ISTA 421/521
Introduction to Machine Learning

Lecture 23: Clustering
K-Means,
Kernelized K-Means
Mixture Models

Clay Morrison

clayton@sista.arizona.edu

Gould-Simpson 819

Phone 621-6609

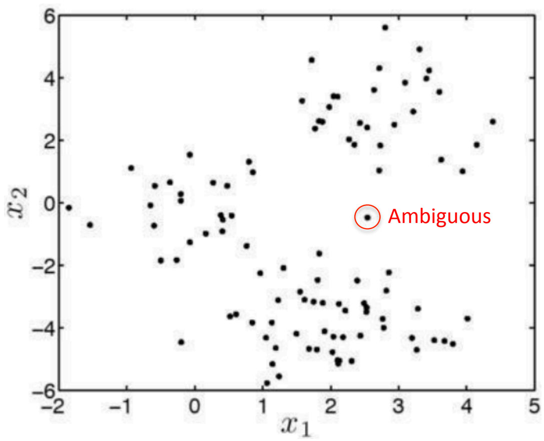
18 November 2014

Clustering

- **Unsupervised learning**: only provided with set of objects **x**
- Goal of **cluster analysis**: create grouping of objects where objects within group are similar and objects in different groups are not similar (or as similar).
- Examples
  - **Customer preference**: lots of data about purchases, used as evidence for personal preferences. Define measure of similarity between customers based on purchasing history: within group similar shopping patterns. Recommend items based on customer similarity. Also cluster items based on customers they were purchased by (items 1 and 2 both bought by customers A, D, E and G).
  - **Gene function prediction**: categorize genes into functional classes based on patterns of mutual interaction in mRNA mircroarray data. Functions of known genes in cluster can be used as predicted function of unknown genes in same cluster.

Example

How many groups?



Many ways of defining similarity in terms of (inverse) distance:

For real valued data…

Euclidean distance:

$$(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)$$

First approach: characterize clusters by centroid; members of clusters are closest to cluster centroid.

K-means Clustering

- Mean point of k^{th} cluster: $\boldsymbol{\mu}_k$
- Binary indicator function: z_{nk}

  1 if object *n* is assigned to cluster *k*, 0 otherwise

  – Each object assigned to one and only one cluster

Algorithm:

(0) Choose *K* (total number of clusters) and initial random cluster means $\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_K$

(1) For each data object, find closest cluster *k* mean and set $z_{nk} = 1$ and $z_{nj} = 0$ for all $j \neq k$.

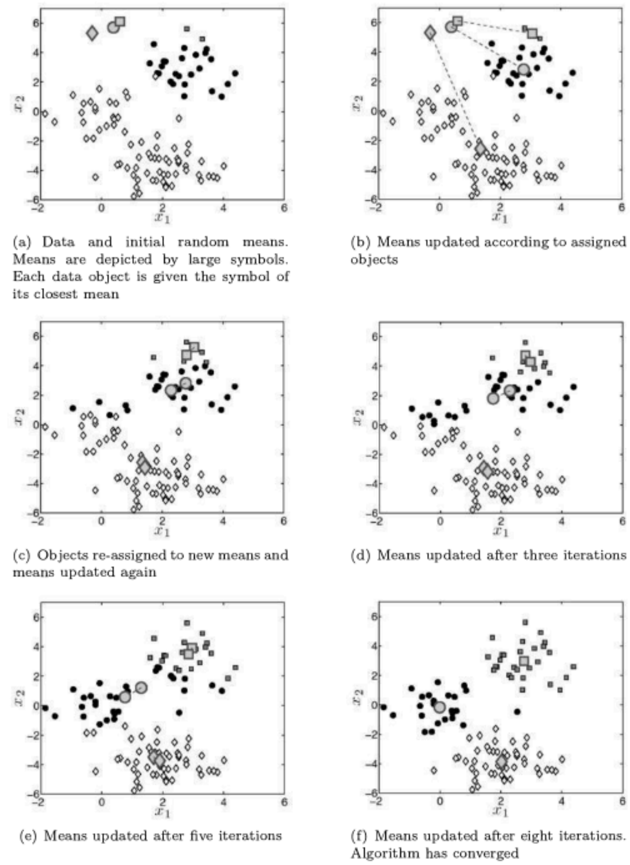
(2) If all the assignments (z_{nk}) are unchanged from the previous iteration, stop.

