IIIT-H

Web Mining
Lecture 5: LSI and EM

Manish Gupta

14th Aug 2013

Slides borrowed (and modified) from
nlp.stanford.edu/IR-book/ppt/18lsi.pptx
www.ics.uci.edu/~lopes/teaching/cs221W12/slides/LSI.pptx
www.csc.villanova.edu/~matuszek/fall2003/DocBased.ppt

Recap of Lecture 4: Link Analysis Algorithms

- PageRank
- Topic-Specific PageRank
- HITS (Hypertext-Induced Topic Selection)
- Spam Detection Algorithms: TrustRank

Announcements

- Assignment 1 will be up by 11:59pm and the submission date is Aug 21 9pm
- Rescheduling of lectures
  - Makeup class for Aug 24 lecture will be on Aug 22 6-7:30pm
  - Makeup class for Aug 28 lecture will be on Sep 2 6-7:30pm

Today's Agenda

- Singular Value Decomposition (SVD)
- Latent Semantic Indexing (LSI)
- K-Means
- Expectation Maximization (EM)

Today's Agenda

- **Singular Value Decomposition (SVD)**
- Latent Semantic Indexing (LSI)
- K-Means
- Expectation Maximization (EM)

Datasets in the form of Matrices

- We are given **n** objects and **d** features describing the objects.
- Each object has **d** numeric values describing it.
- **Dataset**
  - An **n-by-d** matrix **A**, A_{ij} shows the "***importance***" of feature **j** for object **i**.
  - Every row of **A** represents an object.
- **Goal**
  - **Understand** the structure of the data, e.g., the underlying process generating the data.
  - **Reduce the number of features** representing the data

Market Basket Matrices

d products
(e.g., milk, bread, wine, etc.)

n customers $\begin{pmatrix} \\ A \\ \\ \end{pmatrix}$

A_{ij} = quantity of **j**-th product purchased by the **i**-th customer

Find a subset of the products that characterize customer behavior

Social Network Matrices

d groups
(e.g., BU group, opera, etc.)

n users

$$A$$

A_{ij} = partitipation of the **i**-th user in the **j**-th group

Find a subset of the groups that accurately clusters social-network users

Document Matrices

d terms
(e.g., theorem, proof, etc.)

$$
\begin{array}{cc}
 & \textbf{d} \text{ terms (e.g., theorem, proof, etc.)} \\
\textbf{n} \text{ documents} & \left(\begin{array}{c} A \\ \\ A_{ij} = \text{frequency of the } j\text{-th} \\ \text{term in the } i\text{-th document} \end{array} \right)
\end{array}
$$

Find a subset of the terms that accurately clusters the documents

Recommendation Systems

d products

n customers

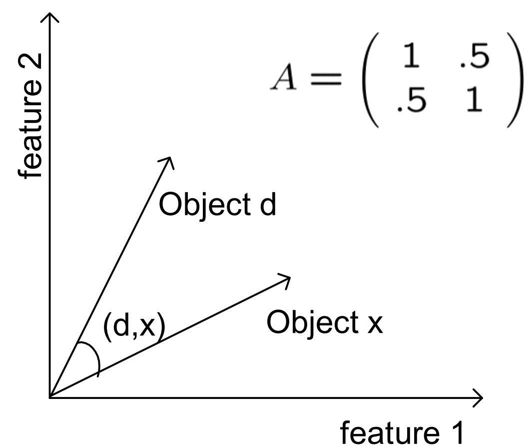
$$A$$

A_{ij} = frequency of the **j**-th product is bought by the **i**-th customer

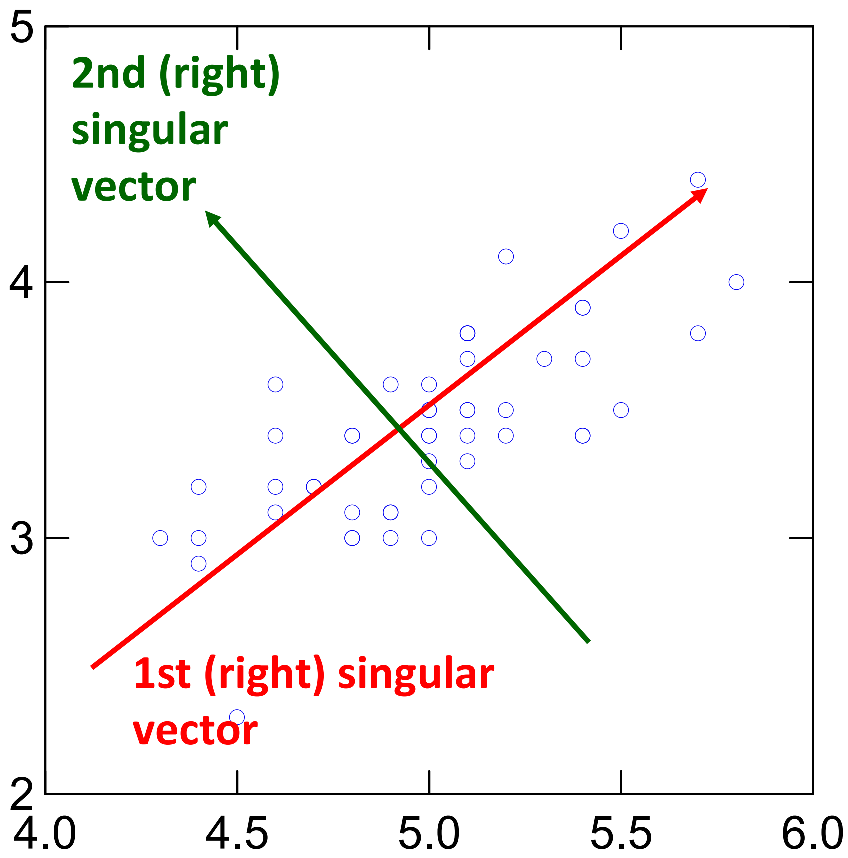
Find a subset of the products that accurately describe the behavior or the customers

The Singular Value Decomposition (SVD)

- Data matrices have **n** rows (one for each object) and **d** columns (one for each feature).
- Rows: vectors in a Euclidean space
- Two objects are "***close***" if the angle between their corresponding vectors is small.



Singular Vectors and Values



- **Input**: 2-d dimensional points
- **Output**:
- 1st (right) singular vector
  - direction of maximal variance
- 2nd (right) singular vector
  - direction of maximal variance, after removing the projection of the data along the first singular vector.
- σ_1: measures how much of the data variance is explained by the first singular vector
- σ_2: measures how much of the data variance is explained by the second singular vector

SVD Decomposition

$$\underset{\mathbf{n\ x\ d}}{\begin{pmatrix} \\ \quad A \quad \\ \\ \end{pmatrix}} = \underset{\mathbf{n\ x\ r}}{\begin{pmatrix} \\ \quad U \quad \\ \\ \end{pmatrix}} \cdot \underset{\mathbf{r\ x\ r}}{\begin{pmatrix} \cancel{\Sigma} \end{pmatrix}} \cdot \underset{\mathbf{r\ x\ d}}{\begin{pmatrix} \quad V \quad \end{pmatrix}^T}$$

