Information Retrieval
WS 2012 / 2013

Lecture 9, Wednesday December 19th, 2012
(Clustering, k-means)

Prof. Dr. Hannah Bast
Chair of Algorithms and Data Structures
Department of Computer Science
University of Freiburg

Overview of this lecture

- Organizational
  - Your results + experiences with Exercise Sheet 8 (LSI)
  - Demo of probabilistic LSI (PLSI)
  - Demo of some of your fancy web applications
  - Date for the exam: **Friday, March 1, 2:00 – 3:30 pm**
- Clustering
  - The k-means algorithm: demo, convergence, complexity, ...
  - Particularities for document clustering
  - Exercise Sheet 9: cluster our example collection using k-means

Experiences with ES#8 (LSI)

- Summary / excerpts last checked December 19, 16:10
  - Could be done in reasonable time for most ... yay !
  - Not more work than this for future sheets please ... ok
  - No major problems, except for some fights with Octave

    e.g., string arrays and sorting not too comfy in Octave
  - Octave is slow ... yes, it's a scripting language
  - Linear algebra: nice!!! ... I couldn't agree more
  - Lecture was interesting, but a bit "freaky"
  - The math stuff should be explained better in the lecture
  - Why most frequent terms, not terms with highest scores?

Your results for ES#8 (LSI)

- Most of you got meaningful results
  - For **k = 10**, mostly pairs of frequent terms

    she – her, und – der, die – der, paris – french, ...
  - For **k = 50**, many "inflection pairs"

    australian – australia, chemist – chemistry, soviet – russian, ...
  - For **k = 100**, similar relations, less inflection pairs

    indian – india, berkeley – california, vol – pp, nobel – prize, ...
  - For **k = 500**, similar relations, more "phrase pairs"

    new – york, grew – up, middle – east, soviet – union, ...
  - Bottom line: it's magic, how this comes out of linear algebra

    ... but then again, the results aren't really that useful

Results for ES#6+7 (web apps)

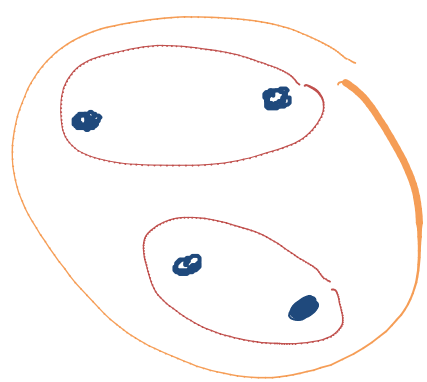
- Let's finally look at some of your fancy web apps
  - Many were special in some or the other aspect
    - suggestions also for multiple keywords
    - result snippets
    - highlighting of query words
    - show more / less
    - super fast
    - particularly colorful
  - Please don't be disappointed if your web app is not shown, I had to make a selection !

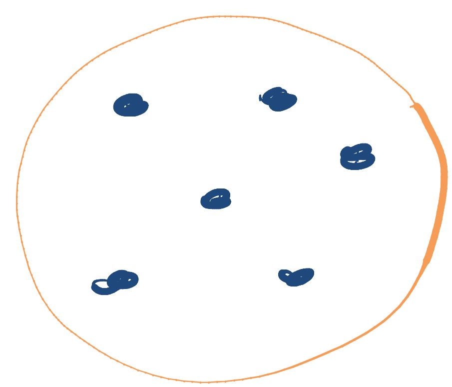
Clustering

- General (informal) definition
  - Given n elements from a metric space = there is a measure of distance between any two elements
  - Group the elements in clusters such that

    Intra-cluster distances are as small as possible

    Inter-cluster distances are as large as possible
  - Note: many ways to make this precise + it depends on the application what is a good clustering and what not





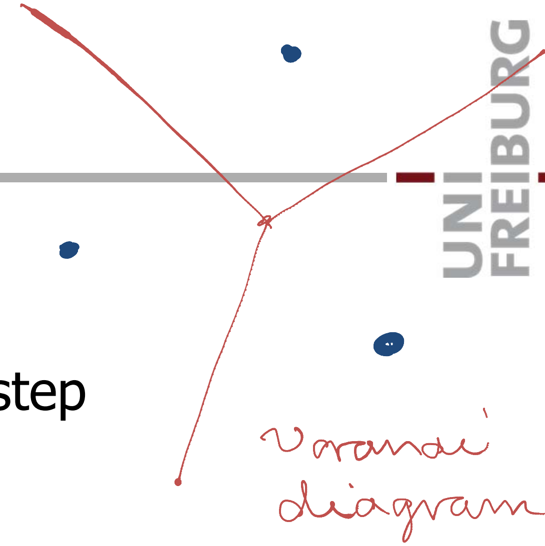
K-Means 1/12

- Setting / Terminology
  - Number of clusters k is given as part of the input
  - Each cluster C_i has a so-called **centroid** μ_i, which is an element from the metric space, but not necessarily (and also not typically) an element from the input set
  - For a given clustering $C_1, \ldots, C_k$ with cluster centroids $\mu_1, \ldots, \mu_k$ define the **residual sum of squares** as

    $$\mathbf{RSS} = \sum_{i=1,\ldots,k} \sum_{x \in C_i} |x - \mu_i|^2$$

    That is, just sum up the squares of all intra-cluster dists
  - The goal of k-means is to minimize the RSS

