Weekly meeting 3

Dr. Doina Bein

Thursday, June 15, 10:30am-**11:40am**

(shorter meeting today due to a family event)

Surveys to be completed

To be done today, before starting research:

CIC-PCUBED Pre-event survey:

https://fullerton.qualtrics.com/jfe/form/SV_6YIVSkC6hLxbunA

Project 1: Data Science

What you need to do: topics & objectives

Objective 1: Learn Python using some textbook or some online courses such as (https://www.codecademy.com/learn/learn-python). Shared by Stephanie Pocci: Learn Python in a couple hours. This YouTuber does a very beginner-friendly crash course about the capabilities of Python and its uses. Here is the link: https://www.youtube.com/watch?v=rfscVS0vtbw

Objective 2: Learn how to use Jupyter Notebook. Start here http://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html

Objective 3: For data science, find a suitable dataset and start training some neural network using with Google tensorflow.

Logistics for all students

- Who is participating: list of current research students and their availability
- Research will be conducted virtually during the week with in-person meetings throughout the week
- Zoom meetings for me to teach new topics and for you to participate in open discussions
- Support:
  - If needed, you can meet me
    Zoom: Mon, Tu, Wed from 8:30-10:25 am
    IN PERSON: Mon, Tu, Wed from 8:30-9:30 am, Thursday 8:30-10am or by email
  - CIC-PCUBED peer mentor: (tentative) availability

Logistics for all students (contd.)

- Make a copy of this GDoc Work schedule, share the Gdoc copy with me, and maintain it weekly and daily; due at the end of Week 2
- Before the end of week 3, make a copy and maintain your Proposed work by individual or teams of up to three; due by the end of Week 3
- Complete your availability here; try to have it consistent over the 7 weeks such that it will be easy to partner in the project
- Group projects: to be decided; sample list here
- Oral or poster presentations: tentatively scheduled for Friday, July 28, from 8:30am-12:30 pm and if needed, from 1:30-4 pm

Please checkout:

- Other websites and ebooks
- Websites with free datasets
- If you find good, free resources, please share it by email or during weekly meetings
- Next meeting: I will lecture on ZOOM on Data Science: Friday, June 16, from 10:30am-12pm

Progress on Learning Python

- Free course: https://www.codecademy.com/learn/learn-python
- Free course: https://www.kaggle.com/learn/python
- Youtube video (about 4 hours): https://www.youtube.com/watch?v=rfscVS0vtbw

Data Science

Clustering

- Let $X = \{x_1, x_2, \dots x_n\}$ denote a data-set like a collection of images
- Each datum $x_1 \in X$ (with X denoting the space of data, usually $X \subset \mathrm{R}^d$) is described as an *attribute vector* $x_i = \left(x_i^1, x_i^2, \dots x_i^d\right)$ called a *feature vector*.
- *Exploratory* data analysis is concerned with efficiently processing of data-sets to find such structural information without any prior knowledge
- *Unsupervised machine learning* is a type of exploratory data analysis and it consists of learning from data without prior knowledge
- *Clustering* is a set of techniques that consists in detecting subsets of data that define groups or clusters
- Those groups should ideally represent *semantic categories* of data

Similarity

(was taken from here https://www.cs.utah.edu/~piyush/teaching/4-10-print.pdf but the link is no available anymore)

- The only information clustering uses is the similarity between examples
- Clustering groups examples based of their mutual similarities
- A good clustering is one that achieves:
  - High within-cluster similarity
  - Low inter-cluster similarity

Similarity

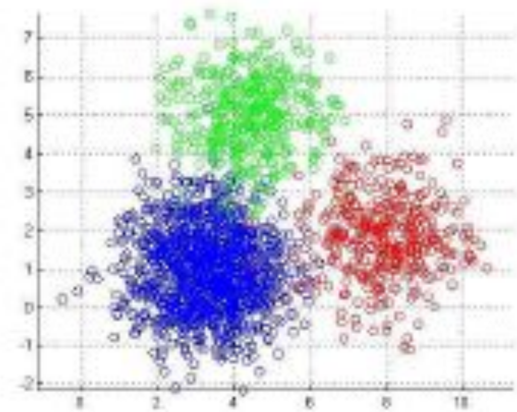
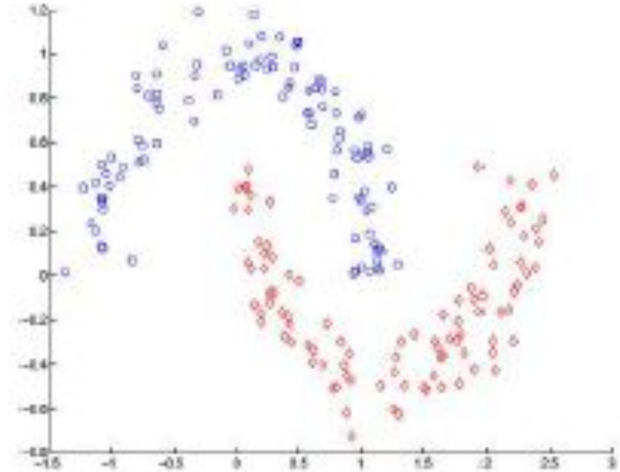
(was taken from here https://www.cs.utah.edu/~piyush/teaching/4-10-print.pdf but the link is no available anymore)

- Choice of the similarity measure is very important for clustering
- Similarity is inversely related to distance
- Different ways exist to measure distances. Some examples:

– Euclidean distance: $d(x, z) = ||x - z|| = \sqrt{\sum_{d=1}^{D}(x_d - z_d)^2}$

– Manhattan distance: $d(x, z) = \sum_{d=1}^{D}|x_d - z_d|$

– Kernelized (non-linear) distance: $d(x, z) = ||\phi(x) - \phi(z)||$

- For the left figure above, Euclidean distance may be reasonable
- For the right figure above, kernelized distance seems more reasonable

Taken from https://www.cs.utah.edu/~piyush/teaching/4-10-print.pdf

Similarity is Subjective

(was taken from here https://www.cs.utah.edu/~piyush/teaching/4-10-print.pdf but the link is no available anymore)

- Similarity is often hard to define

- Different similarity criteria can lead to different clusterings

Some Real-World Examples of Data Clustering

(was taken from here https://www.cs.utah.edu/~piyush/teaching/4-10-print.pdf but the link is no available anymore)

- Clustering images based on their perceptual similarities
- Image segmentation (clustering pixels)

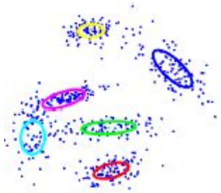
- Clustering webpages based on their content
- Clustering web-search results
- Clustering people in social networks based on user properties or preferences

Picture courtesy: http://people.cs.uchicago.edu/pff/segment/

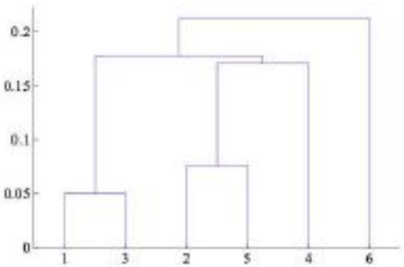
Types of Clustering

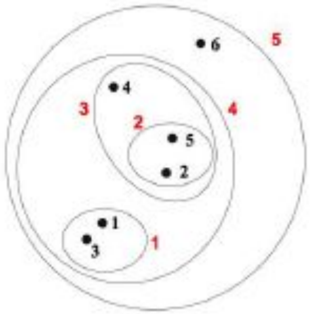
(was taken from here https://www.cs.utah.edu/~piyush/teaching/4-10-print.pdf but the link is no available anymore)