(3) Update each $\boldsymbol{\mu}_k$

(4) Goto 1

$$\boldsymbol{\mu}_k = \frac{\sum_n z_{nk}\mathbf{x}_n}{\sum_n z_{nk}}$$

K-means Example



(a) Data and initial random means. Means are depicted by large symbols. Each data object is given the symbol of its closest mean

(b) Means updated according to assigned objects

(c) Objects re-assigned to new means and means updated again

(d) Means updated after three iterations

(e) Means updated after five iterations

(f) Means updated after eight iterations. Algorithm has converged

Also, simple Java applet
http://www.math.le.ac.uk/people/ag153/homepage/KmeansKmedoids/Kmeans_Kmedoids.html

K-means Clustering

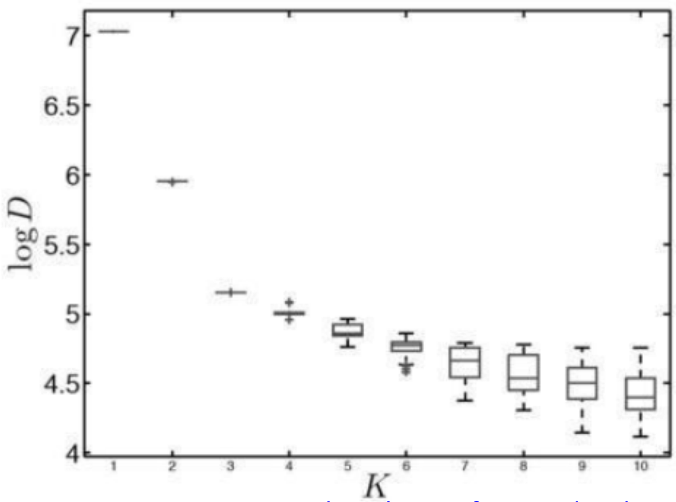
- Finds **local** minimum of total distance of all points to their cluster centers:

$$D = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathsf{T}} (\mathbf{x}_n - \boldsymbol{\mu}_k)$$

- Whether finds ***global*** minimum depends on initial cluster means – can get caught in local minima
- Guaranteed to find global minimum only if evaluate every possible assignment of all *N* points to *K* clusters – intractable
- Typically just run multiple times, select final cluster means with lowest total distance *D*

K-means Clustering

- Choosing the number of clusters: *K*
  - A common problem in cluster analysis



Just minimizing *D* doesn't help

When *K* = *N*, then D = 0 !

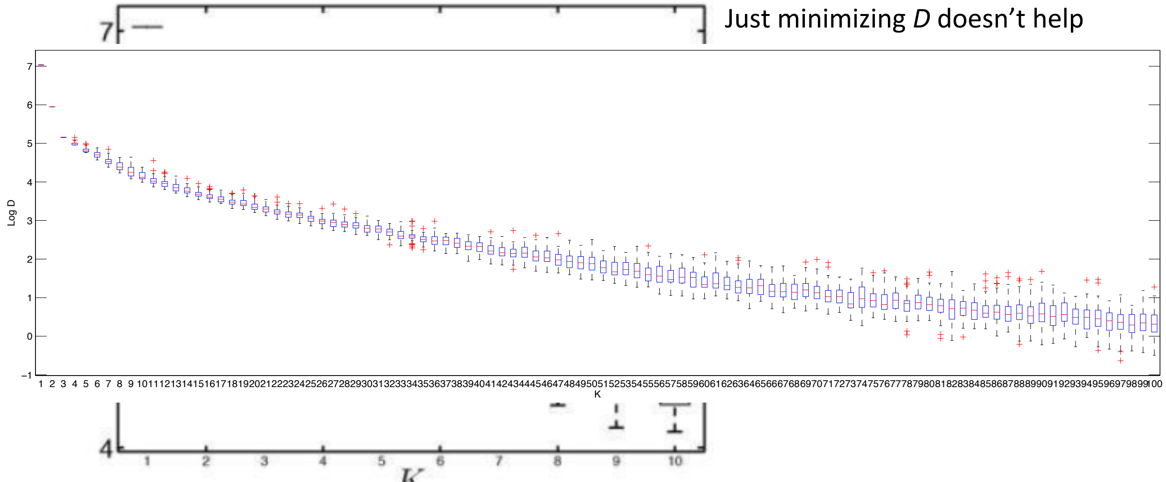
Each box plot is result of 50 random initializations

Since cannot optimize directly, an often used technique for choosing K is to see how use of the clusters works on ***other*** performance tasks

