Expectation Maximization, Continue Clustering

ECE/CS 498 DS U/G

Lecture 8

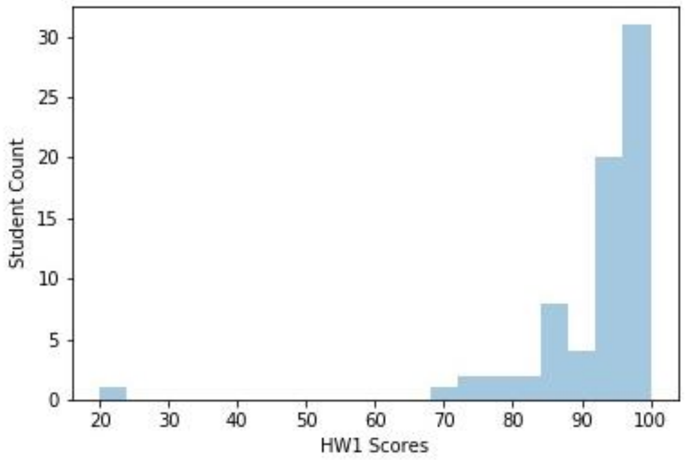
Ravi K. Iyer

Electrical and Computer Engineering

University of Illinois

Announcements

- MP1 Checkpoint 3 due on Monday, Feb 18
- Submit ICA2 today
- Discussion section on Friday, Feb 15
  - Counting parameters required to specify a distribution
- HW2 will be released today
- Poll to sign up for MP1 slot will be released today on Piazza
  - One member can sign up per group
- HW1 grades released
  - Please submit regrade requests within a week
  - Please check solutions before sharing regrade request



Mean	Std	Median	Max	Min
92.25	11.36	95.00	100.00	20.00

GMM Revisited

- Observations: $x_1, x_2, \ldots, x_N$
  - Each observation has 1 feature (1-dimension)
- Data is sampled from one of two Gaussian distributions (K=2)
  - Cluster r: (μ_r, σ_r^2)
  - Cluster b: (μ_b, σ_b^2)

- Source is known but distribution parameters are not known:
  - It is trivial to estimate (μ_r, σ_r^2) and (μ_b, σ_b^2)

- Distribution and its parameters $((\mu_r, \sigma_r^2)$ and $(\mu_b, \sigma_b^2))$ is known but source is now known:
  - Estimate where the each observation is likely to come from using Bayes rule

$$P(b|x_i) = \frac{p(x_i|b)P(b)}{p(x_i|b)P(b) + p(x_i|r)P(r)}$$

Find using PDF of normal distribution

Expectation Maximization Revisited

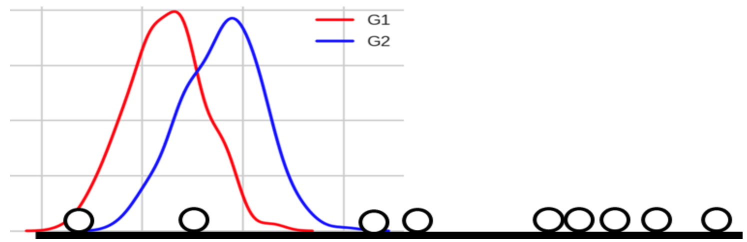
- **What if neither the source nor the distribution parameters are known?**

- **Chicken and Egg problem**
  - Need (μ_b, σ_b^2) and (μ_r, σ_r^2) to guess source of points
  - Need to know source to estimate (μ_b, σ_b^2) and (μ_r, σ_r^2)
  - Use **Expectation Maximization (EM)** algorithm