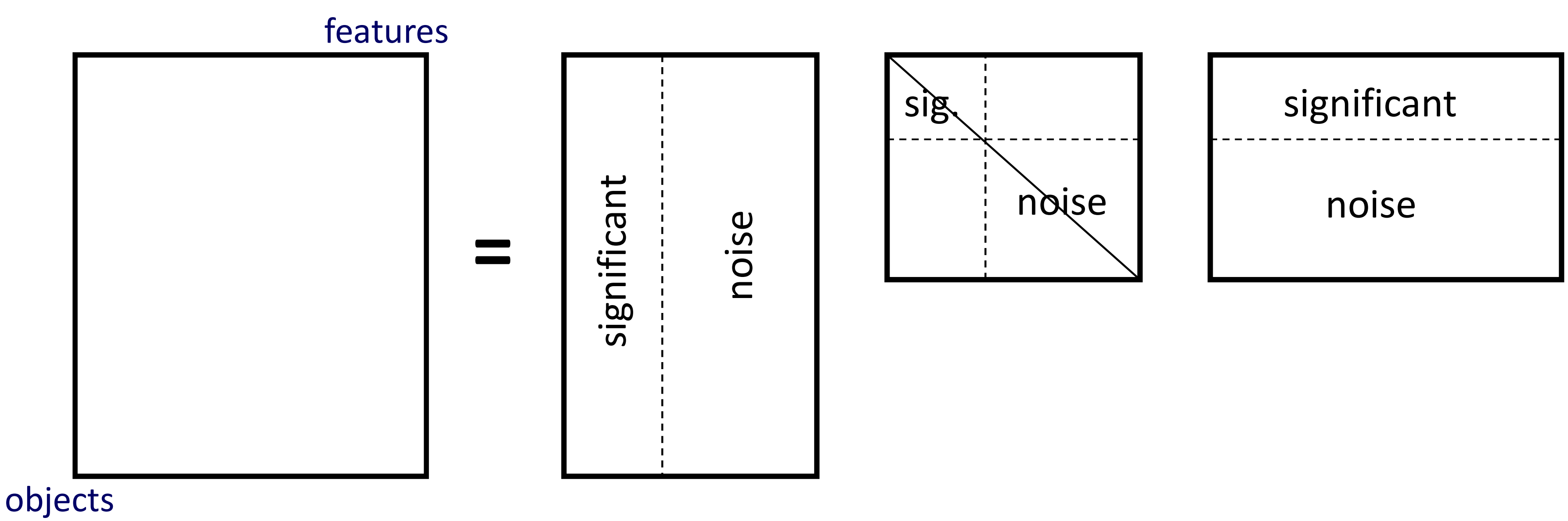
U (V): orthogonal matrix containing the left (right) singular vectors of **A.**

Σ: diagonal matrix containing the **singular values** of **A:**
($\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_\ell$)

Exact computation of the SVD takes **O(min{dn^2 , d^2n})** time.
The top *k* left/right singular vectors/values can be ***computed faster*** using Lanczos/Arnoldi methods.

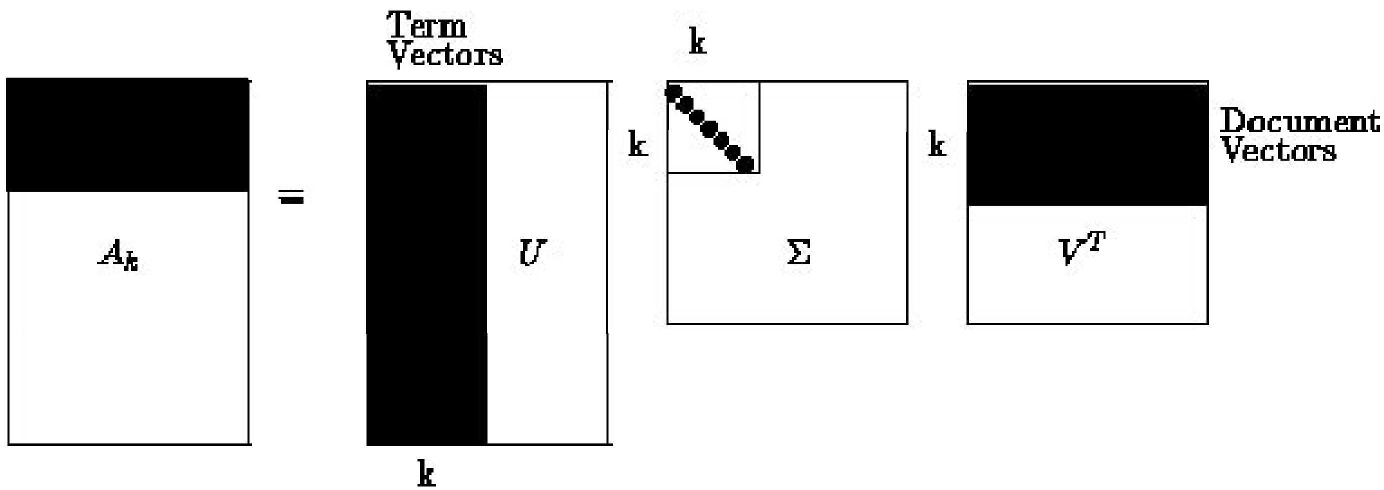
SVD Decomposition

- $A = U\Sigma V^T$ such that
  - $UU^T=I$ and columns $\boldsymbol{U}$ are orthogonal eigenvectors of $\boldsymbol{AA^T}$
  - $VV^T=I$ and columns of $\boldsymbol{V}$ are orthogonal eigenvectors of $\boldsymbol{A^TA}$.
  - Σ = all zeros except diagonal (singular values); singular values decrease along diagonal. They are the square root of the eigenvalues of $\boldsymbol{A^TA}$ ***or*** $\boldsymbol{AA^T}$.



Truncated SVD

- SVD is a means to the end goal.
- The end goal is dimension reduction, i.e. get another version of A computed from a reduced space in $\mathrm{U}\Sigma V^{\mathrm{T}}$
  - Simply zero Σ after a certain row/column k



n x d n x r r x r r x d

Rank-k approximations (A_k)

$$\begin{pmatrix} \\ \\ A_k \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \\ U_k \\ \\ \\ \end{pmatrix} \cdot \begin{pmatrix} \\ \Sigma_k \\ \\ \end{pmatrix} \cdot \begin{pmatrix} \\ V_k^T \\ \\ \end{pmatrix}$$

n x d **n x k** **k x k** **k x d**

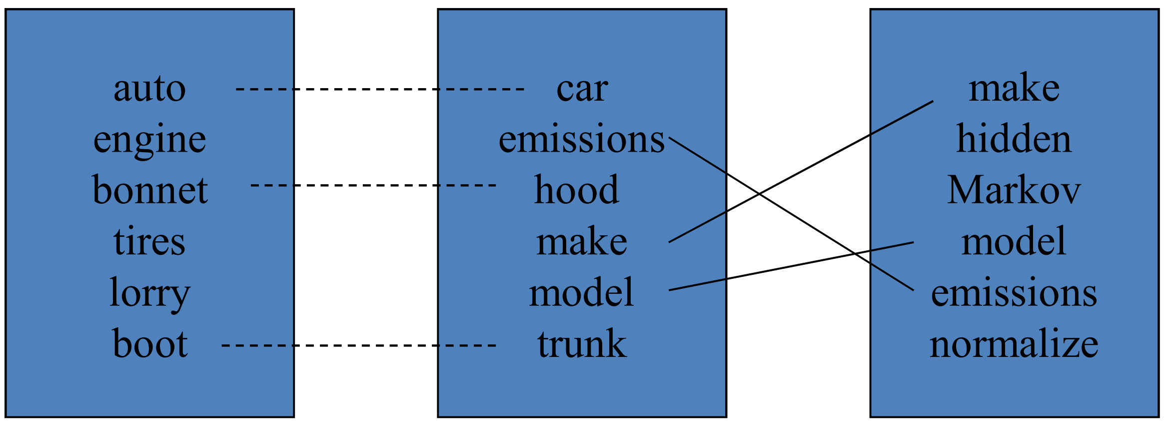
- U_k (V_k): orthogonal matrix containing the top k left (right) singular vectors of **A**. Σ_k: diagonal matrix containing the top k singular values of **A**
- A_k is the best approximation of **A**
- Eckart-Young theorem: Keeping the *k* largest singular values and setting all others to zero gives you the optimal approximation of the original matrix *C* wrt all rank-k matrices and Frobenius norm.
    - $\|A\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{d} |a_{ij}|^2}$

