Information Retrieval

WS 2013 / 2014

Lecture 9, Tuesday December 17th, 2013

(Clustering, K-Means)

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

UNI FREIBURG

Overview of this lecture

UNI FREIBURG

- Organizational
  - Your results + experiences with Exercise Sheet 8 (LSI)
  - Continue LSI Demo "ALWIS" from last lecture
- Clustering
  - Finding groups of "similar" objects / documents
  - Algorithm: K-Means
  - Correctness, Efficiency, Implementation Advice

  Exercise Sheet 9: cluster 100K abstracts about people (subset of our Wikipedia dataset) using K-Means

Experiences with ES#8 (LSI)

UNI FREIBURG

- Summary / excerpts last checked December 17, 16:00
  - Getting used to Octave take time, but good tool to know
  - The many Octave tips on the last slides were very useful
  - Some of you preferred Matlab to Octave … fine!
  - Many term pairs are not really "synonyms" … see next slide
  - Better results, if stopwords (he, may, to, …) were dropped
  - Thanks for the Weihnachtsvorlesung !
  - Many of you are already busy with Christmas otherwise …

Your results for ES#8 (LSI)

UNI FREIBURG

- Main observations
  - Many pairs are not really "synonyms":

    may – refer, refer – to, she – her, football – league,

    new - york, professor – university, party – member, …
  - For **k = 5**, mostly pairs of frequent terms and from sports
  - For **k = 10**, pairs from more different topics (e.g. politics)
  - For **k = 50**, more different topics + some actual synonyms

    batsman – cricketer, pitcher – baseball, singer – songwriter, …

    Bottom line: amazing how this comes out of linear algebra

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

Clustering

UNI FREIBURG

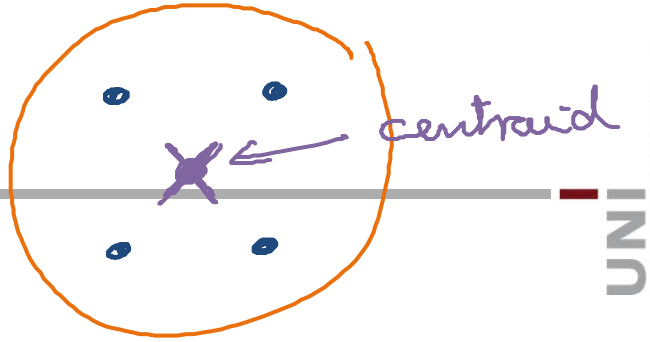
- Informal definition
  - Given n elements from a **metric space** = there is a measure of distance between any two elements
  - Group the elements in clusters such that

    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 1/10



UNI FREIBURG

- Setting / Terminology
  - The 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

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

    The sum of the squares of all intra-cluster distances

    We will see: k-means finds a local minimum of the RSS

K-Means 2/10

UNI FREIBURG

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

    For ES9, pick k random documents 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 ... see slide 8

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

    this too can only decrease the RSS ... see slide 9
  - Let's first look at a demo ...

K-Means 3/10

- Proof of optimality of (A)

$$RSS = \sum_{i=1}^{k_2} \sum_{x \in C_i} (x - \mu_i)^2$$

in Step (A), new $C_1, \dots, C_{k_2}$ are computed

($\mu_1, \dots, \mu_{k_2}$ remain as they are)

$\Rightarrow$ to minimize RSS, assign each x to C_i such that $|x - \mu_i|$ is minimized

that is $\text{cluster}(x) = \operatorname{argmin}_{i=1,\dots,k_2} |x - \mu_i|$

K-Means 4/10

- Proof of optimality of (B)

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

Now the $C_1, \dots, C_k$ is fixed, and we want to find $\mu_1, \dots, \mu_k$ s.t. RSS is minimized

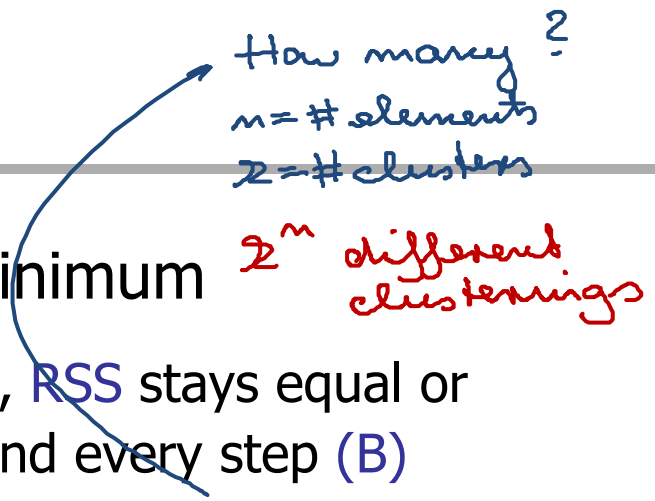
$\Rightarrow$ minimize each of the $\sum_{x \in C_i} (x - \mu_i)^2$

