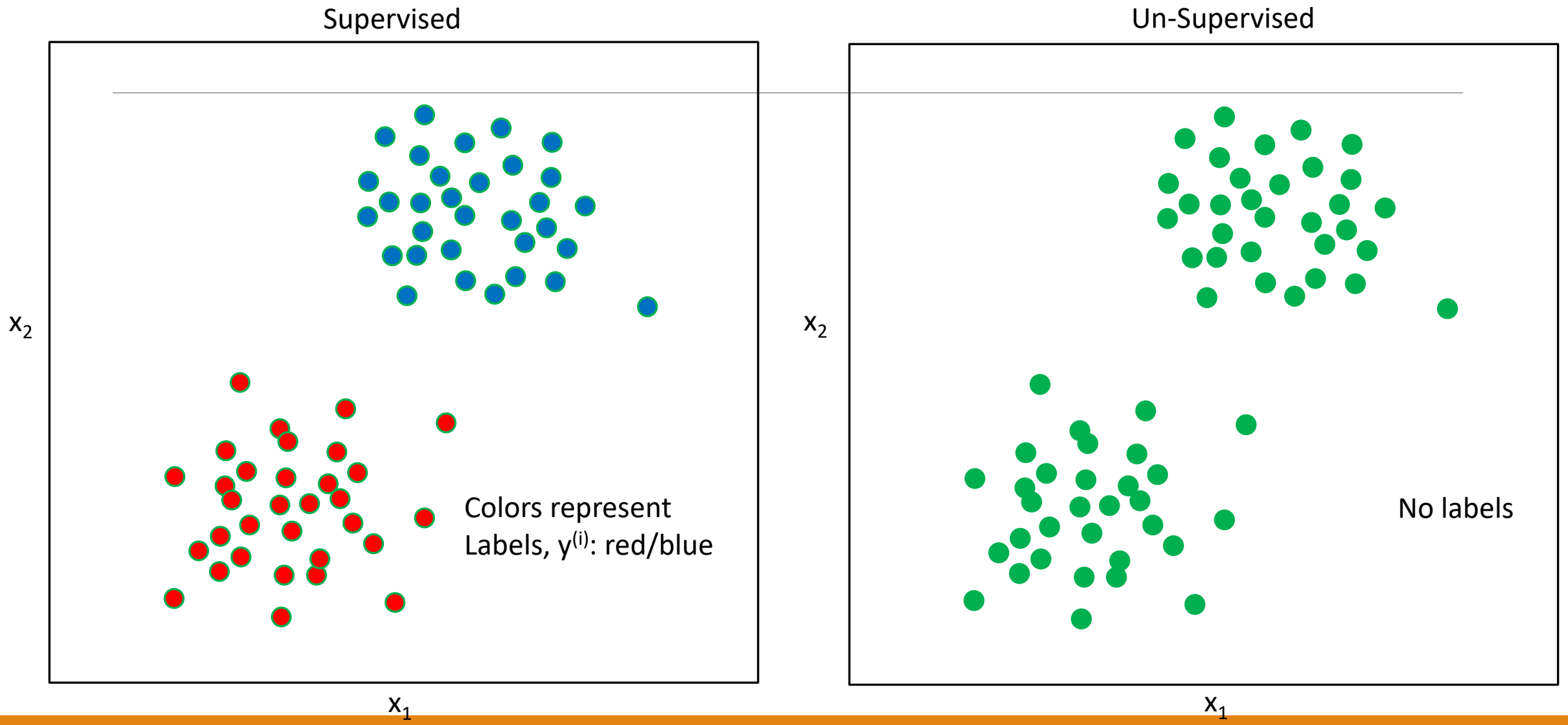
SDSU CS 549 Spring 2024
Machine Learning
Lecture 7: Clustering

IRFAN KHAN

References

- SDSU CS549 Lecture Notes by Prof Yang Xu, Spring 2023. Many slides from his deck.
- Coursera machine learning course by Dr Andrew Ng, Oct 2023

Supervised vs Unsupervised Learning

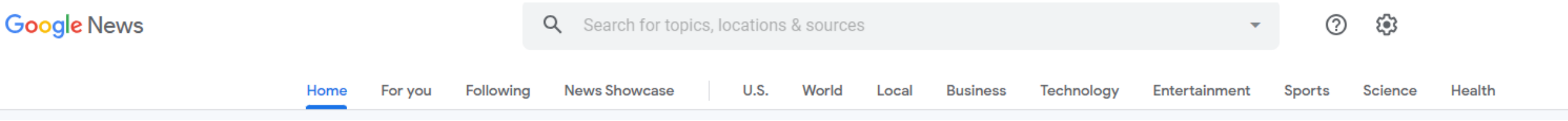


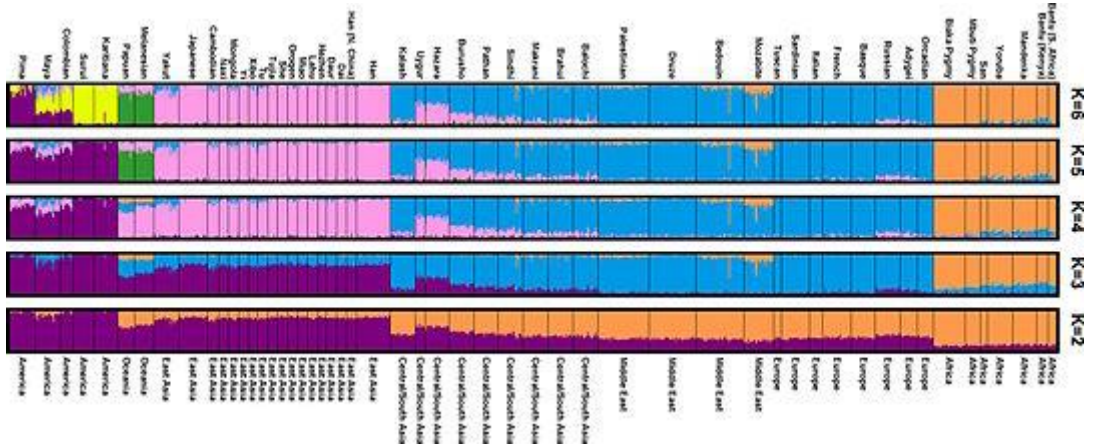
Training Data: $((x_1^{(1)}, x_2^{(1)}, y^{(1)}), (x_1^{(2)}, x_2^{(2)}, y^{(2)}), \ldots (x_1^{(m)}, x_2^{(m)}, y^{(m)}))$

Training Data: $((x_1^{(1)}, x_2^{(1)}), (x_1^{(2)}, x_2^{(2)}), \ldots (x_1^{(m)}, x_2^{(m)}))$

Real World Clustering Examples

- Grouping News Items



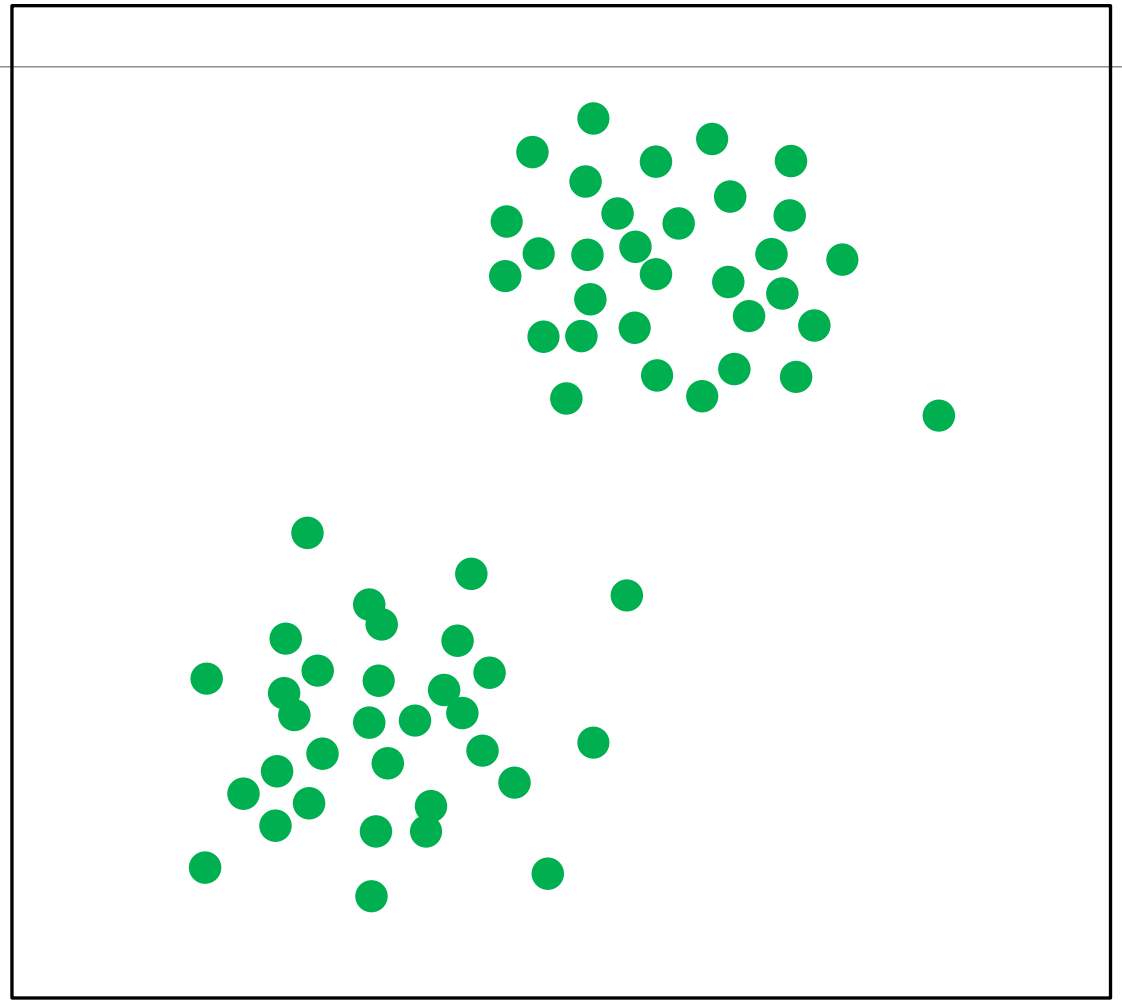


Genetic Clustering
https://en.wikipedia.org/wiki/Human_genetic_clustering

See a comprehensive list of applications that use clustering analysis at:
https://en.wikipedia.org/wiki/Cluster_analysis

K-MEANS CLUSTERING ALGORITHM

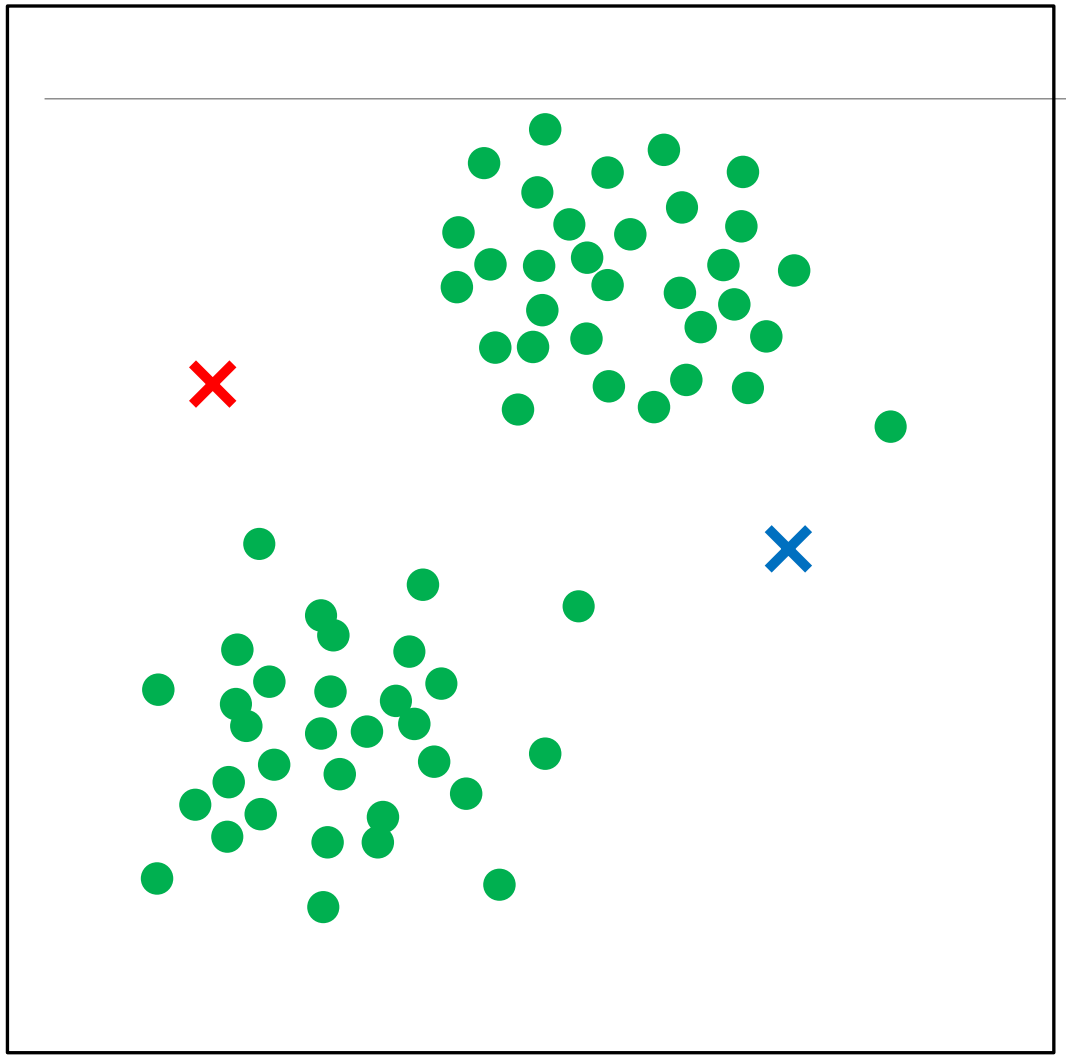
K-means Clustering Algorithm



Term first used by James MacQueen, 1967

Algorithm proposed by Stuart Lloyd, 1957, published in 1982.

K-means clustering algorithm

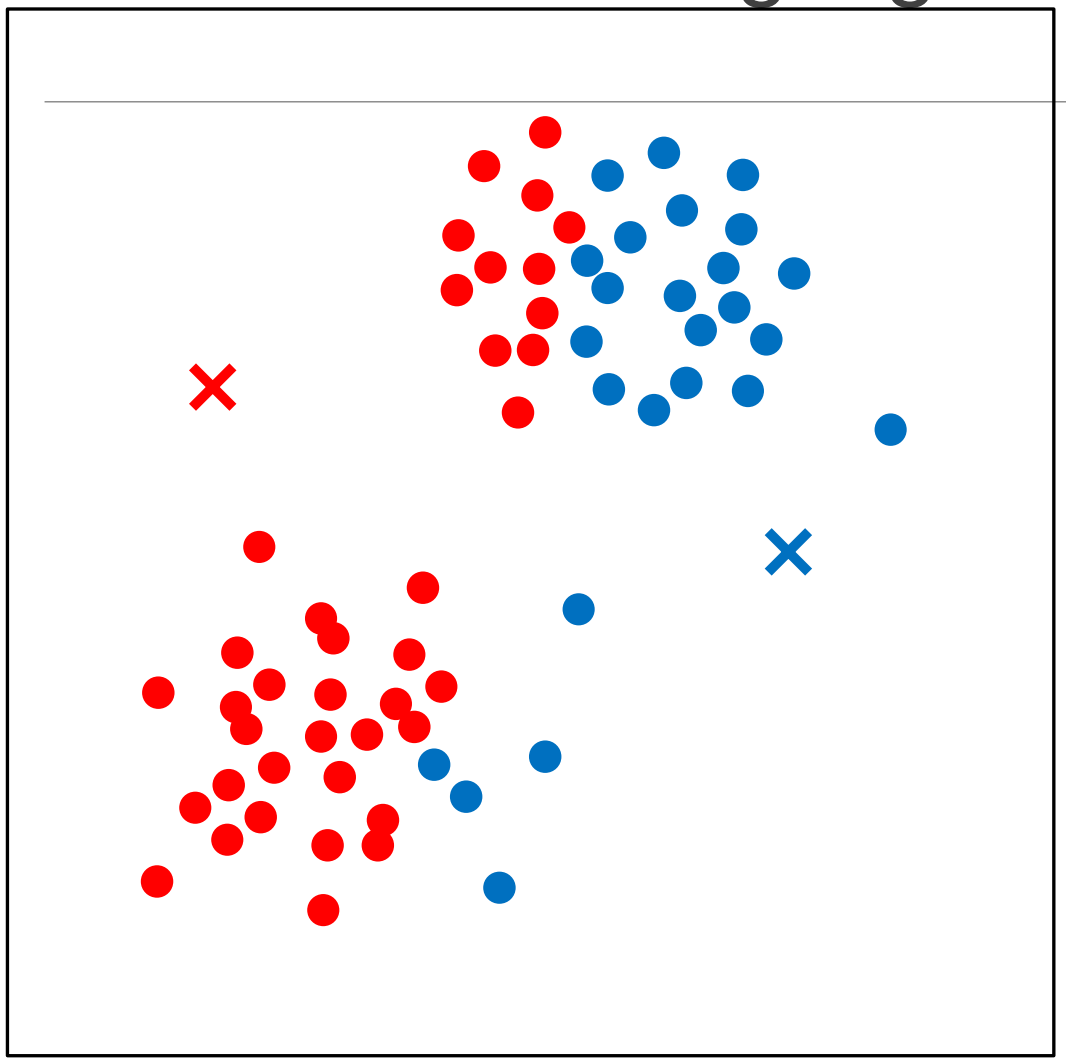


Goal: Assign all data points to 2 clusters

Step 1: Pick 2 *random* initial cluster centroids

Step 2: Paint the data points that are closer to red centroid red, and those closer to blue centroid blue

