Introduction to

Information Retrieval

Hinrich Schütze and Christina Lioma

Chapter 16: Flat Clustering

(Slides modified by Scott Sanner)

Overview

Outline

Recap

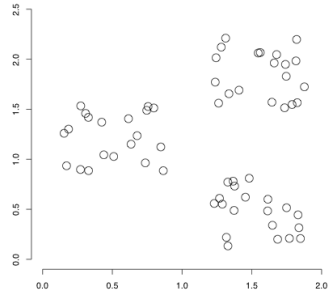
- Up until now…
  - We have built document classifiers given labeled data
  - Classification known as supervised learning
- Today…
  - What if we don't have labeled data?
    - We don't know the class labels?
  - Can we still assign reasonable labels?
    - We could try to cluster documents (cluster=class)
      - What would make a human readable label?
    - Clustering known as unsupervised learning

Outline

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

Data set with clear cluster structure



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

Outline

The cluster hypothesis

Cluster hypothesis. Documents in the same cluster behave similarly with respect to relevance to information needs. All applications of clustering in IR are based (directly or indirectly) on the cluster hypothesis.

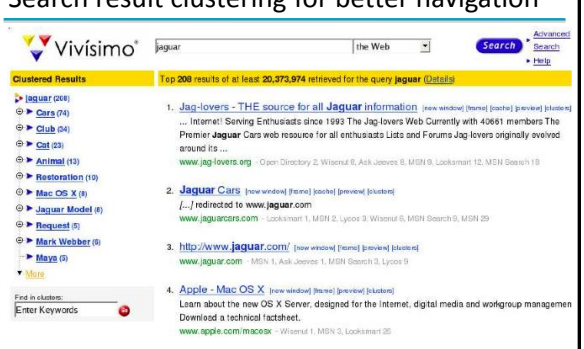
Van Rijsbergen's original wording: "closely associated documents tend to be relevant to the same requests".

Applications of clustering in IR

deduplication
source clustering
diversification

Application	What is clustered?	Benefit
Search result clustering	search results	more effective information presentation to user
Scatter-Gather	(subsets of) collection	alternative user interface: "search without typing"
Collection clustering	collection	effective information presentation for exploratory browsing
Cluster-based retrieval	collection	higher efficiency: faster search

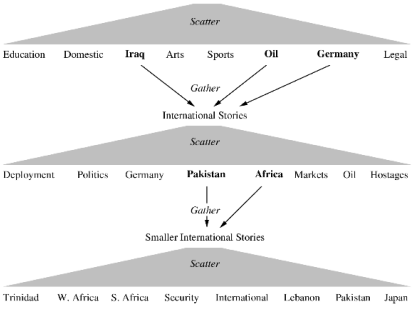
Search result clustering for better navigation



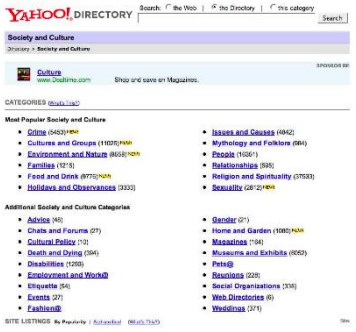
Clustering for better navigation

- Ambiguity like Jaguar a bad example
- Better case of ambiguity: London Heathrow
  - What is my intent?
    - Checkin
    - Ground transport
    - Food availability
    - Security

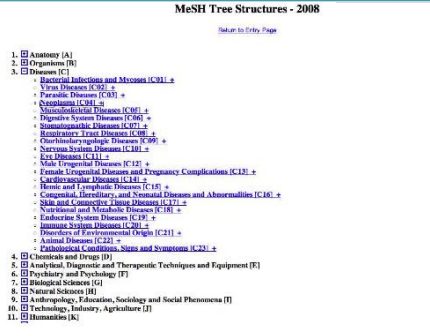
Scatter-Gather



Global navigation: Yahoo



Global navigation: MESH (upper level)