- Flat or Partitional clustering (K-means, Gaussian mixture models, etc.): partitions are independent of each other (hard or soft clustering)



- Hierarchical clustering (e.g., agglomerative clustering, divisive clustering)
  - Partitions can be visualized using a tree structure (a dendrogram)
  - Does not need the number of clusters as input
  - Possible to view partitions at different levels of granularities (i.e., can refine/coarsen clusters)





k-means Clustering

- The *k-means cost function* asks to minimize the sum of squared Euclidean distances of data points to their closest prototype centers:

$$e_k(X;C) = \sum_{i=1}^{n} \min_{j\in\{1,\cdots k\}} \left\|x_i - c_j\right\|^2$$

- k-means cost function seeks compact globular clusters of small variances: the cost of a single cluster $e_1(G) = e_1(G, c)$ is minimized when we choose for the cluster prototype its center of mass c, called the *centroid*:

$$c(G) = \operatorname*{argmin}_{c} \sum_{x\in G} \|x - c\|^2 = \frac{1}{G} \sum_{x\in G} x$$

where $|G|$ denotes the cardinality of G, that is the number of elements contained in group G and $\operatorname{argmin}_x f(x)$ to denote the argument that yields the minimum in case this minimum value is unique

- If instead of choosing the squared Euclidean distance, we had chosen the ordinary Euclidean distance, one obtains the so-called *Fermat-Weber point* that generalizes the notion of median. It is thus also called the *geometric median*.
- Although the Fermat-Weber point is unique and often used in operations research for facility location problems, it does not admit a closed-form solution, but can be arbitrarily finely approximated
- *k-median clustering* is the clustering obtained by minimizing the cost function

$$\min_{C} \sum_{i=1}^{n} \min_{j\in\{1,\cdots k\}} \left\|x_i - c_j\right\|$$

where $\|\cdots\|$ means the regular Euclidean distance

- Partitions from *k*-means and *k*-medians can be very different from each other: the centroid location can be different to the median for a single cluster

- Centroids can be easily corrupted by adding a single outlier point
- We say that the breakdown point of the centroid is 0: A single outlier p_0 diverging to ∞ will impact the centroid to be diverging to ∞ too.
- But the median is more robust since it requires $\left\lfloor \frac{n}{2} \right\rfloor$ outliers (that is, about 50% of outliers) to steer the median point to ∞.
- Therefore k-median clustering is often preferred when there are many outliers in data-sets.

Fig. 7.3 The k-means cost function tend to find globular-shaped clusters that minimize the weighted sum of the cluster variances. k-Means clustering is a model-based clustering where each cluster is associated to a prototype: its center of mass, or centroid. Here, we have choosen $k = 4$ groups for the k-means: Cluster prototypes, centroids, are illustrated with large disks

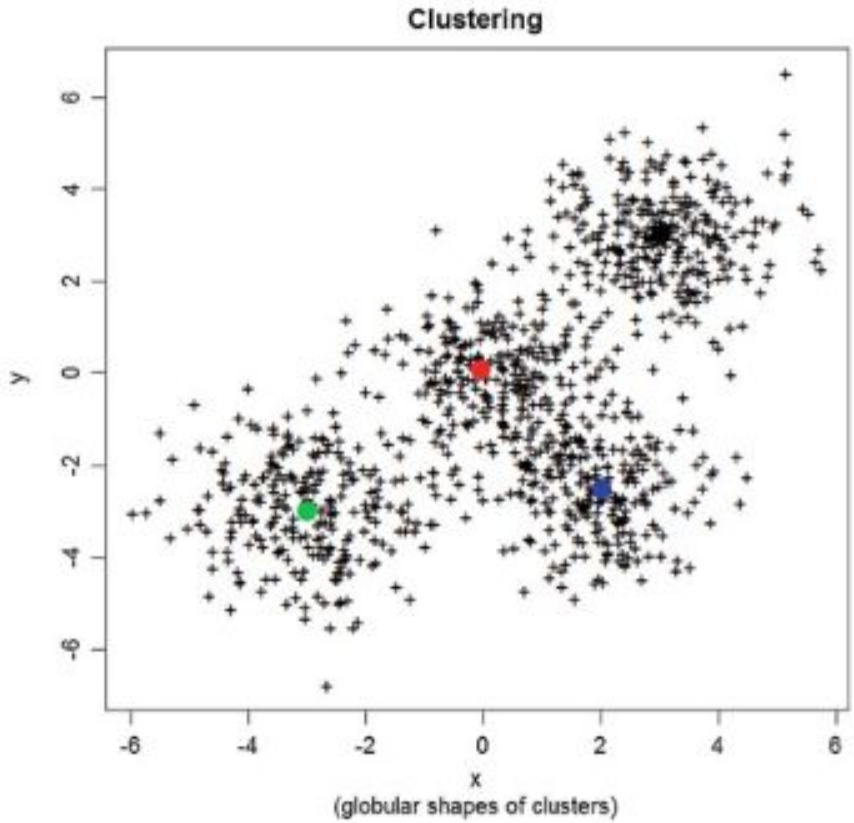


Image taken from textbook, page 169

The k-means Problem (https://www.math.uwaterloo.ca/~cswamy/talks/kmeans-short.ppt)

Given: n points in d-dimensional space

$X \subseteq R^d$: point set with $|X| = n$

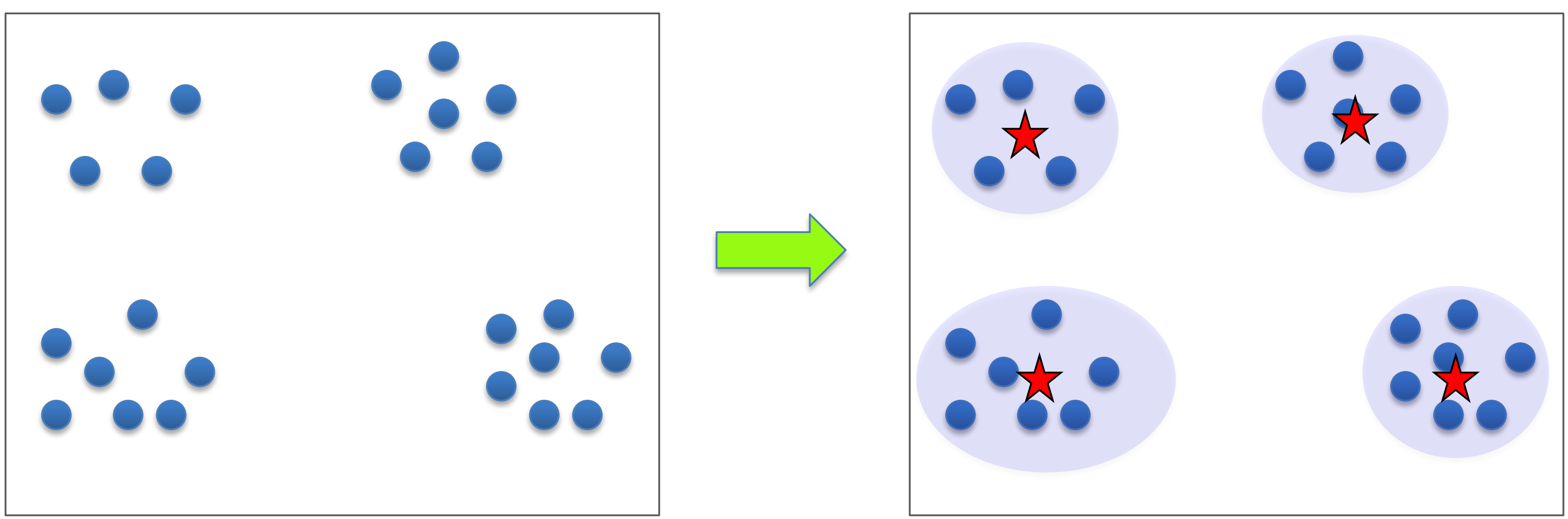
- partition X into k clusters $X_1, \ldots, X_k$
- assign each point in X_i to a common center $c_i \in R^d$