Today's Agenda

- Singular Value Decomposition (SVD)
- **Latent Semantic Indexing (LSI)**
- K-Means
- Expectation Maximization (EM)

Deficiencies with Conventional Automatic Indexing

- **Synonymy:** Various words and phrases refer to the same concept (lowers recall)
- **Polysemy:** Individual words have more than one meaning (lowers precision)
- **Independence:** No significance is given to two terms that frequently appear together
- Latent semantic indexing addresses the first of these (synonymy), and the third (dependence)



Synonymy

Will have small cosine

but are related

Polysemy

Will have large cosine

but not truly related

Latent Semantic Analysis

- Bag of Words methods: A document is only "about" the words in it
- But people interpret documents in a richer context
  - a document is about some domain and concepts in it
  - reflected in the vocabulary but not limited to it
- LSI: developed at Bellcore (now Telcordia) in the late 1980s (1988). It was patented in 1989.
  - Aim: Replace indexes that use **sets of words** by indexes that use **concepts**
  - Latent: Not visible on the surface
  - Semantic: Word meanings
- LSI Demos at http://LSA.colorado.edu

Word Co-Occurrences

- Bag of words approaches assume meaning is carried by vocabulary
- Next step is to look at vocabulary *groups*; what words tend to occur together?
- Still a statistical approach, but richer representation than single terms
- E.g., Looking for articles about Tiger Woods in an API newswire database could bring up stories about the golfer, followed by articles about golf tournaments that don't mention his name.
  - So we are need to recognize that Tiger Woods is *about* golf.

Problem: Very High Dimensionality

- A vector of TF*IDF representing a document is high dimensional.
- Need some way to trim words looked at
  – First, throw away anything “not useful” (stop words)
  – Second, identify clusters and pick representative terms
    - Use SVD

Recall: Term-document matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
anthony	5.25	3.18	0.0	0.0	0.0	0.35
brutus	1.21	6.10	0.0	1.0	0.0	0.0
caesar	8.59	2.54	0.0	1.51	0.25	0.0
calpurnia	0.0	1.54	0.0	0.0	0.0	0.0
cleopatra	2.85	0.0	0.0	0.0	0.0	0.0
mercy	1.51	0.0	1.90	0.12	5.25	0.88
worser	1.37	0.0	0.11	4.15	0.25	1.95
. . .						

Can we transform this matrix, so that we get a better measure of similarity between documents and queries?

SVD: $C=U\Sigma V^T$

- U
  - One row per term
  - One column per min(M,N) where M is the number of terms and N is the number of documents. Each column represents a semantic concept
  - Orthonormal matrix: (i) Row vectors have unit length. (ii) Any two distinct row vectors are orthogonal to each other.
  - How strongly word i is related to the topic/concept represented by semantic dimension j
- Σ
  - Square, diagonal matrix of dimensionality min(M,N) × min(M,N)
  - Diagonal consists of the singular values of C
  - Magnitude of the singular value measures the importance of the corresponding semantic dimension
- V
  - One column per document
  - One row per min(M,N) where M is the number of terms and N is the number of documents. Each row represents a semantic concept
  - Orthonormal matrix: (i) Column vectors have unit length. (ii) Any two distinct column vectors are orthogonal to each other.
  - How strongly document i is related to the topic/concept represented by semantic dimension j .

Example of C = UΣV^T : All Four Matrices

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

$=$

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

$\times$

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

$\times$

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

Dimensionality Reduction

- Key property: Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the "details".
- These details may
  - be noise – in that case, reduced LSI is a better representation because it is less noisy
  - make things dissimilar that should be similar – again reduced LSI is a better representation because it represents similarity better.
- Fewer details is better

Reducing the Dimensionality to 2

U	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

Σ_2	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

- Actually, we only zero out singular values in Σ. This has the effect of setting the corresponding dimensions in U and V^T to zero when computing the product $C = U\Sigma V^T$
- Compute $C_2 = U\Sigma_2 V^T$

Original Matrix C vs. Reduced $C_2 = U\Sigma_2 V^T$

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

We can view C_2 as a two-dimensional representation of the matrix. We have performed a dimensionality reduction to two dimensions.

Why is the LSI-Reduced Matrix "Better"?

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

- Similarity of d2 and d3 in the original space: 0.
- Similarity of d2 und d3 in the reduced space: 0.52 * 0.28 + 0.36 * 0.16 + 0.72 * 0.36 + 0.12 * 0.20 + - 0.39 * - 0.08 ≈ 0.52

- "boat" and "ship" are semantically similar. The "reduced" similarity measure reflects this.
- What property of the SVD reduction is responsible for improved similarity?

Properties of LSI

- Handling Synonymy and Semantic Relatedness
  - The dimensionality reduction forces us to omit a lot of "detail".
  - We have to map different words (= different dimensions of the full space) to the same dimension in the reduced space.
  - The "cost" of mapping synonyms to the same dimension is much less than the cost of collapsing unrelated words. SVD selects the "least costly" mapping.
  - Thus, it will map synonyms to the same dimension. But it will avoid doing that for unrelated words.
- Limitations
  - It cannot capture polysemy i.e words with multiple meanings.
  - The resulting matrix dimension may be difficult to interpret.
  - Finding optimal dimension for semantic space
  - The computational cost of SVD is significant. $O(n^2k^3)$
    - n = number of terms
    - k = number of dimensions in semantic space (typically small ~50 to 350)
  - SVD assumes normally distributed data
    - term occurrence is not normally distributed
- LSI works best in applications where there is little overlap between queries and documents

LSI: Comparison to Other Approaches

- Relevance feedback and query expansion are used to increase recall in information retrieval – if query and documents have (in the extreme case) no terms in common
- LSI increases recall and hurts precision.
- Thus, it addresses the same problems as (pseudo) relevance feedback and query expansion . . .
- . . . and it has the same problems

LSI Implementation

- Compute SVD of term-document matrix
- Reduce the space and compute reduced document representations
- Map the query into the reduced space
  - $q_2^T = \Sigma_2^{-1} U_2^T q^T$
  - This follows from $C_2 = U \Sigma_2 V^T \Rightarrow \Sigma_2^{-1} U^T C_2 = V_2^T$
- Compute similarity of q_2 with all reduced documents in V_2.
- Output ranked list of documents as usual

31

Today’s Agenda

- Singular Value Decomposition (SVD)
- Latent Semantic Indexing (LSI)
- **K-Means**
- Expectation Maximization (EM)

What is Clustering?

- Cluster: A collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- Cluster analysis
  - Grouping a set of data objects into clusters
- Clustering is unsupervised classification: no predefined classes
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

What is a Good Clustering?