K-means Clustering

- Choosing the number of clusters: *K*
  - A common problem in cluster analysis

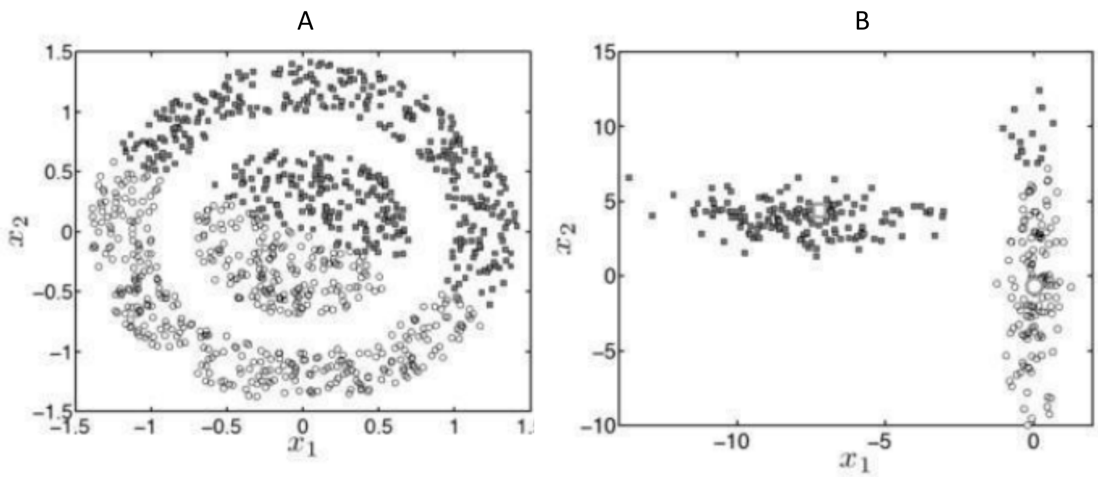
Just minimizing *D* doesn't help



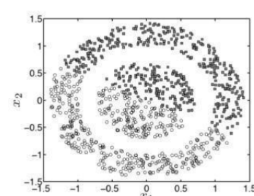
Since cannot optimize directly, an often used technique for choosing K is to see how use of the clusters works on other performance tasks

Where *K*-means fails

- Objects in true clusters do not necessarily conform to current distance-based similarity



Kernelized *K*-means



- A kernelized *K*-means can handle case A
- Derive kernelizable form of distance:

$$d_{nk} = (\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T}(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

$$N_k = \textstyle\sum_{n=1}^{N} z_{nk}$$

$$d_{nk} = \left(\mathbf{x}_n - \frac{1}{N_k}\sum_{m=1}^{N} z_{mk}\mathbf{x}_m\right)^\mathsf{T}\left(\mathbf{x}_n - \frac{1}{N_k}\sum_{r=1}^{N} z_{rk}\mathbf{x}_r\right)$$

Note

NOTE: the means $\boldsymbol{\mu}_k$ doesn't appear here!

Is kernelizable!

$$d_{nk} = K(\mathbf{x}_n, \mathbf{x}_n) - \frac{2}{N_k}\sum_{m=1}^{N} z_{mk}K(\mathbf{x}_n, \mathbf{x}_m) + \frac{1}{N_k^2}\sum_{m=1}^{N}\sum_{r=1}^{N} z_{mk}z_{rk}K(\mathbf{x}_m, \mathbf{x}_r)$$

However, computing $\boldsymbol{\mu}_k$ directly with a transformation…

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} z_{nk}\mathbf{x}_n}{\sum_{n=1}^{N} z_{nk}} \longrightarrow \boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} z_{nk}\phi(\mathbf{x}_n)}{\sum_{n=1}^{N} z_{nk}}$$

Not kernelizable

And when transformation is not explicitly computable, can't even compute the centroid!

Kernelized *K*-means

- Since want to avoid directly needing to compute the means (centroids), need to augment the algorithm:

$$d_{nk} = K(\mathbf{x}_n, \mathbf{x}_n) - \frac{2}{N_k}\sum_{m=1}^{N} z_{mk}K(\mathbf{x}_n, \mathbf{x}_m) + \frac{1}{N_k^2}\sum_{m=1}^{N}\sum_{r=1}^{N} z_{mk}z_{rk}K(\mathbf{x}_m, \mathbf{x}_r)$$

(1) Randomly initialize z_{nk} for each n ← Initializing by object-cluster rather than cluster means!

(2) Compute $d_{n1},\ldots,d_{nk}$ for each object n ← Only requires pairwise inner products (kernelized)

(3) Assign each object to the cluster k with the lowest d_{nk}

- This determines the new z_{nk} for each n

(4) If assignments have changed, goto step 2, otherwise stop.

