Linear Algebra Methods for Data Mining

Saara Hyvönen, Saara.Hyvonen@cs.helsinki.fi

Spring 2007

Clustering & Information Retrieval

A general comment on preprosessing data matrices

- Has great impact as to how methods perform.
- Centering: subtract mean from each column (rows are data points).
- Standardizing: divide each column by std.
- Normalization: normalize each data point to have norm $=1$.
- What to do depends on your problem!!

LSI

- Normalize columns to have length $=1$.
- Compute the SVD of the term-by-document matrix: $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, and approximate $\mathbf{A}$ by

$$\mathbf{A} \approx \mathbf{U}_k(\mathbf{\Sigma}_k\mathbf{V}_k^T) = \mathbf{U}_k\mathbf{D}_k.$$

- $\mathbf{U}_k$: orthogonal basis, that we use to approximate all the documents
- $\mathbf{D}_k$ is the projection of $\mathbf{A}$ onto the subspace spanned by $\mathbf{U}_k$. Column j hold the coordinates of document j in the new basis.

In practice,

- Documents are expressed in the basis consisting by the k first singular vectors.
- Also queries are projected into the subspace spanned by the k first singular vectors.
- Query matching is done in this reduced space.

Why does LSI work?

"Rank reduction removes the noise that obscures the semantic content of the data."

(Jessup and Martin, Taking a new look at the LSA approach to information retrieval, 2001.)

"LSI is based on the assumption that there is some underlying latent semantic structure in the data... that is corrupted by the wide variety of words used..."

(Park et al, Lower dimensional representation of text data in vector space based information retrieval, 2001.)

"automatic association of related terms"

(Berry, Understanding Search Engines, 1999)

LSI

- Data compression by SVD: improved search results.
- Surprisingly low rank gives good results. BUT:
- Answer to the question why is still lacking.
- Difficult to update SVD for new terms and new documents.
- Singular vectors are non-sparse.
- Alternatives to SVD? Cluster-based rank reduction?

Approximation result from SVD

The matrix $\mathbf{B} = \mathbf{U}_k\mathbf{D}_k$ gives the solution to the approximation problem

$$\min_{\mathsf{rank}(\mathbf{B})\leq k} \|\mathbf{A} - \mathbf{B}\|_2.$$

So, if we think of the column vectors of $\mathbf{U}_k$ as a basis, then $\mathbf{D}_k$ gives the least squares solution to

$$\min_{\mathbf{D}} \|\mathbf{A} - \mathbf{U}_k\mathbf{D}\|_2.$$

But what if we would choose a different basis?

Motivation

We can write any rank k approximation of $\mathbf{A} \in \mathbb{R}^{m \times n}$ as a product of two matrices: $\mathbf{BC}$, where $\mathbf{B} \in \mathbb{R}^{m \times k}$, $\mathbf{C} \in \mathbb{R}^{k \times n}$.

SVD gives the best *approximation* of $\mathbf{A}$ in terms of minimizing distance between $\mathbf{A}$ and $\mathbf{BC}$ (in the euclidean norm).

BUT: some other choices of $\mathbf{B}$ and $\mathbf{C}$ might give a better (reduced dimensional) *representation* of the original documents.

Clusters

Assume that the columns of $\mathbf{A}$ have been grouped in k clusters:

$$\mathbf{A} = (\mathbf{A}_1, ..., \mathbf{A}_k).$$

Represent each cluster by its mean, **centroid**:

$$\mathbf{c}_i = \frac{1}{n_i}\mathbf{A}_i\mathbf{e}, \quad \mathbf{e} = (1, 1, ..., 1)^T, \quad i = 1...k,$$

and n_i is the number of columns in $\mathbf{A}_i$. Let $\mathbf{C} = (\mathbf{c}_1, ..., \mathbf{c}_k)$ be the matrix of cluster centroids.

Think of the cluster centroids $(\mathbf{c}_1, ..., \mathbf{c}_k)$ as basis vectors.