- A good clustering method will produce clusters with
  - High <u>intra-class</u> similarity
  - Low <u>inter-class</u> similarity
- Precise definition of clustering quality is difficult
  - Application-dependent
  - Ultimately subjective

K-Means Clustering

$$\min_{C} D = \sum_{k=1}^{K} \sum_{x_i \in C_k} \left\| x_i - m_k \right\|^2$$

- D= total distance
- K = # of clusters
- x are points
- C_k is the set of points in cluster *k*
- m_k is the center of cluster *k*
- ||.|| is a distance

- **Goal:** Given #clusters=K, assign each point to one of the clusters such that the total distance from each point to the center of its cluster is minimized.

K-Means Clustering

- Iterative process to group into *k* clusters.
- Initialize K cluster means
- Repeat until convergence:
  - For each point, find the closest mean and assign it to that cluster
  - Re-compute the mean of all points assigned to the cluster
- Label each point with its current cluster

K-Means Clustering

- Pros:
  - Easy to implement
  - Finds local optimum
- Cons:
  - The number of clusters, *K*, must be known in advance
  - Some clusters might have 0 points
  - Local optimum is not guaranteed to be global optimum
  - The algorithm can only be applied when the mean of a cluster is defined
  - This method is not suitable for clusters with non-convex shapes
  - Sensitive to noise and outliers
- Ideas:
  - Can re-run with several initializations
  - Can choose *K* based on observation or statistical means

K-Means Clustering as an Iterative 2-Step Method

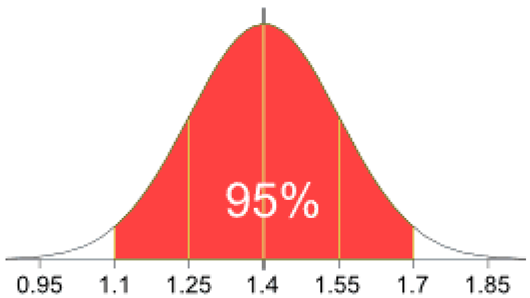
- We are trying to find out where the clusters are and which points are assigned to each cluster. We iteratively solve half the problem. Notice the overall structure.
- Repeat until convergence
  - Assume you know where the cluster centers are. For each point, find the closest mean and assign it to that cluster
  - Assume you know which points belong to each cluster. Re-compute the mean of all points assigned to the cluster
- Label each point with its current cluster

Today’s Agenda

- Singular Value Decomposition (SVD)
- Latent Semantic Indexing (LSI)
- K-Means
- **Expectation Maximization (EM)**

The Gaussian Distribution

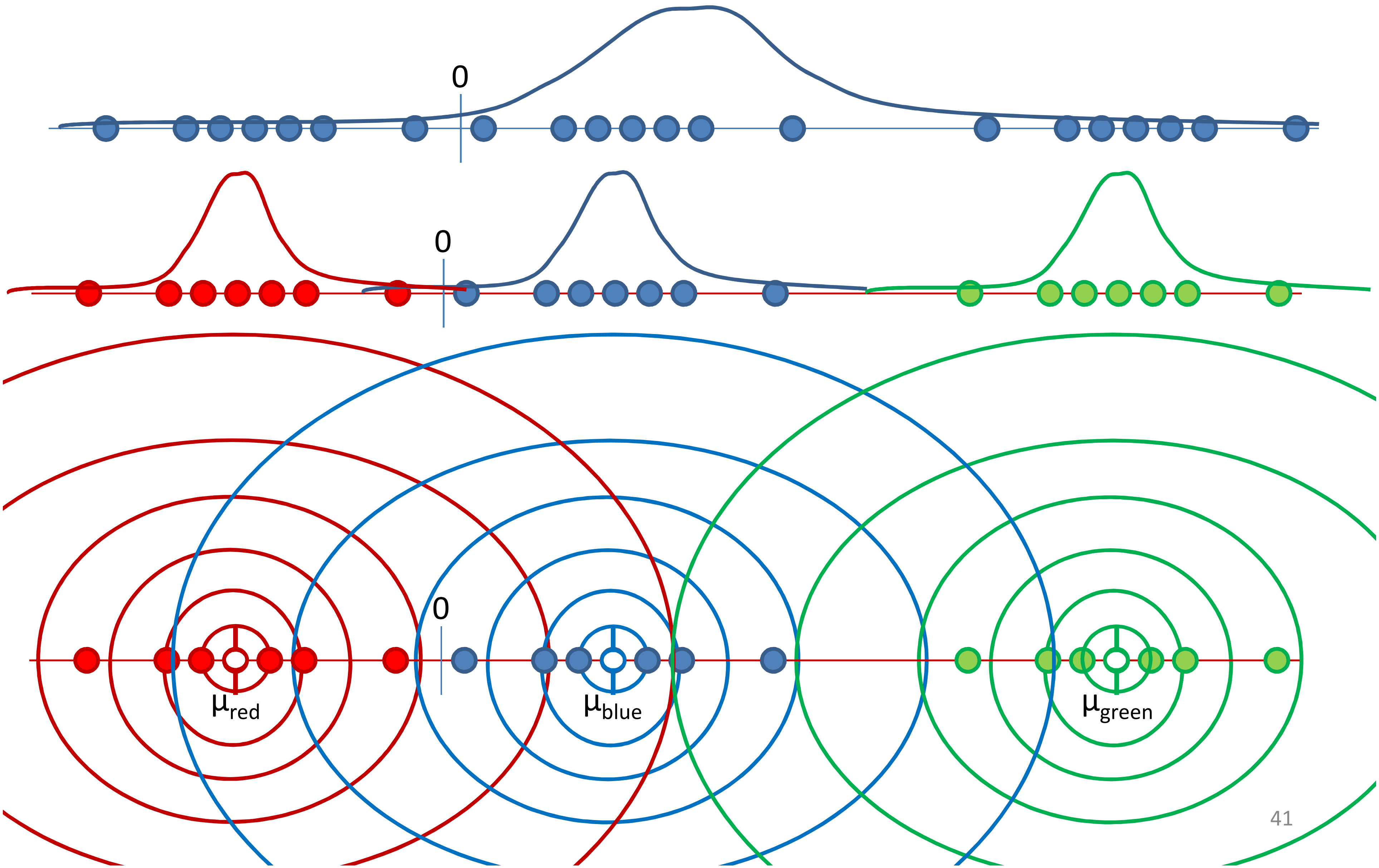
- Gaussian or normal distribution is the most popular continuous probability distribution.
- For example, the distribution of income, distribution of grades in a class.
- Central Limit Theorem
  - Sum of a large number of random variables approaches a Gaussian distribution
- $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
  - μ is the mean or expectation
  - σ is the standard deviation



95% of students at school are between 1.1m and 1.7m tall

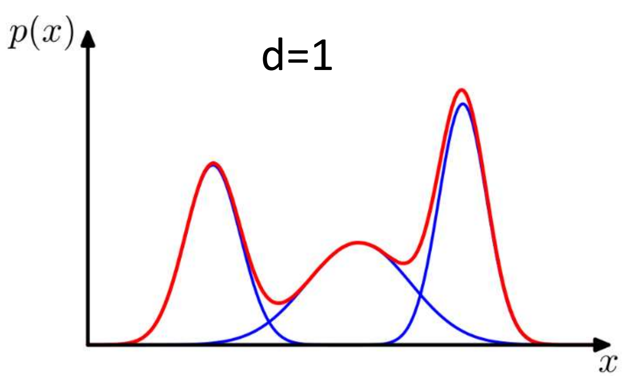
- If a random variable X is distributed normally with mean μ and variance σ^2, it is written as $X \sim N(\mu, \sigma^2)$

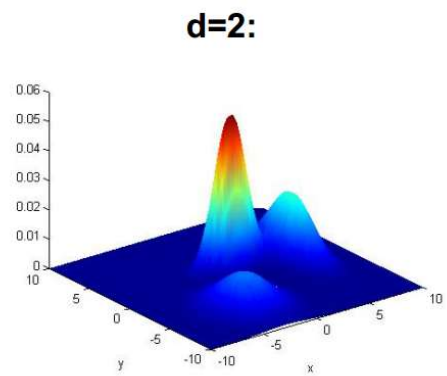
Clustering using Gaussians



Gaussian Mixture Models (GMMs)

- Mixture distribution: It is the probability distribution of a random variable that can be derived from other random variables via simple manipulations.
  - Ex: A Gaussian mixture distribution in 1 dimension as a linear combination of three Gaussians
- Why mixture models?
  - A single Gaussian distribution has limitations when modeling several data sets.
  - If the data has two or more distinct modes as below





Here, a mixture of Gaussians becomes useful.

