(N)GSLT Clustering Course

Day 2

Magnus Rosell and Viggo Kann

2008-10-20

Contents

Course Administration

Individual Project.

Infomat xml.

Other packages.

Representations

Representations

Three Hypothetical Newspaper Headlines

1. Chelsea won the final.
2. China defeated in the Olympic match.
3. Match-making in Olympic final.

Term-document-matrix

	doc1	doc2	doc3
Chelsea	0.33	-	-
China	-	0.2	-
defeat	-	0.2	-
win	0.33	-	-
final	0.33	-	0.25
make	-	-	0.25
match	-	0.2	0.25
Olympic	-	0.2	0.25

Word Representation

Term-document-matrix.

- Document? (Context)
  - Document
  - Paragraph
  - Sentence

How about a word-word-matrix (word-co-occurrence-matrix)?

- Context
  - Document
  - Paragraph
  - Sentence
  - Sliding Window

Word-co-occurrence-matrix

	Chelsea	China	defeat	win	final	make	match	Olympic
Chelsea	*	-	-	1	1	-	-	-
China	-	*	1	-	-	-	1	1
defeat	-	1	*	-	-	-	1	1
win	1	-	-	*	1	-	-	-
final	1	-	-	1	*	1	1	1
make	-	-	-	-	1	*	1	1
match	-	1	1	-	1	1	*	2
Olympic	-	1	1	-	1	1	2	*

Word-co-occurrence-matrix (cont.)

Words before and after.
Weight and normalize.
Each row represents a word. Compare these representations.

When building the word representation, how do we use the context? Magnus Sahlgren:

Syntagmatic use of context – words that co-occur.
term-document-matrix

Paradigmatic use of context – words that are used in similar contexts (that co-occur with the same other words).
word-co-occurrence-matrix

Word-co-occurrence-matrix (cont.)

HAL - Hyper-space Analogous to Language (Lund et al. 1995).

Build a word-word-matrix:

- Context window, for example: 10 words before, 10 words after.
- If word a precedes word b increase the value in the element with column for a and row for b. The row of word x shows how words are used before x and the column how they are used after.
- Increase value according to function of distance between the words, for instance the inverse of the distance.

Let each word be represented by the concatenation of its row and column vector. Two times the number of unique words!

Remove features with low variance. (140 000 → 200 in one reported experiment.)

Representation

The representations are built with the assumption that we can model each word (feature) independently from the others. A very strong assumption!

The matrixes are sparse.

The matrixes are noisy. They may be affected by peculiarities in the data. More data gives better representations.

Dimensionality Reduction

Dimensionality Reduction

A Little Linear Algebra

A square $n \times n$-matrix A has an eigenvector v with corresponding eigenvalue λ if $Av = \lambda v$. A has up to n eigenvectors, which can be ordered by their eigenvalues.

The matrix "prefers" the direction pointed out by the vector v.

PCA – Principle Component Analysis
Find all eigenvalues and eigenvectors. If the matrix rows are considered data points, the eigenvectors are the directions that have most variance.

A Little Linear Algebra (cont.)

A matrix M can be linearly transformed by multiplication by another matrix T to yield a new matrix A:

$$A = MT = \begin{pmatrix} m_{1,1} & m_{1,2} & \dots & m_{1,m} \\ m_{2,1} & m_{2,2} & \dots & m_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,1} & m_{n,2} & \dots & m_{n,m} \end{pmatrix} \begin{pmatrix} t_{1,1} & t_{1,2} & \dots & t_{1,r} \\ t_{2,1} & t_{2,2} & \dots & t_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ t_{m,1} & t_{m,2} & \dots & t_{m,r} \end{pmatrix} =$$

$$= \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,r} \\ a_{2,1} & a_{2,2} & \dots & a_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,r} \end{pmatrix}$$

A Little Linear Algebra (cont.)

Consider M our data matrix, the rows representing objects in a vector space.

The columns of P are in the same vector space as the rows of M. If the columns constitute a basis of a subspace of this vector space, the new matrix T is a linear projection of the data onto this subspace.

Dimensionality Reduction

The representation is sparse and noisy. Can we make it more compact and at the same time remove noise?

- LSA – LSI
- Random Mapping and Random Indexing
- Clustering

One definition of noise: directions with little variance.