Find the coordinates $\mathbf{Y}$ of all the columns of $\mathbf{A}$ in terms of this basis. That is, solve the least squares problem:

$$\min_{\mathbf{Y}} \|\mathbf{A} - \mathbf{C}\mathbf{Y}\|.$$

The solution can be written

$$\mathbf{Y} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{A}.$$

Note: we have k basis vectors, so we get a rank k approximation for $\mathbf{A}$: $\mathbf{A} \approx \mathbf{C}\mathbf{Y}$.

Now we have the coordinates $\mathbf{Y}$ of the columns of $\mathbf{A}$ in the basis defined by the centroid vectors $(\mathbf{c}_1, ..., \mathbf{c}_k)$.

What about the coordinates of the query $\mathbf{q}$ in terms of the centroids? Again, solve:

$$\min_{\mathbf{y}} \|\mathbf{q} - \mathbf{C}\mathbf{y}\|$$

which gives

$$\mathbf{y} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{q}.$$

Cosines

We have

$$\mathbf{A} \approx \mathbf{CY}, \quad \text{where} \quad \mathbf{Y} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{A},$$
$$\mathbf{q} \approx \mathbf{Cy}, \quad \text{where} \quad \mathbf{y} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{q}.$$

What is the cosine between the query and the j^{th} column of $\mathbf{A}$ in terms of the new basis?

What is the cosine between the query and the j^{th} column of $\mathbf{A}$ in terms of the new basis?

$$\cos(\theta(\mathbf{q}, \mathbf{a}_j)) = \frac{\mathbf{q}^T \mathbf{a}_j}{\|\mathbf{q}\| \|\mathbf{a}_j\|} \approx \frac{\mathbf{q}^T \mathbf{C}\mathbf{Y}\mathbf{e}_j}{\|\mathbf{q}\| \|\mathbf{C}\mathbf{Y}\mathbf{e}_j\|} \approx \frac{\mathbf{y}^T \mathbf{C}^T \mathbf{C}\mathbf{y}_j}{\|\mathbf{C}\mathbf{y}\| \|\mathbf{C}\mathbf{y}_j\|}$$

($\mathbf{a}_j = \mathbf{A}\mathbf{e}_j$ and $\mathbf{y}_j = \mathbf{Y}\mathbf{e}_j$ are the j^{th} columns of $\mathbf{A}$ and $\mathbf{Y}$ respectively.)

More natural to orthogonalize the centroid matrix:

$$\mathbf{C} = \mathbf{Q} \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} = \mathbf{Q}_k \mathbf{R}.$$

Now compute the coordinates of the columns of $\mathbf{A}$ in terms of the columns of $\mathbf{Q}_k$. Least squares problem:

$$\min_{\mathbf{Z}} \|\mathbf{A} - \mathbf{Q}_k \mathbf{Z}\|.$$

Solution:

$$\mathbf{Z} = \mathbf{Q}_k^T \mathbf{A}.$$

Cosines

This time

$$\mathbf{A} \approx \mathbf{Q}_k \mathbf{Z}, \quad \text{where} \quad \mathbf{Z} = \mathbf{Q}_k^T \mathbf{A},$$
$$\mathbf{q} \approx \mathbf{Q}_k \mathbf{z}, \quad \text{where} \quad \mathbf{z} = \mathbf{Q}_k^T \mathbf{q}.$$

So the cosines this time are

$$\cos(\theta(\mathbf{q}, \mathbf{a}_j)) = \frac{\mathbf{q}^T \mathbf{a}_j}{\|\mathbf{q}\| \|\mathbf{a}_j\|} \approx \frac{\mathbf{q}^T \mathbf{Q}_k \mathbf{Z} \mathbf{e}_j}{\|\mathbf{q}\| \|\mathbf{Q}_k \mathbf{Z} \mathbf{e}_j\|} \approx \frac{\mathbf{z}^T \mathbf{z}_j}{\|\mathbf{z}\| \|\mathbf{z}_j\|}.$$

The QR based approach is essentially equivalent to the first centroid method:

$$\min_{\mathbf{Y}} \|\mathbf{A} - \mathbf{C}\mathbf{Y}\| = \min_{\mathbf{Y}} \|\mathbf{A} - \mathbf{Q}_k(\mathbf{R}\mathbf{Y})\| = \min_{\mathbf{Z}} \|\mathbf{A} - \mathbf{Q}_k\mathbf{Z}\|.$$

Thus the rank reduction is the same.