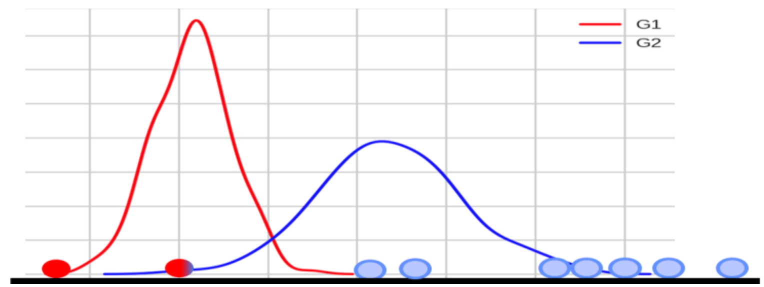
- **EM Algorithm**
  - Start with **two randomly placed Gaussians** (μ_b, σ_b^2) and (μ_r, σ_r^2)
  - For each x_i, calculate $P(b|x_i)$ and $P(r|x_i) = 1 - P(b|x_i)$
  - **Remember it does not assign the point but says here is the probability that it came from the red or from the blue**
  - Adjust (μ_b, σ_b^2) and (μ_r, σ_r^2) to fit points assigned to them

GMM Example: EM in action

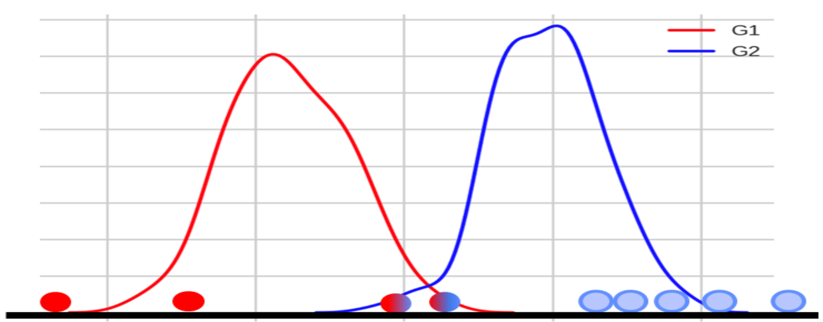
- Repeat the E and M steps iteratively till convergence
- Convergence: When M step gives the same parameters that were used in E



Initialization



1st iteration



Convergence (n^{th} iteration)

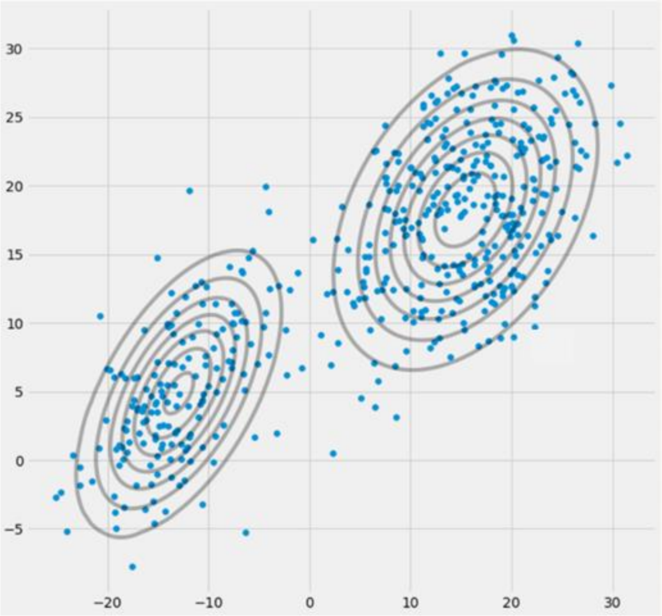
GMM: Multi-dimensional features (1)

- Data with d features i.e., $\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_N} \in \mathbb{R}^d$ from K sources
- Each source $c \in \{1, \dots, K\}$ has a Gaussian distribution, i.e., $\mathcal{N}(\mu_c, \Sigma_c)$ where $\mu_c \in \mathbb{R}^d$ and $\Sigma_c \in \mathbb{R}^{d \times d}$
- Iteratively estimate parameters
  - Prior: What fraction of instances came from source c

$$P(c) = \frac{1}{N}\sum_{i=1}^{N} P(c|\boldsymbol{x_i})$$

  - Mean: Expected value of feature j from source c:

$$\mu_{c,j} = \sum_{i=1}^{N}\left(\frac{P(c|\boldsymbol{x_i})}{NP(c)}\right) x_{i,j}$$