LSA – Latent Semantic Analysis

LSA – SVD on the term-document-matrix

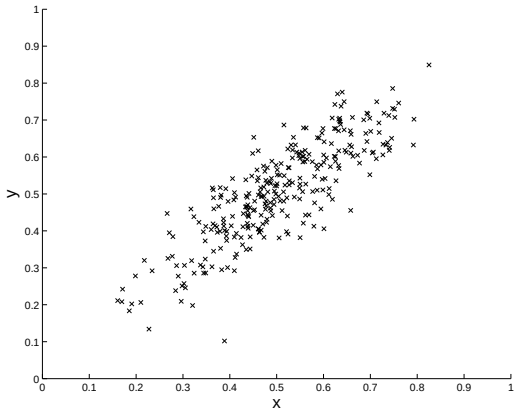
SVD – Singular Value Decomposition

SVD is an extension of PCA for non square matrixes:

- Start with a set of n vectors (words) with dimension m (documents), a $n \times m$-matrix.
- SVD finds a subspace with r dimensions ($r \leq n$, $r \leq m$) so that a projection of the vectors onto it have the least squared distance. Optimal dimension reduction, wrt the variance of the data.
- The basis for the subspace is ordered, so that the largest portion of the variance of the original vectors are in the direction of the first new dimension, the largest part of the rest of the variance is along the the second dimension ...

We can choose r. The reduction is optimal wrt the data variance!

Positive Correlation



SVD – Singular Value Decomposition

$$A_{(n,m)} = U_{(n,R)}\Sigma_{(R,R)}V^T_{(R,m)} = U_{(n,R)}\begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & \sigma_R \end{pmatrix} V^T_{(R,m)} \quad (1)$$

R is the number of eigenvalues of A. We can choose to use only $r < R$, and keep as much as possible of the variance with r dimensions.

We can calculate term-term-similarities $U\Sigma^2U^T$ and the document-document-similarities $V\Sigma^2V^T$. Both in the r-dimensional LSA space. We can compare words and texts!

LSA (cont.)

"Latent" – latent semantic relations. Words that weren't related in the original representation can be deemed related if they have other related words in common.

In this respect LSA makes *paradigmatic use of context*.

$\sim$ 200 dimensions

SVD is computationally heavy.

LSA – LSI

LSI – Latent Semantic Indexing

- LSA used for indexing in a search engine.
- The original usage.
- Goal: increase recall.
- Were not interested in the relations. Only in the methods possibility to improve the search. No semantic interpretations.
- This is a successful (statistical) method for incorporating word relations in the information retrieval model. Most other (linguistic) attempts have failed.

Random Mapping

Observation: random vectors in a multidimensional space is often "almost" orthogonal.

This is usually a problem and is called "The curse of dimensionality". Here we will use it!

Random Mapping (cont.)

Let a $n \times m$-matrix M represent our objects.

Construct a projection $m \times r$-matrix T in a random fashion ($r < m$). The columns of T is considered a new basis for the data. They are "almost" orthogonal.

$A = MT$

Random Mapping (cont.)

If the random matrix T is created carefully the distortion of the data when represented using A compared to M is small.

That is: most of the variance of the data is retained.

The matrix multiplication is very fast compared to LSA (SVD).

Random Indexing

Instead of reducing the space: why not build a smaller from the start?

Random Indexing (cont.)

- Associate every unique word with a multidimensional random vector, with just a few elements taking values other than zero. Call it the *Random Label* of the word. (For instance 1800 dimensions, four 1:s and four -1:s.)
- For every word, create a context vector by adding the Random Labels for the words within a context window centered around each appearance of the word.
  - We decide in advance the size of the context window (number of words before and after) and move it over the text.
  - The addition is weighted as a function of the distance between the word and the others in the context window.
- The context vectors of the words can be compared using a similarity measure, for instance the cosine measure. Words that has a high similarity have some kind of relation (they appear in similar contexts).

Random Indexing (cont.)

Random Indexing is an "incremental" implementation of Random Mapping. We do not need to build the original matrix. The projection matrix is implicitly induced by the Random Labels during the accumulation.

If we would set the Random Labels to be unique vectors with just one 1 in one place, we would get a word-word-matrix.

LSA vs. RI

- The documents of LSA – The context window size of RI
- The dimension reduction step of LSA – The predefined dimensionality of Random Labels in RI.
- LSA starts with the entire matrix – The RI is built incrementally
- LSA finds the optimal subspace – RI picks a subspace at random
- LSA uses SVD which is slow – RI is fast

Random Indexing – paradigmatic use of context.