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

$$\Rightarrow \mu_i = \sum_{x \in C_i} x \,/\, |C_i|$$

$$\frac{\partial^2}{\partial \mu_i^2} = 2|C_i| > 0 \Rightarrow \text{local MIN} \quad \square$$

K-Means 5/10



- Convergence to local RSS minimum
  - By what we have just proven, 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 ...

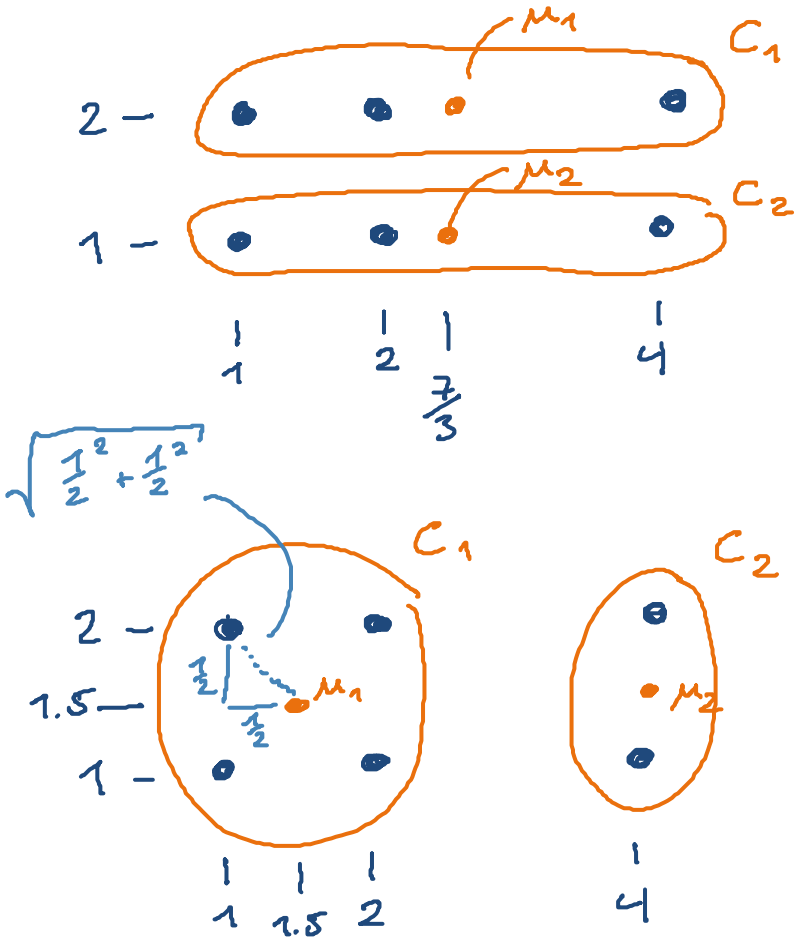
    ... **provided that** we do proper tie breaking in the centroid assignment when two centroids are equally close

    For ES9, simply prefer the centroid with smaller index

    Otherwise we may cycle forever between different clusterings with equal RSS

K-Means 6/10

- A local RSS minimum is not always a global one



$$RSS = 2\cdot\left[\left(1-\tfrac{7}{3}\right)^2+\left(2-\tfrac{7}{3}\right)^2+\left(4-\tfrac{7}{3}\right)^2\right]$$

$$= 2\cdot\tfrac{1}{9}\cdot[16+1+25]$$

$$= 2\cdot\tfrac{42}{9} = \tfrac{84}{9} = 9.333\ldots$$

$$RSS = 4\left(\sqrt{\tfrac{1}{2}}\right)^2 + 2\left(\tfrac{1}{2}\right)^2$$

$$= 4\cdot\tfrac{1}{2} + 2\cdot\tfrac{1}{4} = 2.5$$

SMALLER

K-Means 7/10

UNI FREIBURG

- Termination condition, options
  - **Stop** when no more change in clustering

    Optimal, but this can take a **very** long time
  - **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

