CMPSC 448: Machine Learning

Lecture 13. Clustering

Rui Zhang

Fall 2021

What types of ML are there?

Supervised Learning

Unsupervised Learning

Reinforcement Learning

Supervised Learning

So far the focus was on supervised learning methods such as **regression** and **classification**:

Input: $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$

Output: $f : \mathbb{R}^d \mapsto 1, 2, \ldots, K$ classification

$f : \mathbb{R}^d \mapsto \mathbb{R}$ regression

Regression vs Classification

- Ordinary Least Squares
- Ridge Regression
- Lasso Regression

- Artificial Neural Networks: Perceptron and Deep Neural Networks
- Nearest Neighbor
- Logistic Regression
- Decision Trees
- Support Vector Machines (SVMs)
- Bagging: Random Forests
- Boosting: AdaBoost and Gradient Boosted Trees

Unsupervised Learning

In **unsupervised learning,** no label is available and the goal is to learn hidden structure of the input.

Input: $\mathcal{S} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$

Objective: Learning hidden structure in the data!

Unsupervised Learning Algorithms

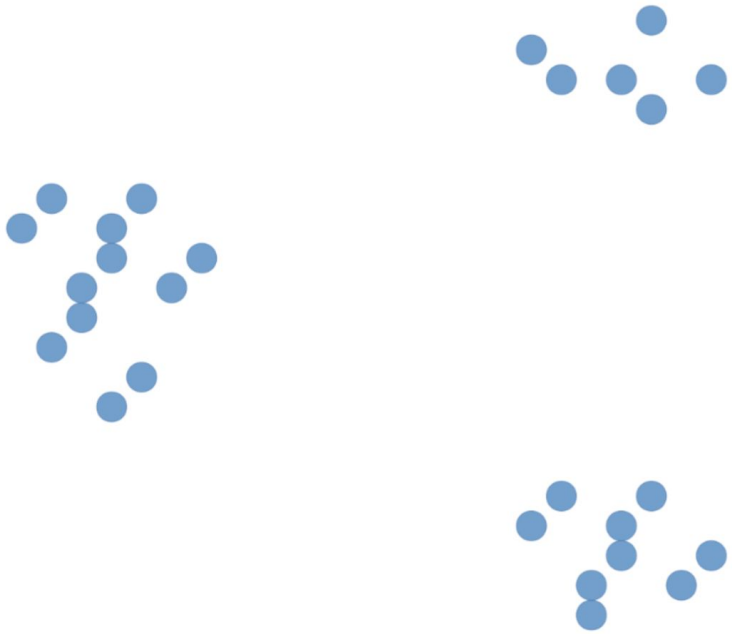
- K-Means
- Gaussian Mixture Models (GMMs, Mixture of Gaussians)
- Principal Component Analysis (PCA)
- Autoencoders

Outline

- The Clustering problem
- K-means and K-means++ algorithm
- Gaussian Mixture Models (GMMs, Mixture of Gaussians)
- EM Algorithm

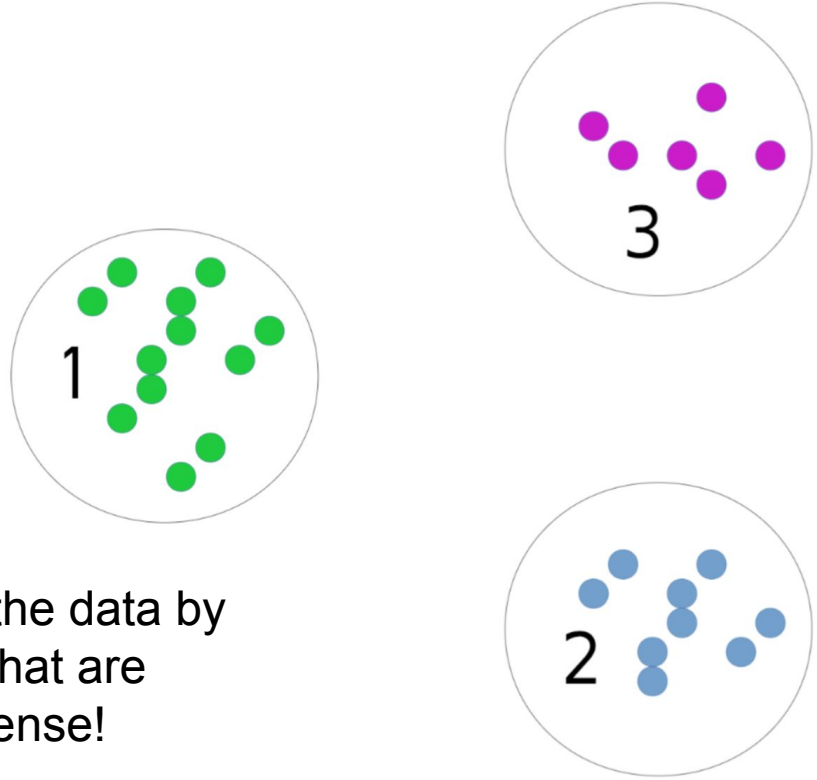
Clustering

The organization of unlabeled data into similarity groups called clusters



Clustering

The organization of unlabeled data into similarity groups called clusters



Idea: we can find structure in the data by isolating groups of examples that are similar in some well-defined sense!

Clustering

The organization of unlabeled data into similarity groups called clusters.

Input: $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$, target cardinality k

Output: partitioning of $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ into k groups

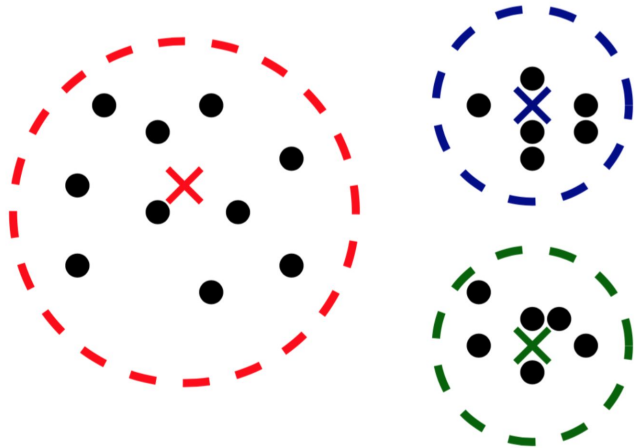
Clustering

The organization of unlabeled data into similarity groups called clusters.

Input: $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$, target cardinality k

Output: partitioning of $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ into k groups

Sometimes we also have a “representative” of each cluster, $\boldsymbol{c}_i \in \mathbb{R}^d$ for each cluster



What is a good clustering?

A good clustering method will produce high quality clusters with

- **high intra-class similarity**: data points in a cluster are near to the cluster centroid
- **low inter-class similarity**: different cluster centroids should be far away from one another

K-means Clustering

Input: $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$, target cardinality k

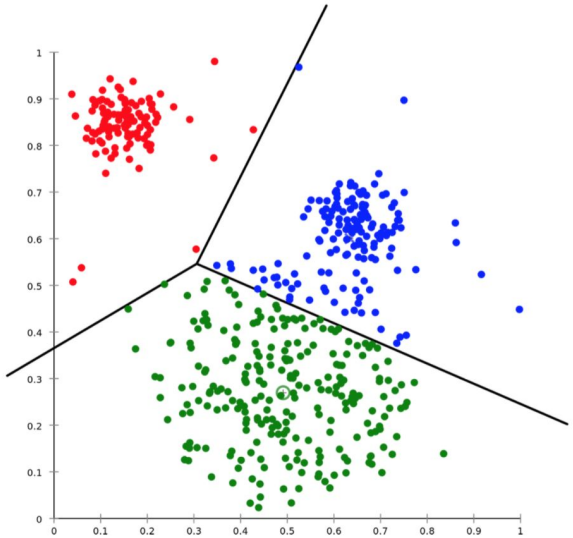
Output: k representatives (“centers”, “means”) $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k \in \mathbb{R}^d$

K-means Clustering

Input: $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$, target cardinality k

Output: k representatives (“centers”, “means”) $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k \in \mathbb{R}^d$

Objective: choose $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k$ to minimize:



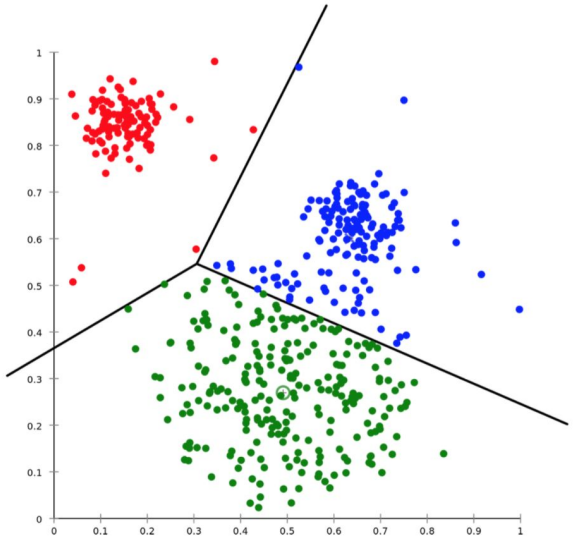
$$\sum_{i=1}^{n} \min_{j \in \{1,2,\ldots,k\}} \|\boldsymbol{x}_i - \boldsymbol{c}_j\|_2^2$$

K-means Clustering

Input: $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$, target cardinality k

Output: k representatives ("centers", "means") $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k \in \mathbb{R}^d$

Objective: choose $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k$ to minimize:



$$\sum_{i=1}^{n} \min_{j \in \{1,2,\ldots,k\}} \|\boldsymbol{x}_i - \boldsymbol{c}_j\|_2^2$$

Natural assignment function:
Assign each data point to its closed centroid

$$f(\boldsymbol{x}) = \arg \min_{j \in \{1,2,\ldots,k\}} \|\boldsymbol{x} - \boldsymbol{c}_j\|_2^2$$

NP-hard, even if k = 2 or d = 2.

An easy case

k-means for k = 1

Input: $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$

Problem: Pick $\boldsymbol{c} \in \mathbb{R}^d$ to minimize $\sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{c}\|_2^2$

Solution:

An easy case

k-means for k = 1

Input: $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$

Problem: Pick $\boldsymbol{c} \in \mathbb{R}^d$ to minimize $\sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{c}\|_2^2$

Solution: $\boldsymbol{c} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i$

Why?

