Information Retrieval

WS 2015 / 2016

Lecture 9, Tuesday December 15th, 2015
(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 experiences with ES8 — Latent Semantic Indexing
  - Quick LSI Demo — "ALWIS"
  - Christmas present — No lecture next week
- Contents
  - Clustering — Definition and example
  - K-Means — Algorithm and analysis
  - K-Means for text — Implementation advice

Exercise Sheet 9: cluster movies dataset using k-means, then report run-time and cluster quality on the Wiki

Christmas present 1/2

- There is **no** lecture next week (December 22)
  - Reason 1: we have one more slot than usual this semester
  - Reason 2: most of you will be away already anyway
  - Reason 3: compensation for repeated overtime
  - **However:** the deadline for ES9 is still December 22

    It makes no sense to give you three weeks for the sheet, it will only lead to procrastination until the very end

    Also, that way you have two weeks of real vacation (at least as far as this course is concerned)
  - We meet again on January 12, **2016** for Lecture 10

Christmas present 2/2

- Cookies
  - No **real** (HTTP) cookies, unfortunately
  - Only chocolate chip cookies
  - I hope you enjoy them anyways
  - I have bought 1000000 of them (in binary)

Experiences with ES8   1/3

- Summary / excerpts
  - Many of you appreciated the "magic" but had trouble understanding the algebra + how it all really works

    I am afraid, one lecture is simply not enough for this stuff, especially if your linear algebra is rusty
  - Getting used to numpy / scipy cost some time
  - Mistake on slide + in the master solution for ES2
  - Problems with large running times or excessive memory use

    For example, iterating over all entries of a large matrix (dense or sparse) is very inefficient in numpy / scipy
  - It seems that many of you are busy with Christmas already

Experiences with ES8 2/3

- Summary of results
  - Results on benchmark improve only marginally, and only with carefully chosen parameters

    This could be considered overfitting

  - Pure LSI gives terrible results → combination is must
  - Many pairs are not "synonyms" in the strict sense:

    about – who, known – as, she – her, composed – music,

    new - york, same – name, jean – french, ...

    Bottom line: amazing application of linear algebra

    But then again: the results aren't really that useful

Experiences with ES8 3/3

- ALWIS
  - ALWIS is a software based on a master's thesis of two of my students (back in Saarbrücken, a long time ago)
  - ALWIS allows an interactive and intuitive exploration of the matrices U and V of the matrix decomposition ($A = U \cdot S \cdot V$)

    U = the underlying "concepts"

    V = the documents expressed in terms of these "concepts"
  - ALWIS implements LSI as well as its probabilistic sibling PLSI

    PLSI gives more intuitive matrices, without negative entries

Clustering 1/3

- Informal definition
  - Given elements $x_1, \ldots, x_n$ from a **metric space**

    metric space = there is a measure of distance between any two elements
  - Group the elements into clusters $C_1, \ldots, C_k$ such that