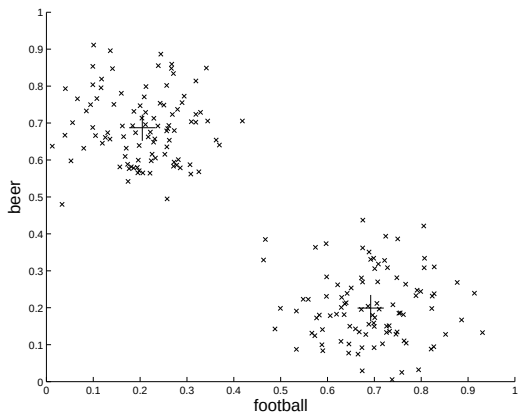
LSA – the term-document-matrix: syntagmatic use of context, the (optimal) dimension reduction tries to capture the paradigmatic use of context.

LSA and RI perform equally on synonym tests.

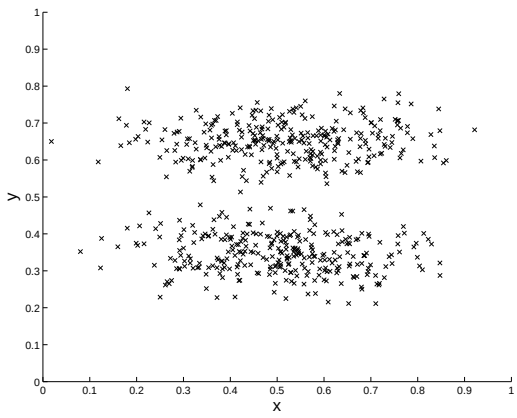
Clustering and Dimensionality Reduction

SVD gives us the directions with the most variance. Clustering can be achieved by splitting data along this direction.

Two Well Separated



Two Along X-axis



Clustering and Dimensionality Reduction (cont.)

- Clustering can be seen as a dimensionality reduction. Objects can be described by which cluster they belong to.
- Represent each object by a vector with similarities to all the clusters. This gives a new representation with k dimensions, which can be compared to a reduction through SVD. The clusters defines areas where their is a lot of objects.

  The cluster centroids could be considered a transformation matrix as in Random Mapping.

  Such clustering dimension reduction (using K-Means) often perform equally to SVD, but can be faster. "Almost" optimal.

Graph Based Clustering

Graph Based Clustering

Six Hypothetical Newspaper Headlines

1. Chelsea won the final.
2. Zimbabwe defeated China in the Olympic match.
3. Match-making in Olympic final.
4. Ericsson stock market winner, increased by 50 per cent.
5. Interest rate at 500 per cent in Zimbabwe.
6. Stock traders nervous as interest rate increases.

Term-document-matrix

	doc1	doc2	doc3	doc4	doc5	doc6
Chelsea	0.33	-	-	-	-	-
China	-	0.2	-	-	-	-
defeat	-	0.2	-	-	-	-
Ericsson	-	-	-	0.16	-	-
win	0.33	-	-	0.16	-	-
final	0.33	-	0.25	-	-	-
increase	-	-	-	0.16	-	0.16
interest	-	-	-	-	0.25	0.16
make	-	-	0.25	-	-	-
market	-	-	-	0.16	-	-
match	-	0.2	0.25	-	-	-
nervous	-	-	-	-	-	0.16
Olympic	-	0.2	0.25	-	-	-
per cent	-	-	-	0.16	0.25	-
rate	-	-	-	-	0.25	0.16
stock	-	-	-	0.16	-	0.16
trade	-	-	-	-	-	0.16
Zimbabwe	-	0.2	-	-	0.25	-

Document-document-similarity-matrix

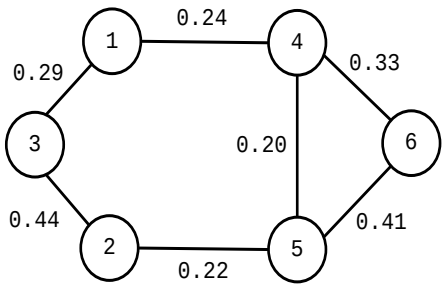
Using the cosine similarity measure between texts we get the following similarity matrix.

	doc1	doc2	doc3	doc4	doc5	doc6
doc1	1.00	0	0.29	0.24	0	0
doc2	0	1.00	0.44	0	0.22	0
doc3	0.29	0.45	1.00	0	0	0
doc4	0.24	0	0	1.00	0.20	0.33
doc5	0	0.22	0	0.20	1.00	0.41
doc6	0	0	0	0.33	0.41	1.00

From this we can construct a weighted graph. $w_{i,j}$ is the weight of the edge between obj_i and obj_j and can be defined in several ways:

1. Let $w_{i,j} = 1$ if $s(i,j) < \epsilon$ (a small constant).
2. Let $w_{i,j} = s(i,j)$ if obj_i is among the k closest neighbors of obj_j and vice versa.
3. Let $w_{i,j} = s(i,j)$ (usually gives a dense graph).