This can be viewed as a convex function of $\boldsymbol{c}$: Take the gradient w.r.t $\boldsymbol{c}$ and set it to zero

K-means Objective

A clustering can be represented by a cluster map, which is a function

$$C : \{1, 2, \ldots, n\} \mapsto \{1, 2, \ldots, k\}$$

n: the number of data points
k: is the number of clusters

k-means objective is to choose to minimize the **within-cluster sum of squares (i.e., variance)**

$$\mathcal{L}(C) = \sum_{l=1}^{k} \sum_{i:C(i)=l} ||\boldsymbol{x}_i - \overline{\boldsymbol{x}}_l||^2$$

$\overline{\boldsymbol{x}}_l$: the mean of $\boldsymbol{x}_i$ belonging to cluster l

K-means Objective

A clustering can be represented by a cluster map, which is a function

$$C : \{1, 2, \ldots, n\} \mapsto \{1, 2, \ldots, k\}$$

n: the number of data points
k: is the number of clusters

It is equivalent to minimize the **the pairwise squared deviations of points in the same cluster**

$$\mathcal{L}(C) = \sum_{l=1}^{k} \sum_{i:C(i)=l} ||\boldsymbol{x}_i - \overline{\boldsymbol{x}}_l||^2$$

$$= \frac{1}{2} \sum_{l=1}^{k} \sum_{i:C(i)=l} \frac{1}{n_l} \sum_{j:C(j)=l} ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2$$

$\overline{\boldsymbol{x}}_l$: the mean of $\boldsymbol{x}_i$ belonging to cluster l
n_l: number of $\boldsymbol{x}_i$ belonging to cluster l

K-means Objective

$$\mathcal{L}(C) = \sum_{l=1}^{k} \sum_{i:C(i)=l} ||\boldsymbol{x}_i - \overline{\boldsymbol{x}}_l||^2$$

$$= \frac{1}{2} \sum_{l=1}^{k} \sum_{i:C(i)=l} \frac{1}{n_l} \sum_{j:C(j)=l} ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2$$

Minimizing this objective is a **combinatorial** optimization problem, the number of possible clustering map is

$$\frac{1}{k!} \sum_{l=1}^{k} (-1)^{k-l} \binom{k}{l} l^n$$

This is a huge search space. There is no known efficient algorithm to solve this. We need to use some iterative and suboptimal algorithm.

K-means algorithm

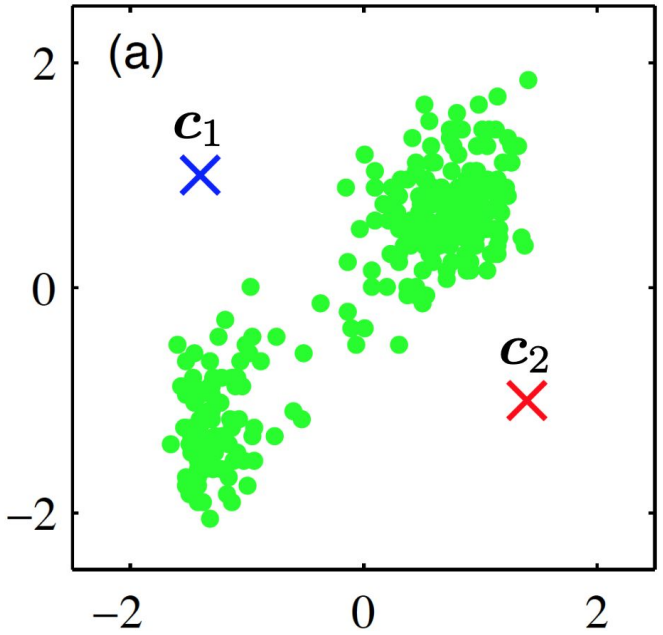
Recall that the mean of the data points minimize the distance to other points in the same cluster, this suggestions an **alternating** algorithm:

Random choose centers $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k$

1. Update clustering assignment: Cluster each point with the center nearest to it
2. Update clustering center: Given the clustering, calculate the mean as the new centers

Repeat the above two steps until the centers converge according to some criterion, such as the clustering don't change.

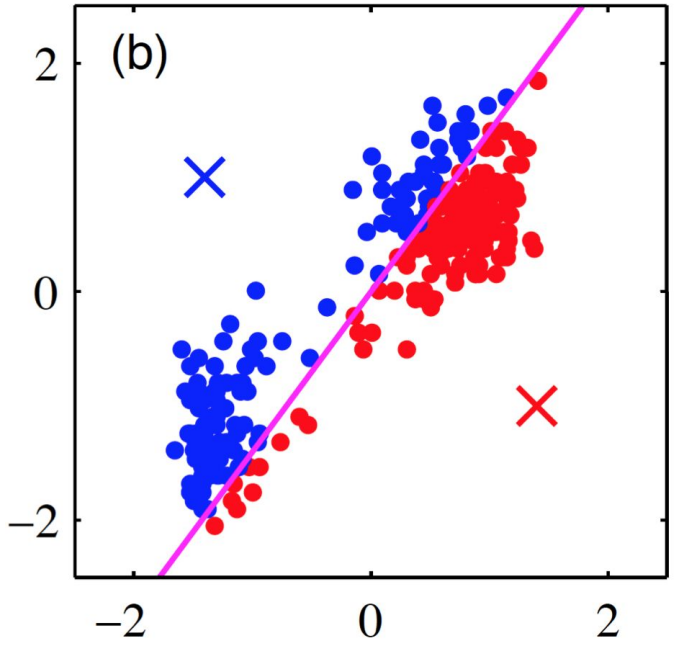
Example



Arbitrary initialization of c_1 and c_2

Note: we usually choose initial centers uniformly at random from the input data points!

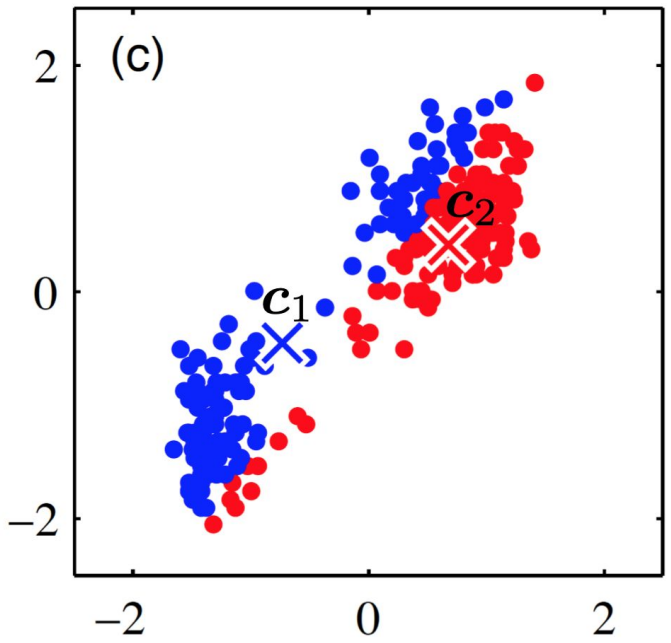
Example



Iteration 1

Optimize assignments (each point is assigned to the closest center)

Example



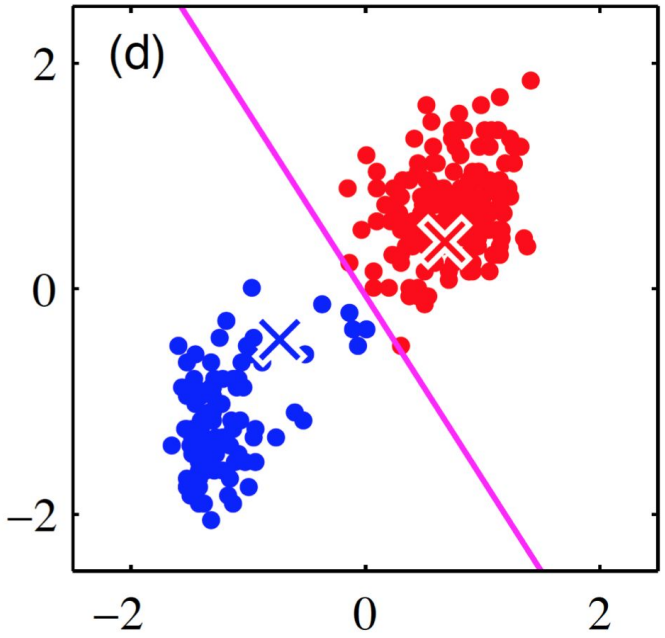
Iteration 1

Optimize representatives

$$c_1 = \frac{\text{sum of all blue points}}{\text{number of blue points}}$$

$$c_2 = \frac{\text{sum of all red points}}{\text{number of red points}}$$

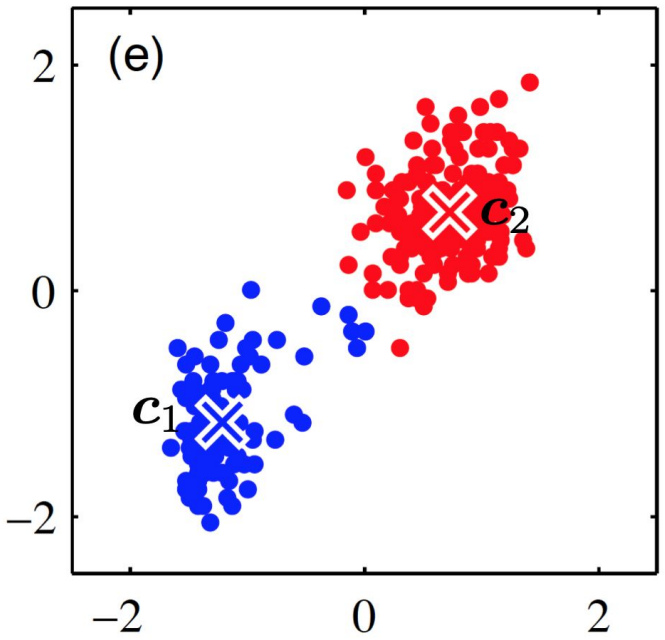
Example



Iteration 2

Optimize assignments (each point is assigned to the closest center)

Example



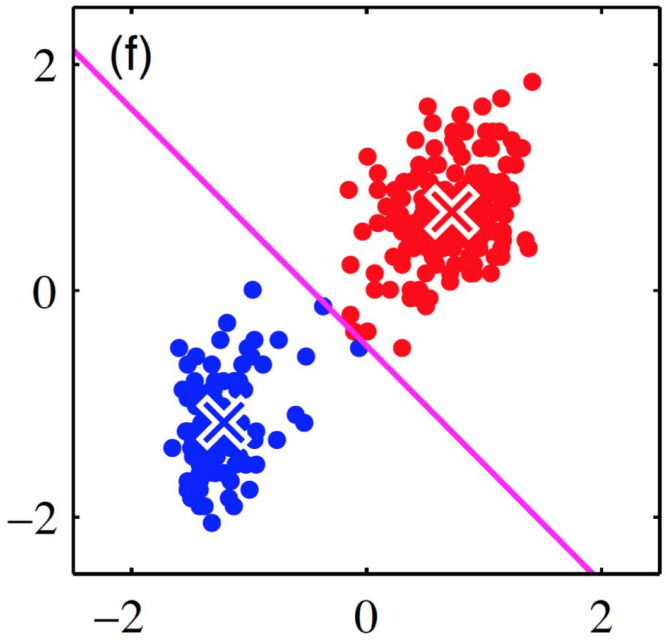
Iteration 2

Optimize representatives

$$c_1 = \frac{\text{sum of all blue points}}{\text{number of blue points}}$$

$$c_2 = \frac{\text{sum of all red points}}{\text{number of red points}}$$

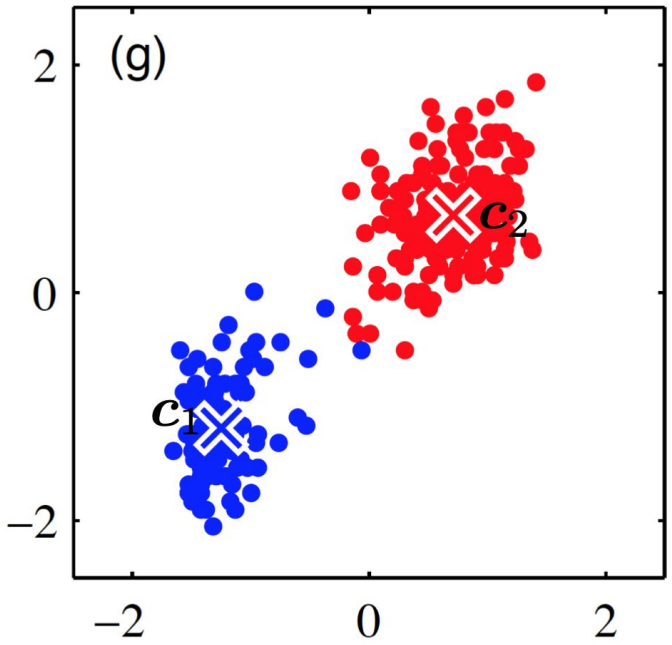
Example



Iteration 3

Optimize assignments (each point is assigned to the closest center)