    Intra-cluster distances are as small as possible

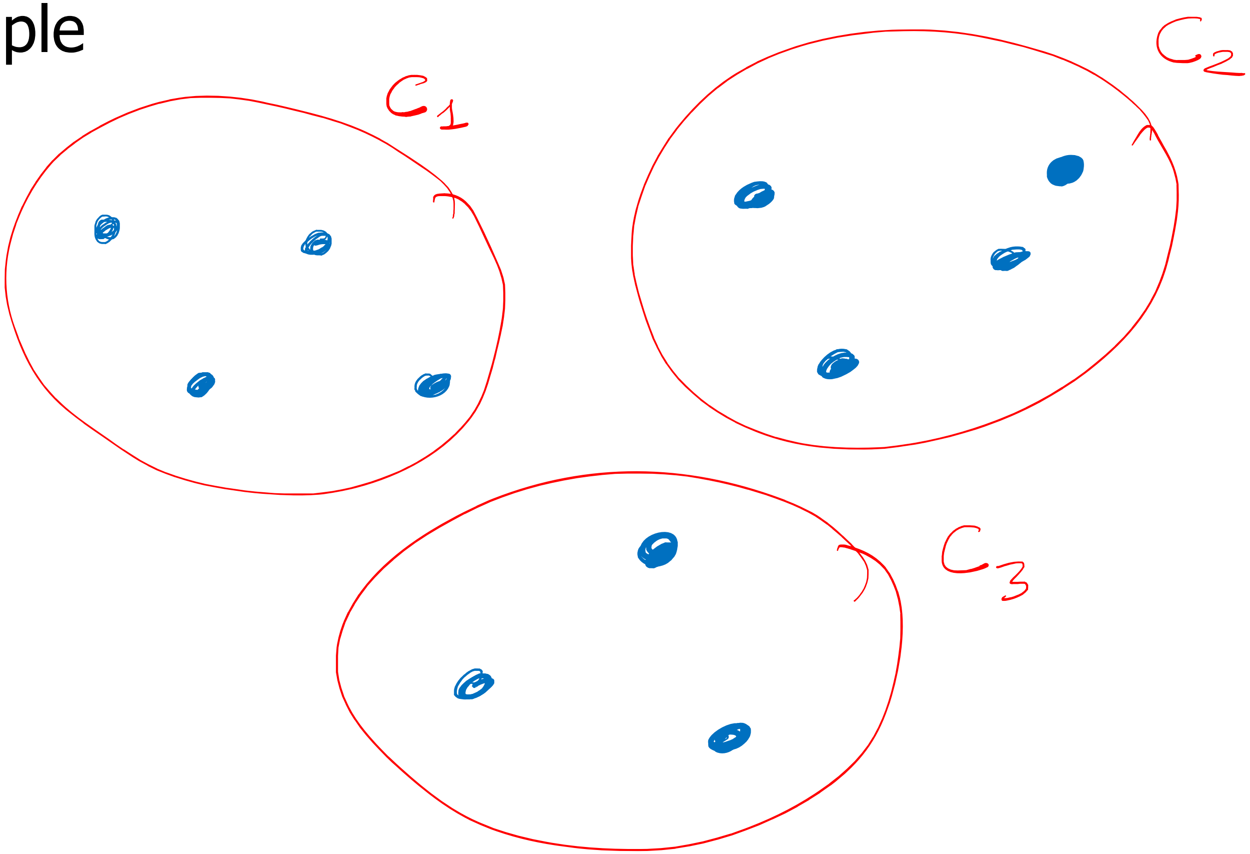
    Inter-cluster distances are as large as possible

    We will make this more precise on slide 10
  - We assume that k is given as part of the input

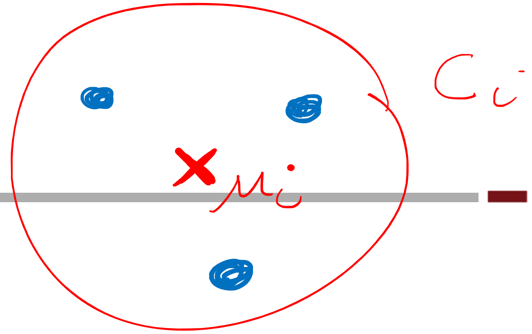
Clustering 2/3

$k = 3$

- Example



Clustering 3/3



- Centroids and RSS
  - Assume we have a **centroid** μ_i for each cluster C_i

    Intuitively: a single element from the metric space "representing the cluster"
  - Let c_i be the index of the cluster to which x_i is assigned

    Each element belongs to exactly one cluster = HARD clustering
  - Then we define the **residual sum of squares** as

    $$RSS = \Sigma_{i=1,\dots,k}\, \Sigma_{x \in C_i} (x - \mu_i)^2 = \Sigma_{i=1,\dots,n} (x_i - \mu_{c_i})^2$$

    The sum of the squares of all intra-cluster distances

K-Means 1/9

- Algorithm
  - Basic idea: find a local optimum of the RSS by greedily minimizing it 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

    (B) compute new centroids as average of elems assigned to it
  - Let's first look at a demo and then show that both steps can only **decrease** the RSS

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

K-Means 2/9

- Step A (assign to nearest centroid)
  - Recall: RSS = $\Sigma_{i=1,\ldots,n} (x_i - \mu_{ci})^2$
  - In Step A, the centroids $\mu_1, \ldots, \mu_k$ are fixed and we want to find those $c_1, \ldots, c_n$ that minimize the RSS:

    $\min_{c1,\ldots,cn} \Sigma_{i=1,\ldots,n} (x_i - \mu_{ci})^2 = \Sigma_{i=1,\ldots,n} \min_{ci} (x_i - \mu_{ci})^2$

    Each summand can be minimized independently
  - $\min_{ci} (x_i - \mu_{ci})^2 = \min_{ci} |x_i - \mu_{ci}|$

    The square distance is min. when the distance is min.
  - $|x_i - \mu_{ci}|$ is minimized for $c_i = \mathrm{argmin}_j |x_i - \mu_j|$

    In words: by assigning x_i to its nearest centroid

K-Means 3/9

- Step B (recompute centroids)
  - Recall: RSS = $\Sigma_{i=1,\ldots,k} \Sigma_{x \in Ci} (x - \mu_i)^2$
  - In Step B, the clusters $C_1, \ldots, C_k$ are fixed and we want to find the centroids $\mu_1, \ldots, \mu_k$ that minimize the RSS:

    $\min_{\mu 1,\ldots,\mu n} \Sigma_{i=1,\ldots,k} \Sigma_{x \in Ci} (x - \mu_i)^2 = \Sigma_{i=1,\ldots,k} \min_{\mu i} \Sigma_{x \in Ci} (x - \mu_i)^2$

    The RSS part for each cluster can be minimized independently
  - We can solve $\min_{\mu i} \Sigma_{x \in Ci} (x - \mu_i)^2$ using simple calculus:

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

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

$$\frac{\partial^2}{\partial \mu_i^2} = 2 \sum_{x \in C_i} 1 = 2|C_i| > 0 \Rightarrow \text{Local Minimum}$$

K-Means 4/9

n elements
k clusters
$\Rightarrow \leq \underbrace{k \cdot \ldots \cdot k}_{n \text{ times}} = k^n$

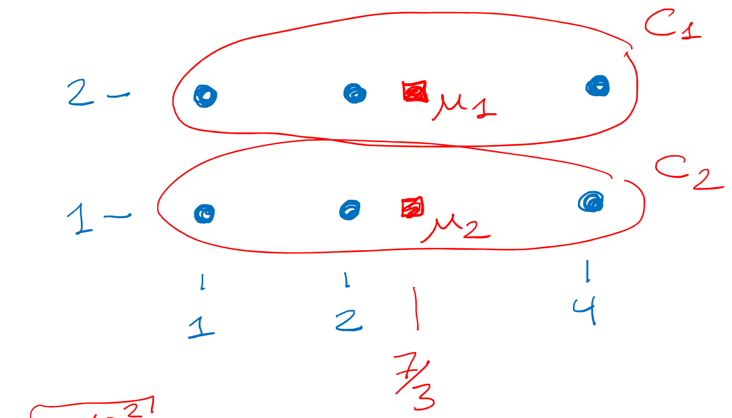
- Convergence to local RSS minimum
  - By what we have just proven, RSS stays equal or decreases in every step (A) and every step (B)
  - There are only finitely many clusterings
  - Therefore, the algorithm will converge if we avoid that it cycles forever between different clusterings with equal RSS
  - Solution: deterministic tie breaking in the centroid assignment, when two centroids are equally close

with the last termination condition on slide 16 $\Rightarrow$ not needed

For ES9, simply prefer the centroid with smaller index

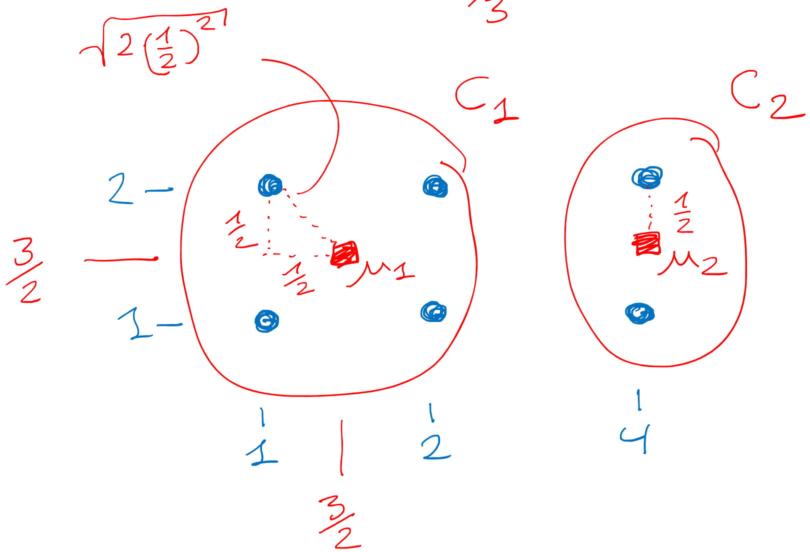
K-Means 5/9

- A local RSS minimum is not always a global one



$$RSS = 2 \cdot \left(\left(\tfrac{1}{3}\right)^2 + \left(\tfrac{4}{3}\right)^2 + \left(\tfrac{5}{3}\right)^2 \right) = 2 \cdot \frac{1+16+25}{9}$$

$$= \frac{84}{9} = 9\tfrac{1}{3}$$



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

$$= 2 + \tfrac{1}{2}$$

$$= 2\tfrac{1}{2}$$

(much) SMALLER

K-Means 6/9

- 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 7/9

- 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

    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 8/9

- 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 9/9

- 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)