d: L_2 distance

Goal: Minimize $\sum_i \sum_{x \in X_i} d(x, c_i)^2$

(web.stanford.edu/group/mmds/slides2012/s-speaker-ppt-2012-temp/Bahmani.pptx)

- K-means clustering is a fundamental problem in data analysis and machine learning
- "By far the most popular clustering algorithm used in scientific and industrial applications" [Berkhin '02]
- Identified as one of the top 10 algorithms in data mining [Wu et al '07]



K = 4

k-Means Optimization Problem

- Finding the center of a single cluster is a particular case of clustering with k = 1 cluster.
- With the squared Euclidean distance cost, the center is the mean of the attributes, hence its naming *k-means*.
- Finding the minimum of a k-means cost function is a NP-hard problem when the dimension $d > 1$ and the number of clusters $k > 1$.
- When $k = 1$, we have shown that we can compute the optimal solution (the centroid) in linear time (computing the mean of the group).
- When $d = 1$, we can compute an optimal k-means solution using dynamic programming: Using $O(nk)$ memory, we can solve the k-means for n scalar values in time $O(n^2k)$

- For NP-hard problems, we seek efficient heuristics to approximate the cost function. We distinguish two classes of such heuristics:

1. Global heuristics that do not depend on initialization, and
2. Local heuristics that iteratively starts from a solution (a partition) and iteratively improves this partition using “pivot rules.”

Lloyd's Batched k-Means Local Heuristic

- Lloyd's heuristic (1957): from a given input and initialization:

Input: $\{x_1, x_2, \ldots x_n\}$, each $x_1 \in \mathbb{R}^d$, the number of partitions *k*

Initialization: Start with *k* cluster centers $\{c_1, c_2, \ldots, c_k\}$ (typically chosen uniformly at random from data points)

iteratively repeat until convergence the following two steps:

1. **Assign points to clusters**: (each $x_i \in X$ is assigned to its closest cluster center) for each $x_i \in X$, let $l_i = \mathrm{argmin}_l \|x_i - c_i\|^2$, and define the *k* cluster groups as $G_j = \{x_i : l_i = j\}$ with $n_j = |G_j|$, the number of elements of X falling into the *j*th cluster.

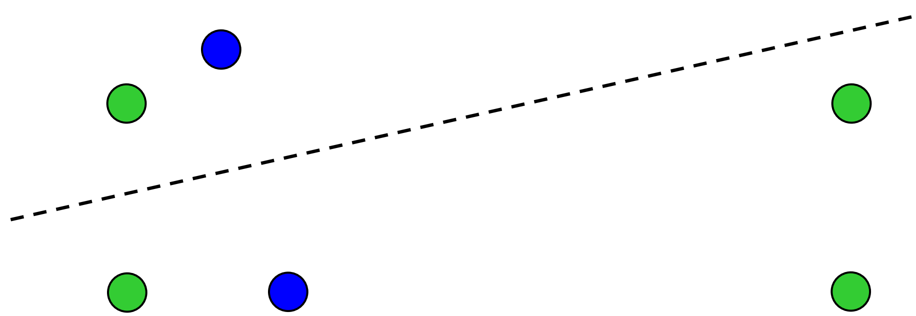
2. **Update centers** (perform an Expected Maximization-type local search): For all $j \in \{1, \ldots, k\}$, update the centers to their cluster centroids : $c_j = \frac{1}{n}\sum_{x \in G_j} x$ (or the barycenters $c_j = \frac{1}{\sum_{x \in G_j} w(x)} \sum_{x \in G_j} w(x) x$ for weighted data-sets).

- A possible convergence criteria: cluster centers do not change anymore

Taken from https://www.math.uwaterloo.ca/~cswamy/talks/kmeans-short.ppt

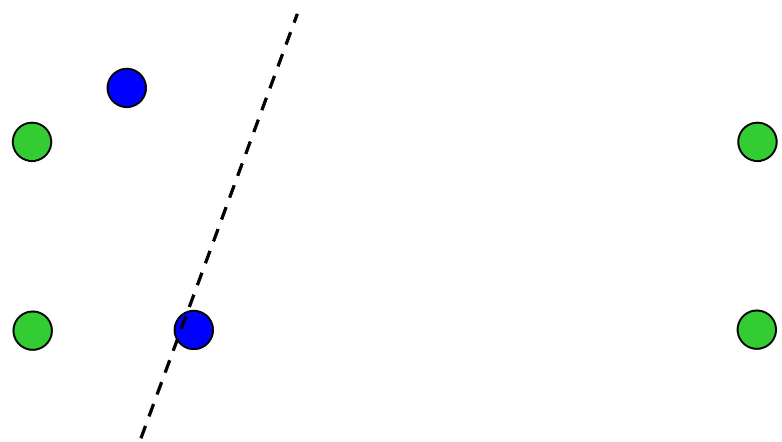
b) Heuristics: Lloyd's method invented in 1957 and remains an extremely popular heuristic even today

1) Start with k initial / "seed" centers $c_1, \ldots, c_k$.
2) Iterate the following Lloyd step
   a) Assign each point to nearest center c_i to obtain clustering $X_1, \ldots, X_k$.
   b) Update $c_i \leftarrow \mathrm{ctr}(X_i) = \sum_{x \in X_i} x/|X_i|$.



b) Heuristics: Lloyd's method invented in 1957 and remains an extremely popular heuristic even today

1) Start with k initial / "seed" centers $c_1, \ldots, c_k$.
2) Iterate the following Lloyd step
   a) Assign each point to nearest center c_i to obtain clustering $X_1, \ldots, X_k$.
   b) Update $c_i \leftarrow ctr(X_i) = \sum_{x \in X_i} x / |X_i|$.



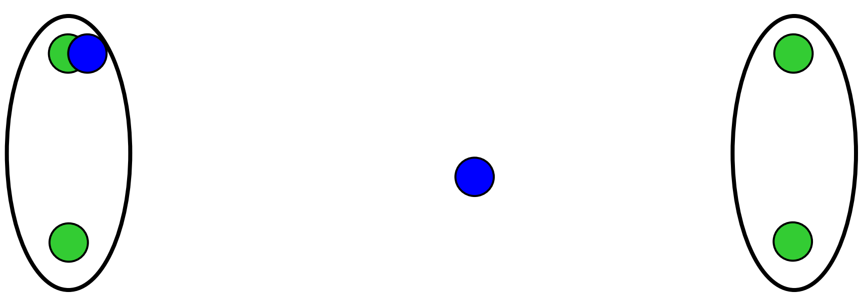
b) Heuristics: Lloyd's method invented in 1957 and remains an extremely popular heuristic even today

1) Start with k initial / "seed" centers $c_1, \ldots, c_k$.