Example



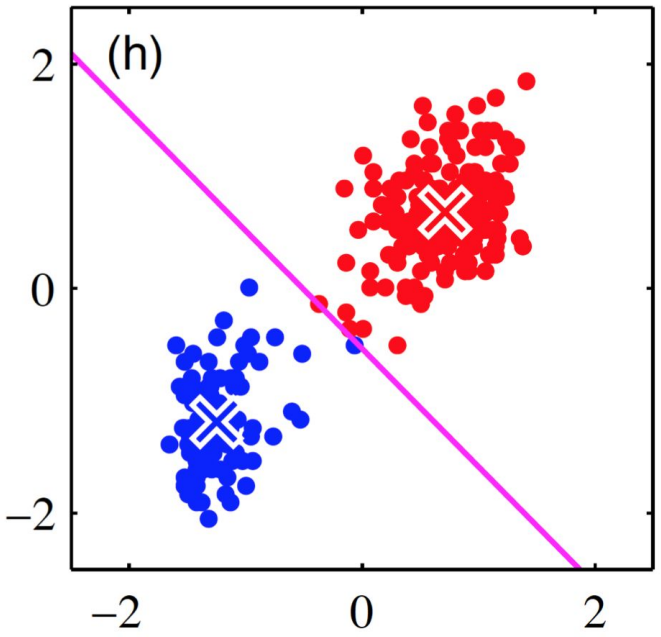
Iteration 3

Optimize representatives

$$c_1 = \frac{\text{sum of all blue points}}{\text{number of blue points}}$$

$$c_2 = \frac{\text{sum of all red points}}{\text{number of red points}}$$

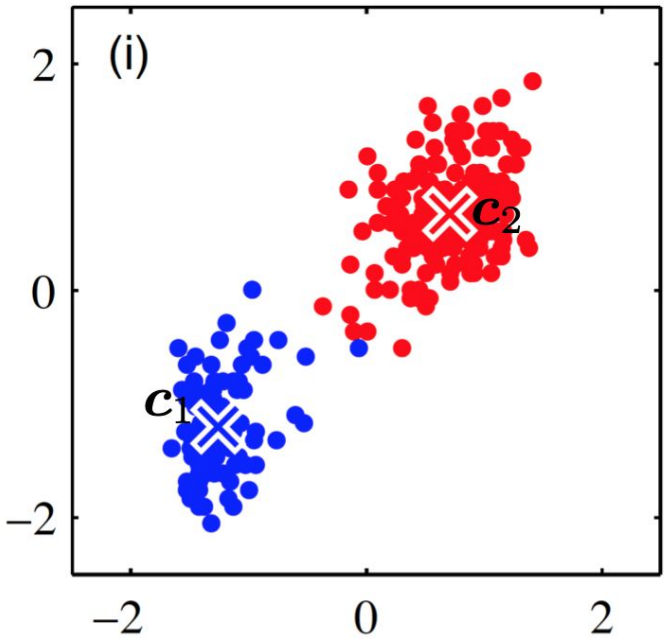
Example



Iteration 4

Optimize assignments (each point is assigned to the closest center)

Example



Iteration 4

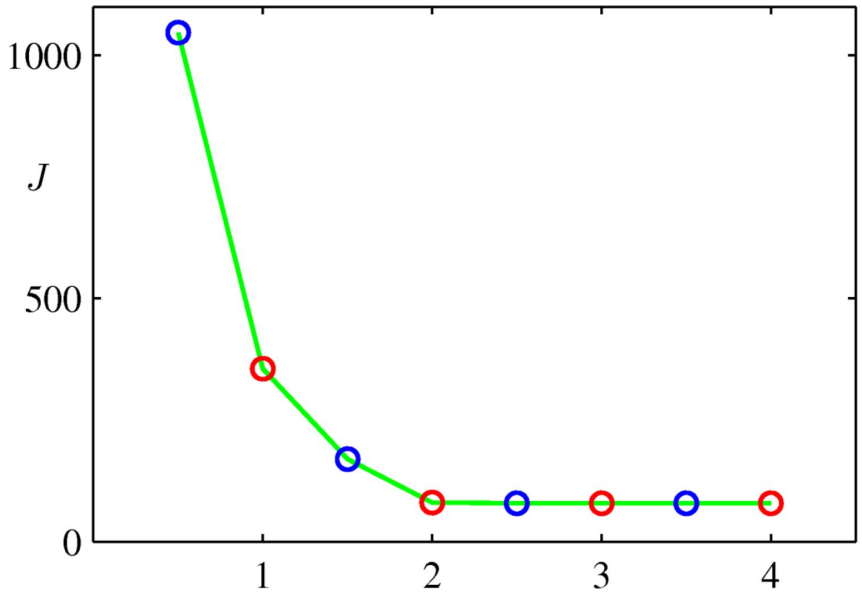
Optimize representatives

$$c_1 = \frac{\text{sum of all blue points}}{\text{number of blue points}}$$

$$c_2 = \frac{\text{sum of all red points}}{\text{number of red points}}$$

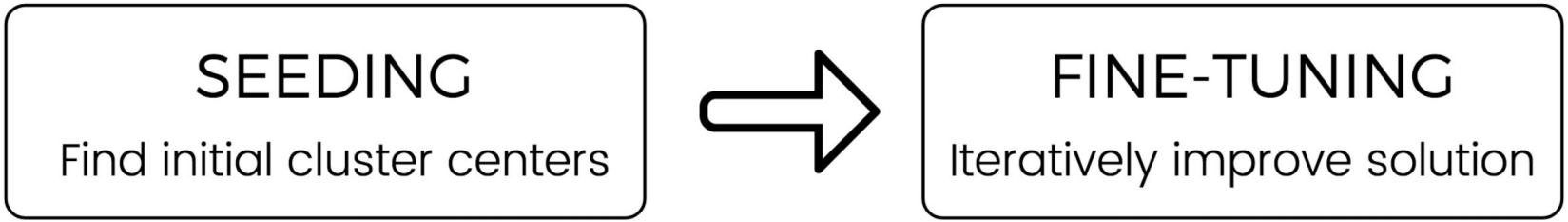
Convergence

Convergence is relatively quick, in number of iterations.
However, all those distance computations are expensive.



k-means algorithm

Most popular clustering approach (non-convex)
Many local minima may exists



Determines which local minimum is searched.

Ensures that local minimum is reached!

K-means Algorithm

The algorithm iterates in an alternating manner.

Closed form solutions exist for each step.

Hard clustering: A data point belongs to exactly one cluster

The objective function is monotonic decreasing.

The algorithm always converges to a local minimum of the objective.

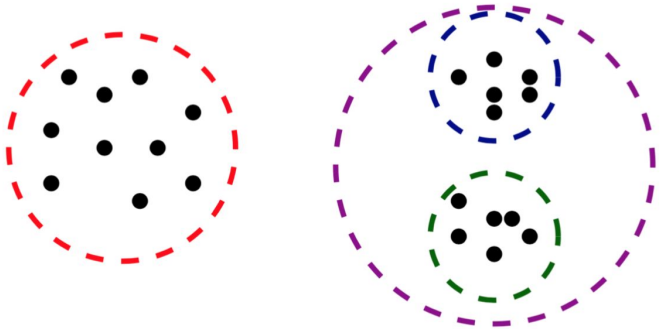
The algorithm might converge to a local solution (due to bad initialization)

Choosing *k*

Usually by hold-out validation/cross-validation on auxiliary task (e.g., supervised learning task).

Heuristic: Find large gap between (k −1)-means cost and k-means cost.

Clustering at multiple scales



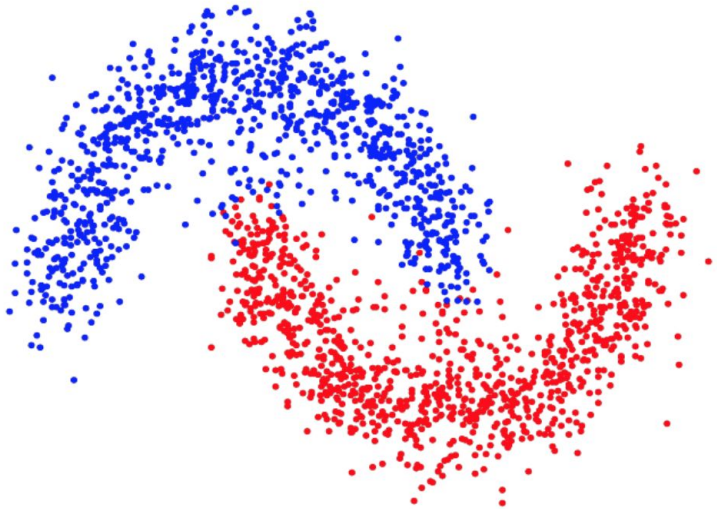
k=2: two large clusters
k=3: one large and two small clusters

$k = 2$ or $k = 3$?

Hierarchical clustering: encode clusterings for all values of k in a tree.