Text Similarity Graph



Graph Based Clustering

There are a lot of problems and algorithms on graphs. Many of them are closely related to clustering, for example:

- Minimum/Maximum Cut
- Minimal Spanning Tree
- Finding
  - Connected Components
  - Cliques

Minimum Cut Problem

What is a good clustering of the vertices of a graph?
The weight of the edges between the clusters should be small!
A *cut* between two disjoint subsets A, B of the vertices is

$$\text{cut}(A,B) = \sum_{i \in A, j \in B} w_{i,j}.$$

For a partition $A_1, \ldots, A_k$ the cut is defined as

$$\text{cut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \text{cut}(A_i, \overline{A}_i)$$

where $\overline{A}$ is the complement of A.

Will the minimum cut be a good clustering?

Minimum Cut Problem (cont.)

The minimum cut often consists of one large cluster and $k-1$ individual vertices, which is clearly not a good clustering.
To get more balanced clusters we could use:

$$\text{RatioCut}(A_1,\dots,A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i,\overline{A}_i)}{|A_i|}$$

$$\text{NCut}(A_1,\dots,A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i,\overline{A}_i)}{\text{vol}(A_i)}$$

where $\text{vol}(A)$ is the sum of the weights of all edges in A.
The minimum of $\sum_{i=1}^{k} 1/|A_i|$ is achieved when all $|A_i|$ coincide.
$\sum_{i=1}^{k} 1/\text{vol}(A_i)$ is minimum when all $\text{vol}(A_i)$ coincide.
Minimum RatioCut and Minimum NCut are NP-hard problems!

Spectral Clustering

Spectral Clustering

Matrices in Spectral Clustering

$$\text{Weight matrix } W = \begin{pmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \dots & w_{n,n} \end{pmatrix}$$

$$\text{Degree matrix } D = \begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & d_n \end{pmatrix}$$

$$\text{where the degree } d_i = \sum_{j=1}^{n} w_{i,j}$$

$$\text{Laplacian matrix } L = D - W$$

Properties of the Laplacian $L = D - W$

1. For each vector $f \in R^n$
   $f^T L f = \frac{1}{2}\sum_{i,j=1}^{n} w_{i,j}(f_i - f_j)^2$.
2. L is symmetric and positive semidefinite.
3. Smallest eigenvalue is 0, corresponding eigenvector is the one vector $\mathbf{1} = (11\dots1)^T$, that is $L\mathbf{1} = 0 \cdot \mathbf{1}$.
4. L has n non-negative real-valued eigenvalues $0 \le \lambda_1 \le \lambda_2 \le \lambda_3 \le \dots \le \lambda_n$.
5. If the graph has k connected components $A_1,\dots,A_k$ then the multiplicity of the eigenvalue 0 is k and the corresponding eigenvectors are $\mathbf{1}_{A_1}, \mathbf{1}_{A_2}, \dots, \mathbf{1}_{A_k}$ where $(\mathbf{1}_{A_i})_j = 1 \Leftrightarrow obj_j \in A_i$.

Normalized Laplacians

$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$
$L_{rw} = D^{-1} L = I - D^{-1} W$
where I is the unitary matrix and

$$D^{-1} = \begin{pmatrix} 1/d_1 & 0 & \dots & 0 \\ 0 & 1/d_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & 1/d_n \end{pmatrix}$$

The normalized Laplacians have similar properties to the vanilla Laplacian.

Spectral Clustering Algorithm

1. Construct L
2. Compute first k eigenvectors $v_1,\dots,v_k$ of L.
3. Let V be the $n \times k$ matrix

$$\begin{pmatrix} | & | & & | \\ v_1 & v_2 & \dots & v_k \\ | & | & & | \end{pmatrix}$$

4. Let $y_i \in R^k$ be the i-th row of V (for $i = 1 \dots n$)
5. Cluster the points $(y_i)_{i=1\dots n}$ with k-means into clusters $C_1,\dots,C_k$
6. Let $A_i = \{obj_j : y_j \in C_i\}$ for $i = 1 \dots k$
7. Return clusters $A_1,\dots,A_k$

If L_{rw} is used we get the *normal spectral clustering algorithm*.