K-means clustering algorithm



Goal: Assign all data points to 2 clusters

Step 1: Pick 2 *random* initial cluster centroids

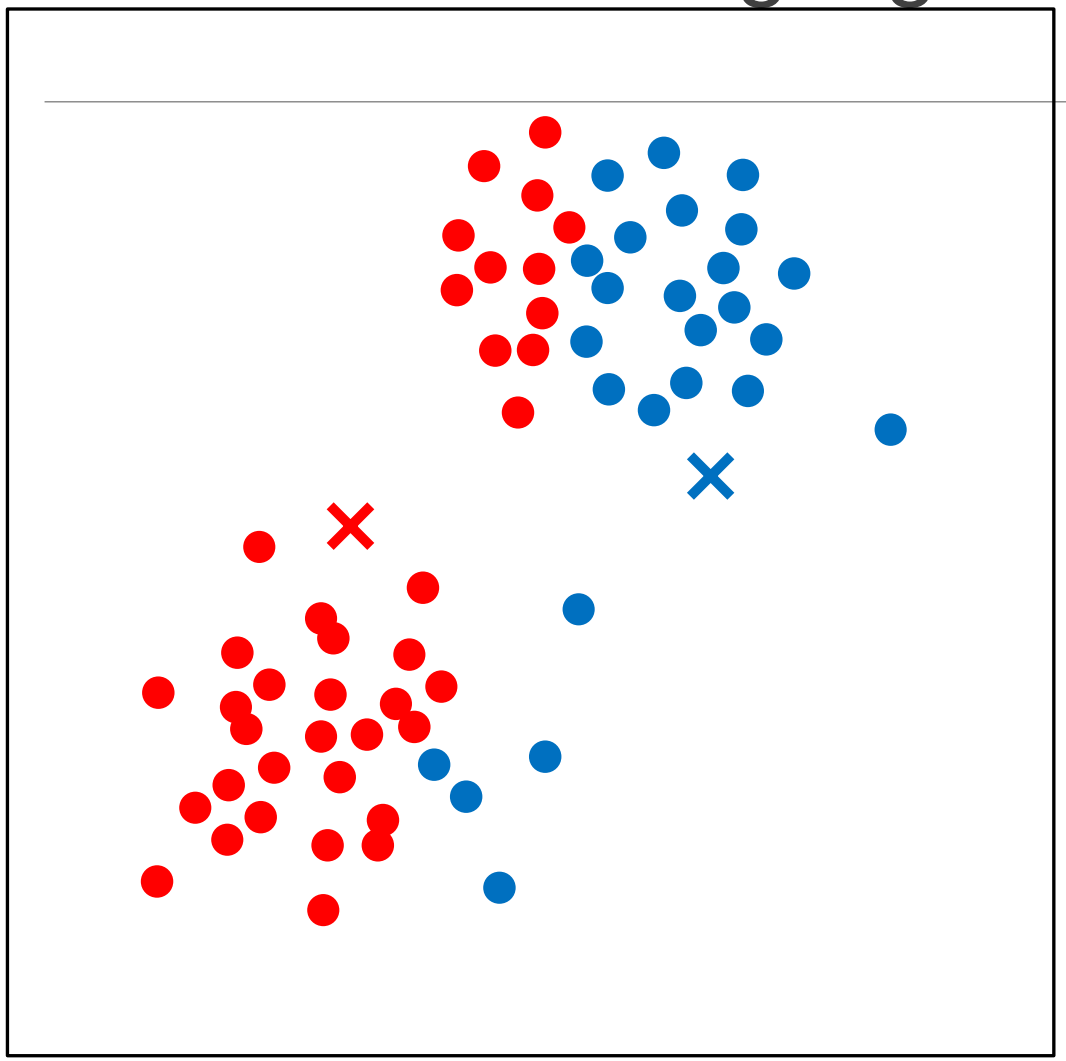
Step 2: Paint the data points that are closer to red centroid red, and those closer to blue centroid blue

Step 3: Update the positions of centroids

Red centroid := average of current red points

Blue centroid := average of current blue points

K-means clustering algorithm



Goal: Assign all data points to 2 clusters

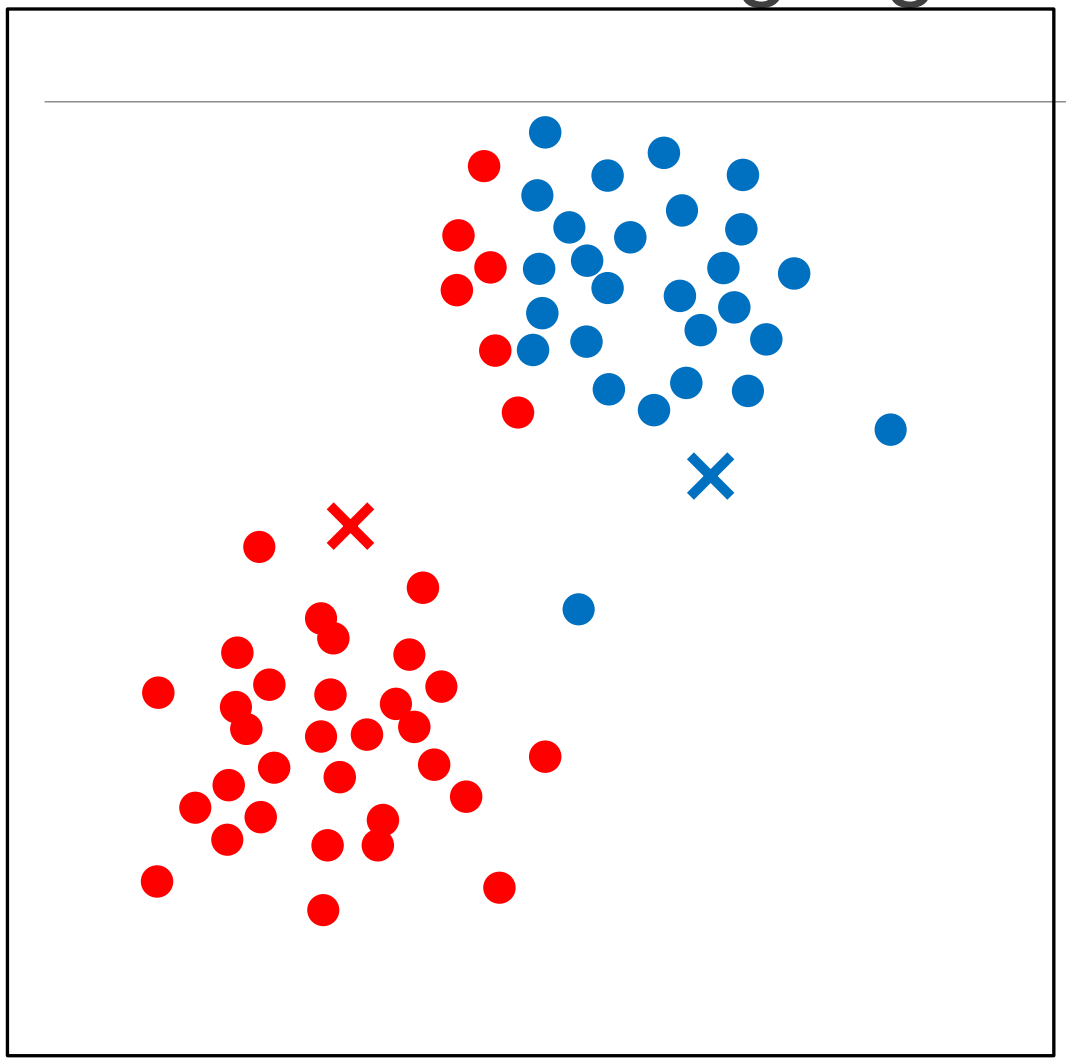
Step 1: Pick 2 *random* initial cluster centroids

Step 2: Paint the data points that are closer to red centroid red, and those closer to blue centroid blue

Step 3: Update the positions of centroids
Red centroid := average of current red points
Blue centroid := average of current blue points

Repeat

K-means clustering algorithm



Goal: Assign all data points to 2 clusters

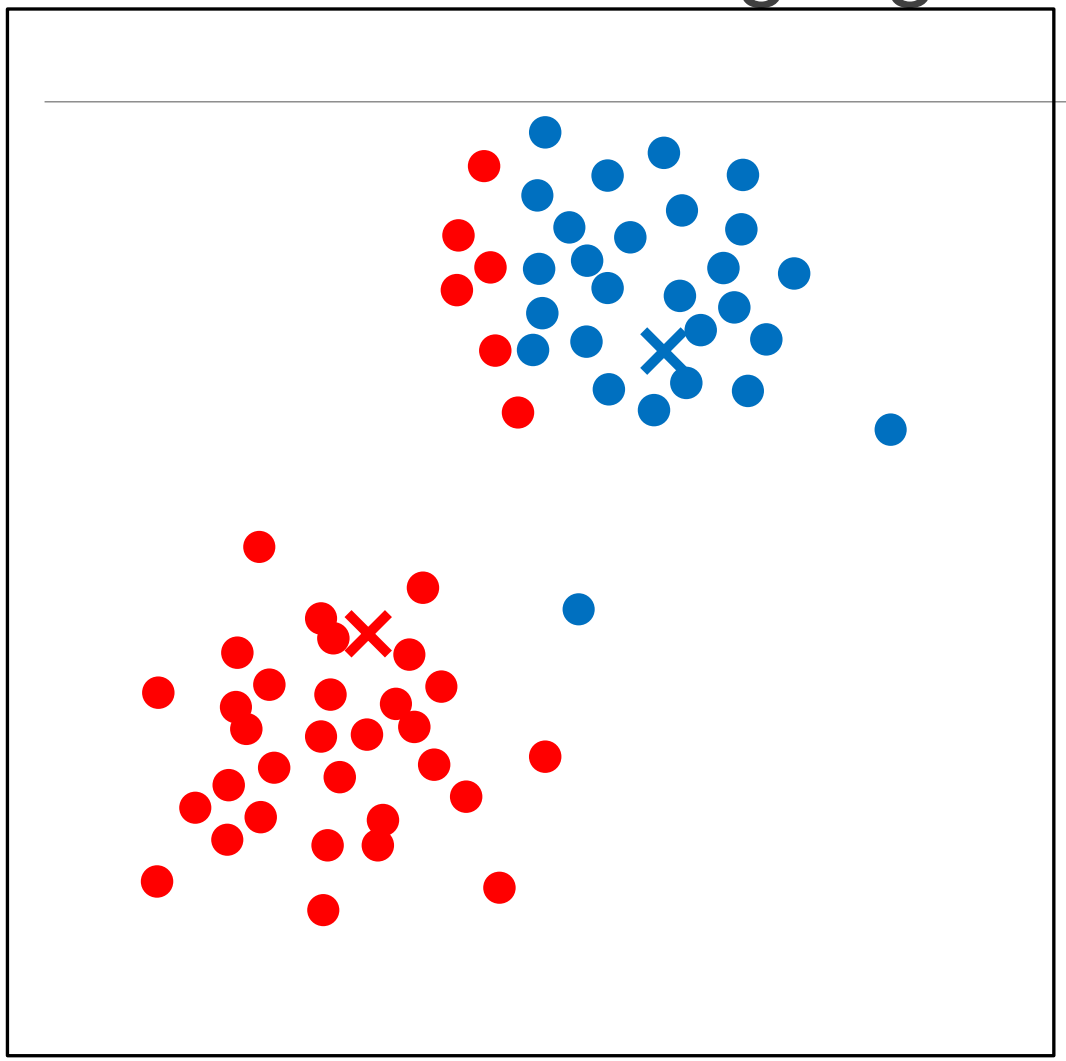
Step 1: Pick 2 *random* initial cluster centroids

Step 2: Paint the data points that are closer to red centroid red, and those closer to blue centroid blue

Step 3: Update the positions of centroids
Red centroid := average of current red points
Blue centroid := average of current blue points

Repeat

K-means clustering algorithm



Goal: Assign all data points to 2 clusters

Step 1: Pick 2 *random* initial cluster centroids

Step 2: Paint the data points that are closer to red centroid red, and those closer to blue centroid blue

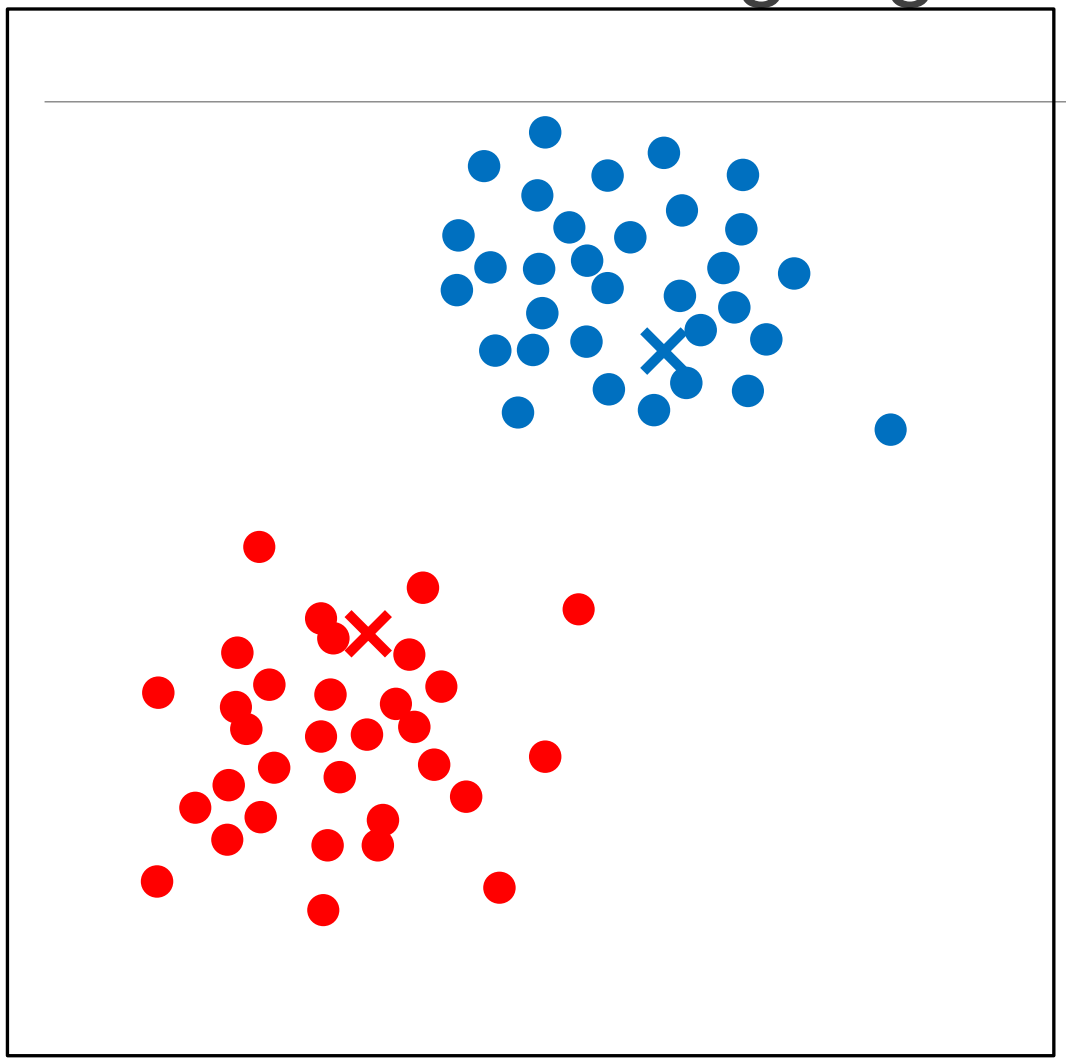
Step 3: Update the positions of centroids