Clustering of Manifold Structure

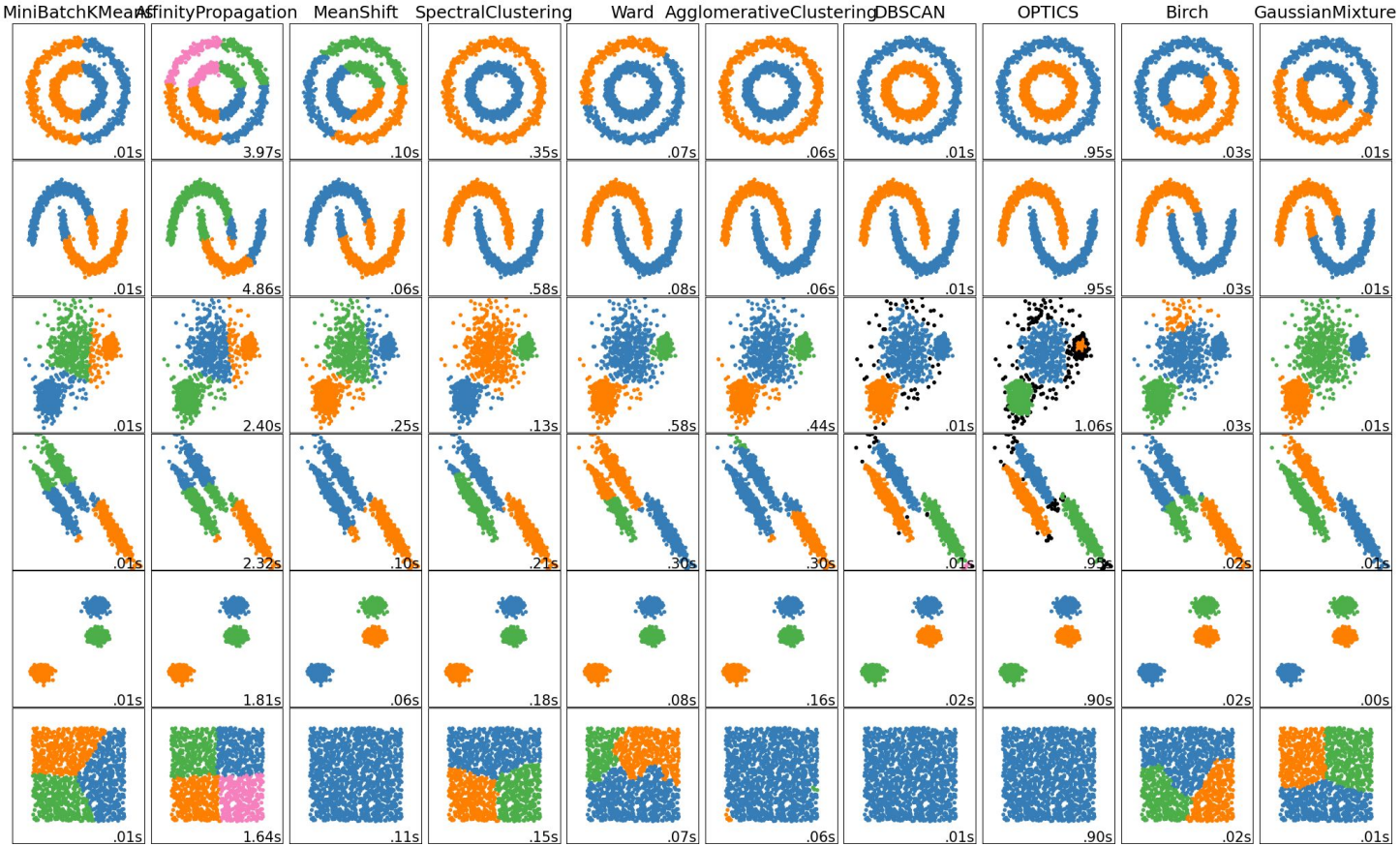
Sometimes it is hard to cluster! (Manifold Structure)



Clustering results are crucially dependent on the measure of similarity (or distance) between the “points” to be clustered! e.g., Euclidean distance is not appropriate for the data.

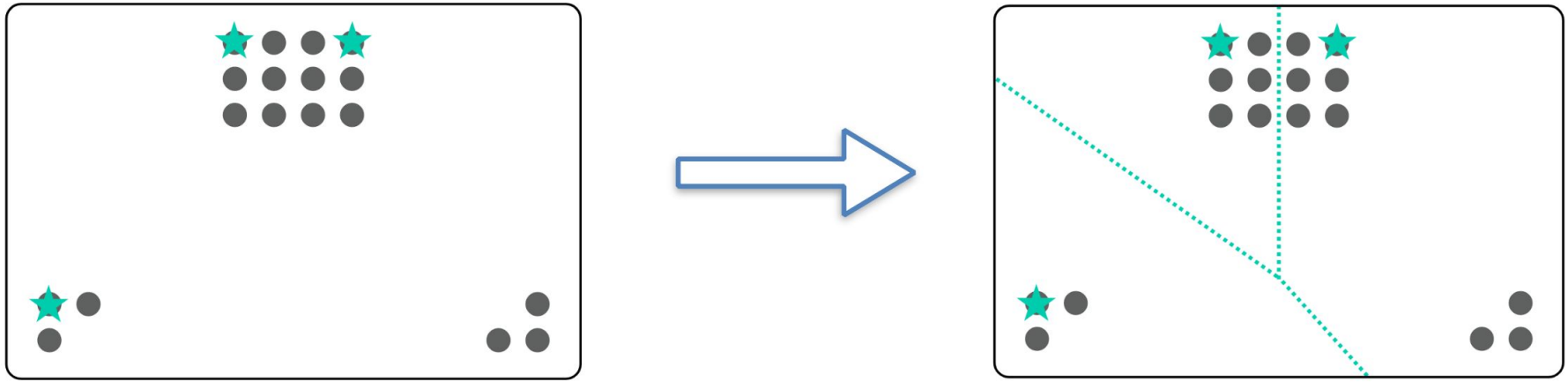
We can utilize kernel trick or map data to a higher dimensions and then do clustering (**kernelized k-means**)!

Clustering algorithms

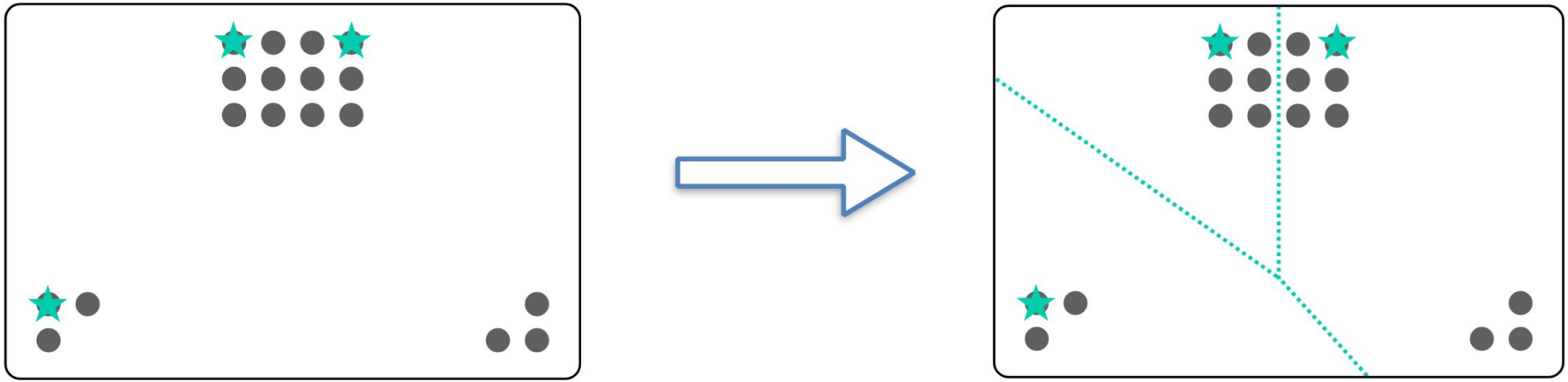


https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

Bad initialization might lead to a bad local minima!



Initialization



- Because the initial centers can substantially influence the quality of the result, there has been significant work on initialization strategies for K-means algorithm.
- One such idea is **K-means++**

K-means++: Smarter Initialization

Smarter initialization (good seedings) by initializing the clusters to be far apart.

Initialization

Pick a sample data point as first cluster center $\boldsymbol{c}_1$

for i = 2, …, k

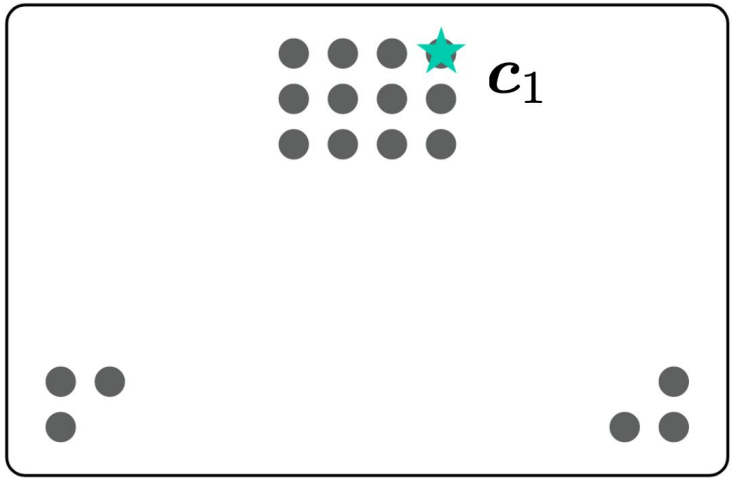
For each $\boldsymbol{x}$, compute the distance from $\boldsymbol{x}$ to the nearest cluster center that has been selected, denote this as $D(\boldsymbol{x})$

Choose one new data point $\boldsymbol{x}$ at random as a new center $\boldsymbol{c}_i$, with probability proportional to $D(\boldsymbol{x})^2$

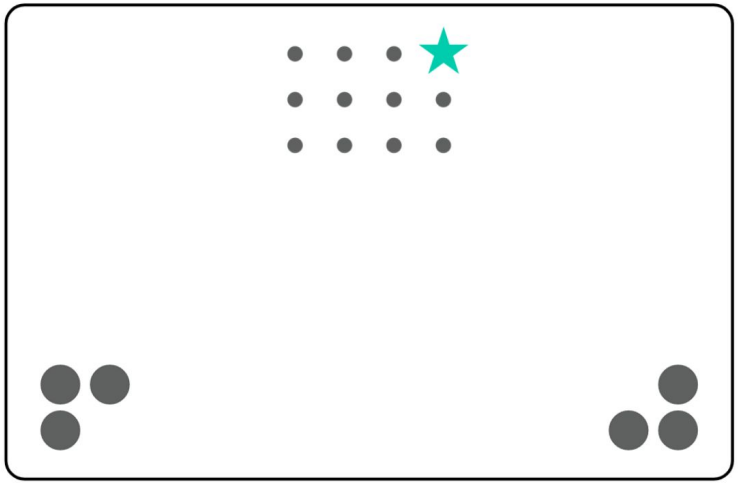
K-means++: Smarter Initialization

We need to decided three cluster centers.

Choose the first cluster center to be a random data point.