Relation to the Minimum RatioCut Problem

Example: For $k = 2$ we want to find

$$\min_{A \subset V} \text{RatioCut}(A, \overline{A})$$

Given A we define the vector $f = (f_1 \dots f_n)^T$ as

$$f_i = \begin{cases} \sqrt{|\overline{A}|/|A|} & \text{if } obj_i \in A \\ -\sqrt{|A|/|\overline{A}|} & \text{if } obj_i \in \overline{A} \end{cases}$$

$$\begin{aligned} f^T L f &= \frac{1}{2}\sum_{i,j=1}^{n} w_{i,j}(f_i - f_j)^2 \\ &= \frac{1}{2}\sum_{i\in A, j\in\overline{A}} w_{i,j}\left(\sqrt{|\overline{A}|/|A|} + \sqrt{|A|/|\overline{A}|}\right)^2 + \\ &+ \frac{1}{2}\sum_{i\in\overline{A}, j\in A} w_{i,j}\left(-\sqrt{|\overline{A}|/|A|} - \sqrt{|A|/|\overline{A}|}\right)^2 \end{aligned}$$

Relation to the Minimum RatioCut Problem (cont.)

Recall that $\text{RatioCut}(A_1, A_2) = \frac{\text{cut}(A_1,\overline{A}_1)}{|A_1|} + \frac{\text{cut}(A_2,\overline{A}_2)}{|A_2|}$.

Thus

$$\begin{aligned} f^T L f &= \text{cut}(A,\overline{A})(|\overline{A}|/|A| + |A|/|\overline{A}| + 2) \\ &= \text{cut}(A,\overline{A})\left(\frac{|A|+|\overline{A}|}{|A|} + \frac{|A|+|\overline{A}|}{|\overline{A}|}\right) \\ &= \text{cut}(A,\overline{A})\left(\frac{|V|}{|A|} + \frac{|V|}{|\overline{A}|}\right) \\ &= |V| \cdot \text{RatioCut}(A,\overline{A}) \end{aligned}$$

Relation to the Minimum RatioCut Problem (cont.)

The vector f can be shown to be orthogonal to the $\mathbf{1}$ vector.
The norm (length) $\| f \|$ of f is $\sqrt{n}$:

$$\| f \| = 2 \sum_{i=1}^{n} = |A| \frac{|\overline{A}|}{|A|} + |\overline{A}| \frac{|A|}{|\overline{A}|} = |\overline{A}| + |A| = n$$

So the problem of minimizing RatioCut$(A, \overline{A})$ can be rewritten

$$\min_{A \subset V} f^T L f \text{ subject to } f \perp\!\!\!\perp \mathbf{1}, f \text{defined as above}, \| f \| = n$$

This is too hard to solve, so we relax it to

$$\min_{f \in R^n} f^T L f \text{ subject to } f \perp\!\!\!\perp \mathbf{1}, \| f \| = n$$

The Rayleigh-Ritz theorem states that the solution of this problem is the second smallest eigenvector (the smallest is $\mathbf{1}$).

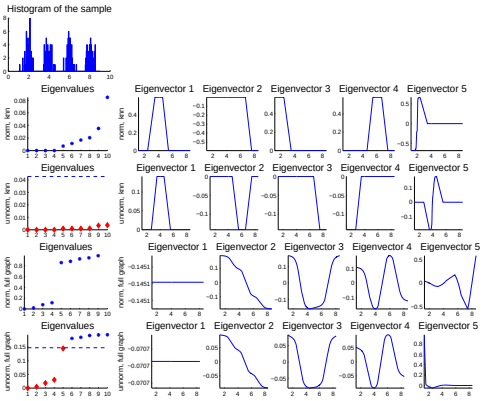
Relation to the Minimum RatioCut Problem (cont.)

In order to obtain a partition of the graph we need to retransform the real-valued solution vector f of the relaxed problem into a discrete indicator vector.
We do this by considering the coordinates f_i as points in R and cluster them into two groups $C, \overline{C}$ by the k-means algorithm.
That is we choose

$$\begin{cases} obj_i \in A \text{ if } f_i \in C \\ obj_i \in \overline{A} \text{ if } f_i \in \overline{C} \end{cases}$$

Similarly we can show that the second smallest eigenvector of the normalized Laplacian L_{sym} approximates the Minimum NCut Problem.