Red centroid := average of current red points

Blue centroid := average of current blue points

Repeat

K-means clustering algorithm



Goal: Assign all data points to 2 clusters

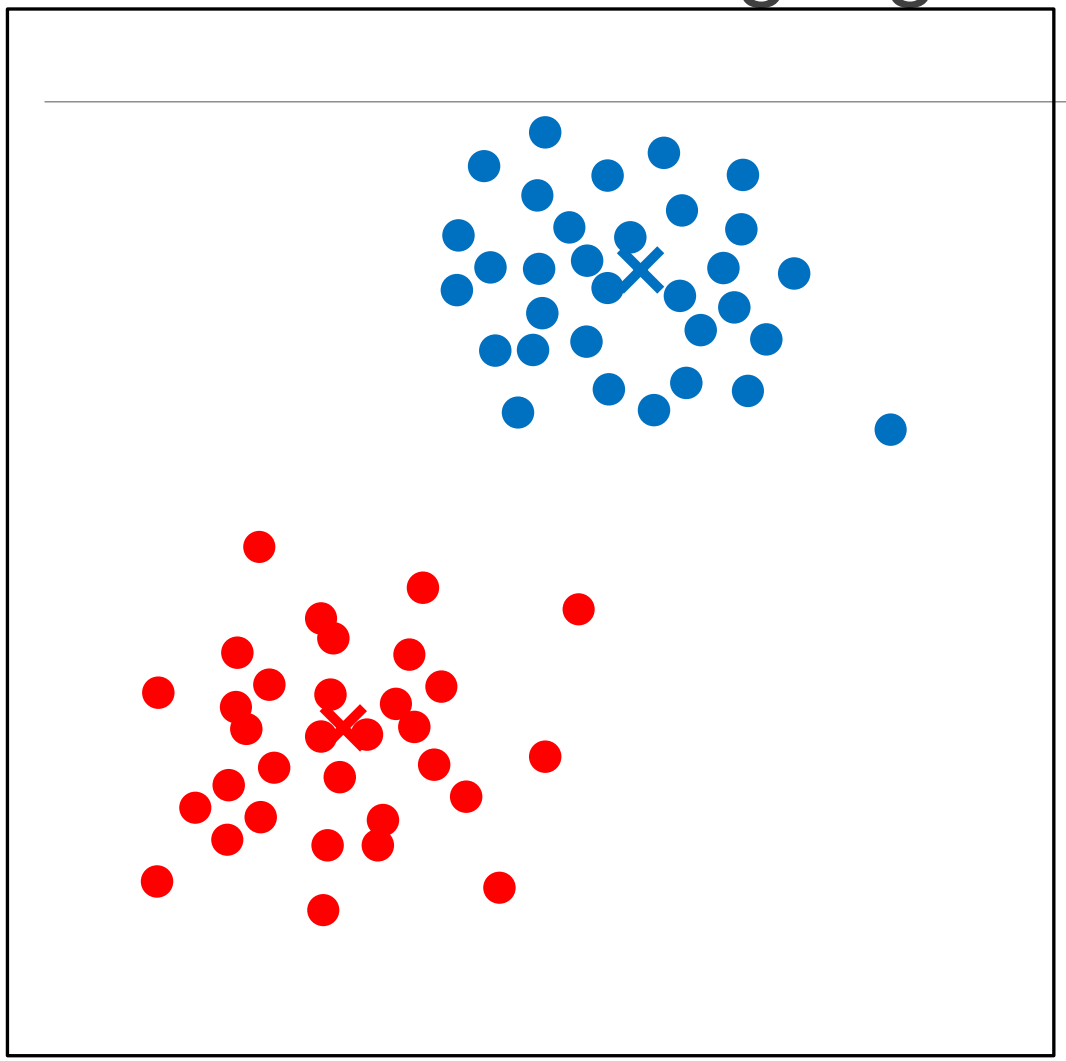
Step 1: Pick 2 *random* initial cluster centroids

Step 2: Paint the data points that are closer to red centroid red, and those closer to blue centroid blue

Step 3: Update the positions of centroids
Red centroid := average of current red points
Blue centroid := average of current blue points

Repeat

K-means clustering algorithm



Goal: Assign all data points to 2 clusters

Step 1: Pick 2 *random* initial cluster centroids

Step 2: Paint the data points that are closer to red centroid red, and those closer to blue centroid blue

Step 3: Update the positions of centroids
Red centroid := average of current red points
Blue centroid := average of current blue points

Repeat
Until no more points need to be repainted, i.e., the centroids no longer change

Clustering is done

K-means formal definition

Given a dataset $\begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(m)} \end{bmatrix}$, $x^{(i)} \in \mathbb{R}^n$, and we want to group the data into k clusters

1. Initialize k **cluster centroids** $\mu_1, \mu_2, \cdots, \mu_k \in \mathbb{R}^n$ randomly

2. Repeat until convergence: {

For $i = 1, \cdots, m$:

$c^{(i)} := \arg\min_j \left\| x^{(i)} - \mu_j \right\|^2$ → Assign $x^{(i)}$ to the closest cluster j

Usually, Euclidean distance is the best option

For $j = 1, \cdots, k$:

$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)}=j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)}=j\}}$ → Update centroid μ_j with mean of all within-cluster data points

}

From a machine learning perspective, K-means minimizes the cost function:

$$J(c, \mu) = \sum_{i=1}^{m} \left\| x^{(i)} - \mu_{c^{(i)}} \right\|^2$$

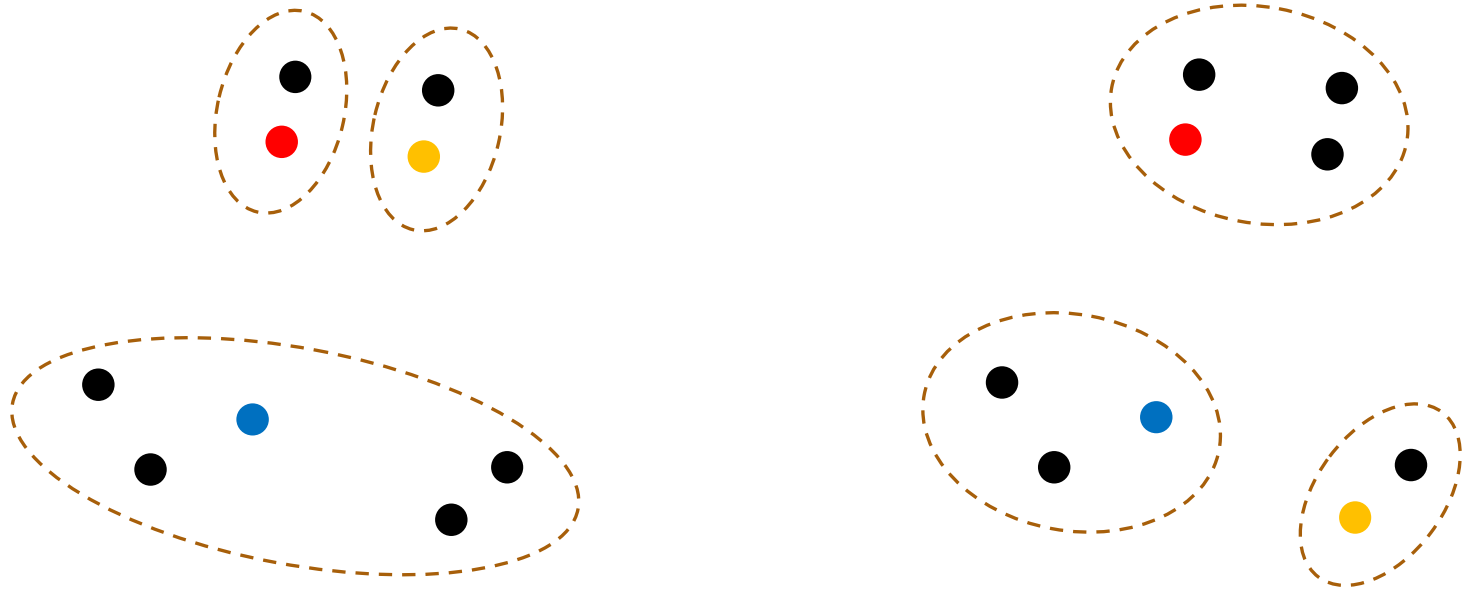
guaranteed to converge

Stable, computationally efficient, differentiable

How to initialize centroids?

Randomly pick k data points as the initial centroids

Sometimes it leads to different clustering results



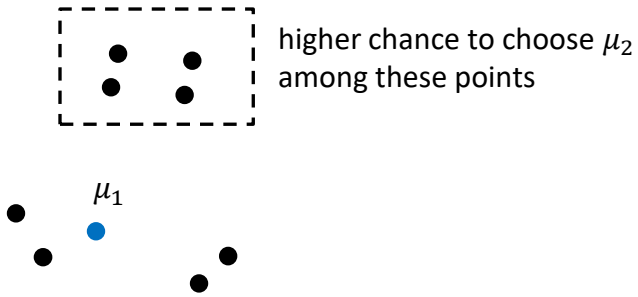
Solutions:

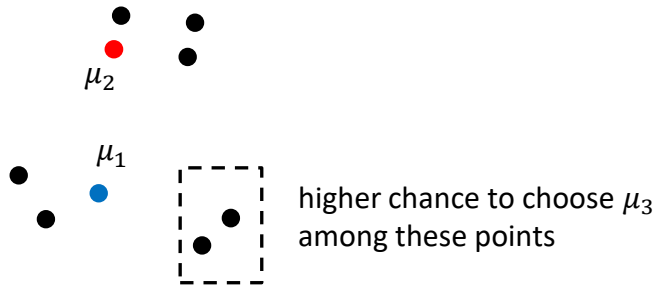
- Run multiple times with different initializations and evaluate
- K-means++ (Arthur & Vassilvitskii, 2007)

Better initialization with *K*-means++

Arthur & Vassilvitskii, 2007

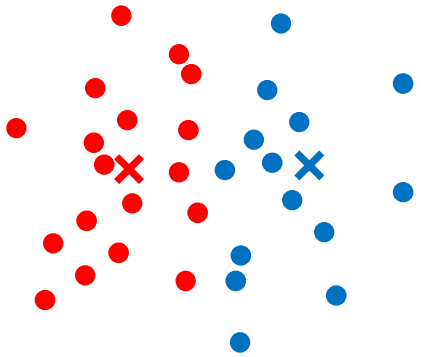
1. Choose one centroid uniformly at random from data points.
2. For each $x^{(i)}$, compute $D(x^{(i)})$, the distance between $x^{(i)}$ and the nearest centroid that has already been chosen.
3. Choose one new data point at random as a new centroid, where the probability of choosing point $x^{(i)}$ is ***proportional*** to $D(x^{(i)})$.
4. Repeat until k centroids have been chosen.



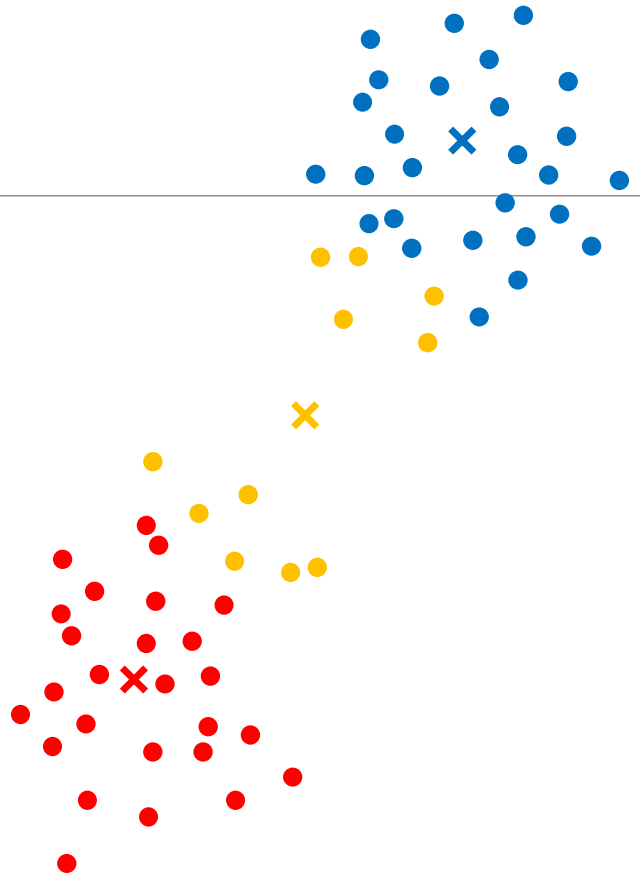


How to choose *K*?

By intuitive observation



Better use one cluster



Better use two clusters

Consider downstream usage: e.g. shirt sizes S,M,L

Is there a systematic evaluation?

Evaluate clustering result

A good clustering algorithm:

- *high* similarity ***within*** cluster
- *low* similarity ***between*** cluster

silhouette coefficient

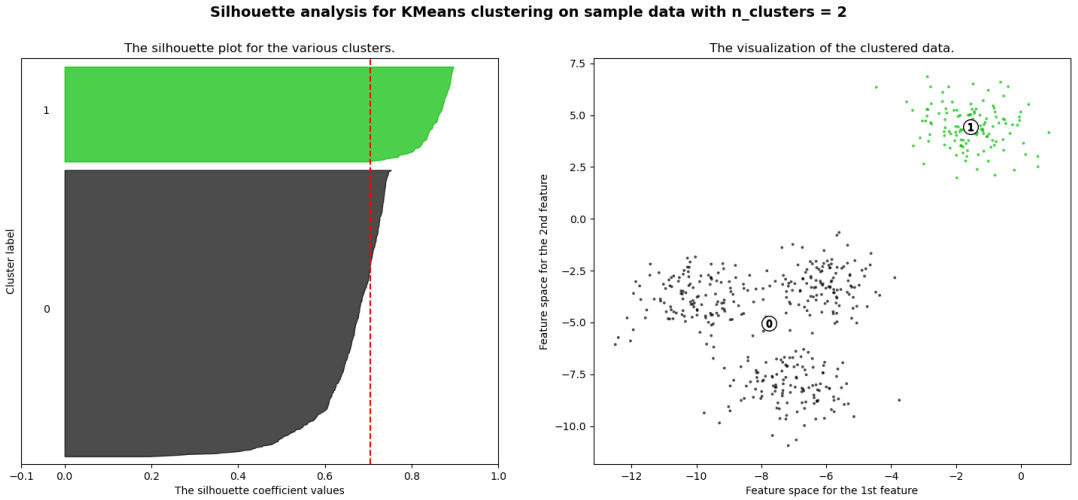
/ˌsilo͞oˈet/: contour, outline

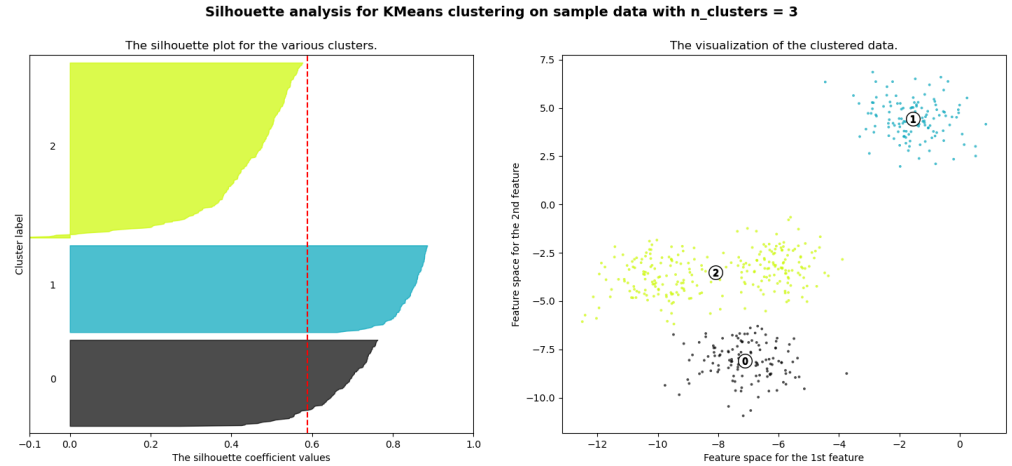
$$s = \frac{b - a}{\max(a, b)} \quad \Rightarrow \quad -1 < s < 1$$

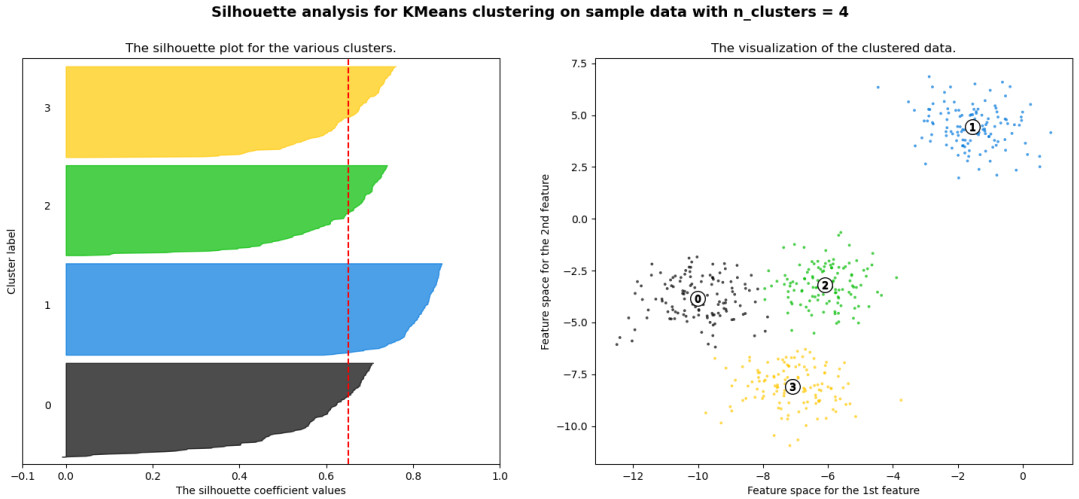
a is the mean distance between a data point and all other points in the same cluster

