $\vec{x}_i$ to closest centroid $\vec{\mu}_a$
- minimize $\sum_{a,i} d(\vec{x}_i, \vec{\mu}_a)$

Algorithm:

- randomly pick centroids, possibly from data points
- assign points to closest centroid
- average assigned points to obtain new centroids
- repeat 2,3 until nothing changes

Issues:

- - takes superpolynomial time on some inputs
- - not guaranteed to find optimal solution
- $+$ converges quickly in practice

Initialization of K-means

- Random seed selection is just one of many ways K-means can be initialized.
- Random seed selection is not very robust: It's easy to get a suboptimal clustering.
- Better heuristics:
  - Select seeds not randomly, but using some heuristic (e.g., filter out 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 sets of seeds, do a K-means clustering for each, select the clustering with lowest RSS

How many clusters?

- Either: Number of clusters K is given.
  - Then partition into K clusters
  - K might be given because there is some external constraint. Example: it was hard to show more than 10–20 clusters on a monitor in the 90s.
- Or: Finding the “right” number of clusters is part of the problem.
  - Given docs, find K for which an optimum is reached.
  - How to define “optimum”?
  - We can’t use RSS or average squared distance from centroid as criterion: always chooses $K = N$ clusters.

Exercise

- Suppose we want to analyze the set of all articles published by a major newspaper (e.g., New York Times or Süddeutsche Zeitung) in 2008.
- Goal: write a two-page report about what the major news stories in 2008 were.
- We want to use K-means clustering to find the major news stories.
- How would you determine K?

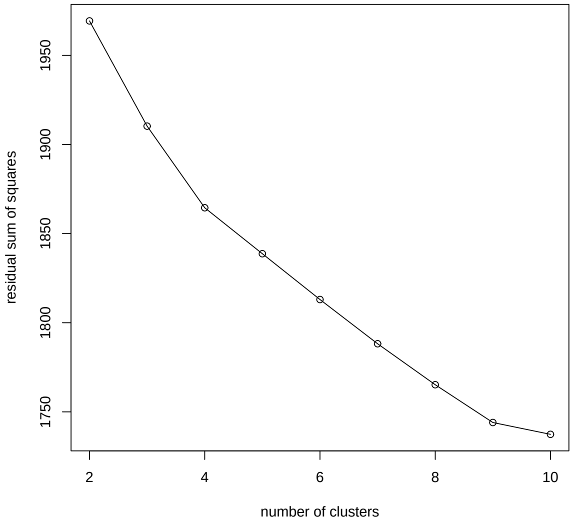
Simple objective function for K (1)

- Basic idea:
  - Start with 1 cluster ($K = 1$)
  - Keep adding clusters ($=$ keep increasing K)
  - Add a penalty for each new cluster
- Trade off cluster penalties against average squared distance from centroid
- Choose K with best tradeoff

Simple objective function for K (2)

- Given a clustering, define the cost for a document as (squared) distance to centroid
- Define total distortion RSS(K) as sum of all individual document costs (corresponds to average distance)
- Then: penalize each cluster with a cost λ
- Thus for a clustering with K clusters, total cluster penalty is $K\lambda$
- Define the total cost of a clustering as distortion plus total cluster penalty: $\text{RSS(K)} + K\lambda$
- Select K that minimizes $(\text{RSS(K)} + K\lambda)$
- Still need to determine good value for $\lambda \ldots$

Finding the “knee” in the curve



Pick the number of clusters where curve “flattens”. Here: 4 or 9.

What is a good clustering?

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

Major issue in clustering – labeling

- After a clustering algorithm finds a set of clusters: how can they be useful to the end user?
- We need a pithy label for each cluster.
- For example, in search result clustering for “jaguar”, The labels of the three clusters could be “animal”, “car”, and “operating system”.
- How can we automatically find good labels for clusters?

Exercise

- Come up with an algorithm for labeling clusters
- Input: a set of documents, partitioned into K clusters (flat clustering)
- Output: A label for each cluster
- Part of the exercise: What types of labels should we consider? Words?

Feature selection

- In text classification, we usually represent documents in a high-dimensional space, with each dimension corresponding to a term.
- In this lecture: axis = dimension = word = term = feature
- Many dimensions correspond to rare words.
- Rare words can mislead the classifier.
- Rare misleading features are called noise features.
- Eliminating noise features from the representation increases efficiency and effectiveness of text classification.
- Eliminating features is called feature selection.