Only cosines are different!

Example: Park et al, 2001

Classification: given a clustering, check whether the same classification is obtained after rank reduction.

Data from MEDLINE, 5 categories: heart attack, colon cancer, diabetes, oral cancer, tooth decay.

500 records in each class.

Misclassification rate %

	Full	Centroid	CentroidQR
Dim	22095×2500	5×2500	5×2500
L_2	11.8	8.5	11.8
Cosine	7.8	8.5	7.8

MEDLINE, same 5 clusters, use 200 documents to compute the centroids.

Self classification and classification of 200 new documents. 7519 terms.

Misclassification rate %

	Centroid		CentroidQR		LSI	
reduced dim	self	new	self	new	self	new
5	2.0	17	3.5	16	25.5	27.5
10					10.5	19.5
20					6.5	17
50					6.0	16.5
100					6.0	14
full(7519)					3.5	16

Example: Medline, query matching

Conculsions (of examples)

If the data is well clustered, then the cluster based rank reduction works much better than LSI (in terms of rank and misclassification rate).

Concept Decomposition

Dhillon & Modha, 2000.

Equivalent to least squares approximation based on centroid vectors.

Concept vector = centroid.

Almost... more specific on clustering algorithm.

Concept decomposition

Given a matrix of concepts=centroids $\mathbf{C}_k = (\mathbf{c}_1, ..., \mathbf{c}_k)$, compute the coordinates of all documents in terms of the centroids by solving

$$\min_{\mathbf{Y}} \|\mathbf{A} - \mathbf{C}_k\mathbf{Y}\|.$$

Exactly the same problem as before!

Only difference: concepts are obtained by using spherical k-means.

Comparison between concept and singular vectors

Dataset: CLASSIC3: documents from MEDLINE, CISI, CRANFIELD: 4099 terms, 3893 documents, 3 clusters.

Angles between subspaces spanned by the concept vectors $\mathbf{c}_k$ and the three first singular vectors:

k	$\cos\theta_1$	$\cos\theta_2$	$\cos\theta_3$
3	.996	.968	.433
4	.996	.989	.557
8	.997	.992	.978

Note:

- Centroids play a very similar role as do singular vectors in LSI.
- **BUT** :Centroids/Concept vectors are sparse. Singular vectors are essentially full.
- With centroids it seems possible to use even smaller reduced dimension than SVD.
- The general idea applicable in other bases as well, BUT you need to have some confidence in that your basis makes sense!

Clustering

- k-means
- spherical k-means (Dhillon and Modha)
- spectral clustering

Aim of clustering:

We wish to partition a data set into groups so that the points in one group are similar to each other and are as different as possible from the points in other groups.

Minimize distance within clusters, maximize distance between clusters.