Each Gaussian can then be considered as a cluster

Gaussian Mixture Models (GMMs)

- GMM density: $p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k)$
- We have a superposition of *K* Gaussian distributions leading to a mixture of Gaussians, $p(\mathbf{x})$.
- Each Gaussian distribution is called a *component* of the mixture and has a mean of $\boldsymbol{\mu}_k$ and covariance of $\boldsymbol{\Sigma}_k$, and mixing coefficient of π_k.
- **Problem:** Given data points (e.g., marks of all students), how do you estimate π_k, μ_k, Σ_k**?**

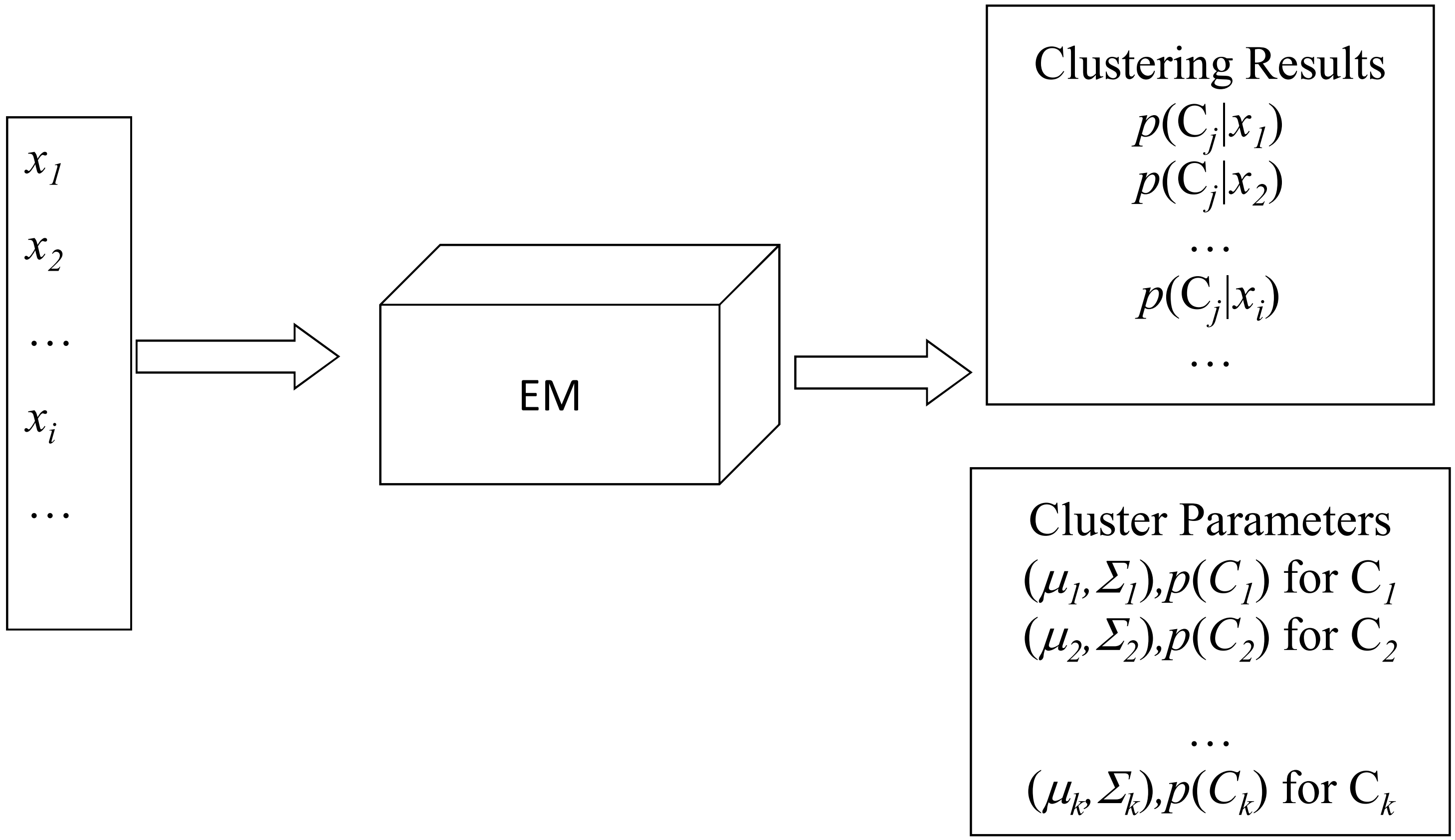
K-Means → EM

- Given N data points $(x_1, x_2, \ldots, x_N)$
- Boot Step:
  - Initialize K clusters: $C_1, \ldots, C_K$
    - (μ_j, Σ_j) and $\pi_j = P(C_j)$ for each cluster j.
- Iteration Step:
  - Estimate the cluster for each data point $p(C_j|x_i)$ ➡ Expectation (EM) Assignment (Kmeans)
  - Re-estimate the cluster parameters ➡ Maximization (EM) Update (Kmeans)
    - $(\mu_j, \Sigma_j), p(C_j)$ for each cluster j

EM Algorithm - Idea

- Initially **guess** the parameters of the model
- Repeat until **convergence**
  - **E step:** Calculate the expectation of the log likelihood function with the current values of the parameters.
  - **M step:** Re-evaluate the parameters of the model by maximizing the expected log likelihood found in the E step.

EM Input/Output



Expectation (E) Step of EM

Input (Known) | Input (Estimation) | Output

x_1
x_2
$\ldots$
x_i
$\ldots$

\+

Cluster Parameters
(μ_1, Σ_1), $p(C_1)$ for C_1
(μ_2, Σ_2), $p(C_2)$ for C_2
$\ldots$
(μ_k, Σ_k), $p(C_k)$ for C_k

$\Rightarrow$

Clustering Results
$p(C_j|x_1)$
$p(C_j|x_2)$
$\ldots$
$p(C_j|x_i)$
$\ldots$

$$p(C_j \mid x_i) = \frac{p(x_i \mid C_j) \cdot p(C_j)}{p(x_i)} = \frac{p(x_i \mid C_j) \cdot p(C_j)}{\sum_j p(x_i \mid C_j) \cdot p(C_j)}$$

$$\mathrm{p}(x_i|C_j) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_j|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1} (x-\mu_j)}$$

Maximization (M) Step of EM

Input (Known)

x_1
x_2
$\dots$
x_i
$\dots$

+

Input (Estimation)