Source: https://www.python-course.eu/expectation_maximization_and_gaussian_mixture_models.php

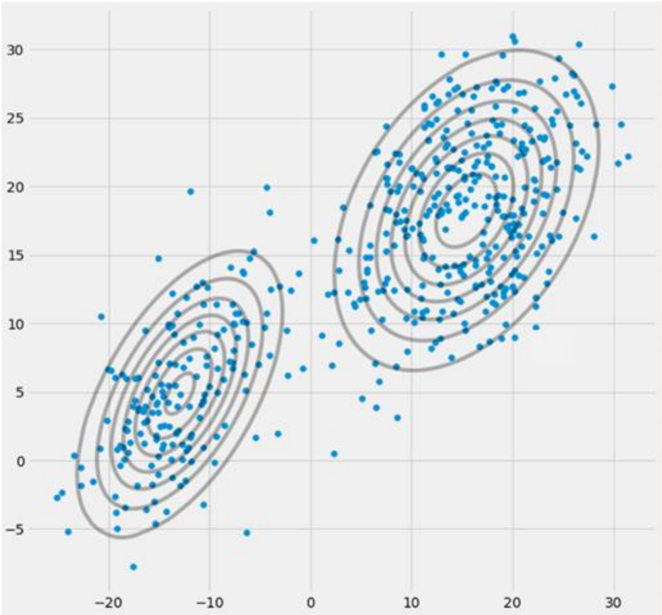
GMM: Multi-dimensional features (2)

- Data with d features i.e., $\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_N} \in \mathbb{R}^d$ from K sources
- Each source $c \in \{1, \dots, K\}$ has a Gaussian distribution, i.e., $\mathcal{N}(\mu_c, \Sigma_c)$ where $\mu_c \in \mathbb{R}^d$ and $\Sigma_c \in \mathbb{R}^{d \times d}$
- Iteratively estimate parameters
  - Covariance: How related are features j and k in source c:

$$(\Sigma_c)_{j,k} = \sum_{i=1}^{N} \left(\frac{P(c|\boldsymbol{x_i})}{NP(c)} \right) (x_{i,j} - \mu_{c,j})(x_{i,k} - \mu_{c,k})$$

  - Assignment: Based on our guess of the source for each instance

$$P(c|\boldsymbol{x_i}) = \frac{p(\boldsymbol{x_i}|c)P(c)}{\sum_{c'=1}^{K} p(\boldsymbol{x_i}|c')P(c')}$$



Source: https://www.python-course.eu/expectation_maximization_and_gaussian_mixture_models.php