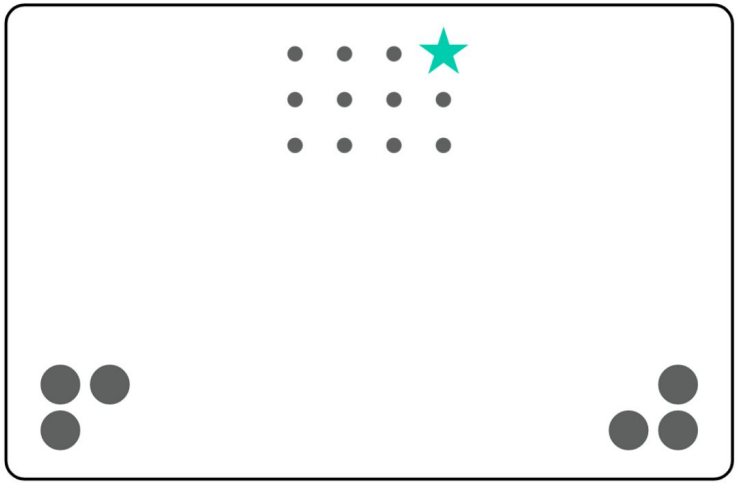


K-means++: Smarter Initialization

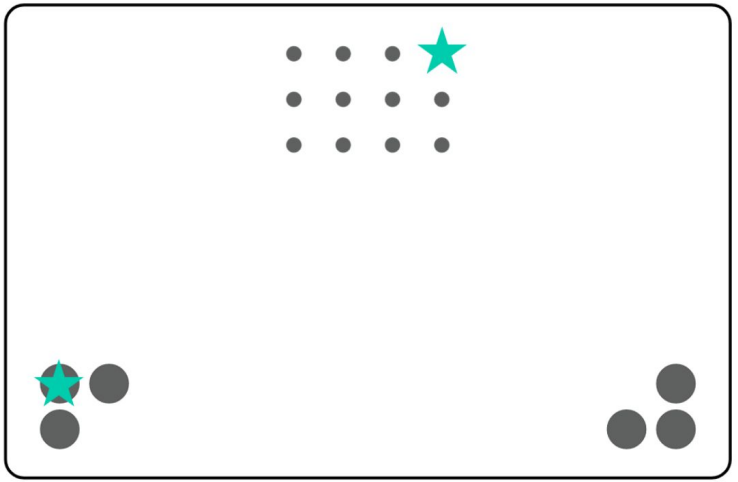


Compute the distance of every point to $\boldsymbol{c}_1$

K-means++: Smarter Initialization

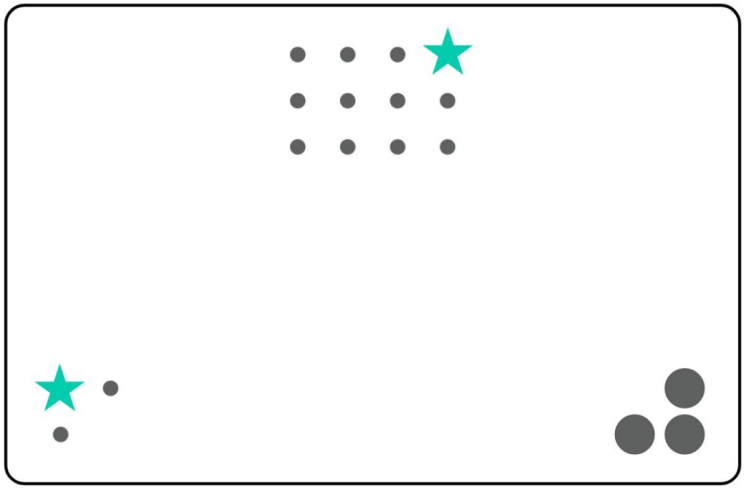


Compute the distance of every point to $\boldsymbol{c}_1$



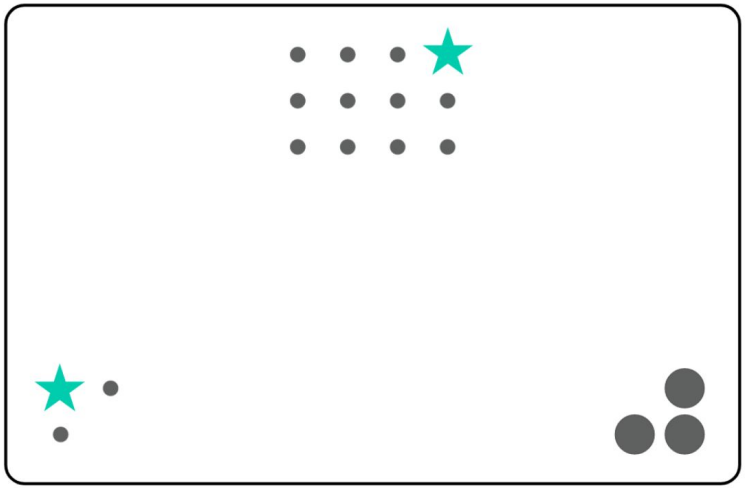
Pick the second center with probability proportional to the distance of a point to the closest center.

K-means++: Smarter Initialization

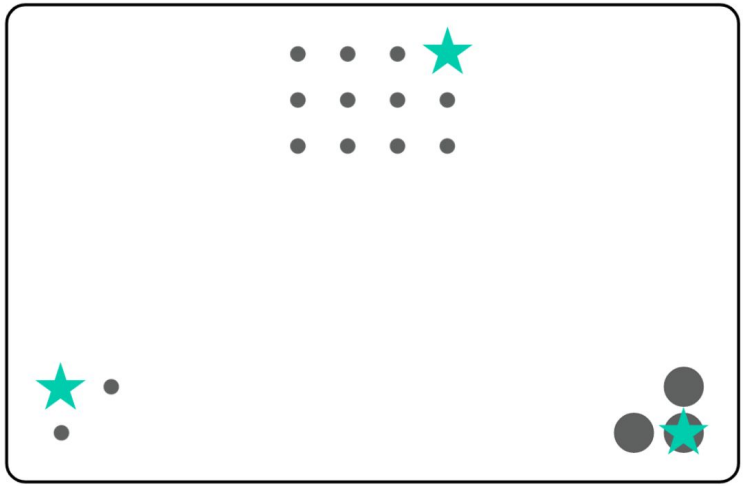


Compute the distance of every point to two centers and pick the smallest.

K-means++: Smarter Initialization



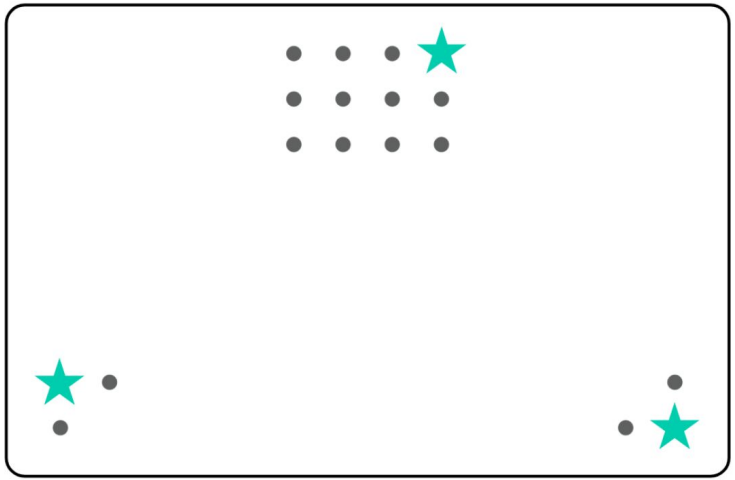
Compute the distance of every point to two centers and pick the smallest.

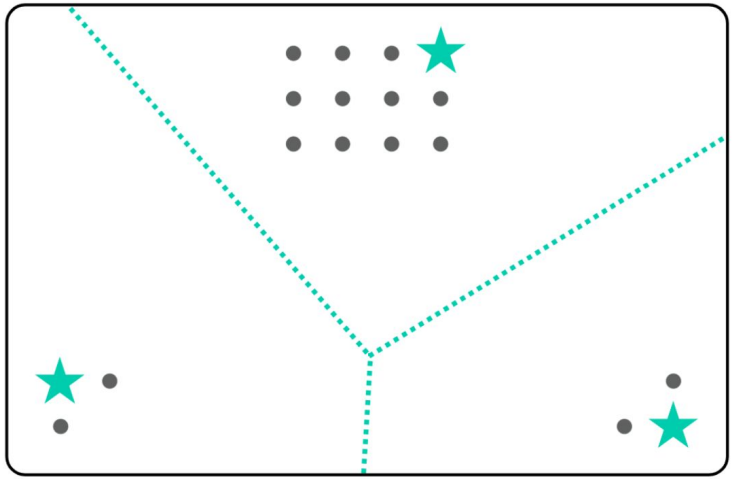


Pick the third center with probability proportional to the distance of point to the closest center we have already seen.

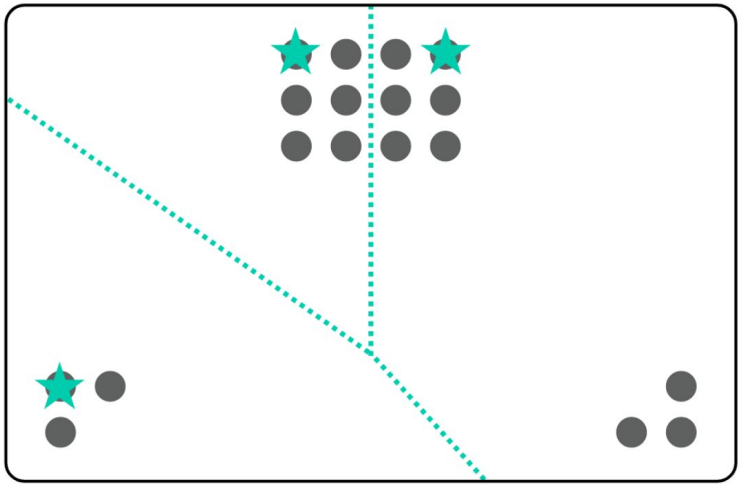
K-means++: Smarter Initialization

After initialization, continue with the k-means algorithm with obtained centers!

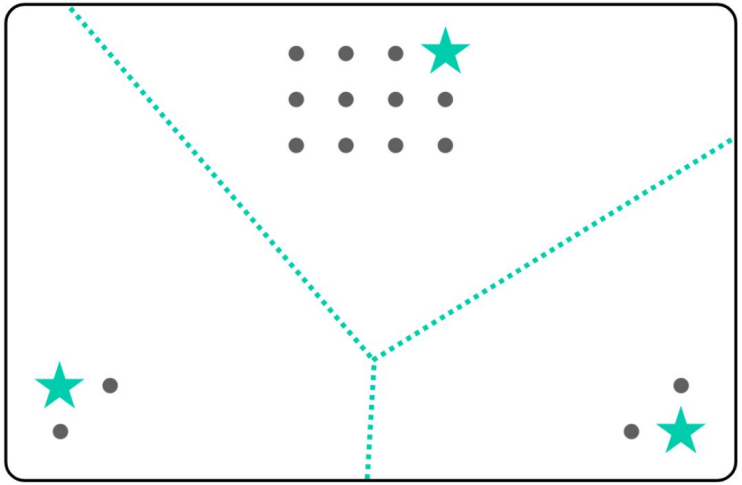




Initialization: bad and good



k-means: (a bad initialization
that results in a local minimum)



k-means++: (smart initialization)

Hard and Soft Clusters

K-Means uses **hard clustering assignment**.