Classification Results
$p(C_j|x_1)$
$p(C_j|x_2)$
$\dots$
$p(C_j|x_i)$
$\dots$

$\Rightarrow$

Output

Cluster Parameters
(μ_1, Σ_1), $p(C_1)$ for C_1
(μ_2, Σ_2), $p(C_2)$ for C_2
$\dots$
(μ_k, Σ_k), $p(C_k)$ for C_k

$$\mu_j = \frac{\sum_i p(C_j \mid x_i) \cdot x_i}{\sum_i p(C_j \mid x_i)} \qquad \Sigma_j = \frac{\sum_i p(C_j \mid x_i) \cdot (x_i - \mu_j) \cdot (x_i - \mu_j)^T}{\sum_i p(C_j \mid x_i)} \qquad p(C_j) = \frac{\sum_i p(C_j \mid x_i)}{N}$$

EM Algorithm

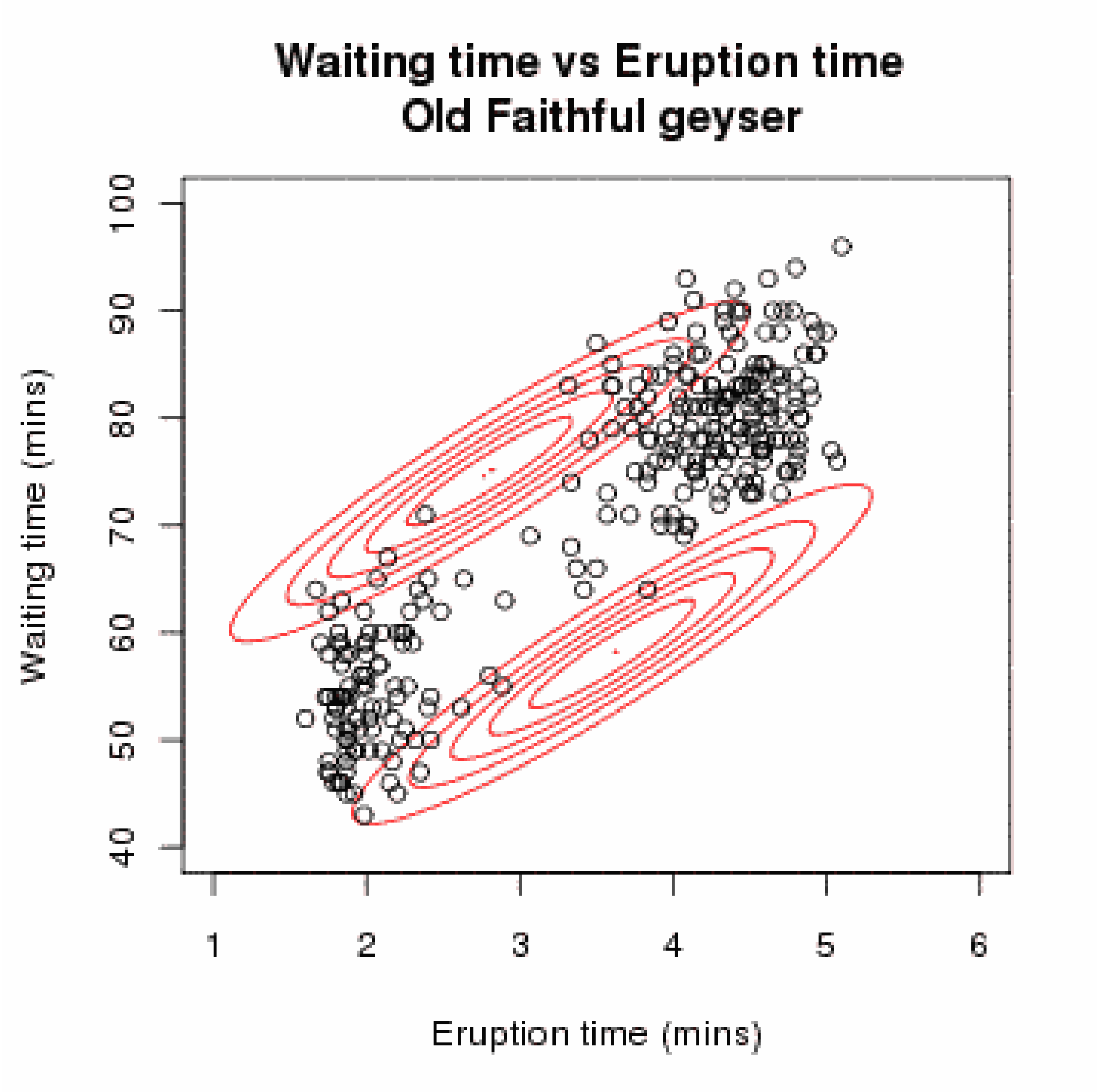
- Given N data points $(x_1, x_2, \ldots, x_N)$
- <u>Boot Step</u>:
  - Initialize K clusters: $C_1, \ldots, C_K$
    - (μ_j, Σ_j) and $\pi_j = P(C_j)$ for each cluster j.
- <u>Iteration Step</u>:
  - Estimate the cluster for each data point ➡ Expectation

$$p(C_j \mid x_i) = \frac{p(x_i \mid C_j) \cdot p(C_j)}{p(x_i)} = \frac{p(x_i \mid C_j) \cdot p(C_j)}{\sum_j p(x_i \mid C_j) \cdot p(C_j)}$$

  - Re-estimate the cluster parameters
    - $(\mu_j, \Sigma_j), p(C_j)$ for each cluster j ➡ Maximization

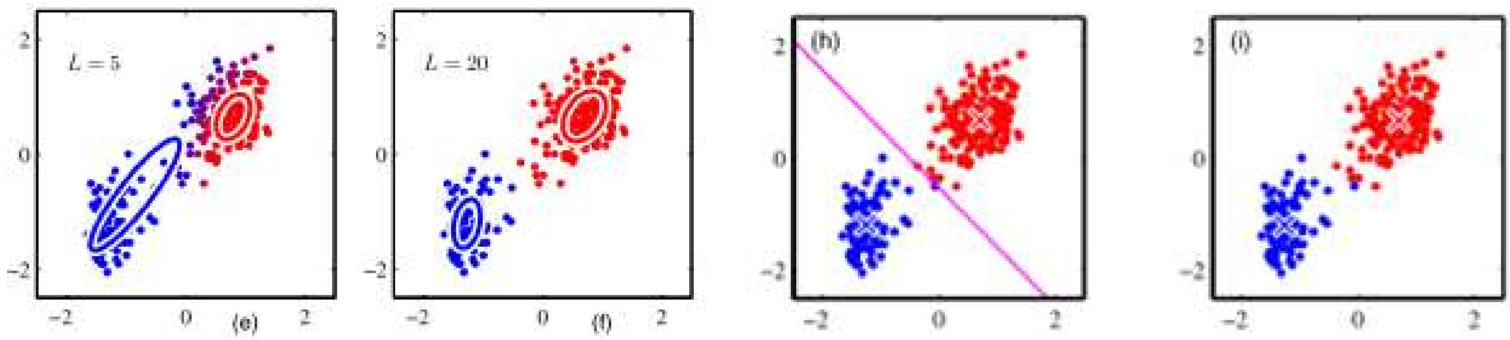
$$\mu_j = \frac{\sum_i p(C_j \mid x_i) \cdot x_i}{\sum_i p(C_j \mid x_i)} \quad \Sigma_j = \frac{\sum_i p(C_j \mid x_i) \cdot (x_i - \mu_j) \cdot (x_i - \mu_j)^T}{\sum_i p(C_j \mid x_i)} \quad p(C_j) = \frac{\sum_i p(C_j \mid x_i)}{N}$$

EM Algorithm Demo



EM and K-Means

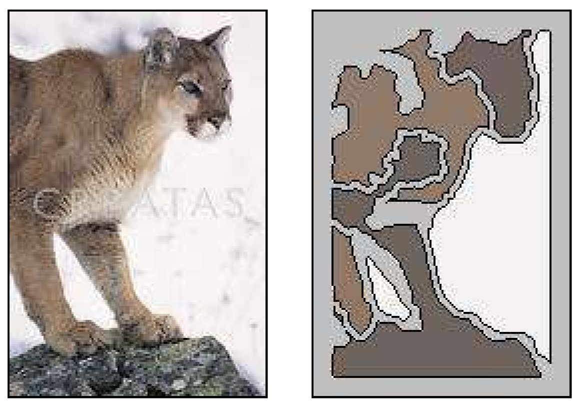
- There is a close similarity.
- *K*-means algorithm performs a hard assignment of data points to clusters.
- EM algorithm makes a *soft* assignment.