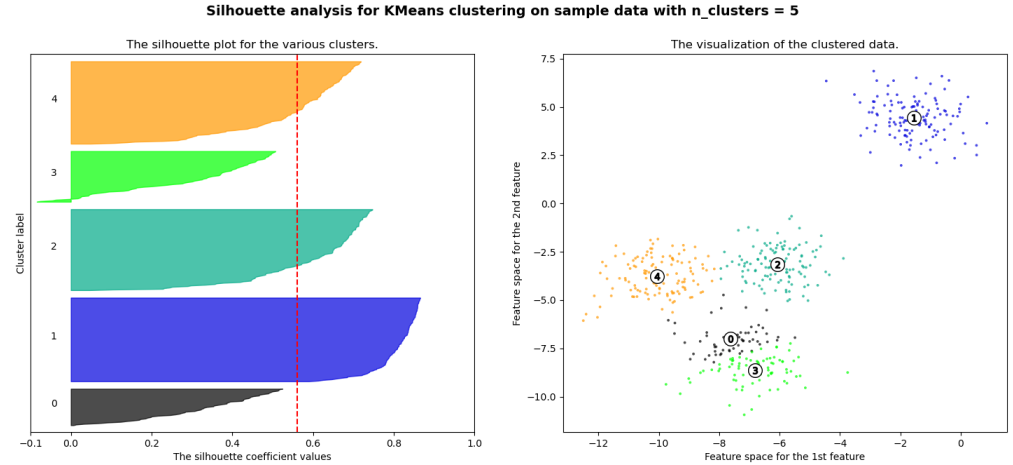
b is the mean distance between a data point and all other points in the next neighbor cluster

Larger s value is better









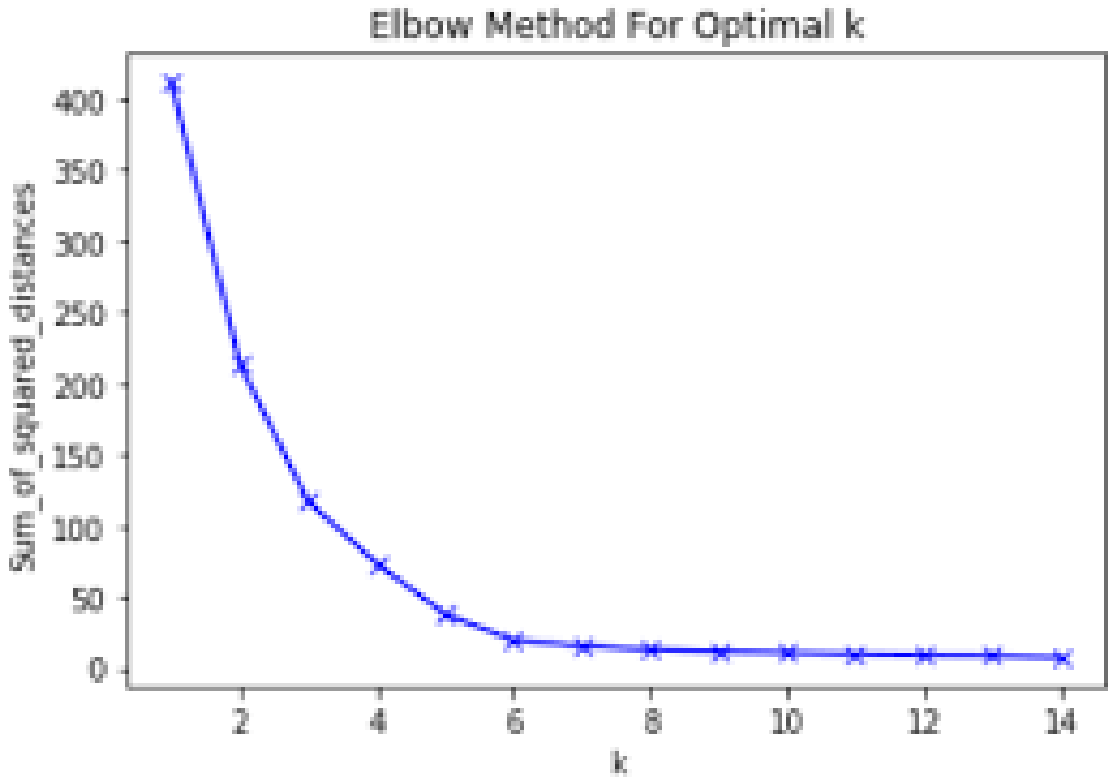
K=2, 4: better, K=3,5: worse. Thickness of the Silhouette plot => Cluster Size

Image source: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Choosing K: Elbow Method

- The basic idea is to run the K-Means clustering algorithm on the dataset for a range of values of K and fo each value of K, calculate the sum of squared distances from each point to its assigned center (inertia).
- By plotting the values of the inertia against the number of clusters, you can observe an "elbow" in the graph, where the rate of decrease in inertia sharply changes.
- The point where this change occurs is considered for choosing K.
- Here's a step-by-step explanation of the Elbow Method:

1. **Run K-Means clustering for different values of K:**
   - Choose a range of values for K (the number of clusters) to explore. Common choices might be from 1 to 10 or more.
   - For each value of K, run the K-Means algorithm on the dataset.
2. **Calculate the inertia for each value of K:**
   - For each value of K, calculate the inertia.
3. **Plot the results:**
   - Create a plot with the number of clusters (K) on the x-axis and the corresponding inertia on the y-axis.
   - The plot will typically show a decreasing inertia as the number of clusters increases.
4. **Identify the "elbow" in the plot:**
   - Look for the point on the plot where the decrease in inertia slows down and forms an "elbow" shape.
   - The "elbow" is considered the optimal number of clusters.
5. **Choose K:**
   - The optimal number of clusters is often where the inertia starts to decrease at a slower rate.
   - It represents a balance between fitting the data well (low inertia) and avoiding overfitting (too many clusters).



K=6

K-means: pros and cons

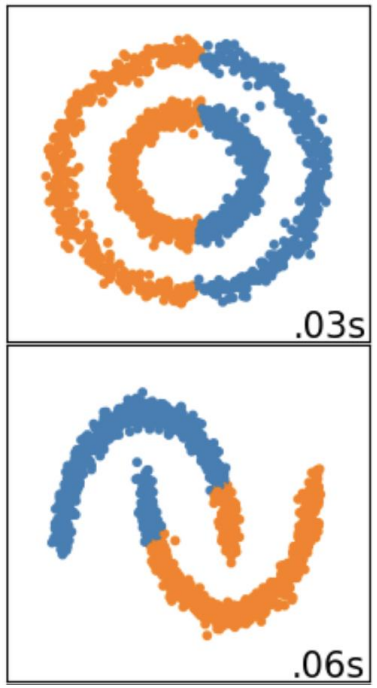
Pros:

- Easy to implement
- Scales to very large datasets

Cons:

- Difficult to choose *K*.
- Only works on spherical, convex clusters.

Where K-means does not work well



HIERARCHICAL CLUSTERING

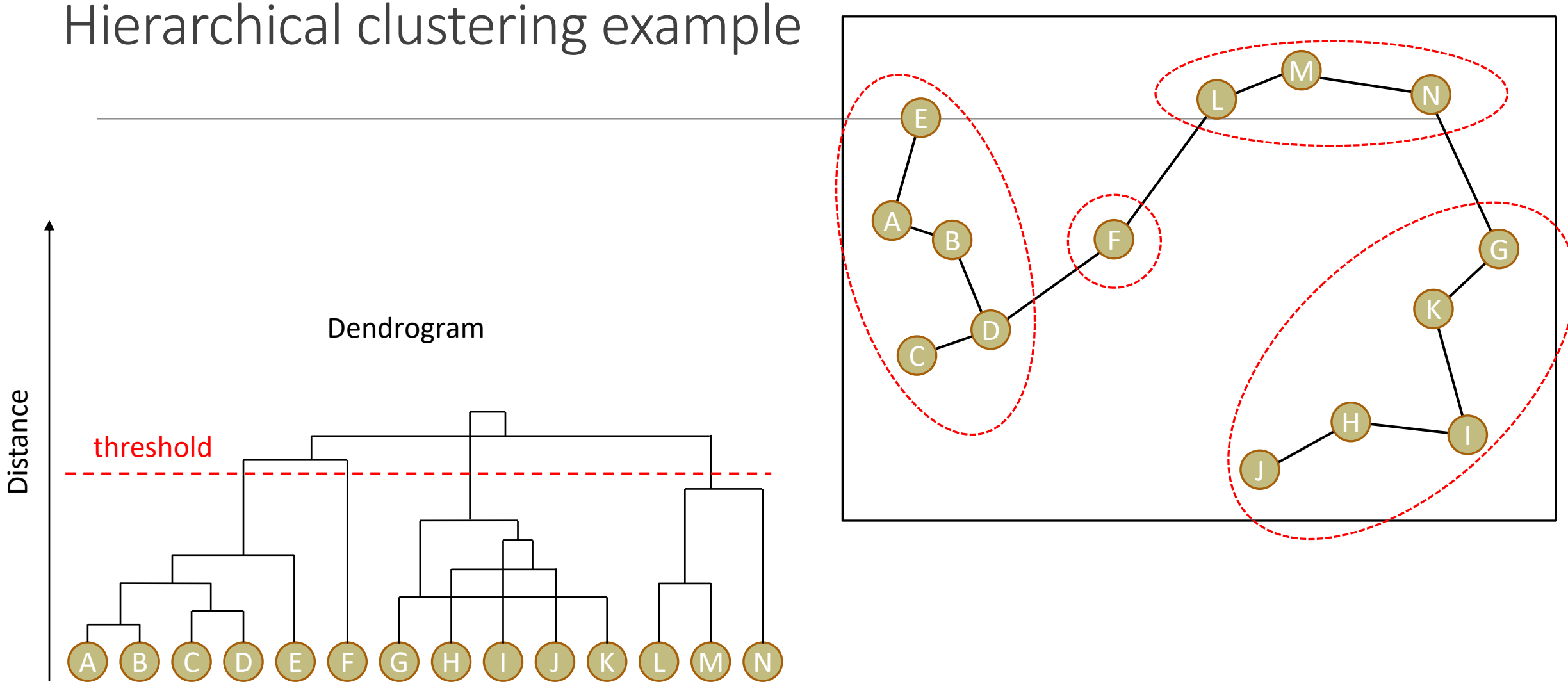
Hierarchical clustering

Idea: make sure *nearby* data points end up in the same cluster

- Initialize a collection C of m singleton clusters, i.e., $c^{(i)} = \{x^{(i)}\}$
- Repeat until only one cluster is left:
  - Find a pair of clusters that is closest: $\arg\min_{i,j} D(c^{(i)}, c^{(j)})$
  - Merge the two clusters $c^{(i)}, c^{(j)}$ into a new cluster $c^{(i\&j)}$
  - Remove $c^{(i)}, c^{(j)}$ from the collection C, and add $c^{(i\&j)}$
- Produce a ***dendrogram***: a hierarchical ***tree*** of clusters