K-Means 2/12



- Algorithm
  - **Idea:** greedily minimize the RSS in every step
  - Initialization: pick a set of centroids

    for example, k random elements from the input set
  - Then alternate between the following two steps

    (A) Assign each element to its nearest centroid

    this can only decrease the RSS ... next slide

    (B) compute new centroids as average of elems assigned to it

    this too can only decrease the RSS ... next slide
  - Let's look at a demo ...

K-Means 3/12

- Proof of optimality of (A) and (B)

$$RSS = \sum_{i=1}^{k} \sum_{x \in C_i} (x - \mu_i)^2$$

(A) assign each el. to centroid

to minimize RSS, obviously assign x to C_i such that $(x-\mu_i)^2$ [and hence $|x-\mu_i|$] is minimal

(B) recompute centroids

for every i, find μ_i such that $\sum_{x \in C_i} |x-\mu_i|^2$ is minimized.

$$\frac{\partial}{\partial \mu_i} \sum_{x \in C_i} (x-\mu_i)^2 = -2 \sum_{x \in C_i} (x-\mu_i) \stackrel{!}{=} 0$$

$$\Rightarrow \mu_i = \sum_{x \in C_i} x \Big/ |C_i| \quad \square$$

K-Means 4/12

- Convergence to local RSS minimum
  - By our optimality proof from the previous slide, RSS stays equal or decreases in every step (A) and every step (B)
  - There are only finitely many clusterings
  - So, eventually, the algorithm will converge ...

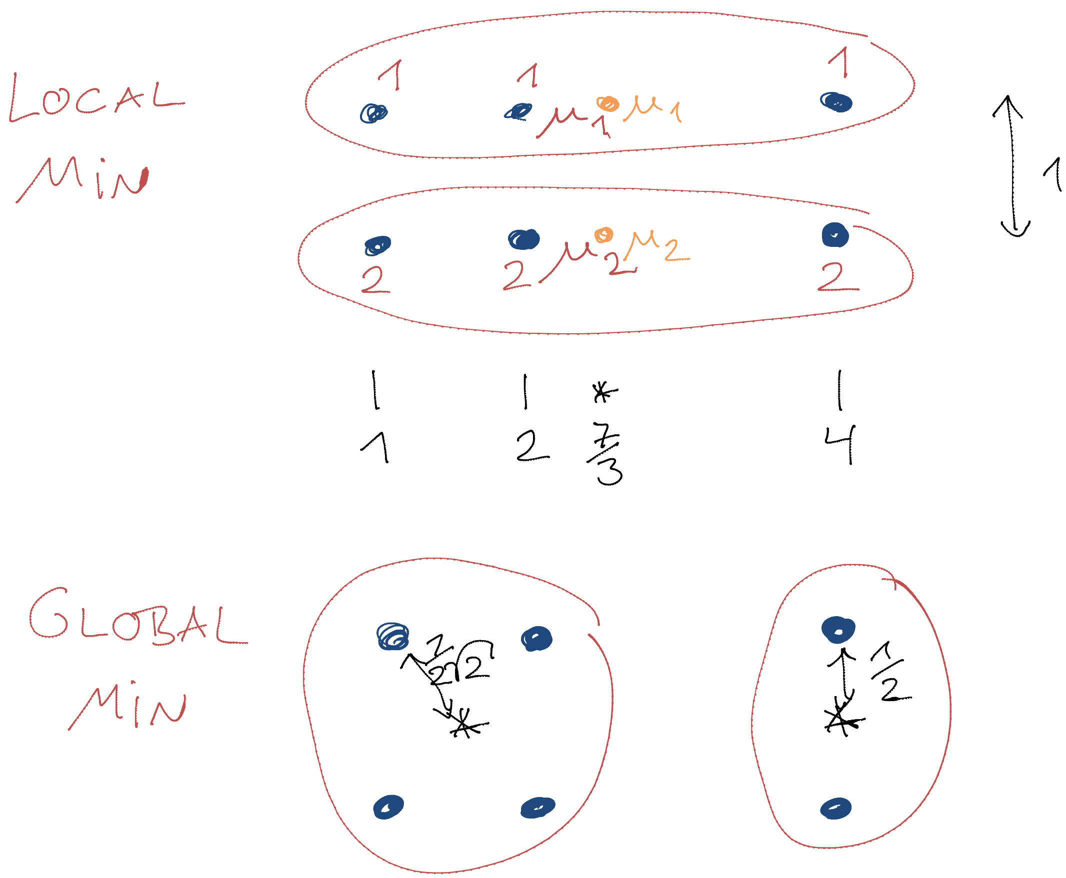
    ... **provided that** we do proper tie breaking in the centroid assignment when two centroids are equally close