K-means

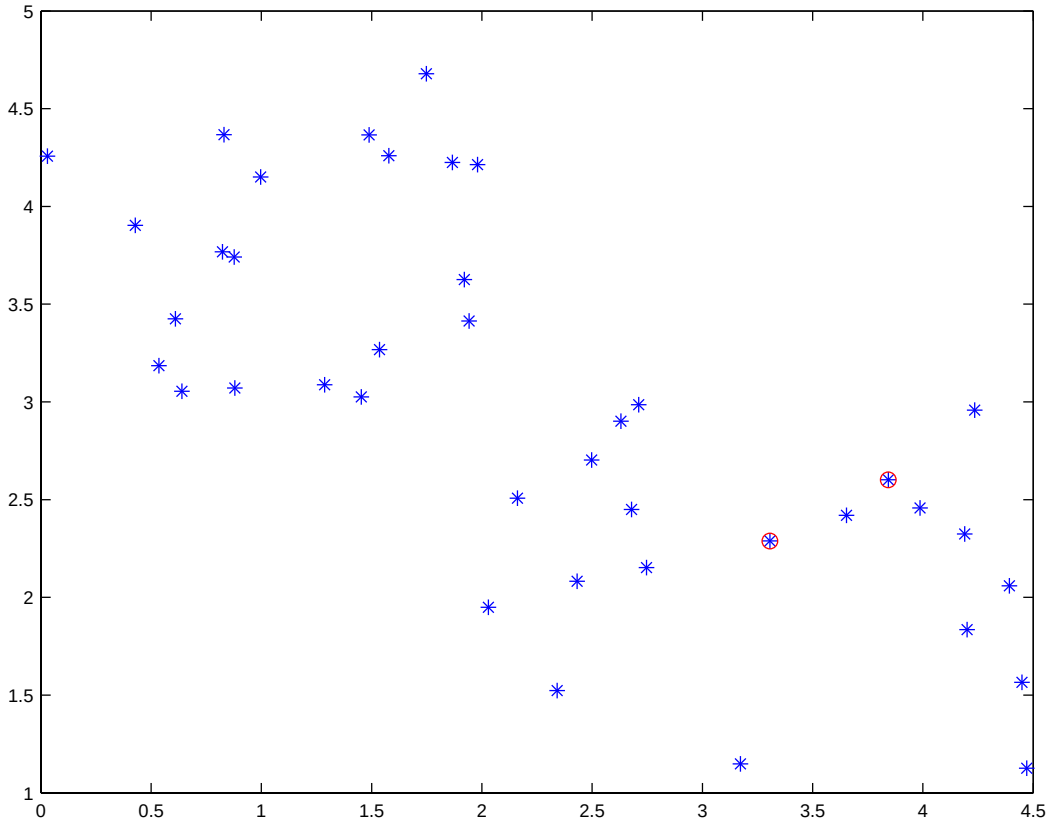
We have n data points $(\mathbf{x}_1,\ ...,\ \mathbf{x}_n)$.
We wish to partition them into k disjoint clusters $C_1, ..., C_k$.

1. Choose at random (from the set of data points) k cluster centers (centroids) $\mathbf{m}_j$, $j = 1...k$.

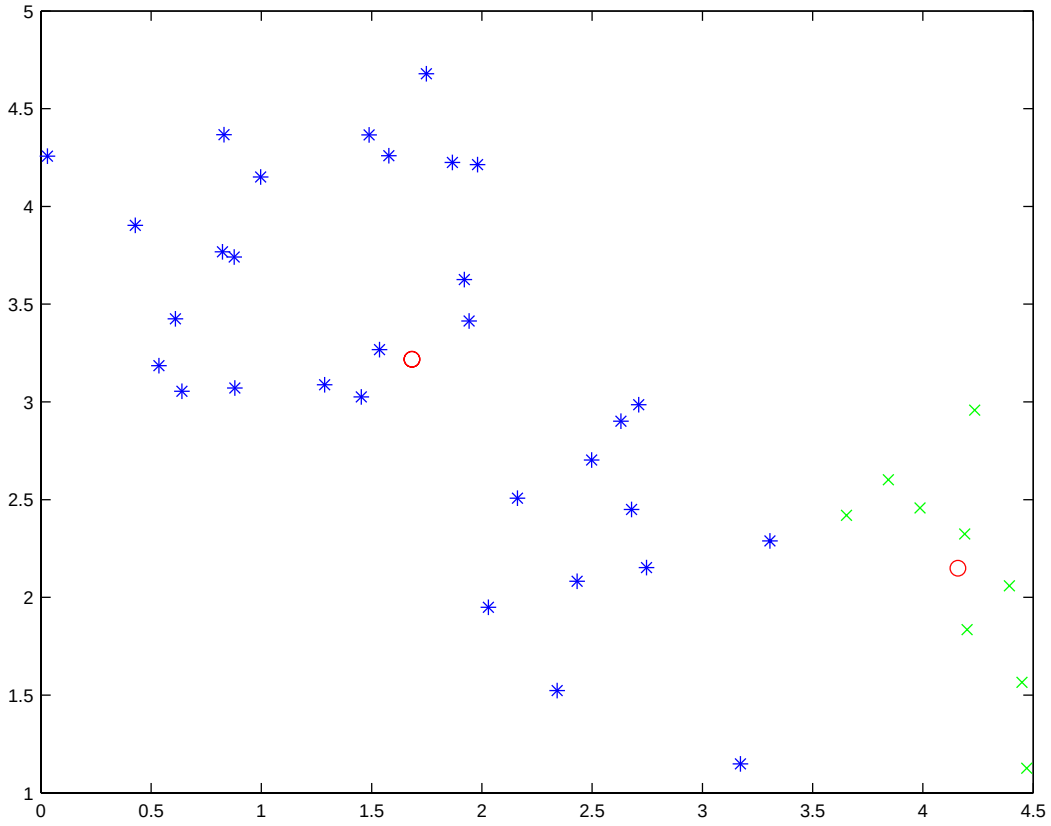
2. While changes in clusters C_j happen,
   - form clusters: assign all points closest to $\mathbf{m}_j$ to the cluster C_j.
   - compute new centroids: $\mathbf{m}_j$ = mean of all points in C_j.