    General-purpose optimization technique that can also be used for soft clustering

K-Means for Text Documents 1/7

- Representation
  - We use the vector space model (VSM), as in Lecture 8

    Each document = one column of our term-doc matrix
  - Centroids are also vectors in this space
  - To computer the centroid of a set of documents, just take the average of the document vectors
  - Important observation: the document vectors are **sparse**, the centroids become **dense** over time

    For ES9, it is critical that you store the document vectors in sparse representation, for the same reasons as in ES8

K-Means for Text Documents 2/7

- Construct from an inverted index
  - The term-document matrix can be constructed from an inverted index just as shown in the last lecture

    For ES9, you can re-use your code from ES8, or from the master solutions if you prefer

    And don't be afraid, you don't need the LSI stuff for this sheet, only the term-document matrix

    So even if you had trouble with ES8, please let that not deter you from enjoying ES9

K-Means for Text Documents 3/7

- Running time
  - Let n = #documents, m = #terms, k = #clusters
  - Assume that each dist computation takes time Θ(D)
  - Then each step (A) takes time **Θ(k · n · D)**

    Compute **dist** between each documents and each cluster
  - 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)

K-Means for Text Documents 4/7

$= \sqrt{z \bullet z}$

$|z| = \sqrt{\sum_{i=1}^{m} z_i^2}$

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

- Distance between two documents
  - We use Euclidean distance: dist(x, y) := |x – y|

    Computing this between a sparse and a dense vector takes time Θ(m), where m is the total number of terms

    $|x-y|^2 = (x-y) \bullet (x-y) = x \bullet x + y \bullet y - 2 \cdot x \bullet y = |x|^2 + |y|^2 - 2 \cdot x \bullet y$

  - **Lemma:** $|x - y|^2 = |x|^2 + |y|^2 - 2 \cdot x \bullet y$, where $x \bullet y$ is the dot product of x and y

    $|x|=|y|=1$: $= 2 - 2 \cdot x \bullet y = 2(1 - x \bullet y)$

    Hence: when $|x| = |y| = 1$, then $½ \cdot |x - y|^2 = 1 - x \bullet y$

    x y

    Computing the dot product between a sparse and a dense vector takes time Θ(M), where M is the number of non-zero entries in the sparse vector

    For ES9, this is critical for the running time, see slide 22

K-Means for Text Documents 5/7

- Using matrix operations
  - Both Steps (A) and (B) can be implemented very efficiently using matrix operations

    Some hints and examples on the next two slides
  - Use the lemma from the previous slides and make sure that document vectors and centroids are normalized

    For ES9, we provide explicit code for the normalization

    Quite tricky to implement efficiently in numpy / scipy

    Explicitly iterating over all entries in a large matrix is very **inefficient** (and hence takes forever) in numpy / scipy

K-Means for Text Documents 6/7

- Using matrix operations, Step (A)
  - For Step (A), we need to compute the dot products between all documents and all centroids
  - Let A be the term-document matrix (one doc per column)
  - Let C be the term-centroid matrix (one centroid per column)
  - Then $C^T \cdot A$ yields a matrix, where the entry at i, j is exactly the dot product between centroid i and document j

C^T A j

$\mu_i \begin{pmatrix} \text{———} \end{pmatrix} \begin{pmatrix} \mid \end{pmatrix} = i \begin{pmatrix} \bullet \end{pmatrix}$

i-th centroid

$k \times m$ $m \times n$ $k \times n$

x_j $\mu_i \bullet x_j$

j-th document

K-Means for Text Documents 7/7

L_1-norm = sum of $|\cdot|$ of entries

- Using matrix operations, Step (B)
  - For Step (B), we need to **add** the vectors of all documents in the same cluster C, and then divide by |C|
  - Let A be the term-document matrix (one doc per column)
  - Let B be a 0-1 matrix where the entry at i, j is 1 iff document i is in cluster j … then L_1-normalize the columns
  - Then A · B yields a matrix, where the j-th column is exactly the average of all documents assigned to cluster j

$$\underbrace{\begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{pmatrix}}_{A,\ m \times n} \cdot \underbrace{\begin{pmatrix} 0 & 1/3 \\ 0 & 1/3 \\ 1/2 & 0 \\ 1/2 & 0 \\ 0 & 1/3 \end{pmatrix}}_{B,\ n \times k} = \underbrace{\begin{pmatrix} \cdot & \cdot \end{pmatrix}}_{m \times k}$$

$\frac{1}{2} \cdot x_3 + \frac{1}{2} \cdot x_4$

$\frac{1}{3} \cdot x_1 + \frac{1}{3} \cdot x_2 + \frac{1}{3} \cdot x_5$

$n = 5$
$k = 2$

References

- Further reading
  - Textbook Chapter 16: Flat clustering

    http://nlp.stanford.edu/IR-book/pdf/16flat.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