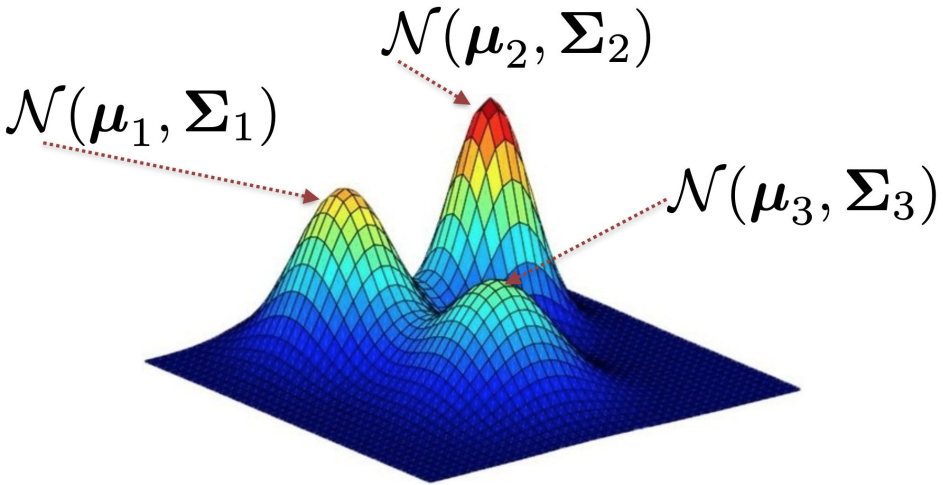
- A data point belongs to exactly one cluster

Mixture of Gaussians uses **soft clustering**.

- A point could be explained by more than one cluster.
- Different clusters take different levels of responsibility (posterior probability) for that point.
- It was actually generated by only one cluster, but we don’t know which one.

Gaussian Mixture Models

Assume the observed data points comes from multiple Gaussian distributions!



But, we do NOT know from which distribution individual data points are generated! (if we knew, then the problem would be a classification problem and we would use MLE to estimate the parameters of each distribution for future predictions!)

Gaussian Mixture Models

Recall the Multivariate Gaussian distribution:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

Gaussian Mixture Models

Recall the Multivariate Gaussian distribution:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

The Gaussian mixture is a **linear superposition** of K Gaussians:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \sum_{k=1}^{K} \pi_k = 1$$

Gaussian Mixture Models

Recall the Multivariate Gaussian distribution:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

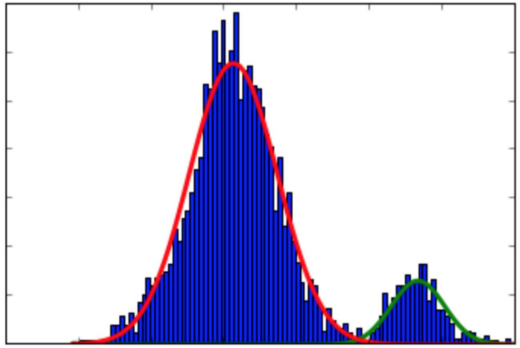
The Gaussian mixture is a **linear superposition** of K Gaussians:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \sum_{k=1}^{K} \pi_k = 1$$

The π_k is non-negative scalars called **mixing coefficients** and they govern the relative importance between the various Gaussians in the mixture density. So model parameters are

$$\theta = (\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), k = 1, \ldots, K$$

Gaussian Mixture Models ($K = 2$, $d = 1$)



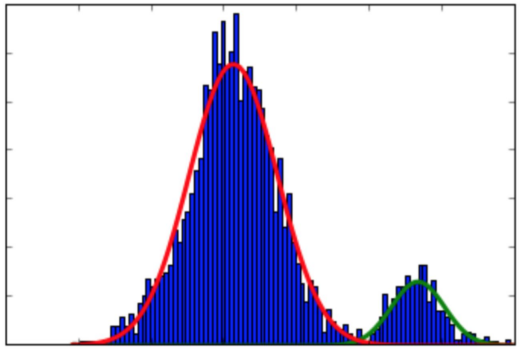
Gaussian Mixture Models (K = 2, d = 1)

Let's S be a random variable to deciding the one of two distributions:

$$S \sim \text{Bernoulli}(\pi)$$

Then, the distribution of each data point is:

$$\text{P}(x) = \pi\mathcal{N}(\mu_1, \sigma_1) + (1 - \pi)\mathcal{N}(\mu_2, \sigma_2)$$



Data generation process using GMMs

Suppose we know the parameters of GMMs:

$$\theta = (\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), k = 1, \ldots, K$$

How to simulate the data generate process?

Data generation process using GMMs

Suppose we know the parameters of GMMs:

$$\theta = (\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), k = 1, \ldots, K$$

How to simulate the data generate process?

1. Select a component k at random, weighted according to π_k
2. Then draw a data point from the k-th Gaussian distribution $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

Data generation process using GMMs

Suppose we know the parameters of GMMs:

$$\theta = (\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), k = 1, \ldots, K$$

How to simulate the data generate process?

1. Select a component k at random, weighted according to π_k
2. Then draw a data point from the k-th Gaussian distribution $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

The first step corresponds to realize the following discrete random variable

$$S \in \{1, \ldots, K\} \qquad \mathbb{P}(S = k) = \pi_k$$

Data generation process using GMMs

Suppose we know the parameters of GMMs:

$$\theta = (\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), k = 1, \ldots, K$$

How to simulate the data generate process?

1. Select a component k at random, weighted according to π_k
2. Then draw a data point from the k-th Gaussian distribution $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

The first step corresponds to realize the following discrete random variable

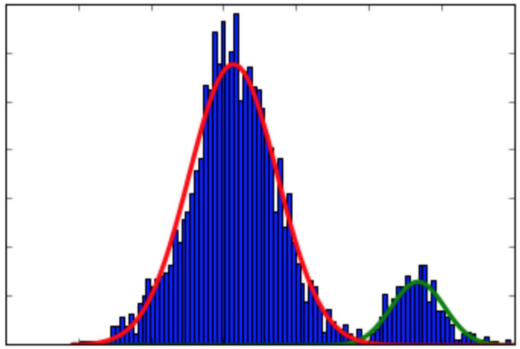
$$S \in \{1, \ldots, K\} \qquad \mathbb{P}(S = k) = \pi_k$$

This random variable S is an example of a state variable, and it is **latent** or **hidden**, because we don't observe it (i.e, we don't know which Gaussian distribution where each data comes from). We will imagine that every generated data point from GMM is associated with a hidden state variable

Gaussian Mixture Models (K = 2, d = 1)

Let's Y be a random variable to deciding the one of two distributions:

$$Y \sim \text{Bernoulli}(\pi)$$



Then, the distribution of each data point is:

$$\text{P}(x) = \pi\mathcal{N}(\mu_1, \sigma_1) + (1 - \pi)\mathcal{N}(\mu_2, \sigma_2)$$

Then, the MLE of observed data points will be:

$$\arg \max_{\pi,\mu_1,\sigma_1,\mu_2,\sigma_2} \prod_{i=1}^{n} \text{P}(x_i)$$

$$= \arg \max_{\pi,\mu_1,\sigma_1,\mu_2,\sigma_2} \prod_{i=1}^{n} \pi\mathcal{N}(x_i|\mu_1, \sigma_1) + (1 - \pi)\mathcal{N}(x_i|\mu_2, \sigma_2)$$

MLE for GMMs

To do clustering $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$, we need to use MLE to infer (K is fixed):

$$\theta = (\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), k = 1, \ldots, K$$

MLE for GMMs

To do clustering $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$, we need to use MLE to infer (K is fixed):

$$\theta = (\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), k = 1, \ldots, K$$

If K = 1: the MLE has the closed-form solution

$$\pi_1 = 1$$

$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i$$

$$\hat{\boldsymbol{\Sigma}}_1 = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})^\top$$

MLE for GMMs

To do clustering $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$, we need to use MLE to infer (K is fixed):

$$\theta = (\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), k = 1, \ldots, K$$

If K > 1, the MLE does not have a closed-form solution.

MLE for GMMs

To do clustering $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$, we need to use MLE to infer (K is fixed):

$$\theta = (\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), k = 1, \ldots, K$$

If K > 1, the MLE does not have a closed-form solution. Recall that the data points follow the following GMM distribution:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

So the data log likelihood is

$$\mathcal{L}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\theta) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

MLE for GMMs

$$\mathcal{L}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\theta) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

However, maximizing this data log likelihood w.r.t. to θ is intractable. It is hard to compute derivatives of this log of sum form concerning model parameters!

MLE for GMMs

$$\mathcal{L}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\theta) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

However, maximizing this data log likelihood w.r.t. to θ is intractable. It is hard to compute derivatives of this log of sum form concerning model parameters!
So we will use an **iterative strategy** similar to k-means algorithm, which hinges critically on the **state variable** associated with each data point:

$$\boldsymbol{s} = (s_1, s_2, \ldots, s_n), \text{where} \;\; s_i \in \{1, \ldots, K\}$$

- Given θ , update the estimate of $\boldsymbol{s}$
- Given $\boldsymbol{s}$, update the estimate of θ

Each step can be performed efficiently

The EM Algorithm for GMMs

$$(\boldsymbol{x}_i, s_i) \quad i = 1, \ldots, n$$

This is called **complete data** = observed variable + hidden variable.

The EM Algorithm for GMMs

$$(\boldsymbol{x}_i, s_i) \quad i = 1, \ldots, n$$

This is called **complete data** = observed variable + hidden variable. To do this iterative algorithm, we are interested in the **complete-data log likelihood**:

$$\mathcal{L}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, s_1, s_2, \ldots, s_n | \theta) = \sum_{i=1}^{n} \log \pi_{s_i} \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{\mu}_{s_i}, \boldsymbol{\Sigma}_{s_i})$$

The EM Algorithm for GMMs

$$(\boldsymbol{x}_i, s_i) \quad i = 1, \ldots, n$$

This is called **complete data** = observed variable + hidden variable. To do this iterative algorithm, we are interested in the **complete-data log likelihood**:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, s_1, s_2, \ldots, s_n | \theta) &= \sum_{i=1}^{n} \log \pi_{s_i} \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{\mu}_{s_i}, \boldsymbol{\Sigma}_{s_i}) \\
&= \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \Delta_{i,k} \pi_k \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right) \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} \Delta_{i,k} \log \pi_k \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} \Delta_{i,k} \left(\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)
\end{aligned}$$

where we define an indicator variable:

$$\Delta_{i,k} = \begin{cases} 1, & s_i = k \\ 0, & s_i \neq k \end{cases}$$

The EM Algorithm for GMMs

$$\mathcal{L}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, s_1, s_2, \ldots, s_n|\theta) = \sum_{i=1}^{n}\sum_{k=1}^{K} \Delta_{i,k} \left(\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right)$$

Recall that we don't know which cluster each data point is from, so we don't know $\Delta_{i,k}$
Otherwise, we can just do MLE for each cluster, which is tractable.

The EM Algorithm for GMMs

$$\mathcal{L}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, s_1, s_2, \ldots, s_n|\theta) = \sum_{i=1}^{n}\sum_{k=1}^{K} \Delta_{i,k} \left(\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right)$$

Recall that we don't know which cluster each data point is from, so we don't know $\Delta_{i,k}$
Otherwise, we can just do MLE for each cluster, which is tractable.