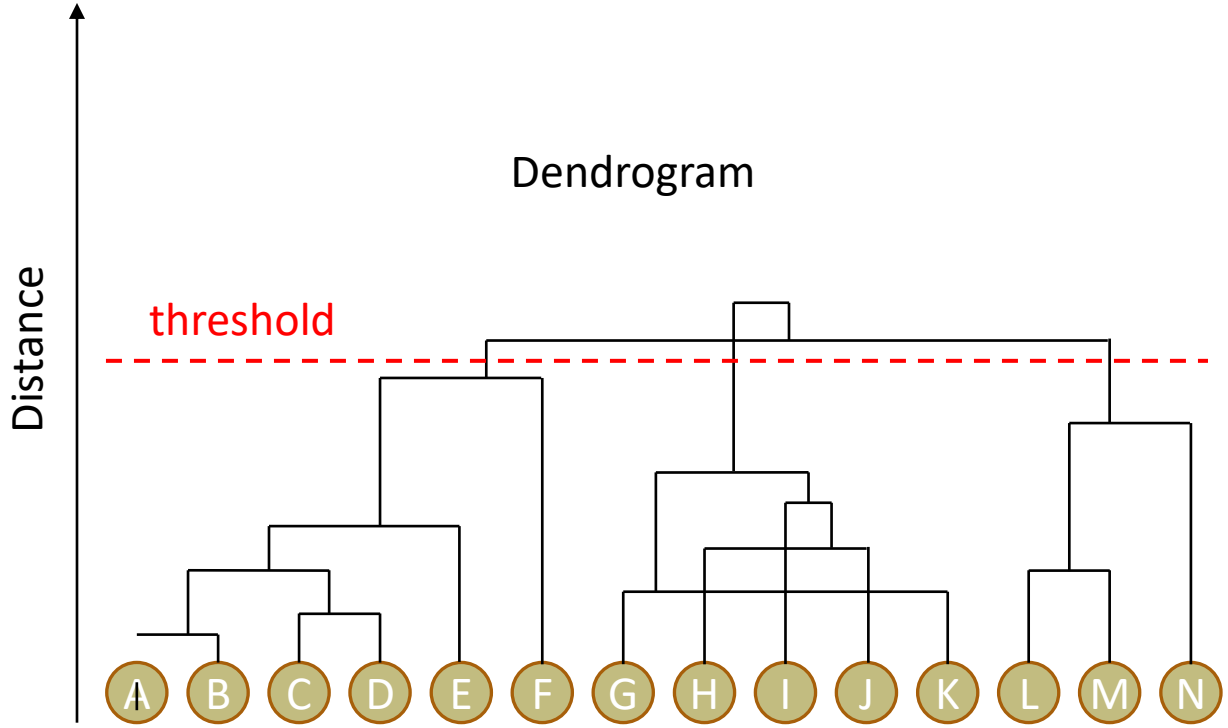
Need to define ***distance***

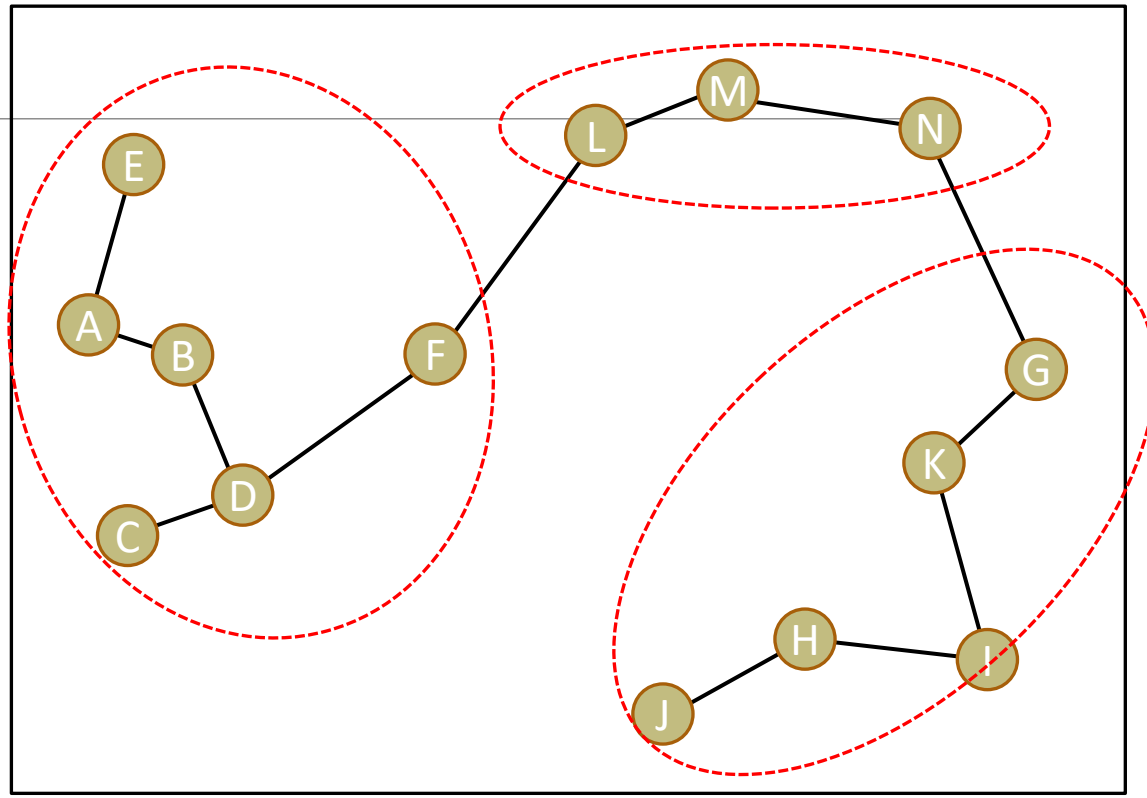
Hierarchical clustering example



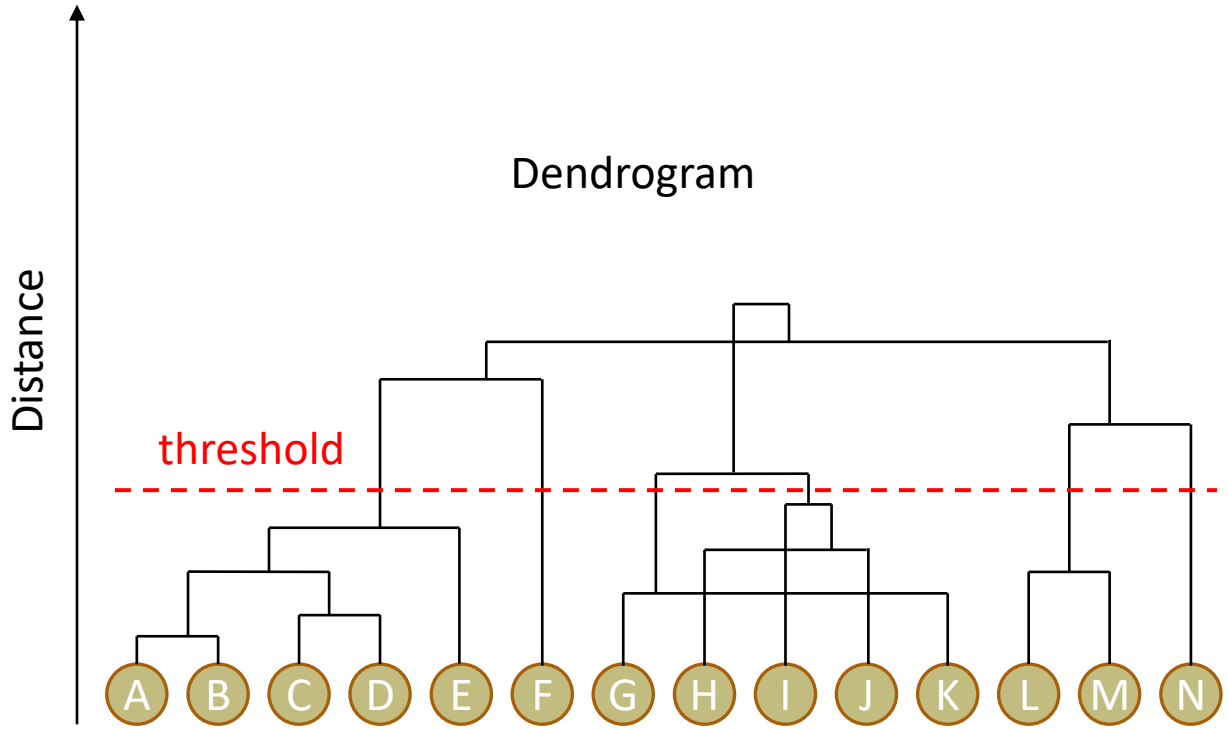
Hierarchical clustering example

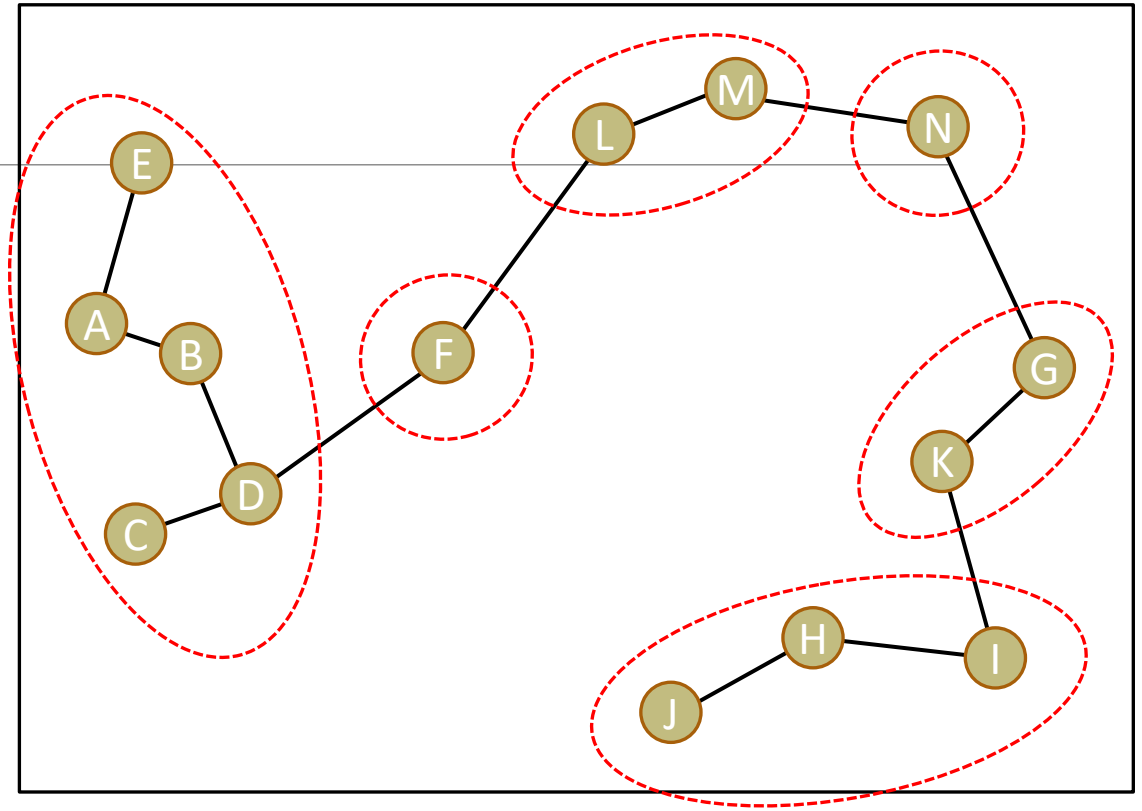
In this case, distance between clusters is defined by the **closest** pair





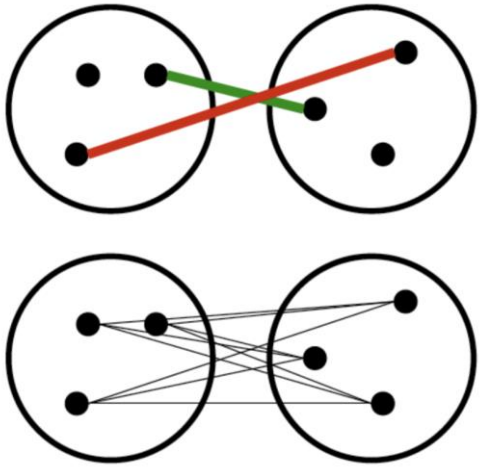
Hierarchical clustering example



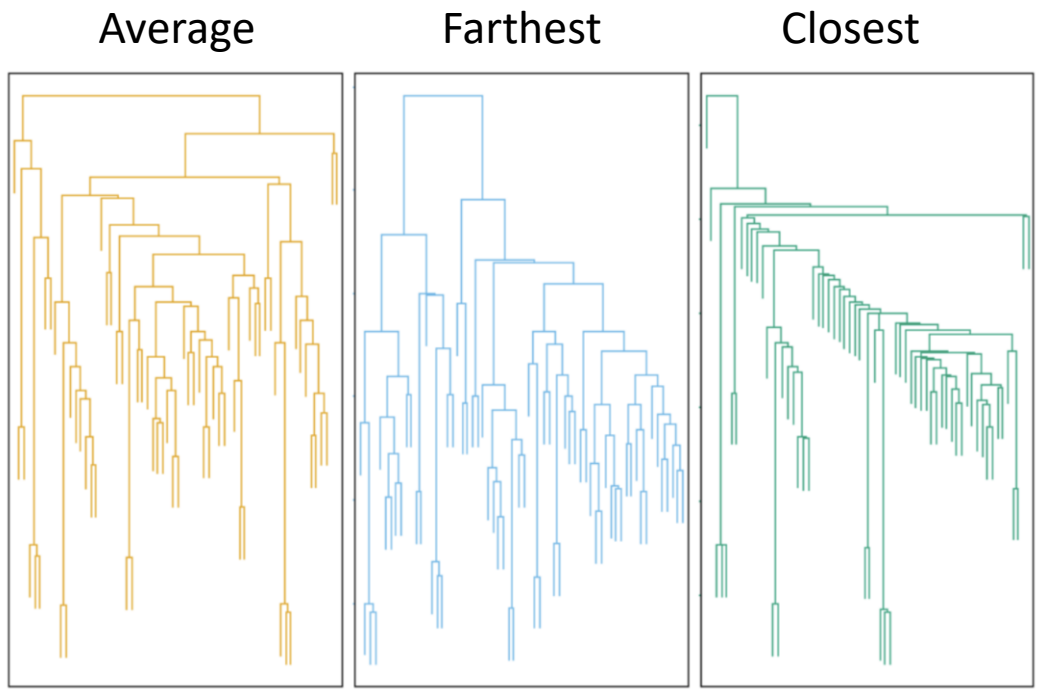


Hierarchical clustering – distance options

Distance options:
closest pair, farthest pair, or average of all pairs



Distance option influences the clustering result



Hierarchical Clustering – Distance Options

Distance Measure	Description	Advantages	Disadvantages
Euclidean Distance	Calculates the straight-line distance between two points	Simple and easy to understand	Sensitive to outliers and can be affected by features with different scales
Manhattan Distance (L1 Distance)	Calculates the sum of the absolute differences between corresponding coordinates of two points	Robust to outliers	Less effective in identifying clusters with different shapes
Minkowski Distance	A generalization of Euclidean and Manhattan distances, where the p-norm is used	Provides a flexible way to balance sensitivity to outliers and different feature scales	More computationally expensive than Euclidean or Manhattan distances
Chebyshev Distance	Calculates the maximum difference between corresponding coordinates of two points	Robust to outliers	Less effective in identifying clusters with different sizes
Correlation Distance	Measures the correlation between two vectors	Suitable for data with non-linear relationships	Sensitive to outliers and can be affected by features with different scales
Mahalanobis Distance	Considers the covariance between features to calculate the distance between two points	Adapts to different feature scales and accounts for correlations	Computationally expensive and requires the inversion of the covariance matrix
Average Distance	Calculates the average distance between all pairs of data points in one cluster and all pairs of data points in the other cluster	Robust to outliers, balances sensitivity to different feature scales, and adapts to different cluster shapes	Computationally expensive, susceptible to noise, and may not capture hierarchical structure well
Farthest Pair Linkage	Calculates the maximum distance between any pair of data points, one from each cluster	Robust to outliers, prevents merging of clusters with different densities, and captures hierarchical structure well	Sensitive to noise, may produce tight clusters, and computationally expensive
Ward's Distance	Minimizes the increase in within-cluster variance when merging clusters	Identifies clusters that are compact and have similar shapes	Computationally expensive for large datasets

Hierarchical clustering: pros and cons

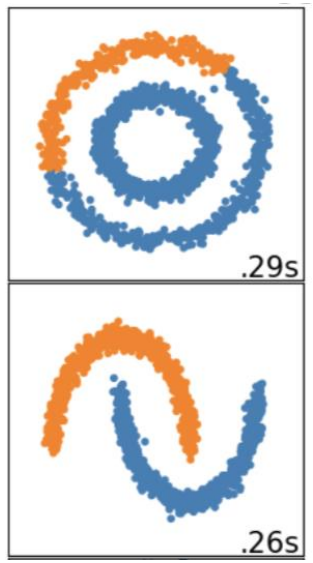
Pros

- Hierarchical structure is more informative than flat clusters (*K*-means)
- Easier to decide the number of clusters

Cons:

- Slow to compute. Time complexity $O(n^3)$.
- Sensitive to outliers, because it tries to connect all data points.

Hierarchical clustering with Ward's method



DBSAN

Density-based clustering

Idea: Clustering based on density (local clustering criterion), e.g., number of *densely* connected points.

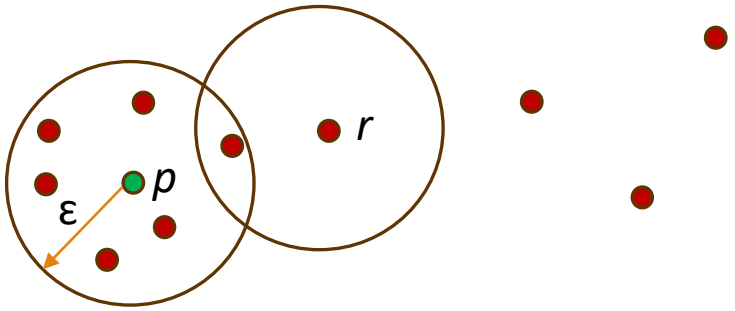
Most well-known algorithm: Density-based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN

Ester, et al (1996)

- DBSCAN classifies all points as ***core*** *points*, (*density*-)***reachable*** *points* and ***outliers*** (or ***noise*** points):

- A point ***p*** is a core point if at least `minPts` points are within distance ***ε*** (***ε*** is the maximum radius)
- A point ***q*** is directly reachable from ***p*** if point ***q*** is within distance ***ε*** from point ***p*** and ***p*** must be a core point.
- A point ***r*** is reachable from ***p*** if there is a path $\boldsymbol{p_1, \ldots, p_n}$ with $p_1 = p$ and $p_n = r$, where each $\boldsymbol{p_{i+1}}$ is directly reachable from $\boldsymbol{p_i}$.
- All points not reachable from any other point are outliers.

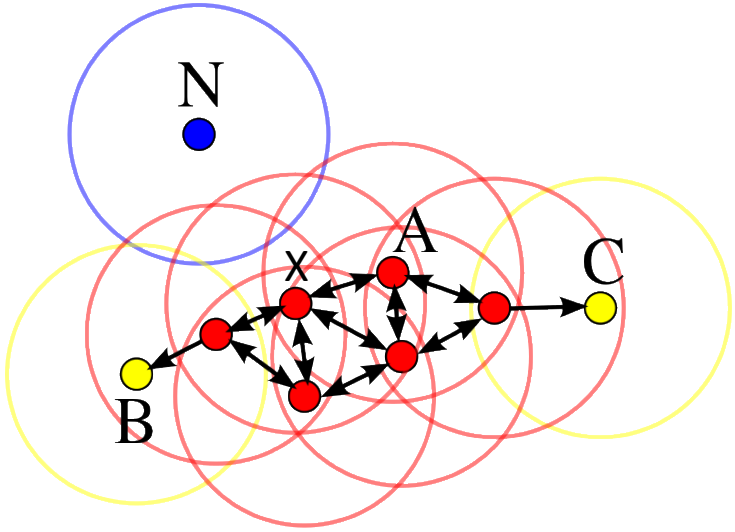


minPts=5

p is a core point, all other points within the circle on the left are directly reachable from *p*. Point *r* is reachable from *p*. All other points are outliers

DBSCAN procedure

1. Find the points in the ε (epsilon) neighborhood of every point, and identify the *core* points (>= minPts)
2. Find the ***directly reachable components*** of *core* points.
3. Assign each non-directly reachable point to a nearby cluster if it is reachable, otherwise assign it to noise.



minPts = 4
X is the only core point.

All other red points are directly reachable points. Because they are reachable from one another, they form a single cluster.

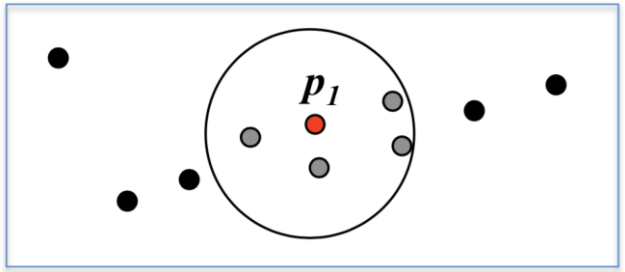
Points **B** and **C** are not *core* points and are not directly reachable from X, but are reachable from **X** and are included in the cluster as well.

Point **N** is a *outlier* point, because it is neither a core nor reachable.

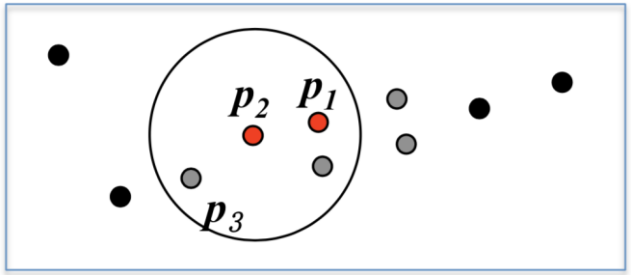
Source: https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

DBSCAN breakdown

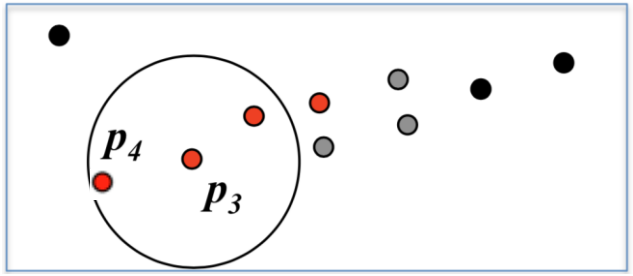
minPts = 3



Start from p_1
p_1 is a *core* point. Create a new **cluster C1**
There 4 neighbor points and they all become candidates to expand

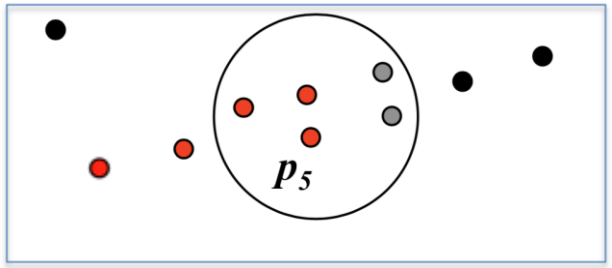


Add p_2 to C1
Found a new candidate p_3

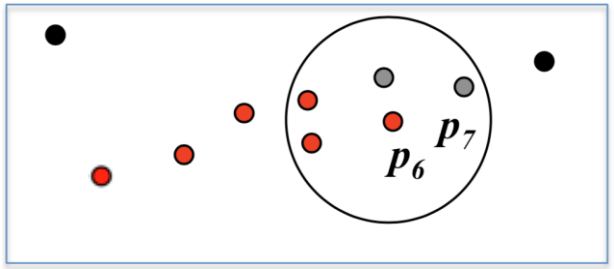


Add p_3 to C1
Found a new neighbor p_4, which also can be added
Because $it\ is\ at\ a\ distance < \epsilon$ from p_3

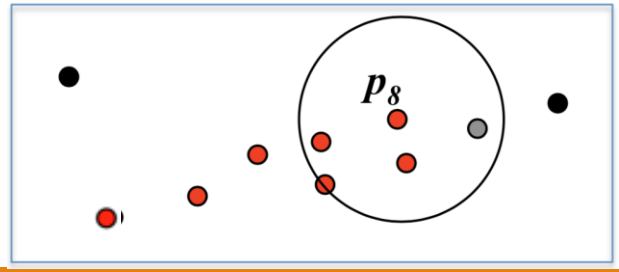
DBSCAN breakdown (cont.)