Example of Spectral Clustering



Social Network Analysis

Social Network Analysis

Social Network Analysis

There are many (social) structures that can be considered a graph.
Bibliometrics is concerned with citations, *citation analysis*.

Just by looking at a link structure, can we decide what/who is important, relevant, authorative?

Here, we will discuss webpages.

Relevance Ranking

Observation: "the creator of page p, by including a link to a page q, has in some measure conferred authority on q."

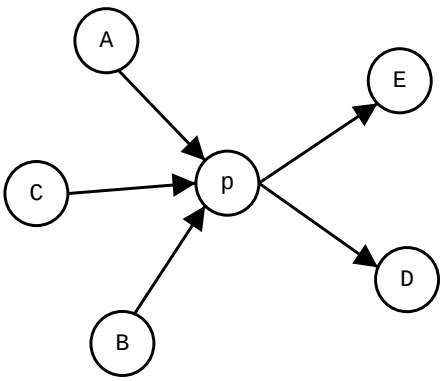
Relevance Ranking Heuristic: Sort by number of in-links.
Problems:

- popularity = relevance?
- many in-links $\Longrightarrow$ high relevance for all words on a web page?

A particular problem:

- It is possible to fool: create a lot of "junk" pages that just link to yours.

Link Structure Example



PageRank by Google

Observation: The relevance of a page is the sum of the relevances for the pages that link to it.

PageRank (first version):

$$PR(p) = c \sum_{q \in B(p)} \frac{PR(q)}{N_q}$$

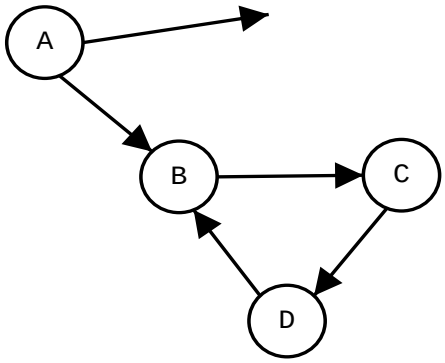
where c is a constant, p and q are pages,
$B(p)$ are the pages that link to p
and N_q is the number of pages that page q links to.

Recursive definition. Need to start with something. Usually a small rank for all pages.

PageRank (cont.)

. . . but a group of pages that only link to each other accumulate rank. They are a *rank sink*.

Rank Sink



PageRank (cont.)

Introduce a damping factor d ($0 < d < 1$):

$$PR(p) = (1 - d) + d \sum_{q \in B(p)} \frac{PR(q)}{N_q}$$

Intuition: a "random surfer" follows links randomly from page to page. With probability $(1 - d)$ she gets bored and picks another page at random.

PageRank (cont.)

PageRank is calculated by iterating matrix multiplication according to the formula. Start with an initial guess of the vector of PageRank for all pages: v_0.

$$\begin{aligned} v_1 &= Av_0 \\ v_2 &= Av_1 \\ &\vdots \end{aligned}$$

The matrix has nice properties and the iteration converges. PageRank is the first eigenvalue of the matrix.

PageRank (cont.)

Google indexes several billion pages.

To calculate PageRank for all takes a couple of weeks.

PageRank is used to rank the returned pages. The rank of a page based on the PageRank and the ordinary similarity of the page and the query.

PageRank is used to many other things than to rank results. For instance to decide which pages that should be updated (crawled) often.

Hubs and Authorities

PageRank summarizes the "relevance" of each page by one value.

Slightly more sophisticated (Kleinberg): *authorities* are are pages that contain good information, *hubs* are pages that have pointers to other pages with good information.

Give each page a rank for each: authority weight $x^{(p)}$, and hub weight $y^{(p)}$, both normalized. An iterative calculation that like for PageRank converges:

$$\begin{aligned} x^{(p)} &= \sum_{q \text{ linking to } p} y^{(p)} \\ y^{(p)} &= \sum_{q \text{ linking from } p} x^{(p)} \end{aligned}$$

Hubs and Authorities

But which pages? Hubs and authorities are calculated after retrieval. However, the set is augmented with:

- all pages that link to the retrieved set,
- and all pages that the retrieved set link to.