"Closest"? Euclidean, cosine, manhattan, ...

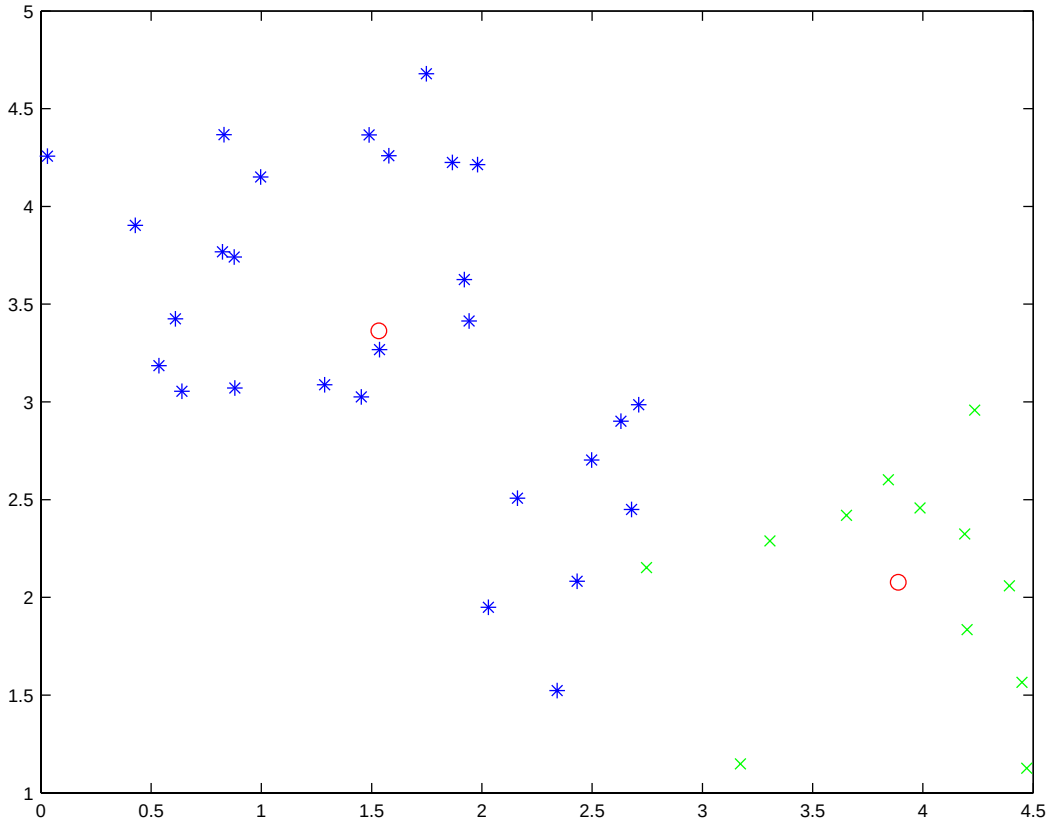
Step 1:



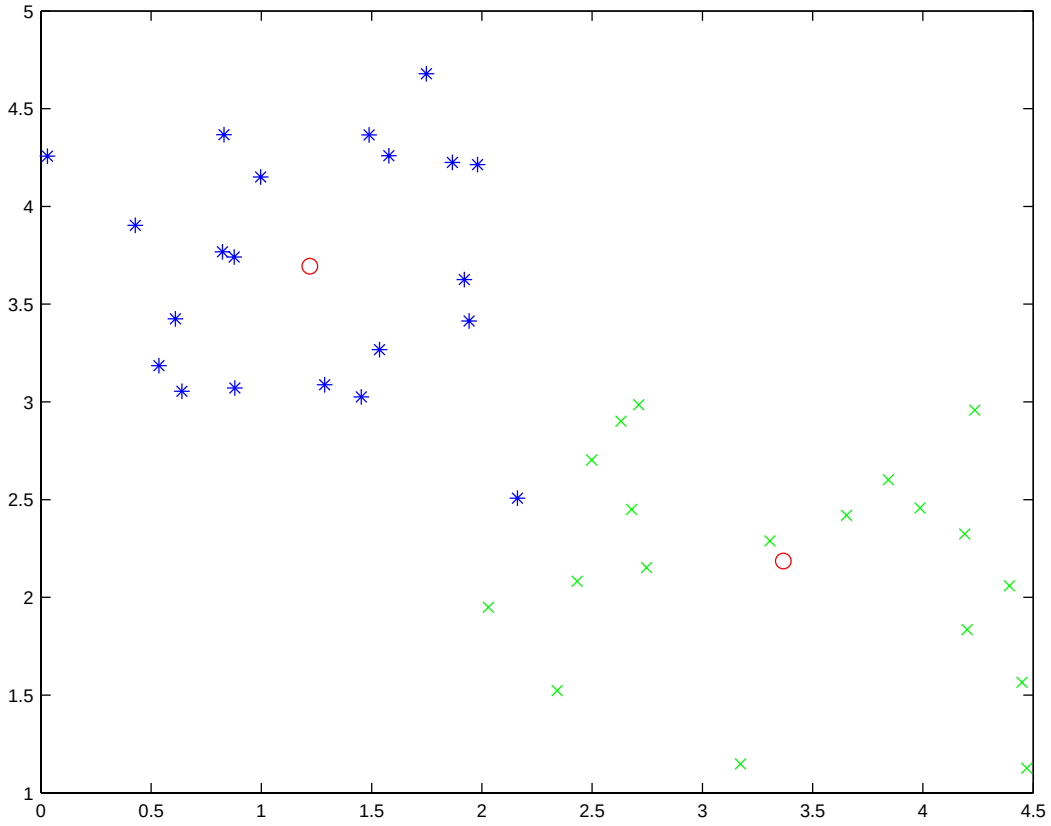
Step 2:



Step 3:



Step 5:



Spherical k-means

Disjoint clusters $C_1, ..., C_k$.

Centroid of cluster j: $\mathbf{m}_j = \frac{1}{n_j} \sum_{\mathbf{x} \in C_j} \mathbf{x}$.

Concept vector: $\mathbf{c}_j = \frac{1}{\|\mathbf{m}_j\|} \mathbf{m}_j$.

The concept vector is closest to all the document vectors in C_j (measured by cosine similarity). Let $\mathbf{z}$ be arbitrary (normalized):

$$\sum_{\mathbf{x} \in C_j} \mathbf{x}^T \mathbf{z} \leq \sum_{\mathbf{x} \in C_j} \mathbf{x}^T \mathbf{c}_j = n_j \|\mathbf{m}_j\| \mathbf{c}_j^T \mathbf{c}_j.$$

Coherence ("quality") of cluster j: $\sum_{\mathbf{x} \in C_j} \mathbf{x}^T \mathbf{c}_j$.

Coherence ("quality") of cluster j: $\sum_{\mathbf{x} \in C_j} \mathbf{x}^T \mathbf{c}_j$.

Quality of given partitioning:

$$Q(C_1, ..., C_k) = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} \mathbf{x}^T \mathbf{c}_j.$$

Determine the partitioning that maximizes Q.

(Optimal solution NP-complete).

Spherical k-means

We have n data points $(\mathbf{x}_1, \ ..., \ \mathbf{x}_n)$.
We wish to partition them into k disjoint clusters $C_1, ..., C_k$.