Add p_5 to C1
No new candidate is found

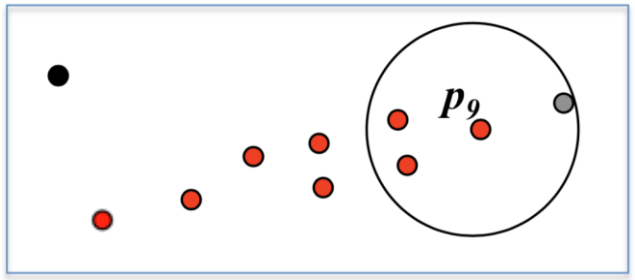


Add p_6 to C1
A new candidate p_7 is found

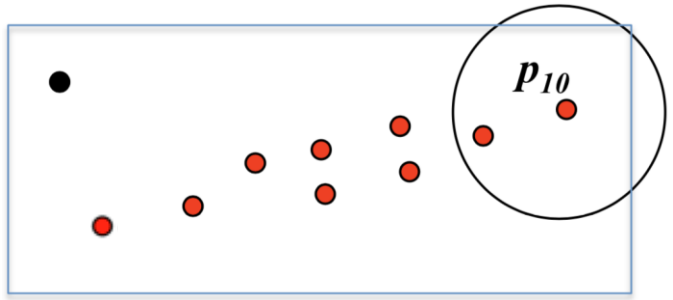


Add p_8 to C1

DBSCAN breakdown (cont.)

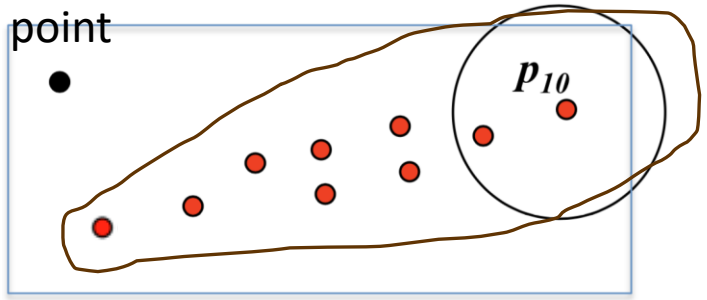


Add p_9 to C1
A new candidate p_{10} is found



Add p_{10} to C1
No more candidates

Noise point

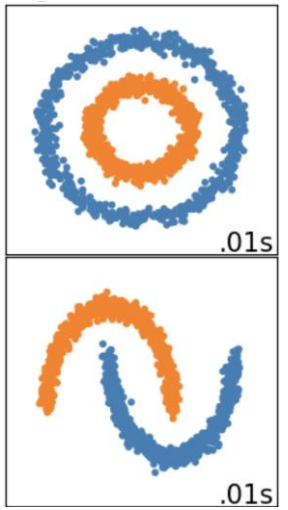


Identify cluster and mark noise points with respect to this cluster

DBSCAN pros and cons

Pros:

- Pretty fast. Time complexity is $O(n \log n)$ when optimized.
- Can find arbitrarily shaped clusters
- Robust to outliers (recognized as noise points)



Cons:

- Cannot work well if density varies in different regions of data
- Choosing a proper distance threshold ε can be difficult

Outline

K-means

Hierarchical clustering

Density-based clustering

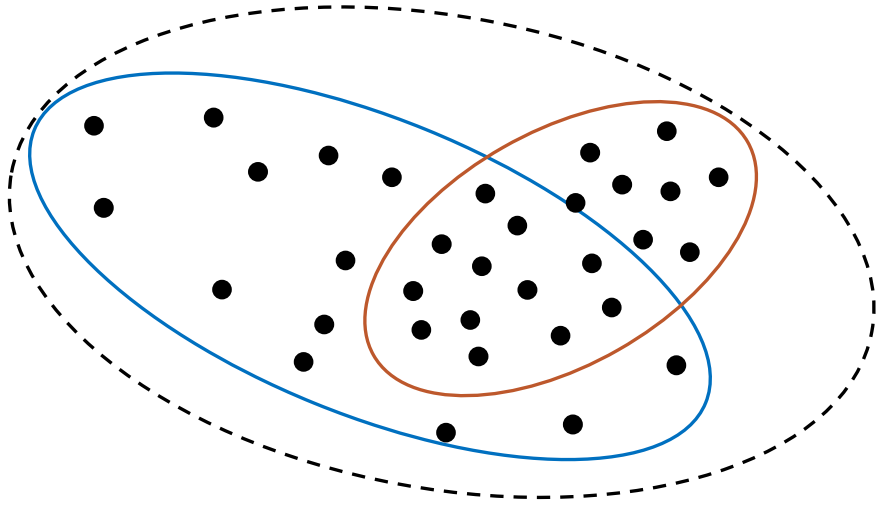
Mixture Gaussian Model and EM method (optional)

- Gaussian distribution
- Mixture of gaussians
- EM (Expectation-Maximization) method

Motivation

K-means makes ***hard*** assignments to data points: $x^{(i)}$ must belong to one of the clusters $1,2,\cdots,K$

Sometimes, one data point can belong to multiple clusters

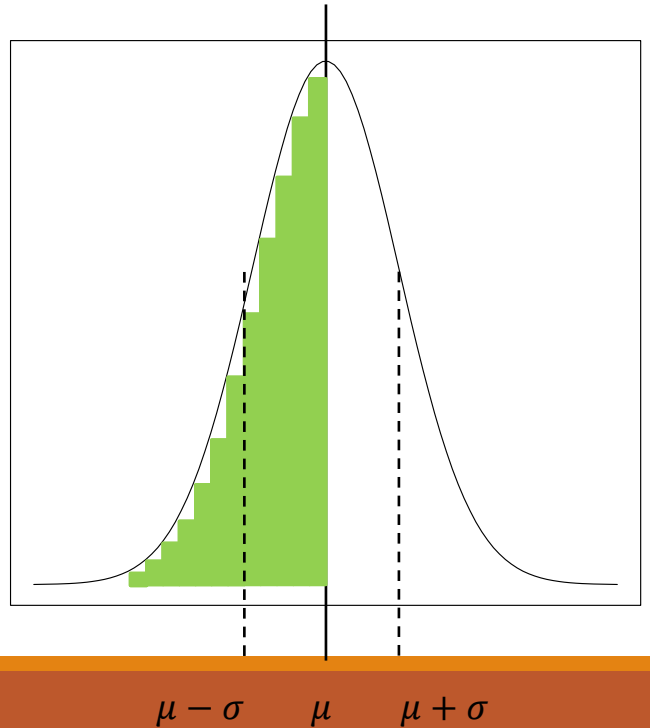


- Clusters may overlap
- Hard assignment may be simplistic
- Need a ***soft*** assignment:
  data points belong to clusters with different ***probabilities***

Gaussian (Normal) distribution

1-D (univariate) Gaussian $\mathcal{N}(\mu, \sigma), \mu \text{ is the mean and } \sigma \text{ is the standard deviation}$

Probability density function (PDF): $p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ μ: mean σ: standard deviation



$$P(x < \mu) = \int_{-\infty}^{\mu} p(x)dx = 0.5 = P(x > \mu)$$

$$P(x < \mu - \sigma) = \int_{-\infty}^{\mu} p(x)dx \approx 0.157 = P(x > \mu + \sigma)$$

Gaussian distribution Use is Ubiquitous

- **Height of Individuals**
- **Test Scores**
- **Measurement Errors**
- **Financial Returns**
- **Natural Processes**
- **Blood Pressure**
- **Body Temperatures**
- **Noise**
- **Illuminance**
- **IQ Scores**
- **Population Characteristics**
- **Logarithms of financial returns/PE ratios etc.**

Gaussian model

μ and σ fully define a gaussian distribution

Use them as parameter $\theta = (\mu, \sigma)$ to define the model:
suppose each data point is randomly *drawn* from the distribution

μ, σ are **unknown**, but they can be learned (estimated) from **data**

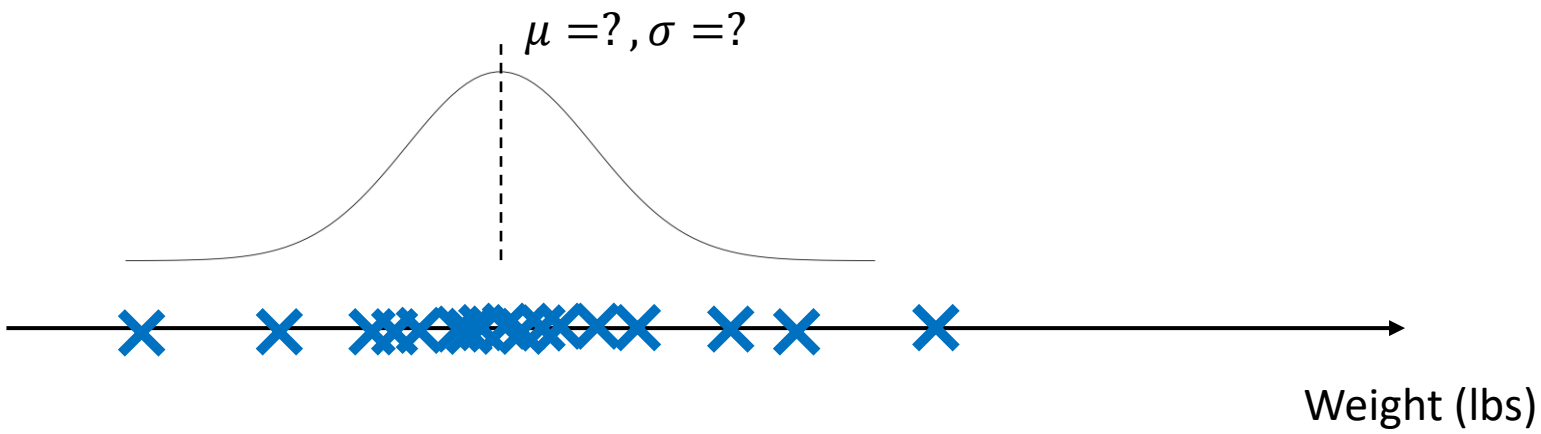
Job: find the parameters that best fit the data

What is "best fit"? → **Maximum Likelihood Estimation** (**MLE**)

Gaussian model example

Data: weight of Salmon fish. Assumption: The weight is from a Gaussian distribution

Task: to estimate the μ, σ of Salman



Maximum Likelihood Estimation (MLE)

Given m data points $X = \{x^{(1)}, \cdots, x^{(m)}\}$ Fit a Gaussian model $\mathcal{N}(\mu, \sigma)$, $\theta = (\mu, \sigma)$

PDF at $x^{(i)}$: $p(x^{(i)}|\theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x^{(i)}-\mu)^2}{2\sigma^2}}$ ⇨ How likely it is to observe $x^{(i)}$ given θ

Assuming all data points are independent, then the likelihood of observing the whole dataset:

$$p(X|\theta) = \prod_{i=1}^{m} p(x^{(i)}|\theta) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x^{(i)}-\mu)^2}{2\sigma^2}}$$

A good estimation of θ needs to maximize $p(X|\theta)$, the **likelihood** of data given the parameters

Maximum Likelihood Estimation (MLE) (cont.)

Likelihood function:

$$\mathcal{L}(\theta) = p(X|\theta) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x^{(i)}-\mu)^2}{2\sigma^2}}$$

It is easier to work with **log-likelihood**:

$$\mathcal{LL}(\theta) = \log\big(\mathcal{L}(\theta)\big) = -\frac{m\log(2\pi)}{2} - m\log(\sigma) - \sum_{i=1}^{m} \frac{(x^{(i)}-\mu)^2}{2\sigma^2}$$

Goal: find the $\theta = (\mu, \sigma)$ that maximizes $\mathcal{LL}(\theta)$

Maximum Likelihood Estimation (MLE) (cont.)

$$\mathcal{LL}(\theta) = \log\big(\mathcal{L}(\theta)\big) = -\frac{m\log(2\pi)}{2} - m\log(\sigma) - \sum_{i=1}^{m}\frac{(x^{(i)}-\mu)^2}{2\sigma^2}$$

Take the derivative of $\mathcal{LL}(\theta)$ w.r.t μ and σ

$$\frac{\partial\mathcal{LL}(\theta)}{\partial\mu} = -\frac{1}{\sigma^2}\sum_{i=1}^{m}(x^{(i)}-\mu) = -\frac{1}{\sigma^2}\left[\sum_{i=1}^{m}x^{(i)} - m\mu\right]$$

$$\frac{\partial\mathcal{LL}(\theta)}{\partial\sigma} = -\frac{m}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{m}(x^{(i)}-\mu)^2$$

$\mathcal{LL}(\theta)$ has extreme values when $\frac{\partial\mathcal{LL}(\theta)}{\partial\mu} = 0$ and $\frac{\partial\mathcal{LL}(\theta)}{\partial\sigma} = 0$

$$\Rightarrow \quad \mu = \frac{1}{m}\sum_{i=1}^{m}x^{(i)} = \bar{X} \qquad \sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)^2}$$

Mean of data
(sample mean)

Stnd deviation of data
(sample variance)

When μ is estimated by $\bar{X}$,

$\sigma = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(x^{(i)}-\bar{X})^2} = \sqrt{Var(X)}$

in order to get an unbiased estimate (Bessel's correction)

These are the reasonable estimates of μ and σ from the data

Take Away

- MLE of Gaussian Distribution parameters is given by its sample mean and sample standard distribution

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} = \bar{X}$$

$$\hat{\sigma} = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(x^{(i)} - \bar{X})^2} = \sqrt{Var(X)}$$

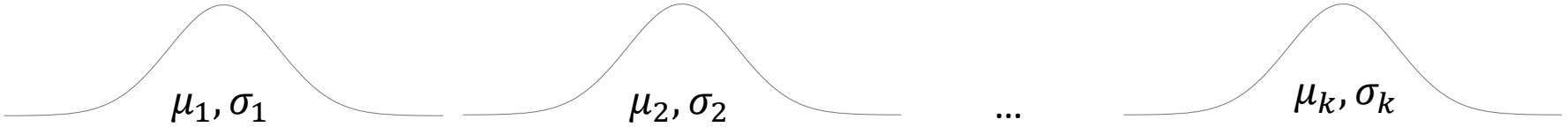
Mixture of Gaussians

Previous example has the assumption that data are drawn from ***one*** Gaussian distribution $\mathcal{N}(\mu, \sigma)$

What if there are ***multiple*** Gaussian distributions: $\mathcal{N}(\mu_1, \sigma_1)$, $\mathcal{N}(\mu_2, \sigma_2)$, …, $\mathcal{N}(\mu_k, \sigma_k)$

How do we generate the data?

Step 1: Draw from k distributions with probabilities $Q_1, Q_2, \cdots, Q_k$

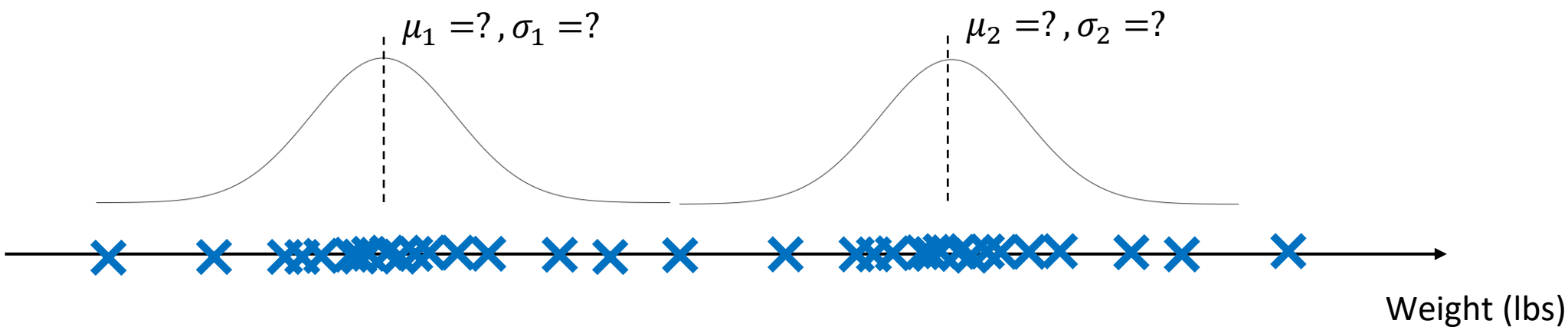


Step 2: Suppose distribution j is chosen, draw a data point from $\mathcal{N}(\mu_j, \sigma_j)$

$$p\left(x^{(i)} \middle| \mu_j, \sigma_j\right) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(x^{(i)}-\mu_j)^2}{2{\sigma_j}^2}}$$

Example of 2 Gaussians

Weights of two kinds of fish: Salmon & Tuna fish



How a data point is generated

A data point $x^{(i)}$ is generated according to the following process:

First, select the fish ***kind*** with

- Probability ϕ_S of being Salmon
- Probability ϕ_T of being Tuna
- $\phi_S + \phi_T = 1$

Given the fish ***kind***, generate the data point from the corresponding Gaussian distribution

- $p\left(x^{(i)} \middle| S\right) \sim \mathcal{N}(\mu_S, \sigma_S)$ for Salmon
- $p\left(x^{(i)} \middle| T\right) \sim \mathcal{N}(\mu_T, \sigma_T)$ for Tuna

Jump to slide 57

Introduce latent (unobserved) variable

Model parameters: $\Theta = (\phi_S, \phi_T, \mu_S, \mu_T, \sigma_S, \sigma_T)$

Parameters for mixture probabilities

Parameters for each Gaussian distribution

For each data point $x^{(i)}$, we don't know if it is a Salmon or Tuna

Let $z^{(i)}$ be the latent random variable indicating which Gaussian distribution $x^{(i)}$ is from

$z^{(i)} = 1$ for Salmon, $z^{(i)} = 2$ for Tuna

Then the likelihood of $x^{(i)}$ is:

$$p(x^{(i)}|\Theta) = \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}|\Theta)$$

Let Q_i be the distribution of $z^{(i)}$
s.t. $\sum_{z^{(i)}} Q_i(z^{(i)}) = 1$
$Q_i(z^{(i)} = j)$ is the probability of $z^{(i)} = j$

Rewrite the likelihood

$$p(x^{(i)}|\Theta) = \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{p(x^{(i)}, z^{(i)}|\Theta)}{Q_i(z^{(i)})}$$

Log likelihood of data

The likelihood of the whole data: $\mathcal{L}(\theta) = p(X|\Theta) = \prod_{i=1}^{m} p\left(x^{(i)}, z^{(i)}|\Theta\right) = \prod_{i=1}^{m} \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{p(x^{(i)}, z^{(i)}|\Theta)}{Q_i(z^{(i)})}$

Log likelihood: $\mathcal{LL}(\theta) = \sum_{i=1}^{m} \log\left(\sum_{z^{(i)}} Q_i(z^{(i)}) \frac{p(x^{(i)}, z^{(i)}|\Theta)}{Q_i(z^{(i)})}\right) = \sum_{i=1}^{m} \log\left(Q_i(z^{(i)}=1) \frac{p\left(x^{(i)}, z^{(i)}|\Theta\right)}{Q_i(z^{(i)}=1)} + Q_i(z^{(i)}=2) \frac{p\left(x^{(i)}, z^{(i)}|\Theta\right)}{Q_i(z^{(i)}=2)}\right)$

It is difficult to take the derivative of $\mathcal{LL}(\theta)$ w.r.t. $\phi_S, \phi_T, \mu_S, \mu_T, \sigma_S, \sigma_T$, and solve them analytically

Solution: Instead of maximizing $\mathcal{LL}(\theta)$, we can maximize the lower bound of $\mathcal{LL}(\theta)$

Idea: Find some expression E, s.t. $\mathcal{LL}(\theta) \geq E$. When we maximize E, $\mathcal{LL}(\theta)$ is also maximized.