Further, pages with similar URL:s are removed from the set.

PageRank vs. Hubs and Authorities

PageRank is one measure of relevance. It is very similar to what is used in bibliometrics. It is a global measure.

Hubs and authorities takes more of the structure into account, defining two different roles for pages. They are local measures, calculated after the retrieval.

Probabilistic Clustering

All methods we have discussed have relied on frequency counts. They are in this respect all sampling a distribution we believe governs a generation of the objects.

Now, an example on how we can explicitly state which distributions that are presupposed.

Probabilistic Clustering (cont.)

Let $\Theta = (L, \{\theta_t\})$ be a document model. That is a set of parameters for a set of probability distributions governing the generation of documents.

L governs the length of the documents: $P(L = l_d|\Theta)$. And θ_t is the probability to write term t. A probability which is modeled independent of all other terms.

Let $n(d, t)$ be the number of times term t appears in document d.

The probability of writing a document with length l_d and frequency counts $\{n(d, t)\}$ becomes:

$$P(\{n(d,t)\}|l_d,\Theta) = \binom{l_d}{\{n(d,t)\}} \prod_{t\in d} \theta_t^{n(d,t)} = \frac{l_d!}{n(d,t_1)!n(d,t_2)!\ldots} \prod_{t\in d} \theta_t^{n(d,t)}$$

Probabilistic Clustering (cont.)

The probability of writing a particular document becomes:

$$P(d|\Theta) = P(L = l_d|\Theta)P(\{n(d,t)\}|l_d,\Theta)$$

We could estimate all parameters of Θ for a set of documents.

But all parameters are different in different groups of texts $\longrightarrow$ clustering! A mixture model with k different groups of texts:

$$P(d|\Theta_{clust}) = \sum_{j=1}^{k} \pi_j P(d|\Theta_j)$$

where $\Theta_{clust} = (k, \pi_1, \pi_2, \ldots \pi_k, \{\Theta_j\}_{j=1}^k)$.

Probabilistic Clustering (cont.)

A set of documents $D = \{d_i\}_{i=1}^n$

The probability of generating the whole set from k different groups:

$$P(D|\Theta_{clust}) = \prod_{i=1}^{n} P(d_i|\Theta_{clust})$$

But we do not know k, the mixture weights, or the Θ_j:s.

For now, lets fix k.

The EM Algorithm

An introduction to the EM Algorithm.

Lets divide Θ_{clust} into two parts:

- A current hypothesis of the document distributions: Θ. Start with a guess (randomly or otherwise): $h = \{h_j\}_{j=1}^k$.
- The hidden variables: $V = (\pi_1, \pi_2, \ldots \pi_k)$. We do not now how the distributions are mixed.

That is: $P(D|\Theta_{clust}) = P(D|\Theta, V)$

The EM Algorithm

EM – Expectation, Maximization

Expectation Decide V using the hypothesis h.

Maximization Find new h using V.

More formally:

Expectation Calculate $Q(h|h') = E[P(V|h')|h, D]$. From this the expected values of V given the current hypothesis h can be obtained.

Maximization $h \leftarrow \operatorname{argmax}_{h'} Q(h'|h)$. The new hypothesis is the one that maximizes the expected values of V.

Probabilistic Clustering: K-Means

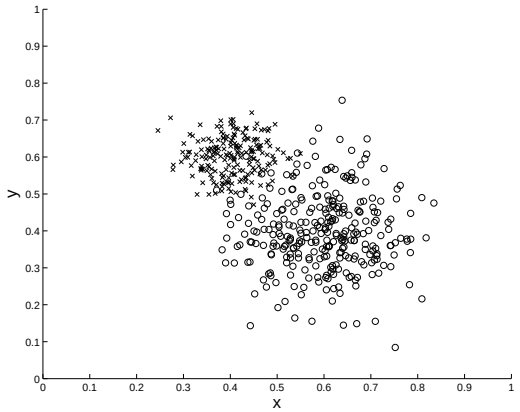
K-Means is an EM Algorithm.

The K-Means algorithm suppose that the data is generated by k normal distributions with undecided variances.
It improves a hypothesis about the means of these distributions.
In every iteration it does two things:

- It estimates which distribution generated which object – the closest mean. *Expectation*.
- It improves the hypothesis – maximizes the estimation of the means based on the cluster assignment estimations. *Maximization*.

This idea can be extended: we could augment the hypothesis with the standard deviations for each distribution.