    For example, prefer centroid with smaller index

    Otherwise we may cycle forever between different clusterings with equal RSS

K-Means 5/12

- A local RSS minimum is not always a global one



$k = 2$

$$RSS = 2\left[\left(\frac{1}{3}\right)^2 + \left(\frac{4}{3}\right)^2 + \left(\frac{5}{3}\right)^2\right] = 2 \cdot \frac{42}{9} = \frac{84}{9} = 9.333..$$

$k = 2$

$$RSS = 4 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} = 2.5$$

K-Means 6/12

n = # elements
k = # clusters

- Termination condition, options
  - **Stop** when no more change in clustering
    optimal, but this can take a **very** long time
    k^n diff. clustering
  - **Stop** after a fixed number of iterations
    easy, but how to guess the right number?
  - **Stop** when RSS falls below a given threshold
    reasonable, but RSS may never fall below that threshold
  - **Stop** when decrease in RSS falls below a given threshold
    reasonable: we stop when we are close to convergence
  - Last two best combined with bound on number of iterations

K-Means 7/12

dot product of x and y = $\sum_{i=1}^{m} x_i \cdot y_i$

L2-norm: $|x| = \sqrt{\sum_{i=1}^{m} x_i^2}$

- K-Means for text documents: particularities
  - Again, documents as vectors in term-space

    each document = one column of our term-doc matrix
  - Distance between two document vectors x and y:

    1 – cos angle(x, y) = 1 – x • y / |x| · |y|
  - Average of a set X of documents is just the (component-wise) sum of the vectors in X, divided by |X|
  - **Tip:** Normalize all docs initially such that |.| = 1, and same for centroids after each recomputation

    then no need to recompute |.| every time

K-Means 8/12

on example collection:
50 1M 300K
$2 \cdot n \cdot m \cdot = 15T = 15 \cdot 10^{12}$

- K-Means for text documents: complexity 1/2
  - Let n = #documents, m = #terms, k = #clusters
  - Then each step (A) takes time **Θ(k · n · m)**

    Compute the distance from each of the n documents to each of the k cluster centroids, Θ(m) time per sim. comp.
  - And each step (B) takes time **Θ(n · m)**

    Each of the n documents is added to one centroid vector, and one vector addition takes time Θ(m)
  - Linear in each of n, m, k but product can become **huge**

K-Means 9/12

- K-Means for text documents: complexity 2/2
  - **But:** our document vectors are sparse:

    each vector has ony << m non-zero elements
  - But centroid vectors become dense after some time ... why?
  - **Idea:** truncate both documents (once initially) and centroids to those M << m terms with highest scores in respect. vector
  - Similarity computation can then be done in time **Θ(M)** and overall cost for step (A) reduces to **Θ(k · n · M)**

    use list intersection of a sparse repr. of the vectors for this
  - Step (B) could be done in time **O(n · M · log n)** using a merge of the sparse vectors ... but probably not faster than the simple **O(n · m)** addition + anyway (A) is the bottleneck

K-Means 10/12

- Choice of a good k
  - **Idea 1:** choose the k with smallest RSS

    Bad idea, because RSS is minimized for $k = n$
  - **Idea 2:** choose the k with smallest $RSS + \lambda \cdot k$

    Makes sense: RSS becomes smaller as k becomes larger

    But now we have λ as a tuning parameter

    However: for a given application (e.g. document clustering), there is often an input-independent good choice for λ, whereas a good k depends on the input

    The formula also has an information-theoretic justification

K-Means 11/12

- When is k-means a good clustering algorithm
  - **Note:** whether it's good or not, k-means is used a **lot** lot lot in practice, just because of it's simplicity
  - k-means tends to produce compact clusters of about equal size

    Indeed, it is optimal for spherical clusters of equal size

K-Means 12/12

- Alternatives
  - **K-Medoids**

    Centroids are elements from the input set
  - **Fuzzy k-means**

    Elements can belong to several clusters to varying degrees ... this is often called soft clustering
  - **EM-Algorithm** (EM = Expectation-Maximization)

    more sophisticated soft clustering that is optimal when elements come from multi-variate Gaussian distribution

References

- Further reading
  - Textbook Chapter 16: Flat clustering
    http://nlp.stanford.edu/IR-book/pdf/18flat.pdf
- Wikipedia
  - http://en.wikipedia.org/wiki/Cluster_analysis
  - http://en.wikipedia.org/wiki/K-means
  - http://en.wikipedia.org/wiki/K-medoids
  - http://en.wikipedia.org/wiki/EM_Algorithm