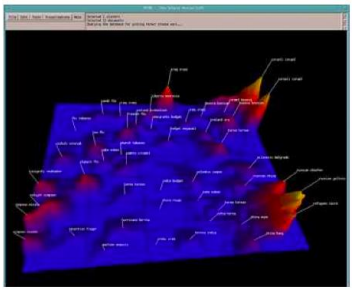
Global navigation: MESH (lower level)

Navigational hierarchies: Manual vs. automatic creation

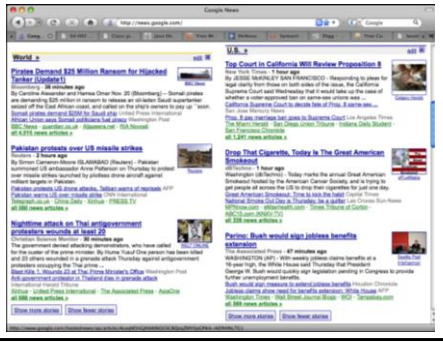
- Note: Yahoo/MESH are not examples of clustering.
- But they are well known examples for using a global hierarchy for navigation.
- Some examples for global navigation/exploration based on clustering:
  - Cartia
  - Themescapes
  - Google News

Global navigation combined with visualization (1)

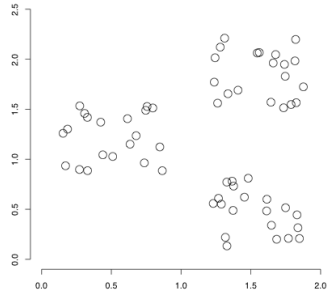
Global navigation combined with visualization (2)



Google News: automatic clustering gives an effective news presentation metaphor



Data set with clear cluster structure



Propose algorithm for finding the cluster structure in this example

Desiderata for clustering

- General goal: put related docs in the same cluster, put unrelated docs in different clusters.
  - How do we formalize 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.
  - Later: Semiautomatic methods for determining *K*
- Secondary goals in clustering
  - Avoid very small and very large clusters
  - Define clusters that are easy to explain to the user
  - Many others . . .

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 in exactly one cluster.
  - More common and easier to do
- Soft clustering: A document can be in more than one cluster.
  - Makes more sense for browsable hierarchies
  - You may want to put sneakers in two clusters:
    - sports apparel
    - shoes
  - You can only do that with a soft clustering approach.
- We will do flat, hard clustering only in this class.
- See IIR 16.5, IIR 17, IIR 18 for soft clustering and hierarchical clustering

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

Outline

K-means

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

Document representations in clustering

- Vector space model
- K-means can use any distance metric
  - Intuition often given via Euclidean distance (i.e., this lecture)
  - But for documents you want to use (1 – cosine similarity)
  - Why?

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:

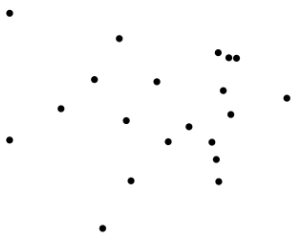
$$\vec{\mu}(\omega) = \frac{1}{|\omega|} \sum_{\vec{x} \in \omega} \vec{x}$$

  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

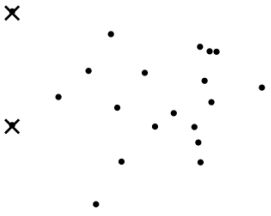
K-means algorithm

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** $\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$

Worked Example: Set of to be clustered

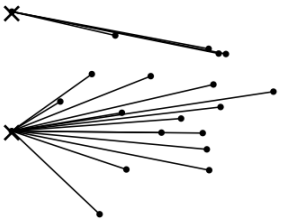


Worked Example: Random selection of initial centroids

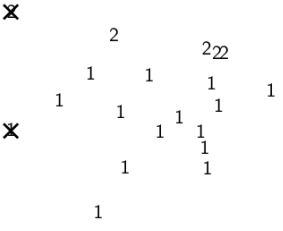


Exercise: (i) Guess what the optimal clustering into two clusters is in this case; (ii) compute the centroids of the clusters

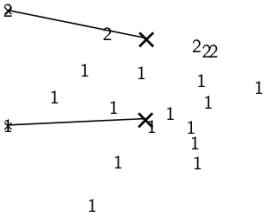
Worked Example: Assign points to closest center