2) Iterate the following Lloyd step

   a) Assign each point to nearest center c_i to obtain clustering $X_1, \ldots, X_k$.

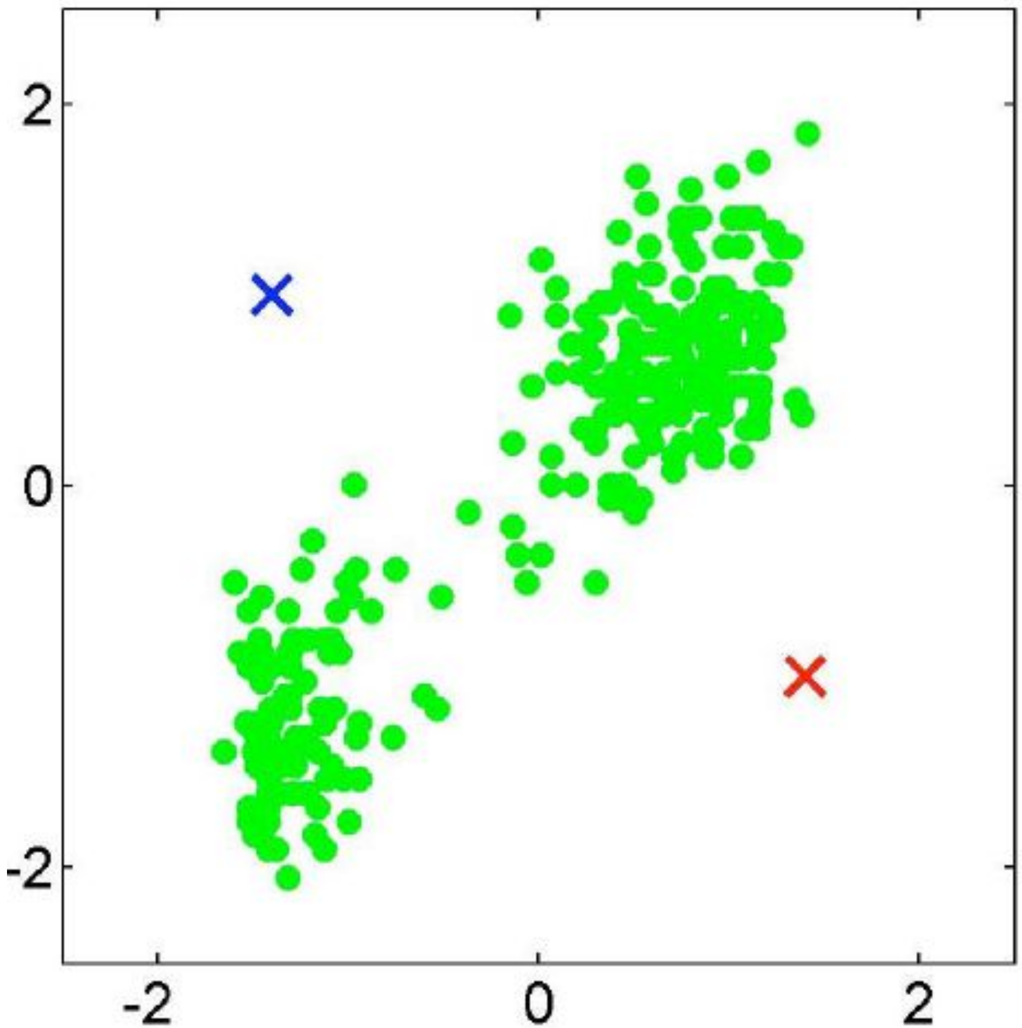
   b) Update $c_i \leftarrow ctr(X_i) = \sum_{x \in X_i} x/|X_i|$.



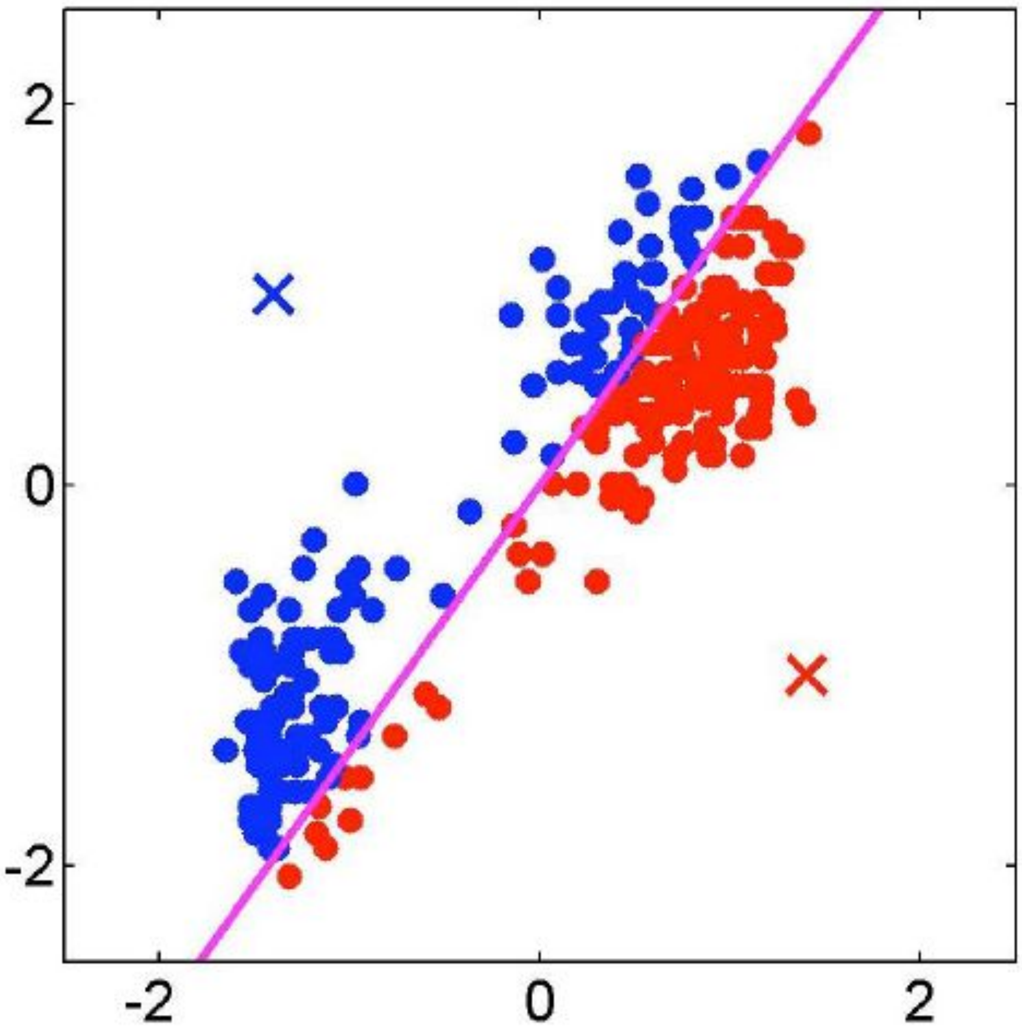
Other example

(https://www.cs.utah.edu/~piyush/teaching/4-10-print.pdf)