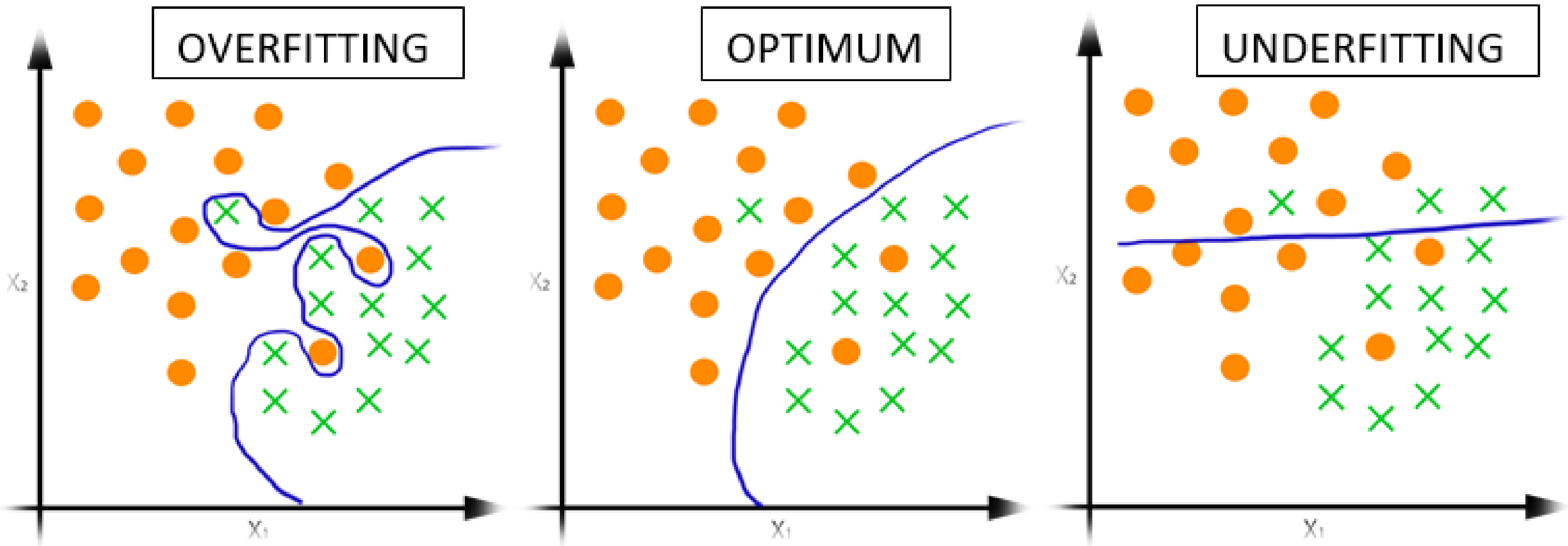
Picking *K* - Gaussian Components

- Maximize the log likelihood of the data given the model

$$L = \log P(x_i, .., x_n) = \sum_{i=1}^{N} \log \sum_{k=1}^{K} p(x_i|k)P(k)$$

- Pick *K* that makes *L* as large as possible $K^* = \max_K L$
  - *K* = *N*: each data point has its own source => overfitting
  - Unlikely to yield meaningful results for new (previously unseen) data points
  - Need to constrain (or regularize) to avoid overfitting

Overfitting



Source: https://medium.com/@srjoglekar246/overfitting-and-human-behavior-5186df1e7d19

Picking *K* - Gaussian Components

Possible to deal with overfitting using the following two ways:

- Split points into training set *T* and validation set *V*
  - For each *K,* fit parameters on *T* and measure likelihood of *V*

- **Occam's Razor**: Pick "simplest" of all models that fit
  - Bayes Inference Criterion (BIC): $(\log(N)\,K\ - 2\log L)$, where K is clusters, L: log likelihood [Fraley et. al , 2002]
  - When picking from several models, the one with the lowest BIC is preferred
  - BIC introduces a penalty term for adding parameters (i.e., #clusters)

K-means clustering

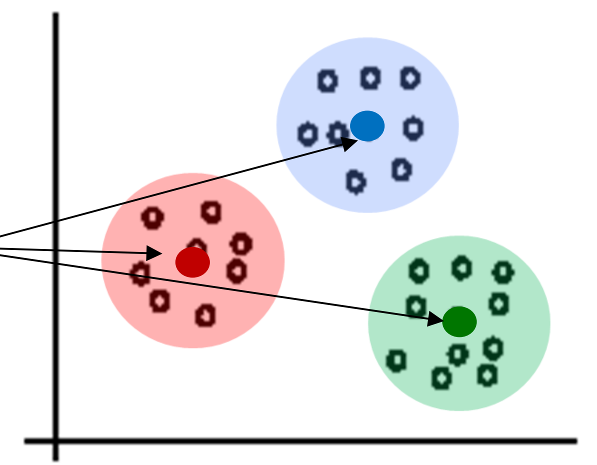
- K-means is a partition-based clustering algorithm

- Let the set of data points (or instances) *D* be

  $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}$,

  where $\boldsymbol{x}_i = (xi_1, xi_2, \dots, x_{id})$ is a vector in a real-valued space $\mathbb{R}^d$, and d is the number of feature (dimensions) in the data

- The *k*-means algorithm partitions the given data into *k* clusters.
  - Each cluster has a cluster **center**, called **centroid**
  - k is specified by the user



K-means clustering with $k = 3$

K-means algorithm

Given *k*, the *k-means* algorithm works as follows:

1) Randomly choose *k* data points (seeds) to be the initial centroids i.e., cluster centers
2) Assign each data point to the closest centroid
3) Re-compute the centroids using the current cluster memberships.
4) If a convergence criterion is not met, go to 2).