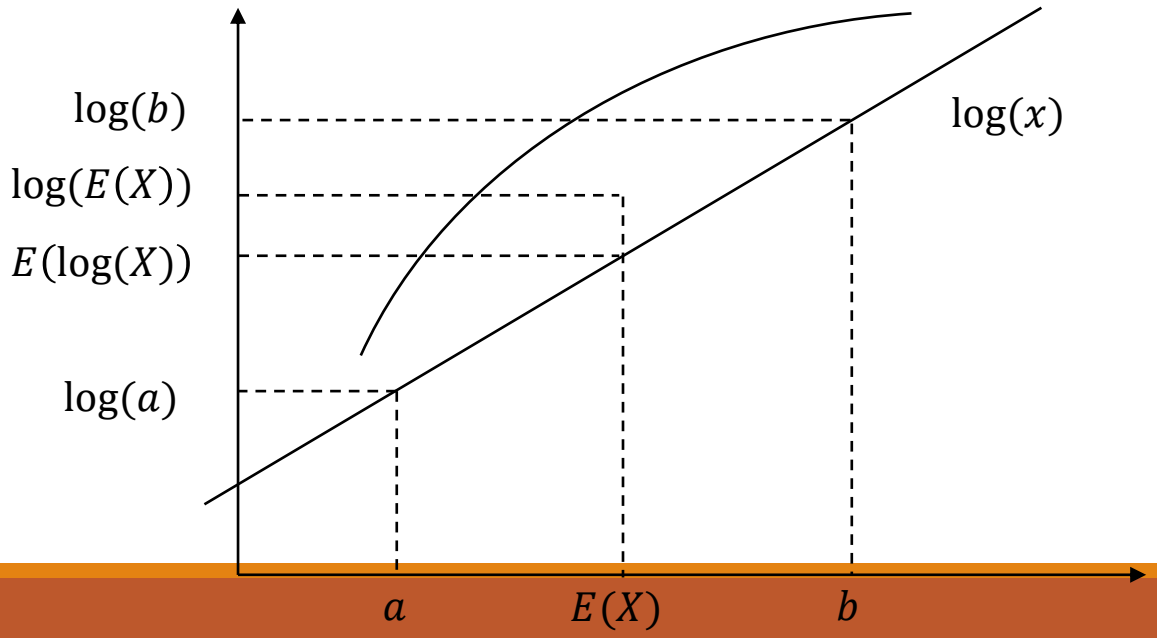
E should have a form that is easier to calculate derivatives

Find the lower bound of $\mathcal{LL}(\theta)$ (optional)

$$\mathcal{LL}(\theta) = \sum_{i=1}^{m} \log\left(\underbrace{Q_i(z^{(i)}=1)}_{\text{Probability}} a + \underbrace{Q_i(z^{(i)}=2)}_{\text{Probability}} b \right)$$

Let a, b be two values of a random variable X

Then $Q_i(z^{(i)}=1)a + Q_i(z^{(i)}=2)b$ is the expectation of $E(X)$



Because log(x) is convex $\log\big(E(X)\big) \geq E(\log(X))$

$$\mathcal{LL}(\theta) \geq \sum_{i=1}^{m} Q_i(z^{(i)}=1)\log(a) + Q_i(z^{(i)}=2)\log(b)$$

$$= \sum_{i=1}^{m}\sum_{z^{(i)}} Q_i(z^{(i)})\log\left(\frac{p(x^{(i)}, z^{(i)}|\Theta)}{Q_i(z^{(i)})}\right)$$

We need to replace $Q_i(z^{(i)})$ with something we know

Jensen's inequality: $f(E(X)) \geq E(f(X))$, when f is convex

How to estimate Q_i (optional)

$$\mathcal{LL}(\theta) \geq \sum_{i=1}^{m} \sum_{z^{(i)}} Q_i(z^{(i)}) \log\left(\frac{p(x^{(i)}, z^{(i)}|\Theta)}{Q_i(z^{(i)})}\right)$$

$Q_i(z^{(i)})$ is unknown, but we can guess it after observing $x^{(i)}$

I.e., after observing a data point $x^{(i)}$, we can "guess" which distribution it is from

$$\frac{1}{\sqrt{2\pi}\sigma_S} e^{-\frac{(x^{(i)}-\mu_S)^2}{2\sigma_S^2}}$$

A **reasonable** way to guess:

If $x^{(i)}$ is drawn from Salmon, then the likelihood of $x^{(i)}$ is $p(x^{(i)}|S)p(S) = p(x^{(i)}|\mu_S, \sigma_S)\phi_S$

If $x^{(i)}$ is drawn from Tuna, then the likelihood of $x^{(i)}$ is $p(x^{(i)}|T)p(T) = p(x^{(i)}|\mu_T, \sigma_T)\phi_T$

Then the chance of $x^{(i)}$ being Salmon is:

$$p(S|x^{(i)}) = \underbrace{\frac{p(x^{(i)}|S)p(S)}{p(x^{(i)}|S)p(S) + p(x^{(i)}|T)p(T)}}_{\text{Posterior, } w_S^{(i)}}$$

The chance of $x^{(i)}$ being Tuna is:

$$p(T|x^{(i)}) = \underbrace{\frac{p(x^{(i)}|T)p(T)}{p(x^{(i)}|S)p(S) + p(x^{(i)}|T)p(T)}}_{\text{Posterior, } w_T^{(i)}}$$

New form of Log-likelihood function (optional)

$$\mathcal{LL}(\theta) \geq \sum_{i=1}^{m} \sum_{z^{(i)}} Q_i\left(z^{(i)}\right) \log\left(\frac{p\left(x^{(i)}, z^{(i)} \middle| \Theta\right)}{Q_i\left(z^{(i)}\right)}\right) = \sum_{i=1}^{m} w_S^{(i)} \log\left(\frac{p\left(x^{(i)}, z^{(i)} = 1 \middle| \Theta\right)}{w_S^{(i)}}\right) + w_T^{(i)} \log\left(\frac{p\left(x^{(i)}, z^{(i)} = 2 \middle| \Theta\right)}{w_T^{(i)}}\right) = \mathcal{LL}'(\theta)$$

$$p\left(x^{(i)}, z^{(i)} = 1 \middle| \Theta\right) = p\left(x^{(i)} \middle| \mu_S, \sigma_S\right)\phi_S = \frac{\phi_S}{\sqrt{2\pi}\sigma_S} e^{-\frac{(x^{(i)}-\mu_S)^2}{2\sigma_S^2}}$$

$$p\left(x^{(i)}, z^{(i)} = 2 \middle| \Theta\right) = p\left(x^{(i)} \middle| \mu_T, \sigma_T\right)\phi_T = \frac{\phi_T}{\sqrt{2\pi}\sigma_T} e^{-\frac{(x^{(i)}-\mu_T)^2}{2\sigma_T^2}}$$

Treating w_S and w_T as known, the derivatives of $\mathcal{LL}'(\theta)$ is much easier to calculate

$$\lfloor \mathcal{LL}(\theta) \rfloor = \mathcal{LL}'^{(\theta)} = \sum_{i=1}^{m} w_S^{(i)} \log\left(\frac{\phi_S}{w_S^{(i)}\sqrt{2\pi}\sigma_S} e^{-\frac{(x^{(i)}-\mu_S)^2}{2\sigma_S^2}}\right) + w_T^{(i)} \log\left(\frac{\phi_T}{w_T^{(i)}\sqrt{2\pi}\sigma_T} e^{-\frac{(x^{(i)}-\mu_T)^2}{2\sigma_T^2}}\right)$$

Maximizing $\mathcal{LL}'(\theta)$ (optional)

$$[\mathcal{LL}(\theta)] = \mathcal{LL}'^{(\theta)} = \sum_{i=1}^{m} w_S^{(i)} \log\left(\frac{\phi_S}{w_S^{(i)}\sqrt{2\pi}\sigma_S} e^{-\frac{(x^{(i)}-\mu_S)^2}{2\sigma_S^2}}\right) + w_T^{(i)} \log\left(\frac{\phi_T}{w_T^{(i)}\sqrt{2\pi}\sigma_T} e^{-\frac{(x^{(i)}-\mu_T)^2}{2\sigma_T^2}}\right)$$

$$\frac{\partial \mathcal{LL}'(\theta)}{\partial \mu_S} = \sum_{i=1}^{m} \frac{\partial}{\partial \mu_S}\left[w_S^{(i)} \log\left(\frac{\phi_S}{\sqrt{2\pi}\sigma_S} e^{-\frac{(x^{(i)}-\mu_S)^2}{2\sigma_S^2}}\right)\right] = \sum_{i=1}^{m} w_S^{(i)}(x^{(i)} - \mu_S) = 0 \quad \Rightarrow \quad \mu_S = \frac{\sum_{i=1}^{m} w_S^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_S^{(i)}}$$

$$\frac{\partial \mathcal{LL}'(\theta)}{\partial \sigma_S} = \sum_{i=1}^{m} \frac{\partial}{\partial \sigma_S}\left[w_S^{(i)} \log\left(\frac{\phi_S}{\sqrt{2\pi}\sigma_S} e^{-\frac{(x^{(i)}-\mu_S)^2}{2\sigma_S^2}}\right)\right] = \sum_{i=1}^{m} w_S^{(i)}[(x^{(i)} - \mu_S)^2 - \sigma_S^2] = 0 \quad \Rightarrow \quad \sigma_S^2 = \frac{\sum_{i=1}^{m} w_S^{(i)}(x^{(i)} - \mu_S)^2}{\sum_{i=1}^{m} w_S^{(i)}}$$

Find the terms that only depends on ϕ_S and ϕ_T $\longrightarrow$ ϕ_S and ϕ_T cannot take any value Under constraint: $\phi_S + \phi_T = 1$

$$\mathcal{LL}'(\theta) = \sum_{i=1}^{m} w_S^{(i)} \log(\phi_S) + w_T^{(i)} \log(\phi_T) \longrightarrow \text{Construct a Lagrangian: } \mathcal{L}(\phi_S) = \left(\sum_{i=1}^{m} w_S^{(i)} \log(\phi_S) + w_T^{(i)} \log(\phi_T)\right) + \beta(\phi_S + \phi_T - 1)$$

$$\frac{\partial \mathcal{L}(\phi_S)}{\partial \phi_S} = \frac{\sum_{i=1}^{m} w_S^{(i)}}{\phi_S} + \beta = 0 \quad \Rightarrow \quad \phi_S = \frac{\sum_{i=1}^{m} w_S^{(i)}}{-\beta} \quad \phi_T = \frac{\sum_{i=1}^{m} w_T^{(i)}}{-\beta} \quad \Rightarrow \quad -\beta = \sum_{i=1}^{m}\left(w_S^{(i)} + w_T^{(i)}\right) = m$$

Solutions of maximizing $\mathcal{L}\mathcal{L}'(\theta)$ (optional)

$$\mu_S = \frac{\sum_{i=1}^{m} w_S^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_S^{(i)}} \qquad \mu_T = \frac{\sum_{i=1}^{m} w_T^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_T^{(i)}}$$

$$\sigma_S^2 = \frac{\sum_{i=1}^{m} w_S^{(i)} \left(x^{(i)} - \mu_S\right)^2}{\sum_{i=1}^{m} w_S^{(i)}} \qquad \sigma_T^2 = \frac{\sum_{i=1}^{m} w_T^{(i)} \left(x^{(i)} - \mu_T\right)^2}{\sum_{i=1}^{m} w_T^{(i)}}$$

$$\phi_S = \frac{\sum_{i=1}^{m} w_S^{(i)}}{m} \qquad \phi_T = \frac{\sum_{i=1}^{m} w_T^{(i)}}{m}$$

Repeatedly update all parameters, $\phi_S, \phi_T, \mu_S, \mu_T, \sigma_S, \sigma_T$ until convergence

In which, $w_S^{(i)} = p\left(S|x^{(i)}\right) = \frac{p\left(x^{(i)}|S\right)\phi_S}{p\left(x^{(i)}|S\right)\phi_S + p\left(x^{(i)}|T\right)\phi_T}$

$$w_T^{(i)} = p\left(T|x^{(i)}\right) = \frac{p\left(x^{(i)}|S\right)\phi_S}{p\left(x^{(i)}|S\right)\phi_S + p\left(x^{(i)}|T\right)\phi_T}$$

E-M (Expectation-Maximization) Algorithm (1-D Gaussian)

Assume the data $\{x^{(i)}\}$ are drawn from *k* Gaussian distributions with probabilities $\phi_1, \phi_2, \cdots, \phi_k$
Each distribution has parameters μ_j, σ_j $(j = 1,2, \cdots, k)$

Randomly initialize all parameters $\phi_1, \phi_2, \cdots, \phi_k$ and μ_j, σ_j $(j = 1,2, \cdots, k)$

Repeat until convergence {

E-step: For each $x^{(i)}$, compute the expectation/prob of which distribution it is from

$$w_j^{(i)} := p\left(z^{(i)} = j \middle| x^{(i)}\right) = \frac{p(x^{(i)}|\mu_j, \sigma_j)\phi_j}{\sum_j p(x^{(i)}|\mu_j, \sigma_j)\phi_j} \qquad \text{For } j = 1,2, \cdots, k$$

M-step: Update the parameters of each cluster j (as if $w_j^{(i)}$ is correct) by maximizing the likelihood:

$$\mu_j := \underbrace{\frac{\sum_{i=1}^m w_j^{(i)} x^{(i)}}{\sum_{i=1}^m w_j^{(i)}}} \qquad \sigma_j^2 := \frac{\sum_{i=1}^m w_j^{(i)} \left(x^{(i)} - \mu_j\right)^2}{\sum_{i=1}^m w_j^{(i)}} \qquad \phi_j := \frac{\sum_{i=1}^m w_j^{(i)}}{m} \qquad \text{For } j = 1,2, \cdots, k$$