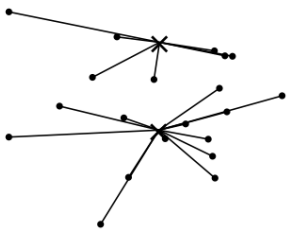
Worked Example: Assignment



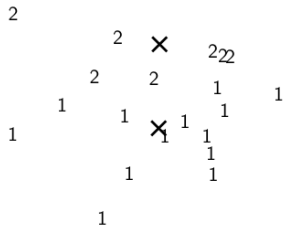
Worked Example: Recompute cluster centroids



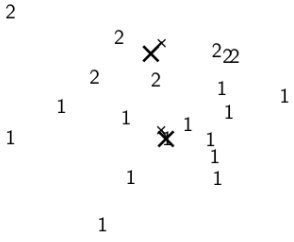
Worked Example: Assign points to closest centroid



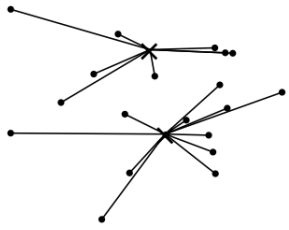
Worked Example: Assignment



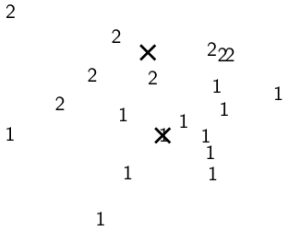
Worked Example: Recompute cluster centroids



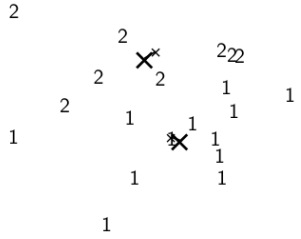
Worked Example: Assign points to closest centroid



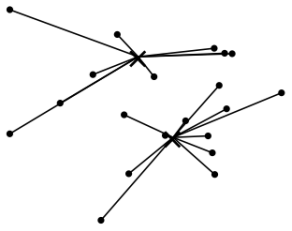
Worked Example: Assignment



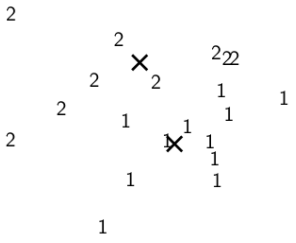
Worked Example: Recompute cluster centroids



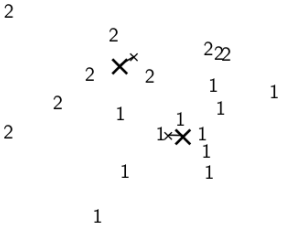
Worked Example: Assign points to closest centroid



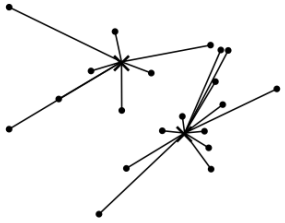
Worked Example: Assignment



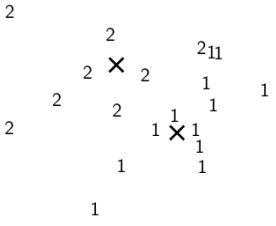
Worked Example: Recompute cluster centroids



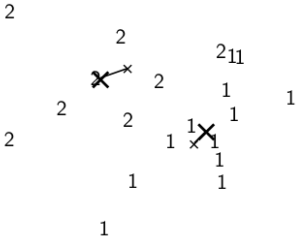
Worked Example: Assign points to closest centroid



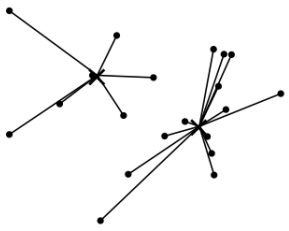
Worked Example: Assignment



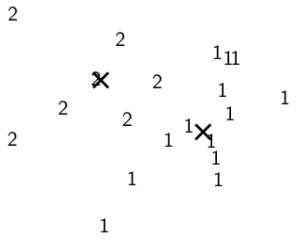
Worked Example: Recompute cluster centroids