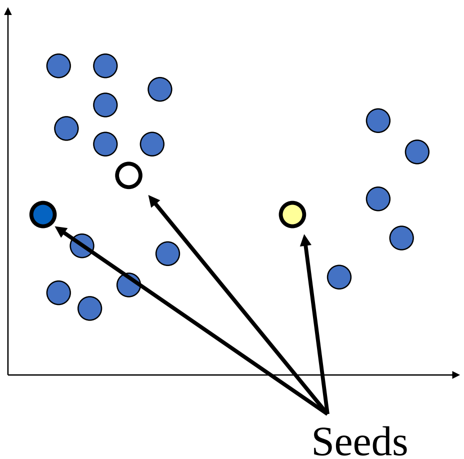
Stopping/Convergence criterion

1. No (or minimal) re-assignments of data points to different clusters,
2. No (or minimal) change of centroids, or
3. Minimal decrease in the **sum of squared error** (SSE),

$$SSE = \sum_{j=1}^{k} \sum_{\boldsymbol{x} \in C_j} dist(\boldsymbol{x}, \boldsymbol{m_j})^2 \qquad \boldsymbol{m_j} = \frac{1}{n_j} \sum_{\boldsymbol{x} \in C_j} \boldsymbol{x}$$

– C_j is the j^{th} cluster, $\boldsymbol{m}_j$ is the centroid of cluster C_j (the mean of all the data points belonging to C_j), n_j is the number of points in cluster C_j, and $dist(\boldsymbol{x}, \boldsymbol{m}_j)$ is the distance between data point $\boldsymbol{x}$ and centroid $\boldsymbol{m}_j$.

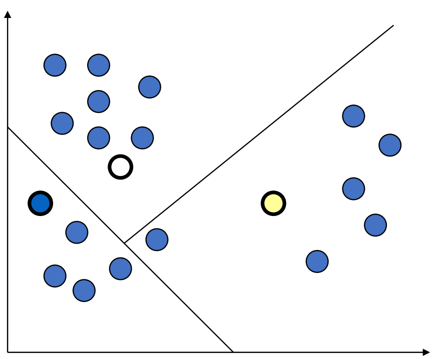
K-Means Example



(a)

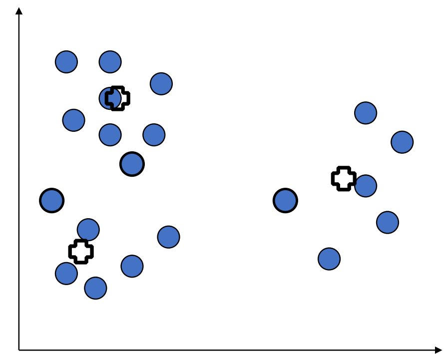
Predetermined number of clusters

Start with seed clusters of one element



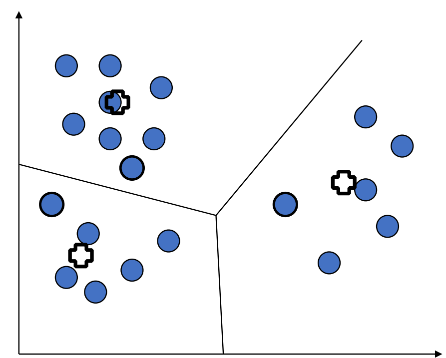
(b)

Assign Instances to Clusters



(c)

Find new centroids



(d)