1. Choose at random k concept vectors $\mathbf{c}_j$, $j = 1...k$.

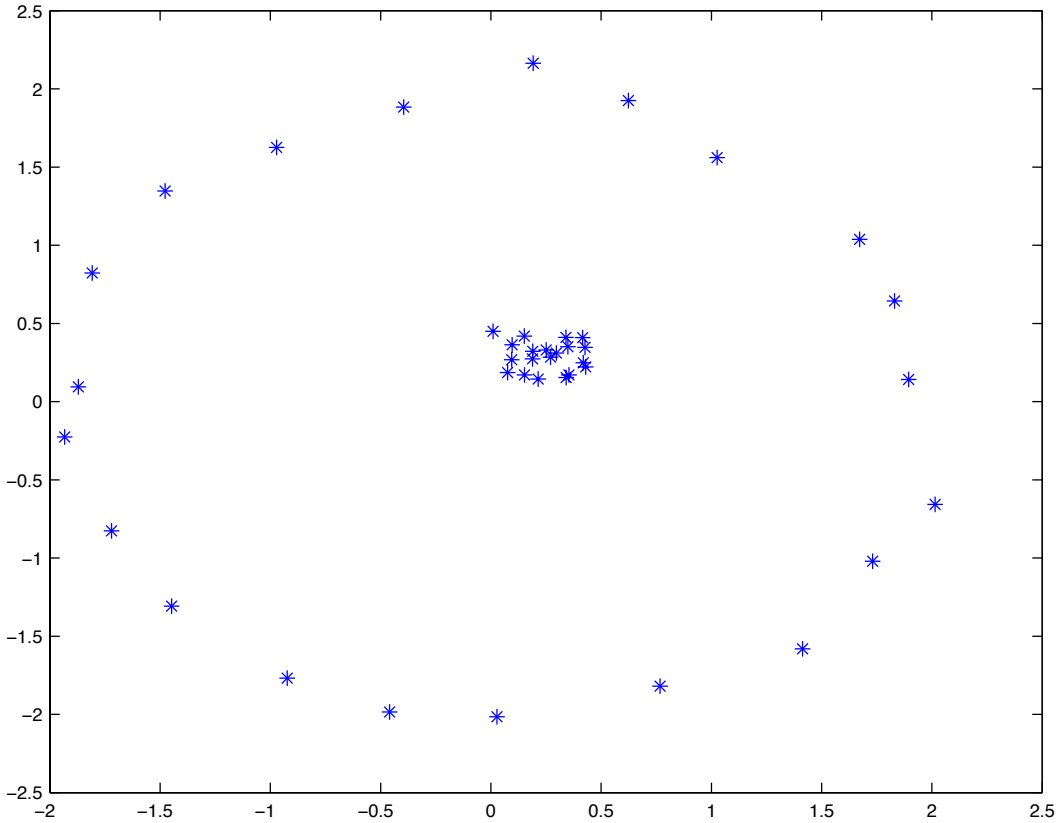
2. While changes in Q-value are above a fixed tolerance,
   - form clusters: assign all points closest (as measured by cosine similarity) to $\mathbf{c}_j$ to the cluster C_j.
   - compute new concept vectors and Q for the new partitioning.

The algorithm finds a local maximum of quality.

Example

Confusion matrix (result of spherical k-means) for CLASSIC3.

MEDLINE	1004	18	11
CISI	5	1440	15
CRANFIELD	4	16	1380


2.5
2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2.5
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
2.5

Spectral clustering

Use methods from spectral graph partitioning to do clustering.

Needed: pairwise distances between data points.

These can be thought of as weights of links in a graph: clustering problem becomes a graph partitioning problem.

Unlike k-means, clusters need not be convex.

Algorithm

We have n data points $(\mathbf{x}_1, \ ..., \ \mathbf{x}_n)$.
We wish to partition them into k disjoint clusters $C_1, ..., C_k$.