Two variants of initialization step:

(A) Run regular *K*-means to convergence, then use z_{nk} as seed to Kernelized version

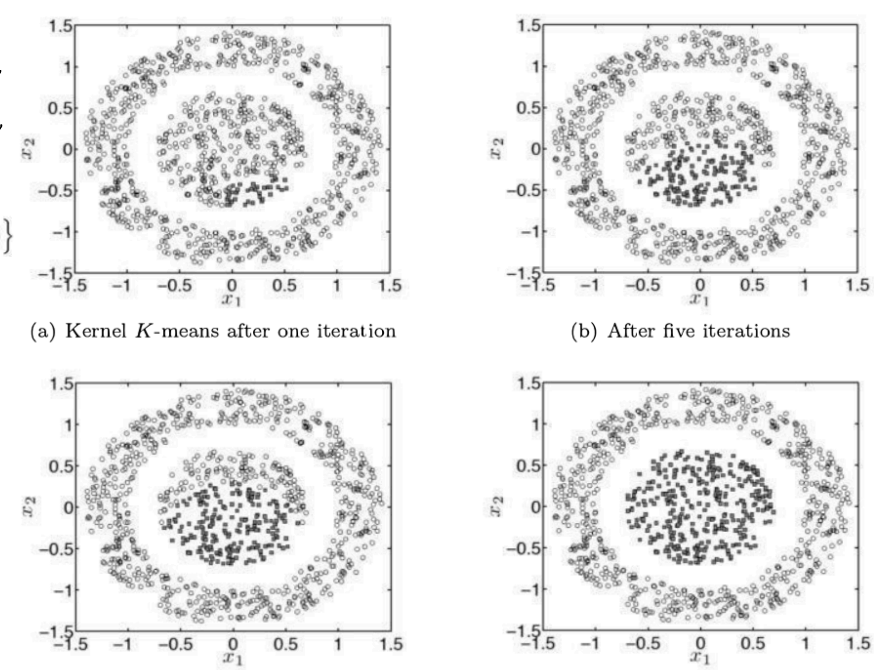
(B) Assign $N - K + 1$ objects to cluster 1 and remaining $K - 1$ objects to separate clusters.

Kernel *K*-Means for case A

Initialization:
Assigned 1 point to "squares" and
the remaining N-1 to "circles"

Gaussian Kernel: γ=1

$$\exp\left\{-\gamma(\mathbf{x}_n - \mathbf{x}_m)^\mathsf{T}(\mathbf{x}_n - \mathbf{x}_m)\right\}$$



(a) Kernel K-means after one iteration

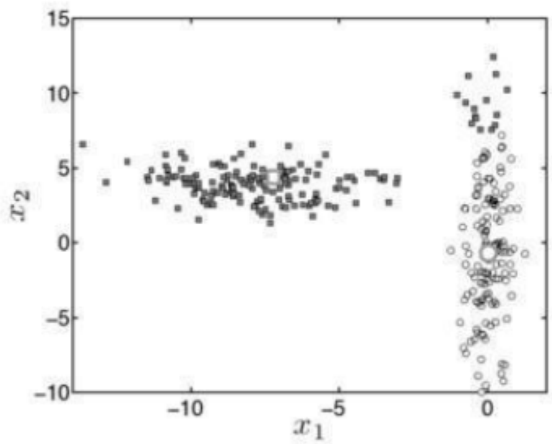
(b) After five iterations

(c) After 10 iterations

(d) At convergence (30 iterations)

Mixture Models

- Some similarities to *K*-means, but much richer representations of the data (rather than points / centroids)

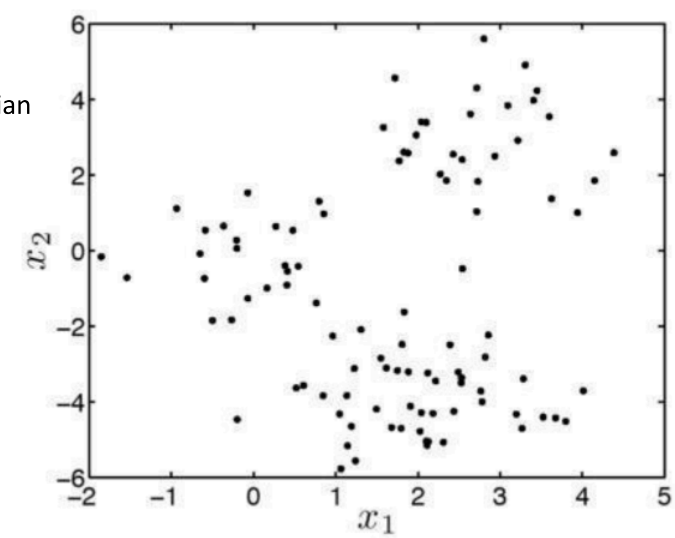


Centroids model of clusters is too simple to capture the structure here

The Generative Picture (again)

- How could we ***generate*** this data?