Assign instances to new cluster

Why k-means is Popular

- Strengths:
  - Simple: Easy to understand and to implement
  - Efficient: Time complexity = $O(tkn)$,

    where n is the number of data points,

    k is the number of clusters, and

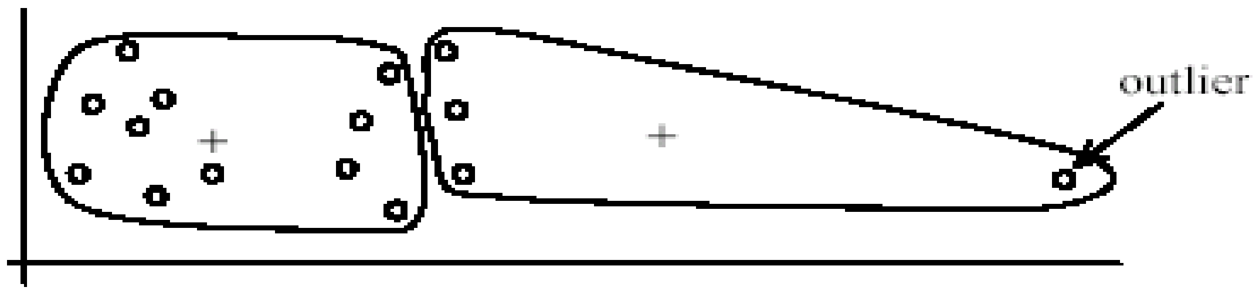
    t is the number of iterations.
  - Since both k and t are small, k-means is considered a linear algorithm

- Note: The algorithm can converge to a local optimum if SSE is used. The global optimum is difficult to find due to complexity.

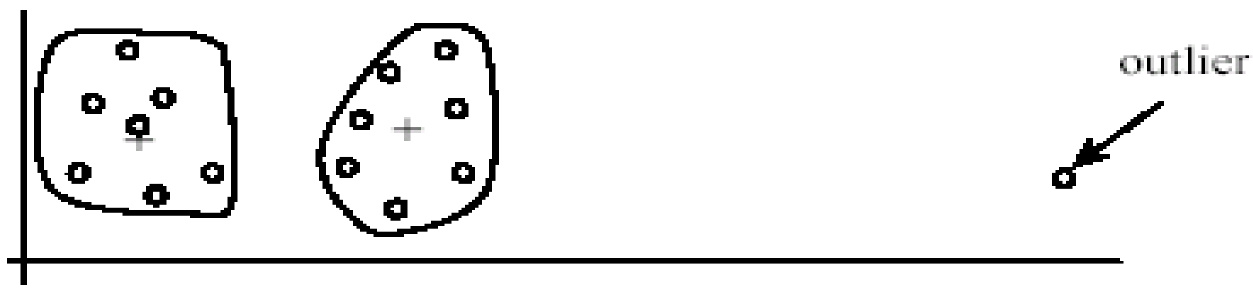
Weaknesses of k-means

- The algorithm is only applicable if the mean is defined
  - For categorical data, *k*-mode - the centroid is represented by most frequent values
  - Can be sensitive to seeds (choice of the initial k centroids)
- The user needs to specify *k*
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points
  - Outliers could be errors in the data recording or some special data points with very different values

Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



(B): Ideal clusters

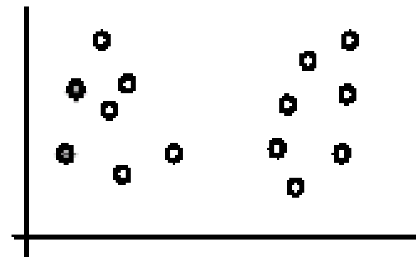
Weaknesses of k-means: To deal with outliers

- One method is to remove some data points in the clustering process that are much further away from the centroids than other data points
  – To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them

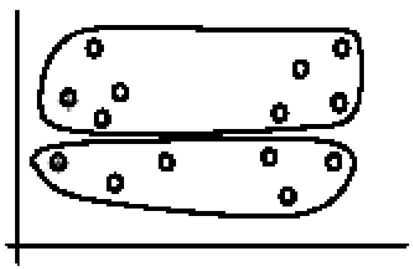
- Another method is to perform random sampling: Since sampling chooses a small subset of the data, the chance of selecting an outlier is small
  – Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

Weaknesses of k-means (cont …)

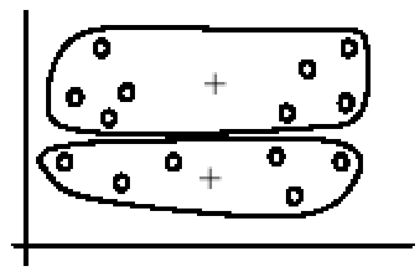
- The algorithm is sensitive to initial seeds.