    For ES9, aim at a combination of small final RSS and a fast running time … post results on the Wiki

K-Means 8/10

UNI FREIBURG

- 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

    But experience shows that for a given kind of application, there is often an input-independent good choice for λ, whereas a good k depends on the input

K-Means 9/10

UNI FREIBURG

- When is K-Means a good clustering algorithm
  - K-Means tends to produce compact clusters of about equal size

    Indeed, it can be shown that K-Means is optimal when the sought for clusters are spherical and of equal size

    Whether it's good or not, k-means is used a **lot lot lot** in practice, just because of it's simplicity

K-Means 10/10

UNI FREIBURG

- Alternatives
  - **K-Medoids**

    Maintain that centroids are elements from the input set
  - **Fuzzy k-means**

    Elements can belong to several clusters to varying degrees ... this is sometimes called "soft clustering"

    Note: LSI computed a kind of soft clustering
  - **EM-Algorithm** (EM = Expectation-Maximization)

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

K-Means for Text Documents 1/6

UNI FREIBURG

- Representation
  - We represent documents as vectors in term space

    Each document = one column of our term-doc matrix

    For ES9, two ways to represent this:

    DENSE = Array<float> of size m, where m = #terms

    SPARSE = Map<int, float> = only the non-zero scores

    For efficiency reasons, after the first iteration, we will further truncate the SPARSE representation to only those entries with the highest scores, see slide 19

K-Means for Text Documents 2/6

UNI FREIBURG

- Construct from an inverted index
  - This is easy if we store the documents in SPARSE representation = Map<int, float>

    Just go over the inverted lists one by one, and add the term id – score pairs to the corresponding documents

    For ES9, assign consecutive term ids 0, 1, 2, … as you process the inverted lists (one per term), and remember the terms in an Array<String> of size m

    This is needed, so that in the end you can write out the terms corresponding to the highest centroid entries

K-Means for Text Documents 3/6

- Distance between two documents
  - We use the opposite of cosine similarity

    $\text{dist}(x, y) := 1 - \cos \text{angle}(x, y) = 1 - x \bullet y \,/\, |x| \cdot |y|$

    where $x \bullet y$ is the dot product of x and y

    Check: when $x = y$, then **dist(x, x) = 0** … and when x and y have no words in common then **dist(x, y) = 1**

    For ES9, normalize all documents x such that $|x| = 1$
    Then we can very easily compute dist(x, y) as $1 - x \bullet y$
  - Alternative: use Euclidean distance $|x - y|$ or $|x - y|^2$

    Fact: $|x - y|^2 = 2 \cdot (1 - x \bullet y) = 2 \cdot \text{dist}(x, y)$

$$|x-y|^2 = (x-y) \bullet (x-y) = \underbrace{x \bullet x}_{|x|^2 = 1} + \underbrace{y \bullet y}_{|y|^2 = 1} - 2 \cdot x \bullet y$$
$$= 2 \cdot (1 - x \bullet y) \qquad \square$$

K-Means for Text Documents 4/6

UNI FREIBURG

- Running time
  - Let n = #documents, m = #terms, k = #clusters
  - Each step (A) takes time **$\Theta(k \cdot n \cdot m)$**

    Compute **dist** from each of the n documents to each of the k cluster centroids, with Θ(m) time per **dist** computation
  - Each step (B) takes time **$\Theta(n \cdot m)$**

    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 for Text Documents 5/6

UNI FREIBURG

- Speed-up tricks for Step (A)
  - Initially, our document vectors are sparse
  - But centroid vectors become dense after some time
  - Idea: after each iteration, truncate each document and each centroid vector to the M entries with the highest scores

    For ES9, choose M = 1000

    If a vector has less than M non-zero scores, just do nothing
  - A **dist** computation can then be done in time $\Theta(M)$
  - Overall cost for Step (A) then reduces to $\Theta(k \cdot n \cdot M)$

    Provided that we use a SPARSE representation

K-Means for Text Documents 6/6

UNI FREIBURG

- Speed-up tricks for step (B)
  - Step (B) could be done in time **O(n · M · log n)** using a merge of the document vectors in SPARSE representation

    For ES9, use a DENSE representation to re-compute the centroids

    This takes time O(n · m) and is very simple and is as fast as the (more complicated) merge, unless m is very large

    Anyway, step (A) is the bottleneck with respect to runtime

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
- Demo web application
  - http://www.onmyphd.com/?p=k-means.clustering