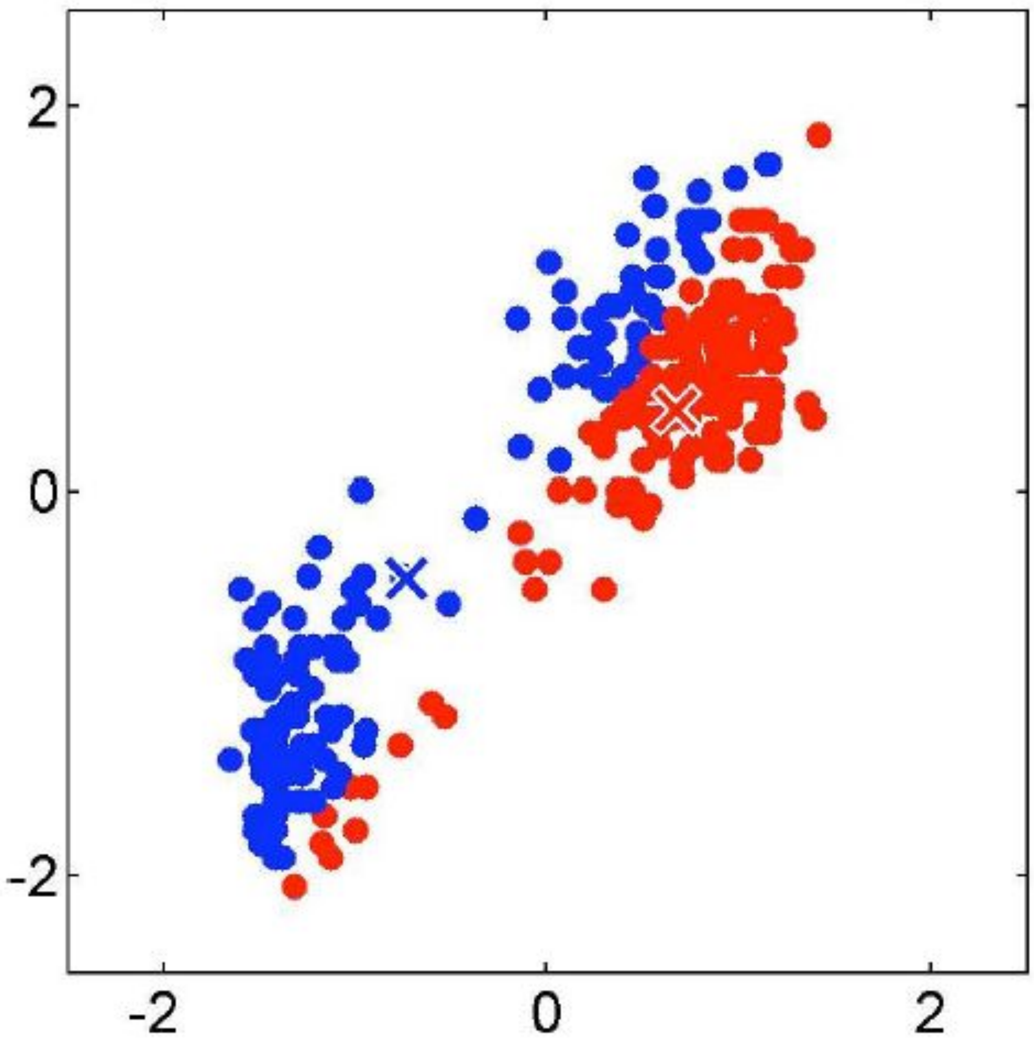
Initialization (assume K = 2)