Two Different in Size



The EM Algorithm

We could introduce any number of parameters in the EM algorithm. Like for instance the number of clusters.

For each new parameter the calculations become more heavy.

Feature Selection and Extraction

Feature Selection – Choose among existing. Example: Remove stop words, high and low frequency words

Feature Extraction – Construct features from the data (for instance combine more naïve features). Example: LSA, or any other linear projection, or more complicated projections.

Dimensionality Reduction – Feature Selection/Extraction.

To interpret a clustering result perhaps we need features with obvious "meaning".

Feature Selection and Extraction (cont.)

Feature Selection/Extraction is very important for supervised learning. Only features that are good at separating the classes are retained.

It is much harder for unsupervised learning. We do not know which the classes are. Domain knowledge can be used.

One methodology: cluster, remove features that have the same distribution in all clusters. Cluster again.

In probabilistic clustering it is possible to model how important the different features are, by for instance their distribution over the different clusters.

Collaborative Filtering

Consider two persons A and B that have similar opinions/preferences about several movies. If A likes a movie that B have not seen, it is to some extent likely that B would like it too.

If several other persons C_i that share many opinions with B, also likes the movie, it is more likely that B will like it.

Collaborative Filtering tries to extract such information from databases. Examples: movie databases, customer information, for instance "since you are interested in this book, you may want to consider these also", etc.

Collaborative Filtering (cont.)

A (people × movie)-matrix:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix}$$

where

$$a_{i,j} = \begin{cases} 1 \text{ if person i liked movie j} \\ 0 \text{ if person i did not like movie j} \\ - \text{ otherwise} \end{cases}$$

A sparse matrix: most people have not seen all movies!

Collaborative Filtering (cont.)

A simple method:

- Cluster the persons.
- Suggest common movies to those in each cluster that have no opinion about it.

Extension:

- Cluster the persons based on matrix A.
- Cluster the movies based on matrix A.
- Iterate:
  - Cluster the movies based on person clusters.
  - Cluster the persons based on movie clusters.

This can be good, but might over-generalize: clusters are generalizations. Let's instead look at one person or movie at the time.

Collaborative Filtering (cont.)

Start with a random clustering of the persons with n' clusters and a random clustering of the movies with m' clusters.

Let $c(i)$ and $c(j)$ be the current cluster of person i and movie j, respectively.

Calculate estimations:

$p(i')$ the probability of a person picked at random belonging to person cluster i'

$p(j')$ the probability of a movie picked at random belonging to movie cluster j'

$p(i',j')$ the probability that a person in person cluster i' likes a movie in movie cluster j'

Collaborative Filtering (cont.)

Assignment:
Choose one person i. Calculate the probability that he/she should belong to person cluster i' based on the current clusterings:

$$\pi_{i \to i'} = p(i') \prod_{j \in \text{movies i have seen}} \begin{cases} p(i', c(j)) & \text{if } a_{i,j} = 1 \\ 1 - p(i', c(j)) & \text{if } a_{i,j} = 0 \end{cases}$$

Assign person i randomly to a new cluster according to the distribution $\{\pi_{i \to i'}\}$.
Similarly for movies.

Collaborative Filtering (cont.)

Repeat:

1. Pick a person or a movie at random.
2. Assign it to a cluster.
3. Calculate new estimations.

This is an example of Gibbs sampling, which is guaranteed to converge. But the it could take time.

Clustering as an Optimization Problem

Clustering as an Optimization Problem

Clustering as an Optimization Problem

Optimization problem in general: a set of states and a score function for the states. Find state with best score.

The minimum/maximum cut problem on the similarity graph.

Each possible clustering is a state. We can define an internal measure as a score.

If we define a neighborhood around each state we can use hill climbing to find a local optimum. Chose repeatedly the neighboring state with the best score.

The EM Algorithm (and K-Means) and Gibbs sampling are kinds of hill climbing approaches.

Last Words

Clustering is the partition of a set of objects into clusters (groups, parts) of objects that are as similar as possible in some sense. We might also want the objects from different clusters to as be dissimilar as possible.

The similarity has to be defined. The quality of the clustering result is dependent on the similarity definition.

There are many and varied methods for clustering. The more sophisticated a method, the more knowledge about the data has to be incorporated into it, and the slower the execution.

If we know a lot about the data (and/or have a precise definition of similarity), it might be possible to use some other kind of method to obtain a (correct) grouping.