1. Form affinity matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ defined by

$$\mathbf{A}_{ij} = \begin{cases} \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2) & \text{if} \quad i \neq j \\ = 0 & \text{if} \quad i = j. \end{cases}$$

2. Define $\mathbf{D}$ to be the diagonal matrix whose i^{th} diagonal element is the sum of $\mathbf{A}$'s i^{th} row, and construct the matrix

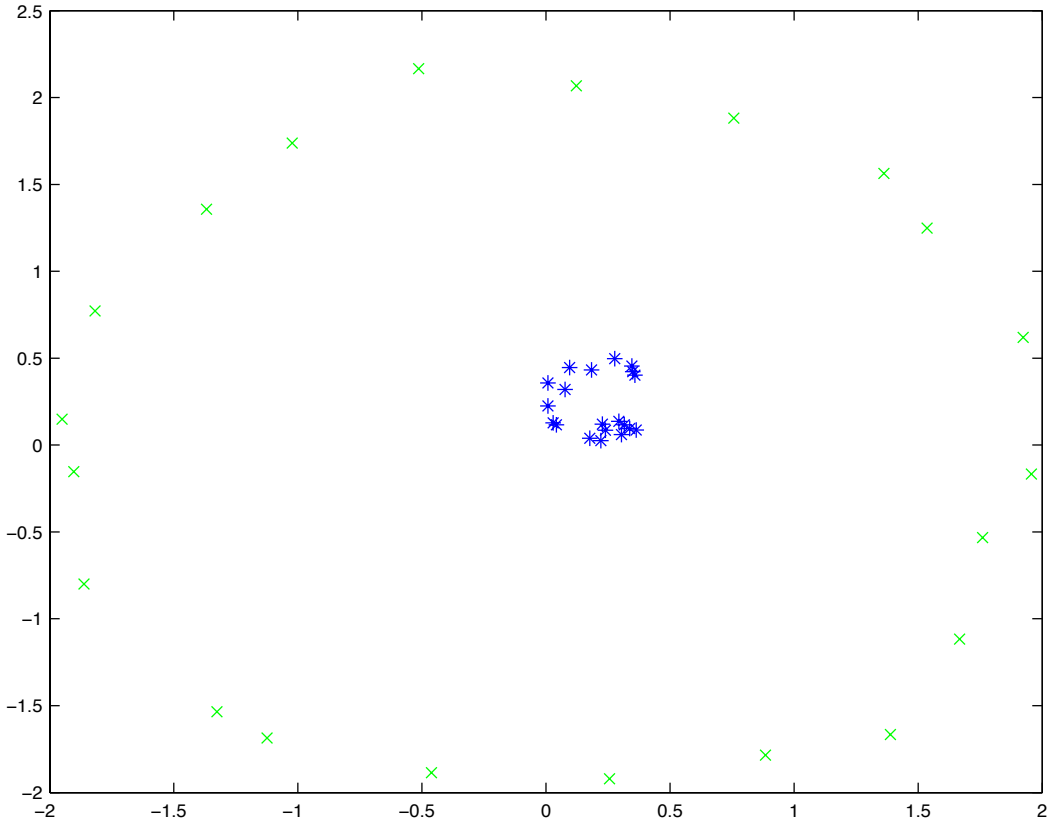
$$\mathbf{L} = \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}.$$

3. Find the eigenvectors $\mathbf{v}_j$ of $\mathbf{L}$ corresponding to the k largest eigenvalues, and form the matrix $\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2\ ...\ \mathbf{v}_k] \in \mathbb{R}^{n \times k}$.

4. Form the matrix $\mathbf{Y}$ from $\mathbf{V}$ by renormalizing each of $\mathbf{V}$'s *rows* to have unit length.

5. Treating each row of $\mathbf{Y}$ as a point in $\mathbb{R}^k$, cluster them into k clusters via k-means (or any other clustering algorithm).

6. If the row i of the matrix $\mathbf{Y}$ was assigned to cluster j, assign the data point $\mathbf{x}_i$ to the cluster j.


2.5
2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2

References

[1] Lars Eldén: Numerical Linear Algebra and Data Mining Applications, Draft.

[2] H. Park, M. Jeon, J.B. Rosen, Lower Dimensional Representation of Text Data based on Centroids and Least Squares, BIT 43, 4427-448, 2003.

[3] I. Dhillon, D. Modha, Concept Decompositions for Large Sparse Text Data using Clustering, 2001.

[4] I A. Ng, M. Jordan, Y. Weiss, On Spectral Clustering: Analysis and and Algorithm, NIPS 2002.