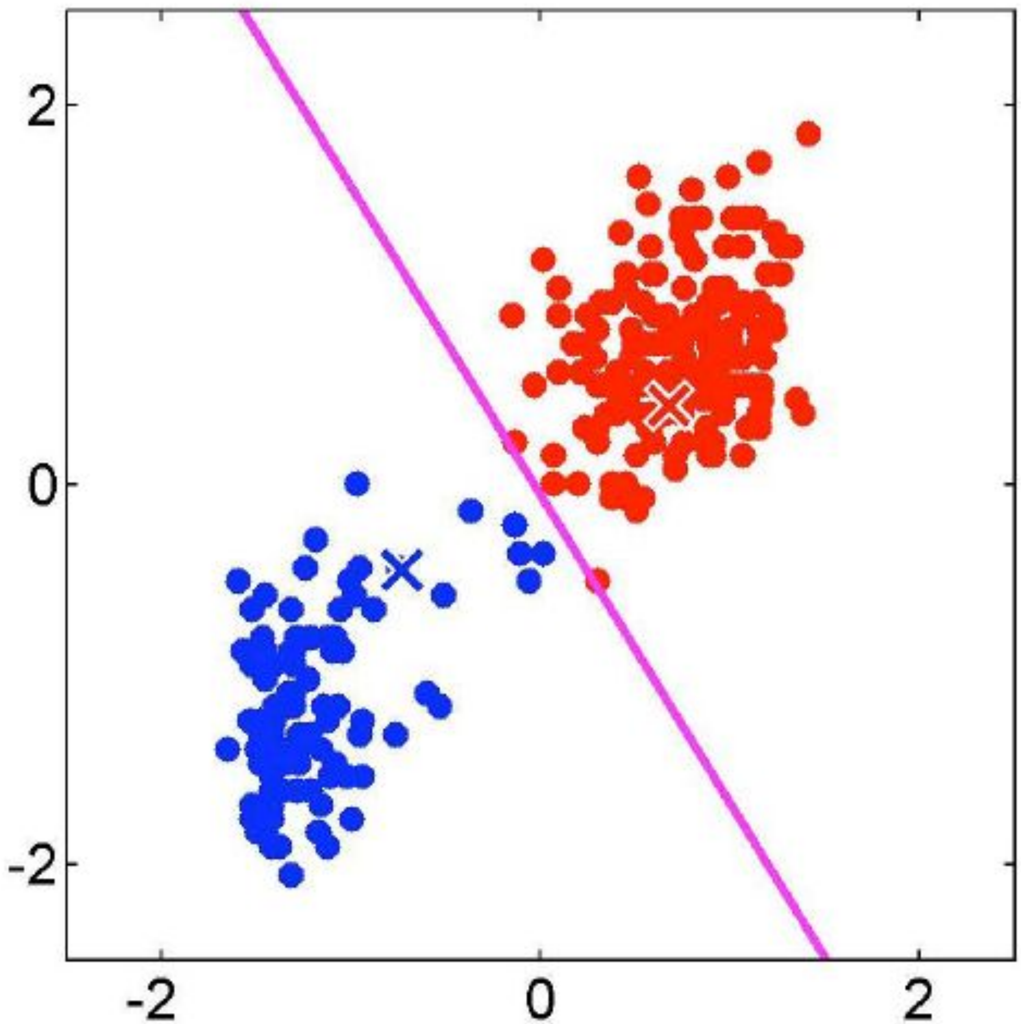
Iteration 1: Assign points to clusters



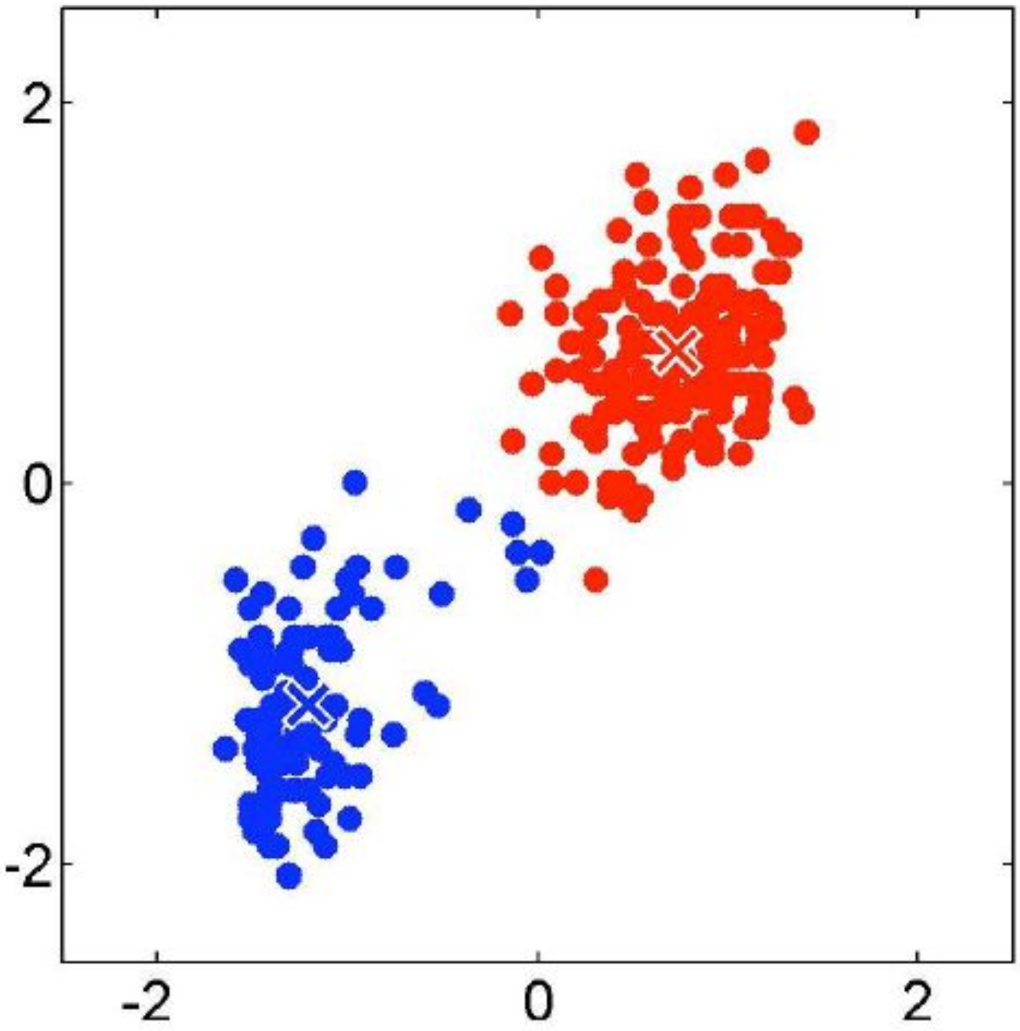
Iteration 1: Update centers



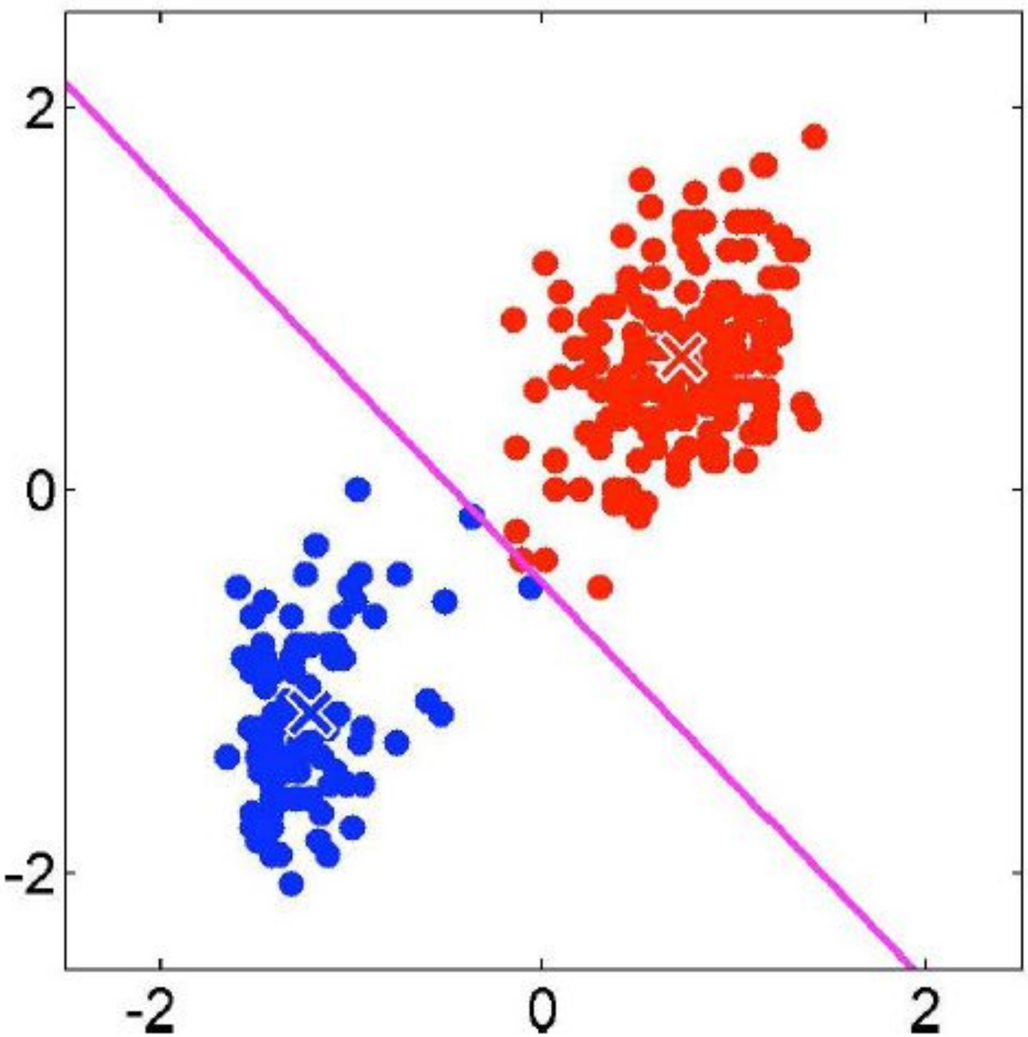
Iteration 2: Assign points to clusters



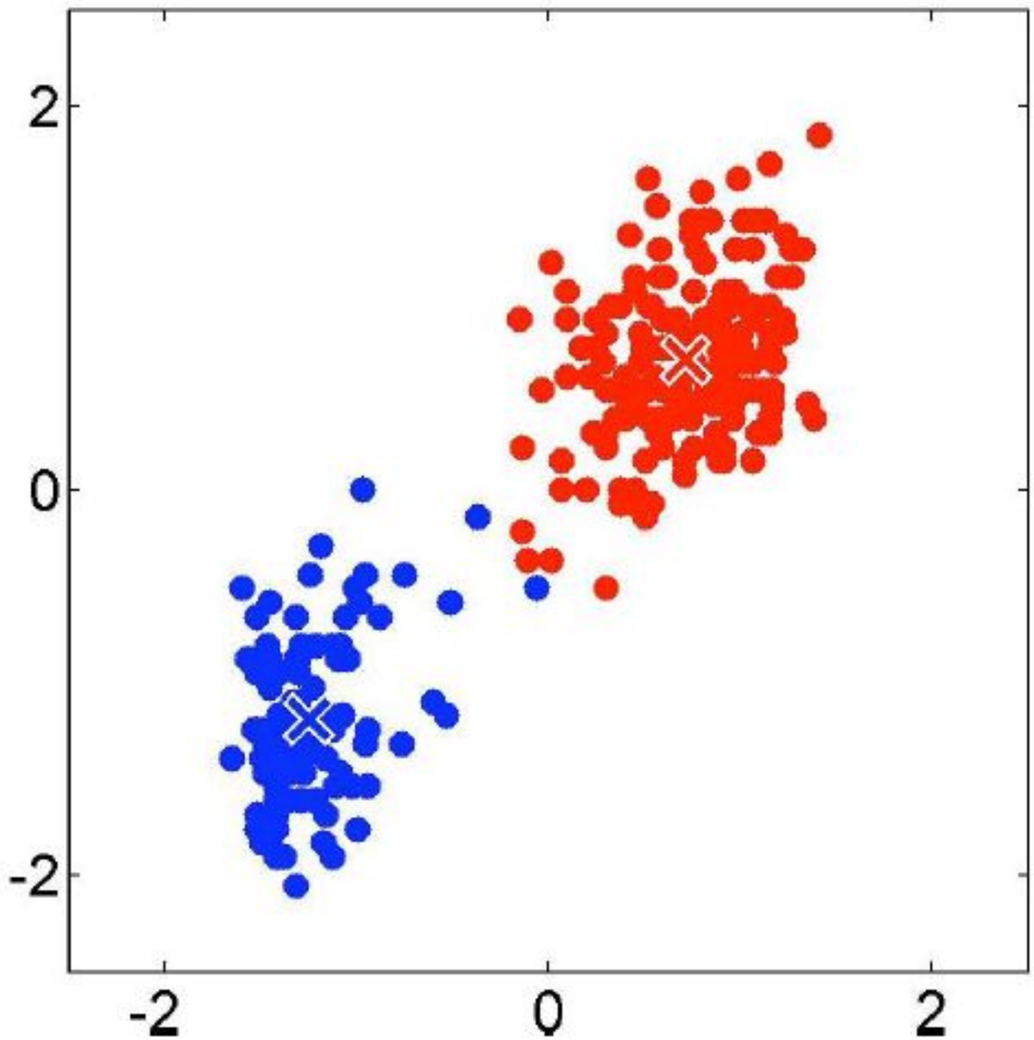
Iteration 2: Update centers



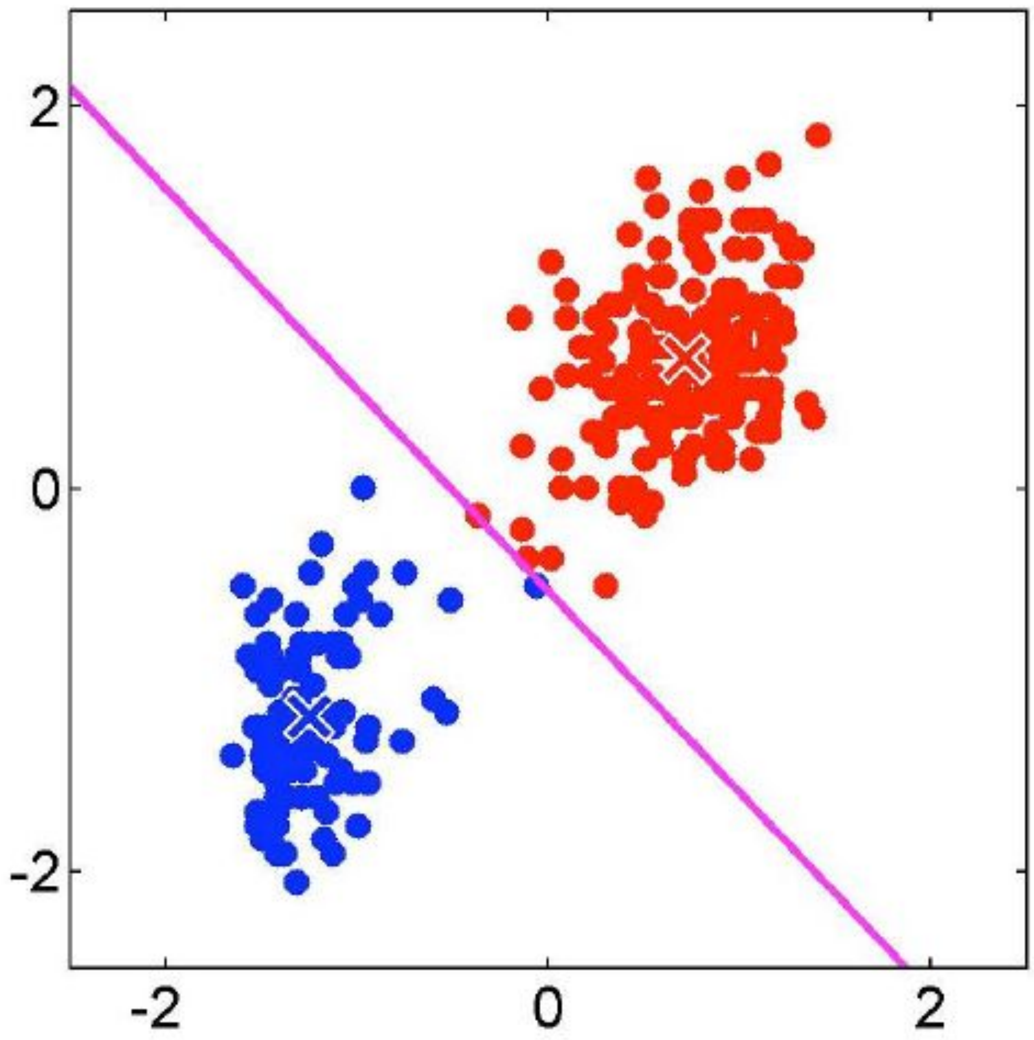
Iteration 3: Assign points to clusters



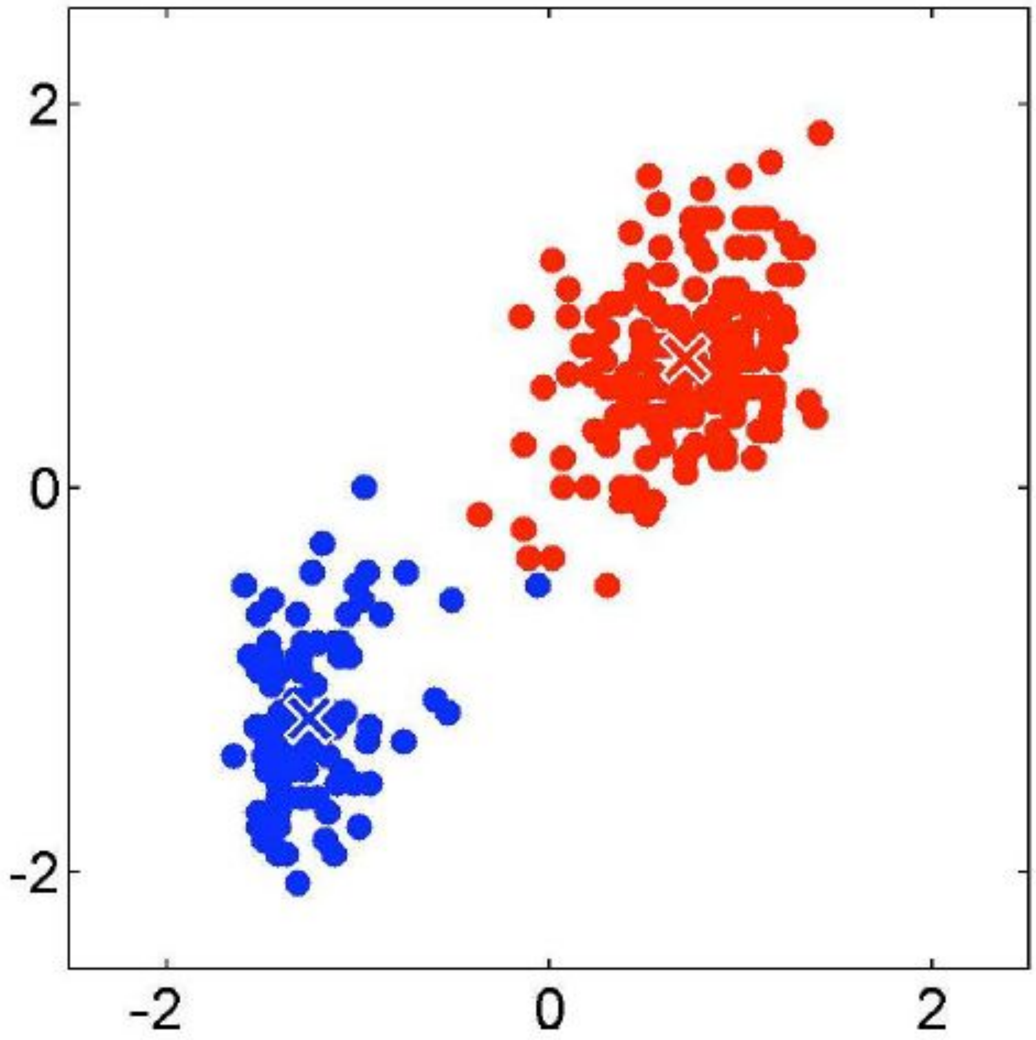
Iteration 3: Update centers



Iteration 4: Assign points to clusters



Iteration 4: Update centers



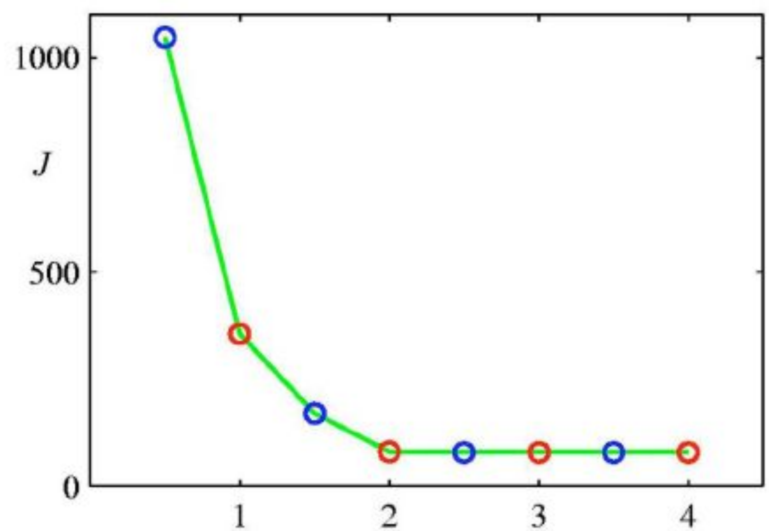
Choosing the number of clusters K

(https://www.cs.utah.edu/~piyush/teaching/4-10-print.pdf)