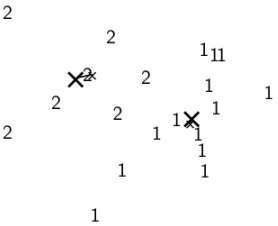
Worked Example: Assign points to closest centroid



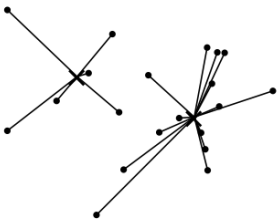
Worked Example: Assignment



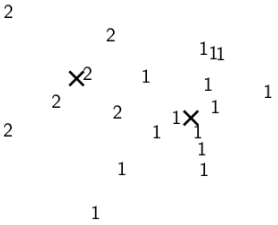
Worked Example: Recompute cluster centroids



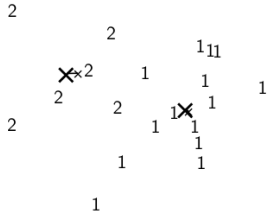
Worked Example: Assign points to closest centroid



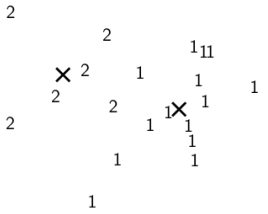
Worked Example: Assignment



Worked Example: Recompute cluster caentroids



Worked Ex.: Centroids and assignments after convergence



K-means is guaranteed to converge: Proof

- RSS = sum of all squared distances between document vector and closest centroid
- RSS decreases during each reassignment step.
  - because each vector is moved to a closer centroid
- RSS decreases during each recomputation step.
  - see next slide
- There are only a finite number of clusterings.
- Thus: We must reach a fixed point.
- Assumption: 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)

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

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

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

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.

Mean also reduces RSS for cosine distance – spherical K-means!

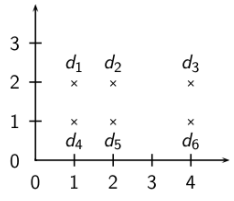
K-means is guaranteed to converge

- 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.

Convergence Exercise: Suboptimal clustering



- What is the optimal clustering for $K = 2$?
- Do we converge on this clustering for arbitrary seeds d_i, d_j?

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 ways of computing initial centroids:
  - 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 random sets of seeds, do a *K*-means clustering for each, select the clustering with lowest RSS

Time complexity of *K*-means

- Computing one distance of two vectors is $O(M)$.
- Reassignment step: $O(KNM)$ (we need to compute KN document-centroid distances)
- Recomputation step: $O(NM)$ (we need to add each of the document's < M values to one of the centroids)
- Assume number of iterations bounded by I
- Overall complexity: $O(IKNM)$ – linear in all important dimensions
- However: This is not a real worst-case analysis.
- In pathological cases, complexity can be worse than linear.

Outline

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

External criteria for clustering quality

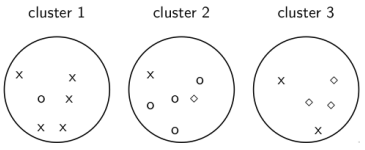
- Based on a gold standard data set, e.g., the Reuters collection we also used for the evaluation of classification
- Goal: Clustering should reproduce the classes in the gold standard
- (But we only want to reproduce how documents are divided into groups, not the class labels.)
- First measure for how well we were able to reproduce the classes: purity

External criterion: Purity

$$\text{purity}(\Omega, C) = \frac{1}{N}\sum_k \max_j |\omega_k \cap c_j|$$

- $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ is the set of clusters and $C = \{c_1, c_2, \ldots, c_J\}$ is the set of classes.
- For each cluster ω_k : find class c_j with most members n_{kj} in ω_k
- Sum all n_{kj} and divide by total number of 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); and 3 = $\max_j |\omega_3 \cap c_j|$ (class ⋄, cluster 3). Purity is (1/17) × (5 + 4 + 3) ≈ 0.71.