For each $\mathbf{x}_n$:
  (1) Select one of three Gaussians
    probability π_k for each Gaussian
    (where $\Sigma_k\, \pi_k = 1$)
  (2) Sample $\mathbf{x}_n$ from that Gaussian

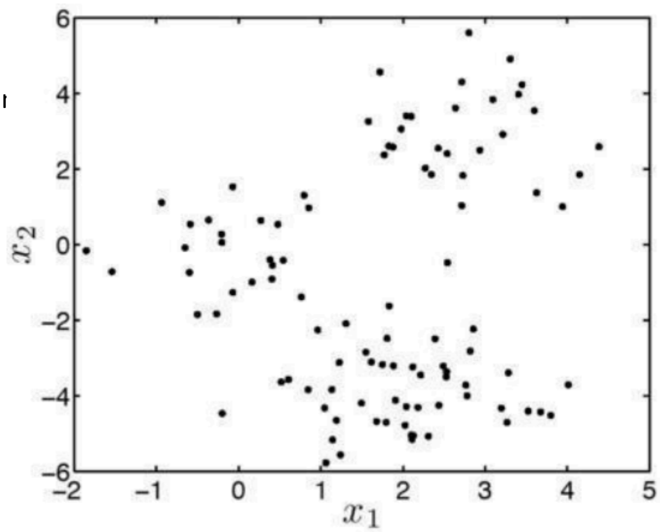


The Generative Picture (again)

- How could we ***generate*** this data?

For each $\mathbf{x}_n$:
  (1) Select one of three Gaussians
    probability π_k for each Gaussiai
    (where $\Sigma_k\,\pi_k = 1$)
  (2) Sample $\mathbf{x}_n$ from that Gaussian

• Use $z_{nk} = 1$ to mean individual n was “sampled from” generator k ($z_{nj} = 0$ for all other $j \neq k$)
• Each Guassian is modeled with mean and covariance $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$

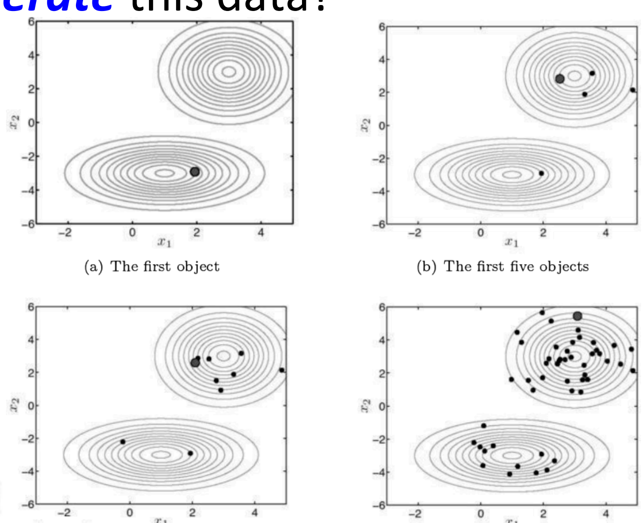


The Generative Picture (again)

- How could we ***generate*** this data?

For each $\mathbf{x}_n$:
  (1) Select one of three Gaussians
    probability π_k for each Gaussian
    (where $\Sigma_k\,\pi_k = 1$)
  (2) Sample $\mathbf{x}_n$ from that Gaussian

• Use $z_{nk} = 1$ to mean individual n was “sampled from” generator k ($z_{nj} = 0$ for all other $j \neq k$)
• Each Guassian is modeled with mean and covariance $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$



(a) The first object
(b) The first five objects
The first 10 objects
(d) The first 50 objects

$$p(\mathbf{x}_n | z_{nk} = 1, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$\boldsymbol{\mu}_1 = [3, 3]^{\mathsf{T}},\ \boldsymbol{\Sigma}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \qquad \boldsymbol{\mu}_2 = [1, -3]^{\mathsf{T}},\ \boldsymbol{\Sigma}_2 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\pi_1 = 0.7,\ \pi_2 = 0.3$$

Note axis scale; x_2 is being squashed