- Recall that *k*-means objective is to minimize the cost function, aka the sum of distances of points from their cluster centers

$$J(C,r) = \sum_{i=1}^{n}\sum_{j=1}^{k} r_{nk}\left\|x_i - c_j\right\|^2$$

- One way to select the number of clusters *k* is to try different values of *k*, plot the *k*-means objective versus k, and choose the "elbow-point"
- One drawback of this method is that it is computationally very expensive, and sometimes (depending on the data-sets), the inflexion point between the sharp decrease and the plateau is not so well defined



For the plot to the left, k = 2 is the elbow point

Initialization Issues:

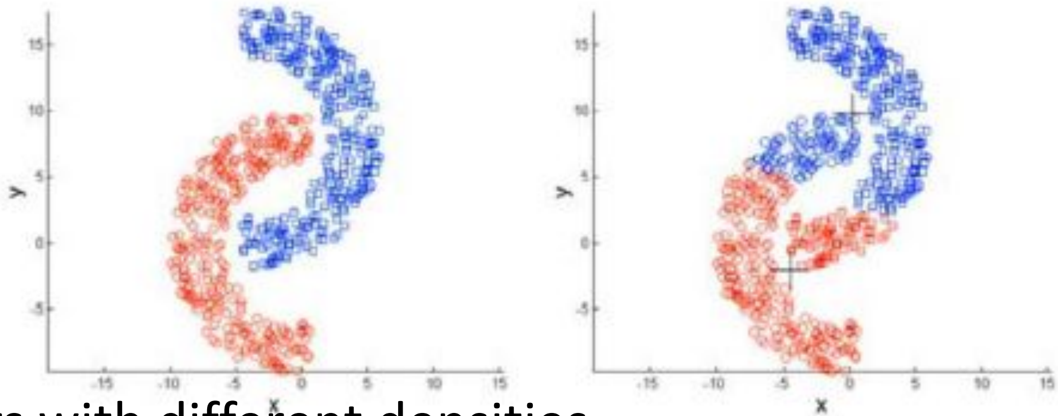
(https://www.cs.utah.edu/~piyush/teaching/4-10-print.pdf)

- K-means is extremely sensitive to cluster center initialization
- Bad initialization can lead to
  - Poor convergence speed
  - Bad overall clustering
- Safeguarding measures:
  - Choose first center as one of the examples, second which is the farthest from the first, third which is the farthest from both, and so on.
  - Try multiple initializations and choose the best result
- Other smarter initialization schemes (e.g., look at the K-means++ algorithm by Arthur and Vassilvitskii)

Limitations Illustrated

(https://www.cs.utah.edu/~piyush/teaching/4-10-print.pdf)

- K-Means might not find the best possible assignments and centers.
- Non-convex/non-round-shaped clusters: Standard K-means fails!



- Clusters with different densities