- We can derive *K*-means algorithm as a limiting case of EM for Gaussian mixtures.

Image Segmentation using EM

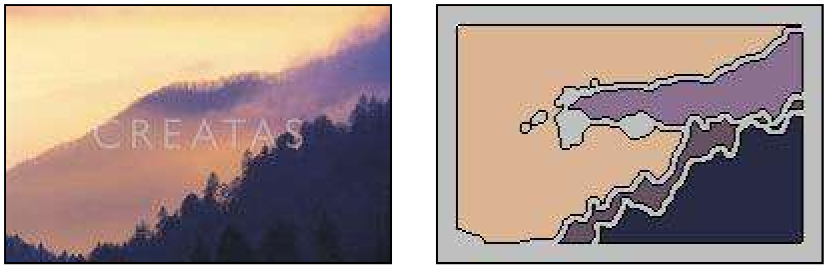
- Step 1: Feature Extraction
- Step 2: Image Segmentation using EM: Break up the image into meaningful or perceptually similar regions

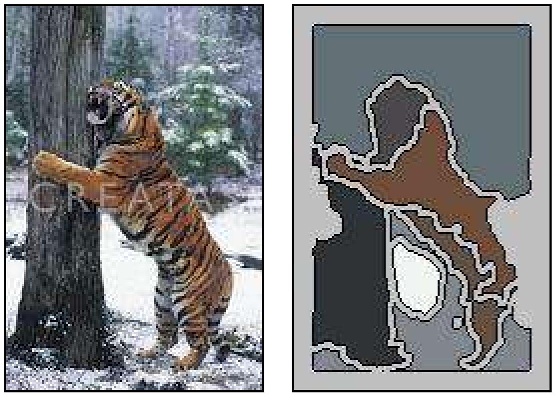


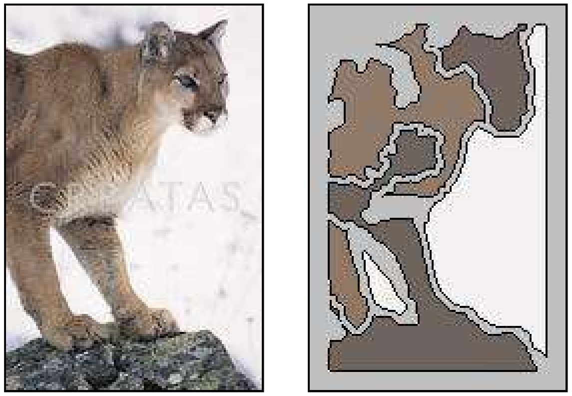
Symbols

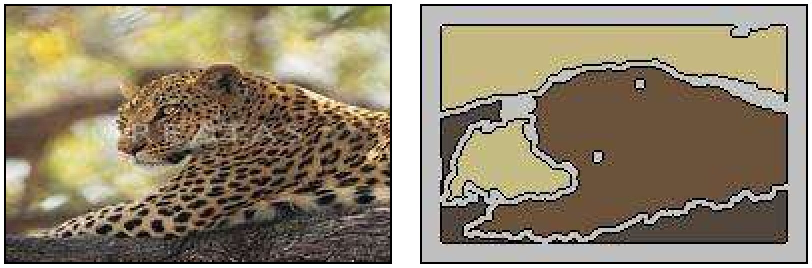
- The feature vector for pixel i is called x_i.
- There are going to be K segments; K is given.
- Gaussian Mixture Model
  - The j-th segment has a Gaussian distribution with parameters $\theta_j=(\mu_j,\Sigma_j)$.
  - π_j's are the weights (which sum to 1) of Gaussians. Θ is the collection of parameters
    - $\Theta = (\pi_1, \ldots, \pi_k, \theta_1, \ldots, \theta_k)$
- Initialization
  - The covariance matrices are initialized to be the identity matrix.
  - The means can be initialized by finding the average feature vectors in each of K windows in the image; this is data-driven initialization.

Sample Results









More Segmentation Results




More Segmentation Results

http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Evaluation Metric for GMMs and EM

- Likelihood = p(X, clusters$|\mu_1, \sigma_1^2, \pi_1$, …, μ_K, σ_K^2, π_K)
- Marginal likelihood = p(X$|\mu_1, \sigma_1^2, \pi_1$, …, μ_K, σ_K^2, π_K)
- EM aims at finding the MLE of the marginal likelihood
  - Indirectly using the iterative formulation
- EM (locally) maximizes the "Marginal" Likelihood
  - EM(X_1, …, X_M) = argmax$[\mu_1, \sigma_1^2, \pi_1$, …, $\mu_K, \sigma_K^2, \pi_K]$ p(X_1,…X_M | μ_1, σ_1^2, π_1, …, μ_K, σ_K^2, π_K)

Analysis of EM Performance

EM is guaranteed to find a local optimum of the Likelihood function.

Theorem: After one iteration of EM, the Likelihood of the new GMM >= the Likelihood of the previous GMM.

(Dempster, A.P.; Laird, N.M.; Rubin, D.B. 1977. "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society. Series B (Methodological)* **39** (1): 1–38.JSTOR 2984875.)

Applications of EM

- NLP
  - Forward-backward algorithm related to Hidden Markov Models (HMM)
  - Inside-outside algorithm with Probabilistic Context Free Grammars (PCFG)
  - Parameter estimation for machine translation
- IR
  - Estimation of weights in interpolation for language modeling
  - Collection clustering
  - Cluster-based retrieval
- Cracking 250 year old codes (http://www.wired.com/dangerroom/2012/11/ff-the-manuscript/all/)
- Computational Biology
  - Gene expression clustering
  - Motif finding
  - Haplotype inference problem
  - Learning profiles of protein domains and RNA families
  - Discovery of transcriptional modules
  - Tests of linkage disequilibrium
  - Protein identification
  - Medical imaging
- Image Processing
  - Image segmentation
  - Object class recognition
  - Object detection
  - Analyzing articulated motion
- Many more …

Take-away Messages

- We studied two main techniques today
  - SVD
  - EM
- SVD is used for latent semantic indexing, while EM has a number of applications
- Latent semantic indexing handles the problem of synonymy and word co-occurrences thereby helping us to move from word-based doc representation to concept-based one.
- With respect to EM
  - We saw how EM can be used to estimate GMMs
  - Next lecture we will see another application of EM