Therefore, we can instead use an iterative approach by alternating between

- E-Step: replacing $\Delta_{i,k}$ with its **expected value**, and then
- M-Step: **maximize** w.r.t.θ

The EM Algorithm for GMMs

$$\mathcal{L}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, s_1, s_2, \ldots, s_n|\theta) = \sum_{i=1}^{n}\sum_{k=1}^{K} \Delta_{i,k} \left(\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right)$$

Recall that we don't know which cluster each data point is from, so we don't know $\Delta_{i,k}$
Otherwise, we can just do MLE for each cluster, which is tractable.

Therefore, we can instead use an iterative approach by alternating between

- E-Step: replacing $\Delta_{i,k}$ with its **expected value**, and then
- M-Step: **maximize** w.r.t.θ

The EM algorithm is an iterative algorithm that produces a sequence of

$$\theta^{(1)}, \theta^{(2)}, \ldots$$

E-Step: replacing $\Delta_{i,k}$ with its expected value

Calculate the **expected complete-data log likelihood**

$$\mathcal{Q}(\theta, \theta^{(t)}) = \mathbb{E}_{S|\boldsymbol{x}} \left[\mathcal{L}(\boldsymbol{x}, s|\theta) | \boldsymbol{x}; \theta^{(t)} \right]$$

Conditional Probability w.r.t. $S|\boldsymbol{x}$
Here S is capitalized since it is viewed as a random variable

The pdf of $S|\boldsymbol{x}$ depends on the GMM parameters using the current estimate $\theta^{(t)}$

E-Step: replacing $\Delta_{i,k}$ with its expected value

Calculate the **expected complete-data log likelihood**

$$\begin{aligned}\mathcal{Q}(\theta, \theta^{(t)}) &= \mathbb{E}_{S|\boldsymbol{x}}\left[\mathcal{L}(\boldsymbol{x}, s|\theta)|\boldsymbol{x}; \theta^{(t)}\right] \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} \gamma_{i,k}^{(t)} \left(\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right)\end{aligned}$$

$$\begin{aligned}\gamma_{i,k}^{(t)} &= \mathbb{E}[\Delta_{i,k}|\boldsymbol{x}_i; \theta^{(t)}] \\ &= \mathbb{P}(\Delta_{i,k} = 1|\boldsymbol{x}_i; \theta^{(t)})\end{aligned}$$

E-Step: replacing $\Delta_{i,k}$ with its expected value

Calculate the **expected complete-data log likelihood**

$$\begin{aligned}\mathcal{Q}(\theta, \theta^{(t)}) &= \mathbb{E}_{S|\boldsymbol{x}}\left[\mathcal{L}(\boldsymbol{x}, s|\theta)|\boldsymbol{x}; \theta^{(t)}\right] \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} \gamma_{i,k}^{(t)} \left(\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right)\end{aligned}$$

Using the current model $\theta^{(t)}$, $\gamma_{i,k}^{(t)}$ is

the Responsibility of cluster k for $\boldsymbol{x}_i$, i.e.,
the soft membership of $\boldsymbol{x}_i$ belonging to cluster k

$$\begin{aligned}\gamma_{i,k}^{(t)} &= \mathbb{E}[\Delta_{i,k}|\boldsymbol{x}_i; \theta^{(t)}] \\ &= \mathbb{P}(\Delta_{i,k} = 1|\boldsymbol{x}_i; \theta^{(t)}) \\ &= \mathbb{P}(s_i = k|\boldsymbol{x}_i; \theta^{(t)}) \\ &= \frac{\mathbb{P}(s_i = k; \theta^{(t)}) f(\boldsymbol{x}_i|s_i = k; \theta^{(t)})}{f(\boldsymbol{x}_i; \theta^{(t)})} \\ &= \frac{\pi_k^{(t)} \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})}{\sum_{l=1}^{K} \pi_l^{(t)} \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_l^{(t)}, \boldsymbol{\Sigma}_l^{(t)})}\end{aligned}$$

M-Step: maximize w.r.t. θ

Compute

$$\theta^{(t+1)} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{(t)})$$

M-Step: maximize w.r.t. θ

Compute

$$\theta^{(t+1)} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{(t)})$$

where

$$\begin{aligned}
\mathcal{Q}(\theta, \theta^{(t)}) &= \mathbb{E}_{S|\boldsymbol{x}}\left[\mathcal{L}(\boldsymbol{x}, s|\theta)|\boldsymbol{x}; \theta^{(t)}\right] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} \gamma_{i,k}^{(t)} \left(\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} \gamma_{i,k}^{(t)} \left(\log \pi_k - \frac{d}{2}\log 2\pi - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| - \frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)\right)
\end{aligned}$$

To maximize this, take the gradient with respect to the parameters and set them to zero (why this is tractable now? Because there is no marginalization inside log!)

M-Step: maximize w.r.t. θ

Compute

$$\theta^{(t+1)} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{(t)})$$

Solution:

$$\pi_k^{(t+1)} =$$

$$\boldsymbol{\mu}_k^{(t+1)} =$$

$$\boldsymbol{\Sigma}_k^{(t+1)} =$$

M-Step: maximize w.r.t. θ

Compute

$$\theta^{(t+1)} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{(t)})$$

Solution:

fraction of all data that is explained by k-th component

$$\pi_k^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} \gamma_{i,k}^{(t)}$$

$$\boldsymbol{\mu}_k^{(t+1)} =$$

$$\boldsymbol{\Sigma}_k^{(t+1)} =$$

M-Step: maximize w.r.t. θ

Compute

$$\theta^{(t+1)} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{(t)})$$

Solution:

fraction of all data that is explained by k-th component

$$\pi_k^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} \gamma_{i,k}^{(t)}$$

weighted sample mean for k-th component

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \gamma_{i,k}^{(t)} \boldsymbol{x}_i}{\sum_{i=1}^{n} \gamma_{i,k}^{(t)}}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} =$$

M-Step: maximize w.r.t. θ

Compute

$$\theta^{(t+1)} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{(t)})$$

Solution:

fraction of all data that is explained by k-th component

$$\pi_k^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} \gamma_{i,k}^{(t)}$$

weighted sample mean for k-th component

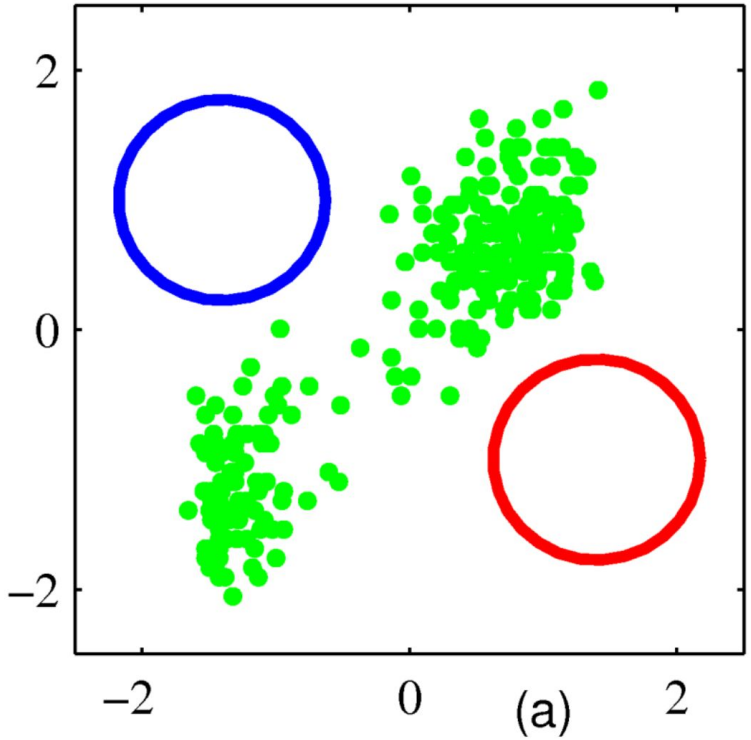
$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \gamma_{i,k}^{(t)} \boldsymbol{x}_i}{\sum_{i=1}^{n} \gamma_{i,k}^{(t)}}$$

weighted sample covariance for k-th component

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \gamma_{i,k}^{(t)} (\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(t+1)})(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(t+1)})^\top}{\sum_{i=1}^{n} \gamma_{i,k}^{(t)}}$$

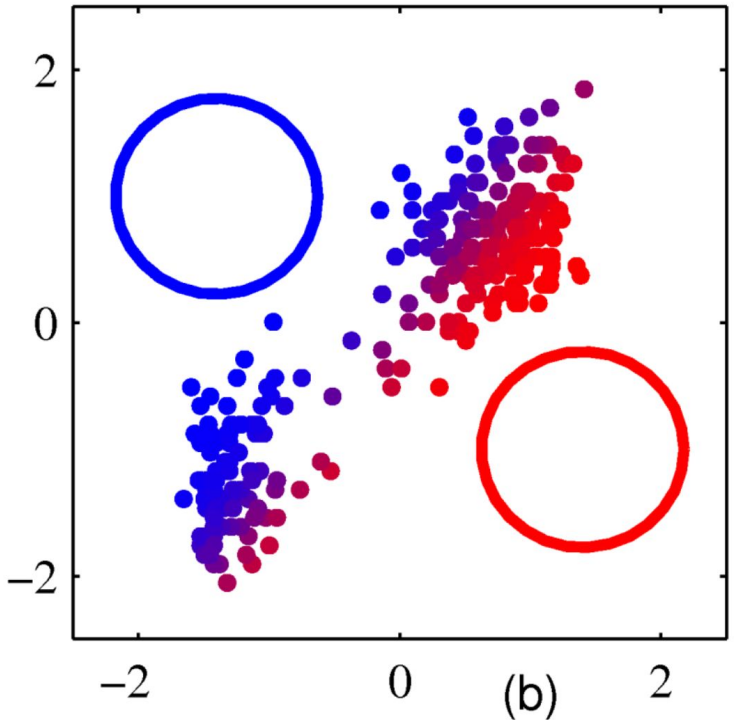
EM Example ($K = 2$, $d = 2$)

Initialize parameters: means, covariances, and mixing coefficients.