(A). Random selection of seeds (centroids)



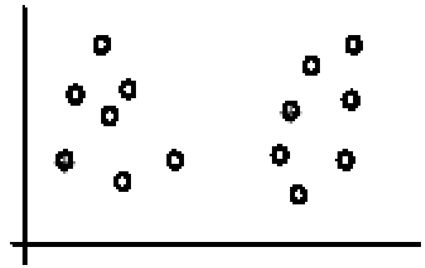
(B). Iteration 1



(C). Iteration 2

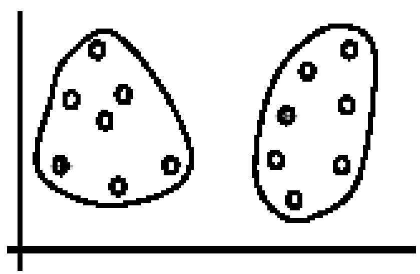
Weaknesses of k-means (cont …)

- Use different seeds: Good results

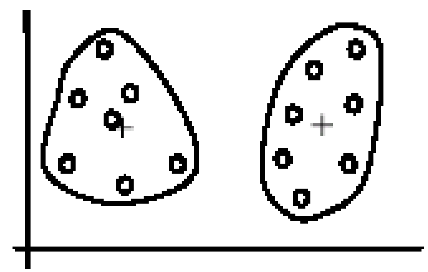


(A). Random selection of *k* seeds (centroids)

There are some methods to help choose good seeds



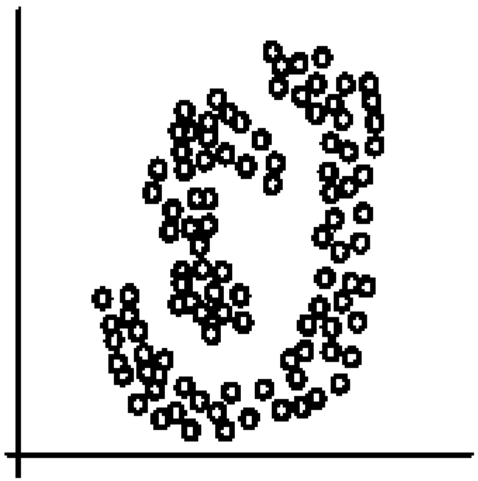
(B). Iteration 1



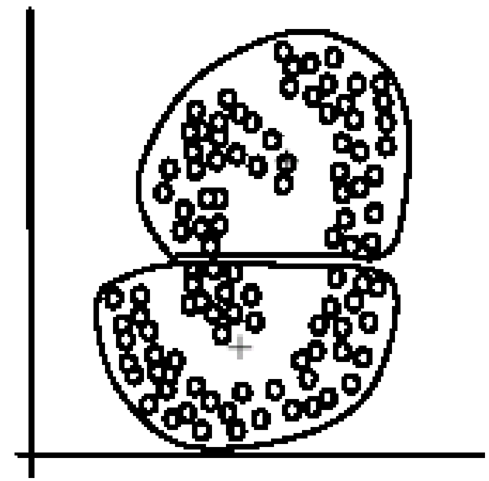
(C). Iteration 2

Weaknesses of k-means (cont …)

- The *k*-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres)



(A): Two natural clusters



(B): *k*-means clusters

K-means summary

- Despite the weaknesses, *k*-means is a very useful algorithm due to its simplicity and efficiency
  - Other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general
  - Although they may be more suitable for some specific types of data or applications
- Comparing different clustering algorithms is a difficult task
  - Problem dependent insights are very useful