Rand index

- Definition: $\mathrm{RI} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{FN}+\mathrm{TN}}$
- Based on 2x2 contingency table of all pairs of documents:

	same cluster	different clusters
same class	true positives (TP)	false negatives (FN)
different classes	false positives (FP)	true negatives (TN)

- TP+FN+FP+TN is the total number of pairs.
- There are $\binom{N}{2}$ pairs for N documents.
- Example: $\binom{17}{2}$ = 136 in o/⋄/x example
- Each pair is either positive or negative (the clustering puts the two documents in the same or in different clusters) . . .
- . . . and either "true" (correct) or "false" (incorrect): the clustering decision is correct or incorrect.

Rand Index: Example

As an example, we compute RI for the o/⋄/x example. We first compute TP + FP. The three clusters contain 6, 6, and 5 points, respectively, so the total number of "positives" or pairs of documents that are in the same cluster is:

$$\mathrm{TP}+\mathrm{FP} = \binom{6}{2} + \binom{6}{2} + \binom{5}{2} = 40$$

Of these, the x pairs in cluster 1, the o pairs in cluster 2, the ⋄ pairs in cluster 3, and the x pair in cluster 3 are true positives:

$$\mathrm{TP} = \binom{5}{2} + \binom{4}{2} + \binom{3}{2} + \binom{2}{2} = 20$$

Thus, FP = 40 − 20 = 20. FN and TN are computed similarly.

Rand measure for the o/⋄/x example

	same cluster	different clusters
same class	TP = 20	FN = 24
different classes	FP = 20	TN = 72

RI is then

(20 + 72)/(20 + 20 + 24 + 72) ≈ 0.68.

Two other external evaluation measures

- Two other measures
- Normalized mutual information (NMI)
  - How much information does the clustering contain about the classification?
  - Singleton clusters (number of clusters = number of docs) have maximum MI
  - Therefore: normalize by entropy of clusters and classes
- F measure
  - Like Rand, but "precision" and "recall" can be weighted

Evaluation results for the o/⋄/x example

	purity	NMI	RI	F_5
lower bound	0.0	0.0	0.0	0.0
maximum	1.0	1.0	1.0	1.0
value for example	0.71	0.36	0.68	0.46

All four measures range from 0 (really bad clustering) to 1 (perfect clustering).

Internal criteria: PMI

- For **text documents**, coherency of top cluster unigrams can be a measure of coherency
- How to measure coherency without labels?
  - Compute average pointwise mutual informatin (PMI) of top-10 unigrams in each cluster (averaged over clusters)

$$\text{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}.$$

  - Probabilities p(.) computed from background sources (e.g., Wikipedia)

Outline

How many clusters?

- Number of clusters *K* is given in many applications.
  - E.g., there may be an external constraint on *K*. Example: In the case of Scatter-Gather, it was hard to show more than 10–20 clusters on a monitor in the 90s.
- What if there is no external constraint? Is there a "right" number of clusters?
- One way to go: define an optimization criterion
  - Given docs, find *K* for which the optimum is reached.
  - What optimiation criterion can we use?
  - We can't use RSS or average squared distance from centroid as criterion: always chooses *K* = *N* clusters.

Exercise

- Your job is to develop the clustering algorithms for a competitor to news.google.com
- You want to use *K*-means clustering.
- How would you determine *K*?

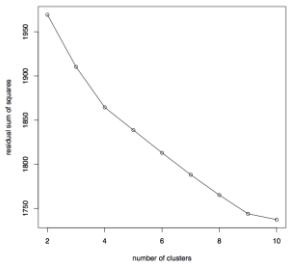
Simple objective function for *K* (1)

- 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 the value of *K* with the 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 clustering with *K* clusters, total cluster penalty is *K*λ
- Define the total cost of a clustering as distortion plus total cluster penalty: RSS(K) + *K*λ
- Select K that minimizes (RSS(K) + *K*λ)
- Still need to determine good value for λ . . .

Finding the "knee" in the curve



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

Take-away today

- What is clustering?
- Applications of clustering
- *K*-means algorithm
- Evaluation of clustering
- How many clusters?

Resources

- Chapter 16 of IIR
- Resources at http://ifnlp.org/ir
  - *K*-means example
  - Keith van Rijsbergen on the cluster hypothesis (he was one of the originators)
  - Bing/Carrot2/Clusty: search result clustering