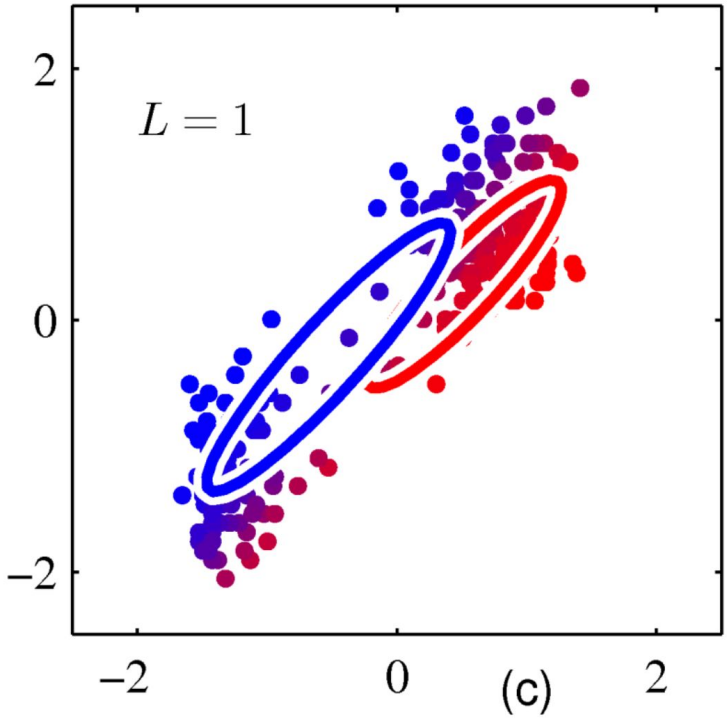
EM Example

First E-Step



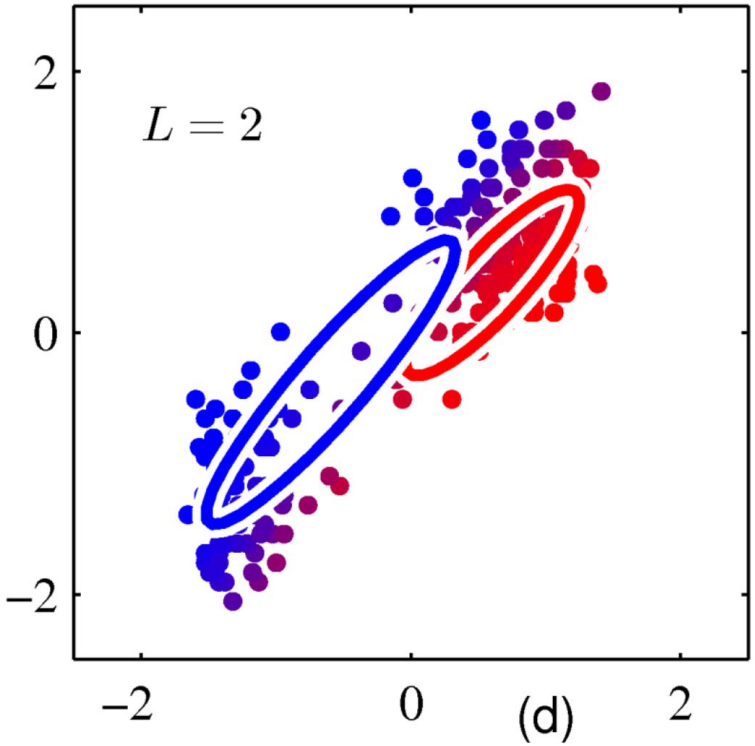
EM Example

First M-Step



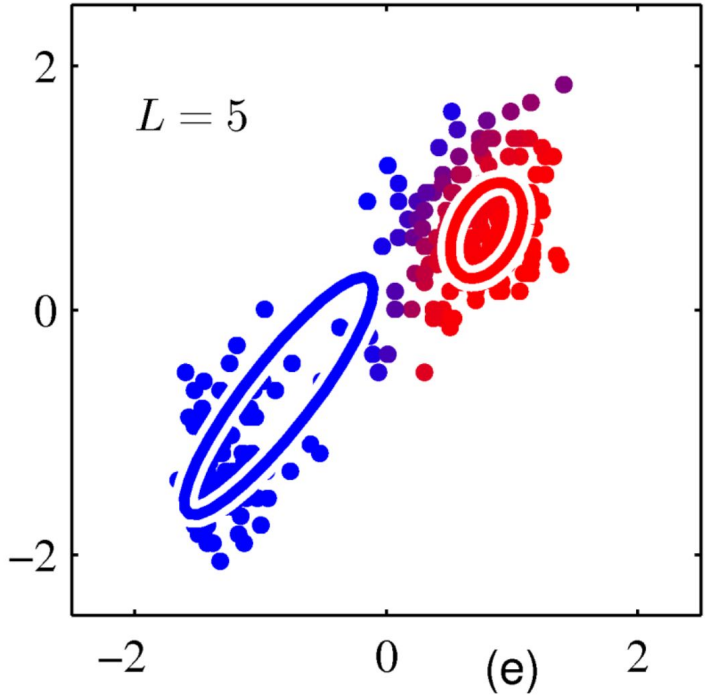
EM Example

Second E and M step



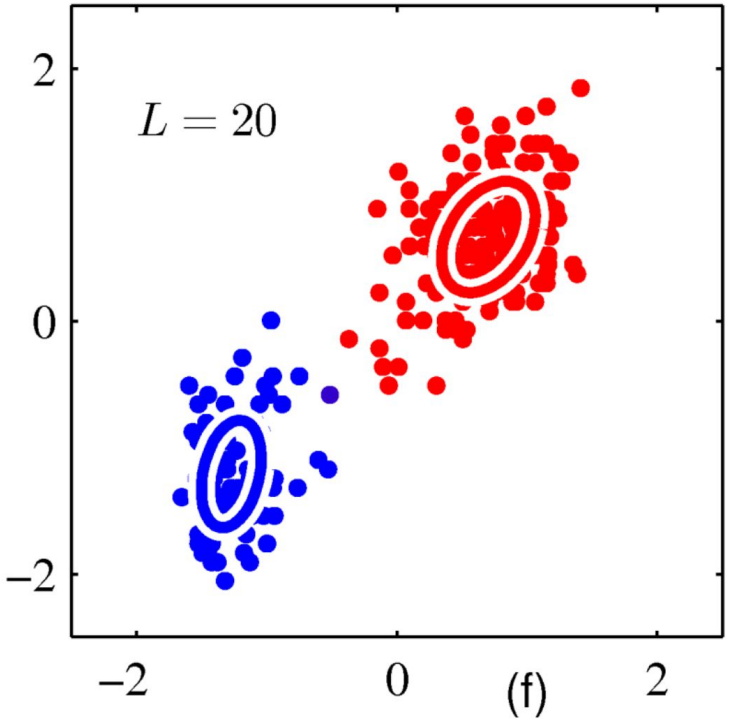
EM Example

Three more E and M steps



EM Example

After 20 E and M steps



Relation to K-means

EM for GMM looks really like K-means.

The K-means algorithm is a special case of EM algorithm. There is a precise correspondence:

- First, fix each cluster covariance matrix to be $\sigma^2 I$, i.e., Gaussian distributions for each cluster have a common, isotropic covariance
- As $\sigma^2 \to 0$, the update equations converge to doing k-means:

$$\gamma_{i,k} \to \begin{cases} 1, & k = \arg\min_l ||\boldsymbol{x}_i - \boldsymbol{\mu}_l||^2 \\ 0, & \text{otherwise} \end{cases}$$

EM Algorithm in General

The EM algorithm is not specific to GMMs. It can be applied to many other **maximum likelihood estimation problems where using hidden variables can make computation easier**.

EM Algorithm in General

The EM algorithm is not specific to GMMs. It can be applied to many other **maximum likelihood estimation problems where using hidden variables can make computation easier**. As before, we can denote

$$(\boldsymbol{x}_i, s_i) \quad i = 1, \ldots, n$$

where s_i is a hidden variable that explains how $\boldsymbol{x}_i$ was generated. Also, denote:

Data Log Likelihood:

$$\mathcal{L}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n | \theta)$$

Complete-Data Log Likelihood:

$$\mathcal{L}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, s_1, s_2, \ldots, s_n | \theta)$$

EM Algorithm in General

Initialize $\theta^{(0)}$

Repeat $t = 0, \ldots$

E-Step: Form

$$\mathcal{Q}(\theta, \theta^{(t)}) = \mathbb{E}_{S|\boldsymbol{x}} \left[\mathcal{L}(\boldsymbol{x}, s|\theta) | \boldsymbol{x}; \theta^{(t)} \right]$$

M-Step: Compute

$$\theta^{(t+1)} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{(t)})$$

Until termination criteria satisfied

Will it always converge to optimal?
no! It depends on the initialization!

Theorem. For each $t = 0, \ldots,$ the data log likelihood is non-decreasing, i.e.,

$$\mathcal{L}(\boldsymbol{x}; \theta^{(t+1)}) \geq \mathcal{L}(\boldsymbol{x}; \theta^{(t)})$$

For proof, check https://see.stanford.edu/materials/aimlcs229/cs229-notes8.pdf

EM as a Minimization-Maximization Algorithm

E-Step: Form a Minimization function

M-Step: Maximize the Minimization function

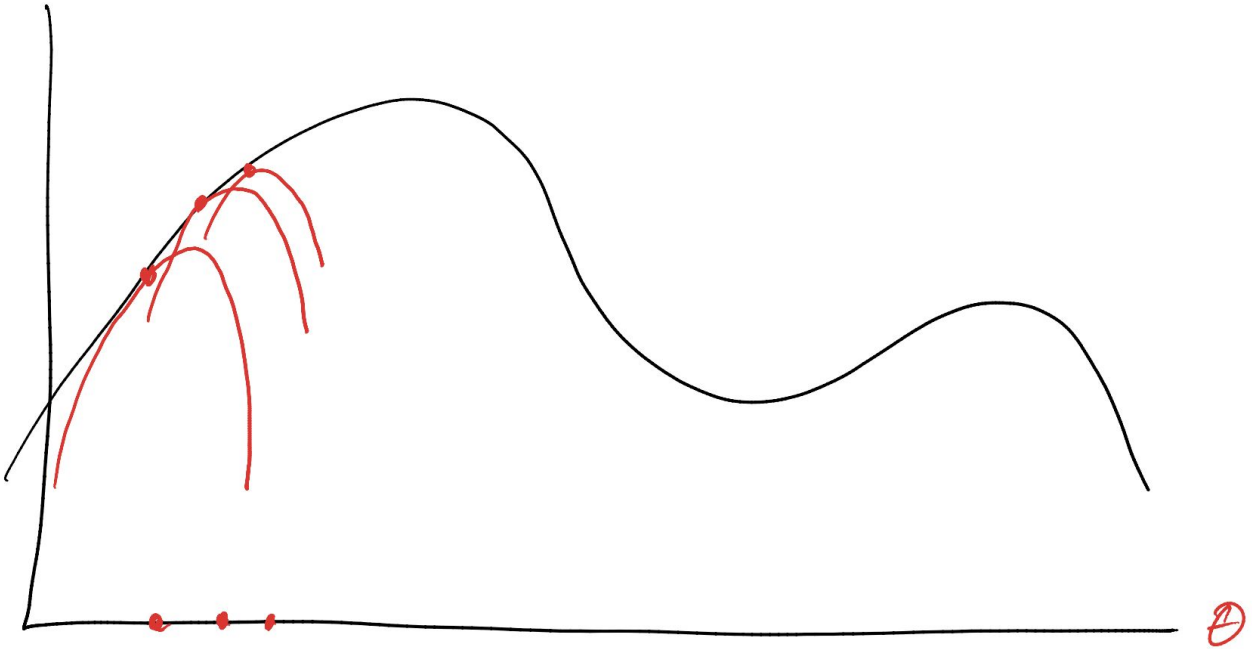


Figure from Clay Scott