Further Reading

- Dumais, S. T., Furnas, G. W., Landauer, T. K. and Deerwester, S. (1988), "Using latent semantic analysis to improve information retrieval." In Proceedings of CHI'88: Conference on Human Factors in Computing, New York: ACM, 281-285.
- Deerwester, S., Dumais, S. T., Landauer, T. K., Furnas, G. W. and Harshman, R.A. (1990) "Indexing by latent semantic analysis." Journal of the Society for Information Science, 41(6), 391-407.
- Foltz, P. W. (1990) "Using Latent Semantic Indexing for Information Filtering". In R. B. Allen (Ed.) Proceedings of the Conference on Office Information Systems, Cambridge, MA, 40-47.
- Chapter 16 and 18 of Manning-Raghavan-Schuetze book
  - http://nlp.stanford.edu/IR-book/
- http://en.wikipedia.org/wiki/Singular_value_decomposition
- http://en.wikipedia.org/wiki/Latent_semantic_indexing
- http://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm
- EM for NLP: http://www.cs.jhu.edu/~jason/465/PowerPoint/lect26-em.ppt
- EM for Computational Biology: http://www.nature.com/nbt/journal/v26/n8/full/nbt1406.html

Preview of Lecture 5:Topic Models

- Probabilistic Latent Semantic Analysis (PLSA)
- Latent Dirichlet Allocation (LDA)
- Dirichlet Process
- Pachinko Allocation

Disclaimers

- This course will represent opinions of the instructor. It does not reflect views of Microsoft or any other entity.
- Algorithms, techniques, features, etc mentioned here might or might not be in use by Microsoft or any other company.
- Lot of material covered in this course is borrowed from slides across many universities and conference tutorials. These are gratefully acknowledged.

Thanks!

LSI Example (1)

Apply the LSA method to the following technical memo titles

c1: *Human* machine *interface* for ABC *computer* applications

c2: A *survey* of *user* opinion of *computer system response time*

c3: The *EPS user interface* management *system*

c4: *System* and *human system* engineering testing of *EPS*

c5: Relation of *user* perceived *response time* to error measurement

m1: The generation of random, binary, ordered *trees*

m2: The intersection *graph* of paths in *trees*

m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering

m4: *Graph minors*: A *survey*

LSI Example (2)

First we construct the term-document matrix

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Compute U, Σ, V and then perform rank-2 approximation and reconstruct the term-document matrix

LSI Example (3)

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

*The word **user** seems to have presence in the documents where the word **human** appears*

Special Case of Jensen's Inequality

Lemma: **If** p(x) **and** q(x) **are two discrete probability distributions, then:**

$$\sum_x p(x)\log p(x) \geq \sum_x p(x)\log q(x)$$

with equality if and only if p(x) = q(x) **for all** x.

Proof:

$$\sum_x p(x)\log p(x) - \sum_x p(x)\log q(x) \geq 0$$

$$\sum_x p(x)\log\big(p(x) - q(x)\big) \geq 0$$

$$\sum_x p(x)\log\left(\frac{p(x)}{q(x)}\right) \geq 0$$

$$\sum_x p(x)\log\frac{q(x)}{p(x)} \leq 0$$

$$\sum_x p(x)\log\frac{q(x)}{p(x)} \leq \sum_x p(x)(\frac{q(x)}{p(x)} - 1)$$

The last step follows using a bound for the natural logarithm: $\ln(x) \leq x - 1$

Special Case of Jensen's Inequality

Continuing in efforts to simplify:

$$\sum_x p(x)\log\frac{q(x)}{p(x)} \le \sum_x p(x)(\frac{q(x)}{p(x)}-1) = \sum_x p(x)\left(\frac{q(x)}{p(x)}\right) - \sum_x p(x) = \sum_x q(x) - \sum_x p(x) = 0$$

We note that since both of these functions are probability distributions, they must sum to 1.0. Therefore, the inequality holds.

The general form of Jensen's inequality relates a convex function of an integral to the integral of the convex function and is used extensively in information theory.

The EM Theorem

Theorem: If $\sum_t P_{\theta'}(t|y)\log(P_\theta(t,y)) > \sum_t P_{\theta'}(t|y)\log(P_{\theta'}(t,y))$ **then** $P_\theta(y) > P_{\theta'}(y)$ **.**

Proof: Let y **denote observable data. Let** $P_{\theta'}(y)$ **be the probability distribution of** y **under some model whose parameters are denoted by** θ'**.**

Let $P_\theta(y)$ **be the corresponding distribution under a different setting** θ **.**

Our goal is to prove that y **is more likely under** θ **than** θ' **.**

Let t **denote some hidden, or latent, parameters that are governed by the values of** θ **. Because** $P_{\theta'}(t|y)$ **is a probability distribution that sums to** 1**, we can write:**

$$\log P_\theta(y) - \log P_{\theta'}(y) = \sum_t P_{\theta'}(t|y)\log P_\theta(y) - \sum_t P_{\theta'}(t|y)\log P_{\theta'}(y)$$

Because we can exploit the dependence of y on t and using well-known properties of a conditional probability distribution.

Proof Of The EM Theorem

We can multiply each term by "1":

$$\log P_\theta(y) - \log P_{\theta'}(y) = \sum_t P_{\theta'}(t|y)\log\left(P_\theta(y)\frac{P_\theta(t,y)}{P_\theta(t,y)}\right) - \sum_t P_{\theta'}(t|y)\log\left(P_{\theta'}(y)\frac{P_{\theta'}(t,y)}{P_{\theta'}(t,y)}\right)$$

$$= \sum_t P_{\theta'}(t|y)\log\left(\frac{P_\theta(t,y)}{P_\theta(t|y)}\right) - \sum_t P_{\theta'}(t|y)\log\left(\frac{P_{\theta'}(t,y)}{P_{\theta'}(t|y)}\right)$$

$$= \sum_t P_{\theta'}(t|y)\log(P_\theta(t,y)) - \sum_t P_{\theta'}(t|y)\log(P_{\theta'}(t,y))$$

$$+ \sum_t P_{\theta'}(t|y)\log(P_{\theta'}(t|y)) - \sum_t P_{\theta'}(t|y)\log(P_\theta(t|y))$$

$$\geq \sum_t P_{\theta'}(t|y)\log(P_\theta(t,y)) - \sum_t P_{\theta'}(t|y)\log(P_{\theta'}(t,y))$$

where the inequality follows from our lemma.

Explanation: What exactly have we shown? If the last quantity is greater than zero, then the new model will be better than the old model. This suggests a strategy for finding the new parameters, θ – choose them to make the last quantity positive!

Discussion

- **If we start with the parameter setting θ', and find a parameter setting θ for which our inequality holds, then the observed data, y, will be more probable under θ than θ'.**

- **The name Expectation Maximization comes about because we take the expectation of $P_\theta(t,y)$ with respect to the old distribution $P_{\theta'}(t,y)$ and then maximize the expectation as a function of the argument θ.**

- **We can find a $\boldsymbol{\theta}$ that maximizes $\sum_t \boldsymbol{P_{\theta'}(t|y)}$log $\boldsymbol{P_\theta(t,y)}$ by taking the partial derivatives wrt parameters in $\boldsymbol{\theta}$**

- **Critical to the success of the algorithm is the choice of the proper intermediate variable, t, that will allow finding the maximum of the expectation of $\sum_t P_{\theta'}(t|y)\log(P_\theta(t|y))$.**

EM Theorem: why?

- Why optimizing $\sum_{\mathbf{t}} \mathbf{P}_{\boldsymbol{\theta}'}(\mathbf{t}|\mathbf{y}) \mathbf{log}\ \mathbf{P}_{\boldsymbol{\theta}}(\mathbf{t}, \mathbf{y})$ is easier than optimizing $\log \mathrm{P}_{\theta}(\mathrm{y})$
- $P_\theta(t, y)$ involves the complete data and is usually a product of a set of parameters. $P_\theta(y)$ usually involves summation over all hidden variables.