Weighted average

}

Compare with *K*-means

Randomly initialize all *k* centroids $\mu_1, \mu_2, \cdots, \mu_k$

Repeat until convergence {

E-step: For each $x^{(i)}$, assign it to the closest centroid

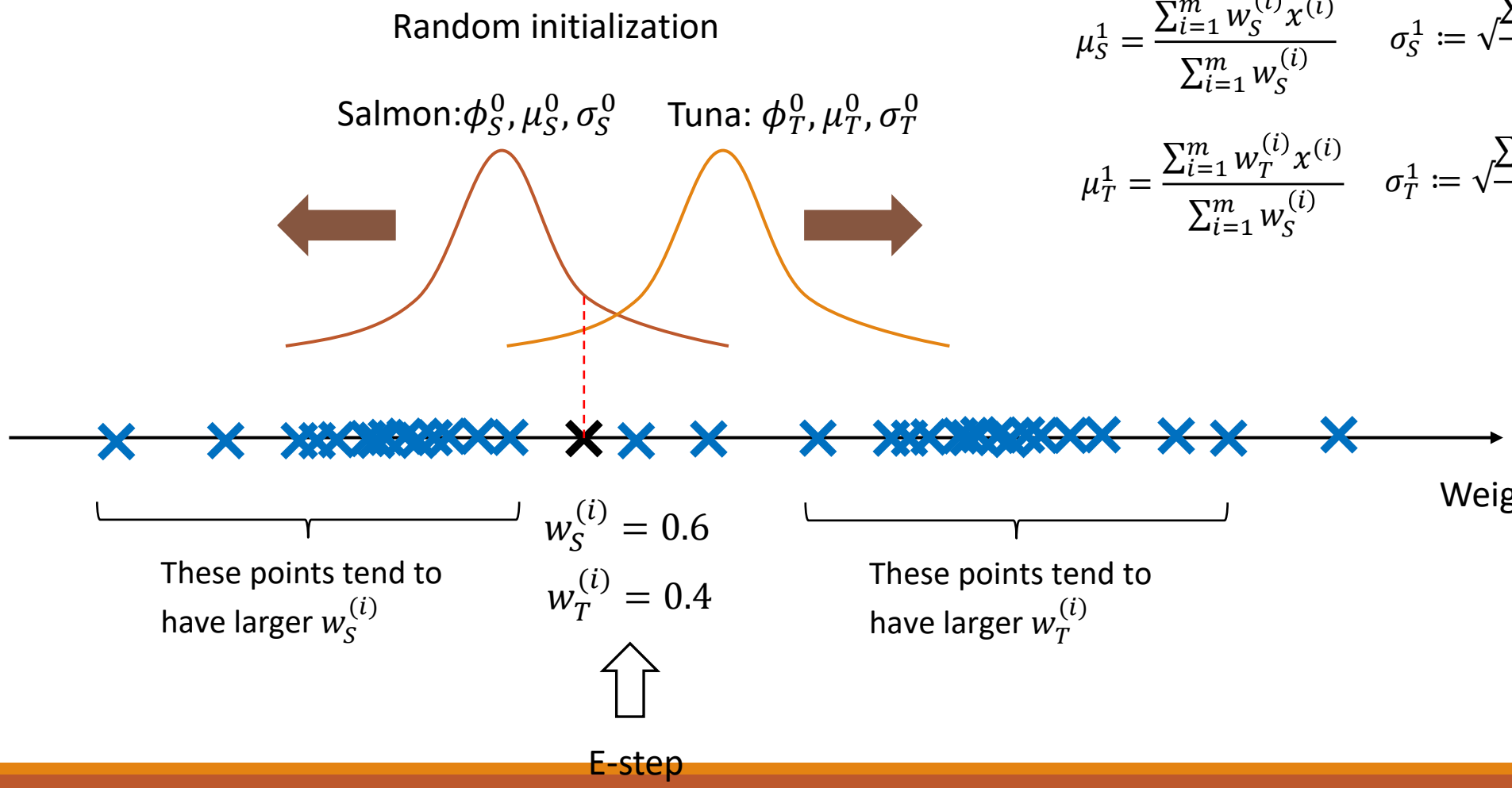
$$c^{(i)} := \arg\min_j \left\| x^{(i)} - \mu_j \right\|^2$$

M-step: Update the positions of centroids

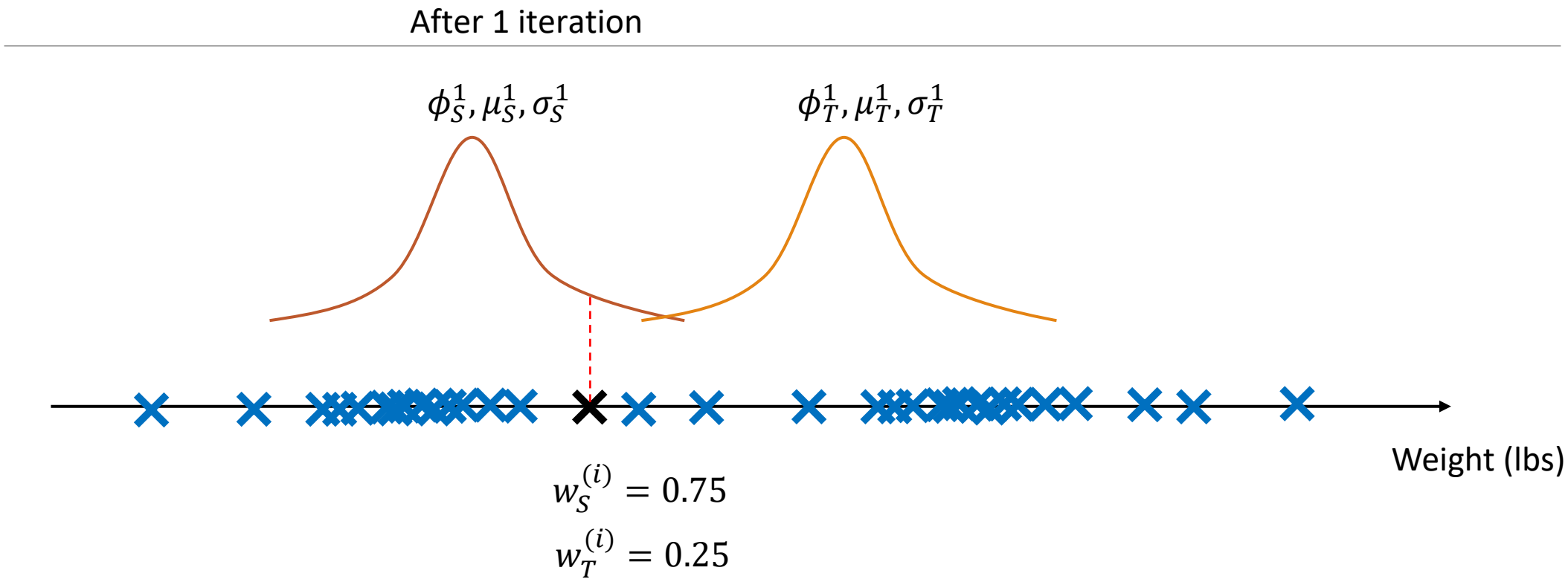
$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$

}

Demonstration with k = 2, 1-D Gaussian

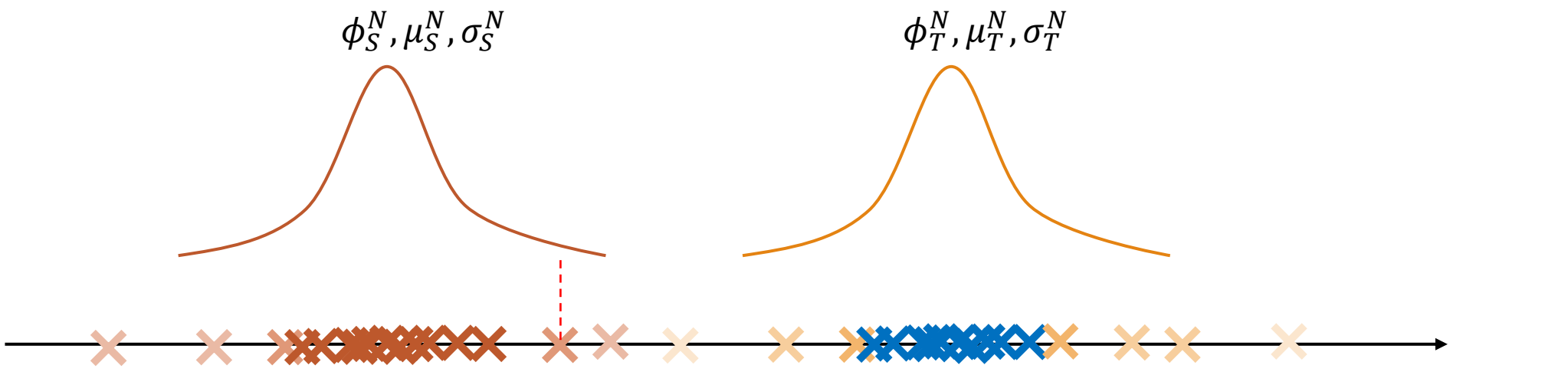


Demonstration with k = 2, 1-D Gaussian



Demonstration with k = 2, 1-D Gaussian

After N iterations, all parameters converge



$w_S^{(i)} = 0.77$

$w_T^{(i)} = 0.23$

This data point is 0.77 chance a Salmon, and 0.23 chance a Tuna

No hard assignment!

What about multivariate Gaussians?

A random vector $X = \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix}$ is said to have a multivariate Gaussian distribution

If its probability density function is: $$p(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Mean: $\mu \in \mathbb{R}^n$ Covariance matrix: Σ

Property: $$\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) dx_1 dx_2 \cdots dx_n = 1.$$

Covariance matrix

If X_i, Y_j are a pair of 1-D random variables

Then the covariance is defined as: $Cov[X_i, Y_j] = E\left[\left(X - E(X_i)\right)\left(Y - E(Y_j)\right)\right] = E[X_i Y_j] - E(X_i)E(Y_j)$

If $X = \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix}$, $Y = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix}$ are a pair of n-D random variables

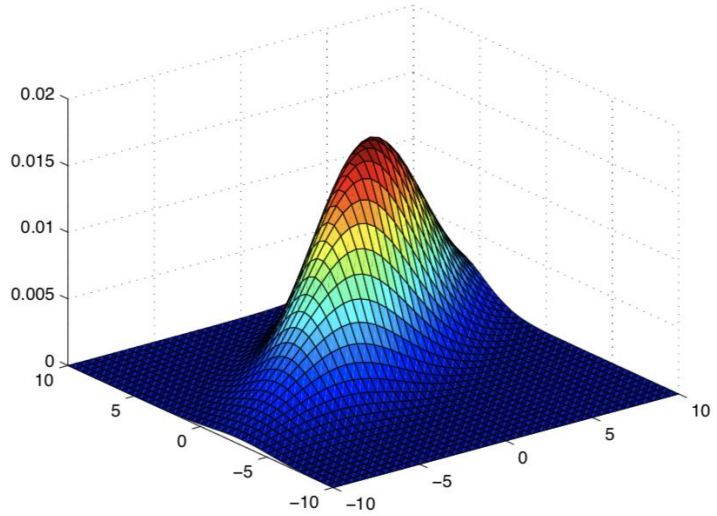
Then the covariance matrix Σ is a $n \times n$ symmetric matrix whose (i, j) th entry is $Cov[X_i, Y_j]$

$$\Sigma = \begin{bmatrix} Cov[X_1, Y_1] & Cov[X_1, Y_2] & \cdots & Cov[X_1, Y_n] \\ Cov[X_2, Y_1] & Cov[X_2, Y_2] & & Cov[X_2, Y_n] \\ \vdots & \vdots & & \vdots \\ Cov[X_n, Y_1] & Cov[X_n, Y_2] & & Cov[X_n, Y_n] \end{bmatrix}$$

When n=2, 2-D Gaussian distribution

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \qquad \Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2 \\ \sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

$$p(x) = \frac{1}{2\pi \begin{vmatrix} \sigma_1^2 & \sigma_1\sigma_2 \\ \sigma_1\sigma_2 & \sigma_2^2 \end{vmatrix}^{1/2}} \exp\left(-\frac{1}{2} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix}^T \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2 \\ \sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}^{-1} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix} \right)$$



Special case: covariance matrix is diagonal (0 cross-correlation)

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

$$p(x) = \frac{1}{2\pi \begin{vmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{vmatrix}^{1/2}} \exp\left(-\frac{1}{2} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix}^T \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}^{-1} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix} \right)$$

$$= \frac{1}{2\pi\sqrt{\sigma_1^2 \sigma_2^2}} \exp\left(-\frac{1}{2} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix}^T \begin{bmatrix} 1/\sigma_1^2 & 0 \\ 0 & 1/\sigma_2^2 \end{bmatrix}^{-1} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix} \right)$$

$$= \frac{1}{2\pi\sigma_1\sigma_2} \exp\left(-\frac{1}{2\sigma_1^2}(x_1 - \mu_1)^2 - \frac{1}{2\sigma_2^2}(x_2 - \mu_2)^2 \right)$$

$$= \underbrace{\frac{1}{2\pi\sigma_1} \exp\left(-\frac{1}{2\sigma_1^2}(x_1 - \mu_1)^2 \right)}_{\text{PDF for } x_1} \cdot \underbrace{\frac{1}{2\pi\sigma_2} \exp\left(-\frac{1}{2\sigma_2^2}(x_2 - \mu_2)^2 \right)}_{\text{PDF for } x_2} \Rightarrow$$

Product of two independent 1-D Gaussian distribution

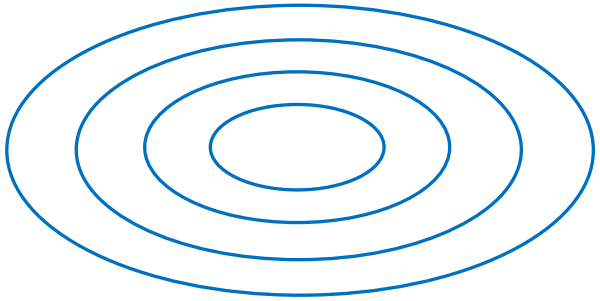
Contours of 2-D Gaussians

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad p(x) = \frac{1}{2\pi\sigma_1\sigma_2} \exp\left(-\frac{1}{2\sigma_1^2}(x_1 - \mu_1)^2 - \frac{1}{2\sigma_2^2}(x_2 - \mu_2)^2 \right)$$

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$$

To draw contours, let $p(x)$ be a constant

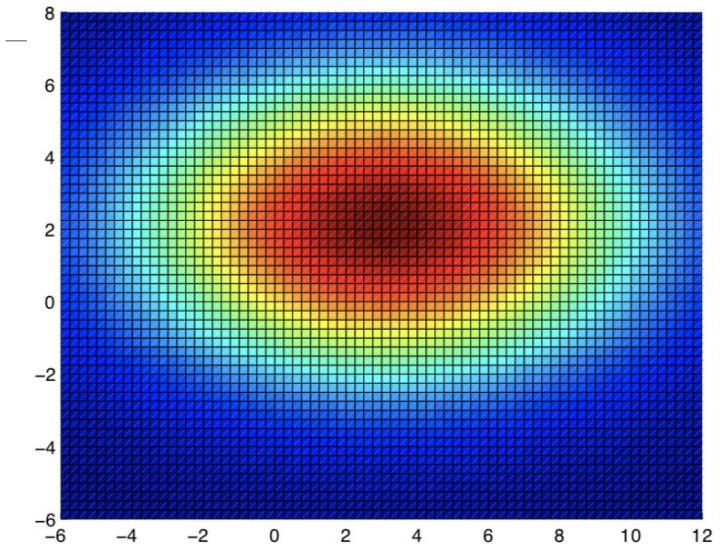
$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \qquad p(x) = c \quad \Rightarrow \quad 1 = \frac{(x_1 - \mu_1)^2}{2\sigma_1^2 \log\left(\frac{1}{2\pi c\sigma_1\sigma_2}\right)} + \frac{(x_2 - \mu_2)^2}{2\sigma_2^2 \log\left(\frac{1}{2\pi c\sigma_1\sigma_2}\right)}$$



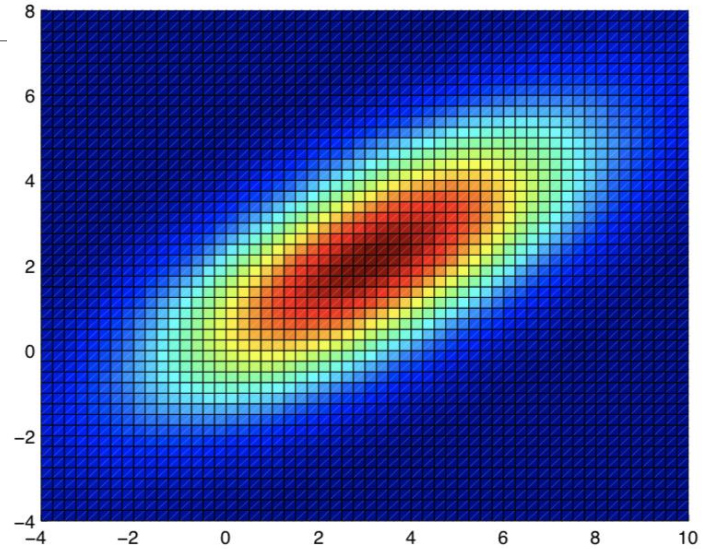
$$1 = \frac{(x_1 - \mu_1)^2}{r_1^2} + \frac{(x_2 - \mu_2)^2}{r_2^2}$$

An ellipse!

Covariance matrix decides the shape of ellipse



$$\mu = \begin{pmatrix} 3 \\ 2 \end{pmatrix} \qquad \Sigma = \begin{bmatrix} 25 & 0 \\ 0 & 9 \end{bmatrix}$$



$$\mu = \begin{pmatrix} 3 \\ 2 \end{pmatrix} \qquad \Sigma = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix}$$

E-M algorithm for mixture of Multivariate Gaussians

Assume the data $\{x^{(i)}\}$ are drawn from *k n*-D Gaussian distributions with probabilities $\phi_1, \phi_2, \cdots, \phi_k$
Each distribution has parameters $\mu_j, \Sigma_j \ (j = 1,2, \cdots, k)$

Randomly initialize all parameters $\phi_1, \phi_2, \cdots, \phi_k$ and $\mu_j, \Sigma_j \ (j = 1,2, \cdots, k)$

Repeat until convergence {

E-step: For each $x^{(i)}$, compute the expectation of which distribution it is from

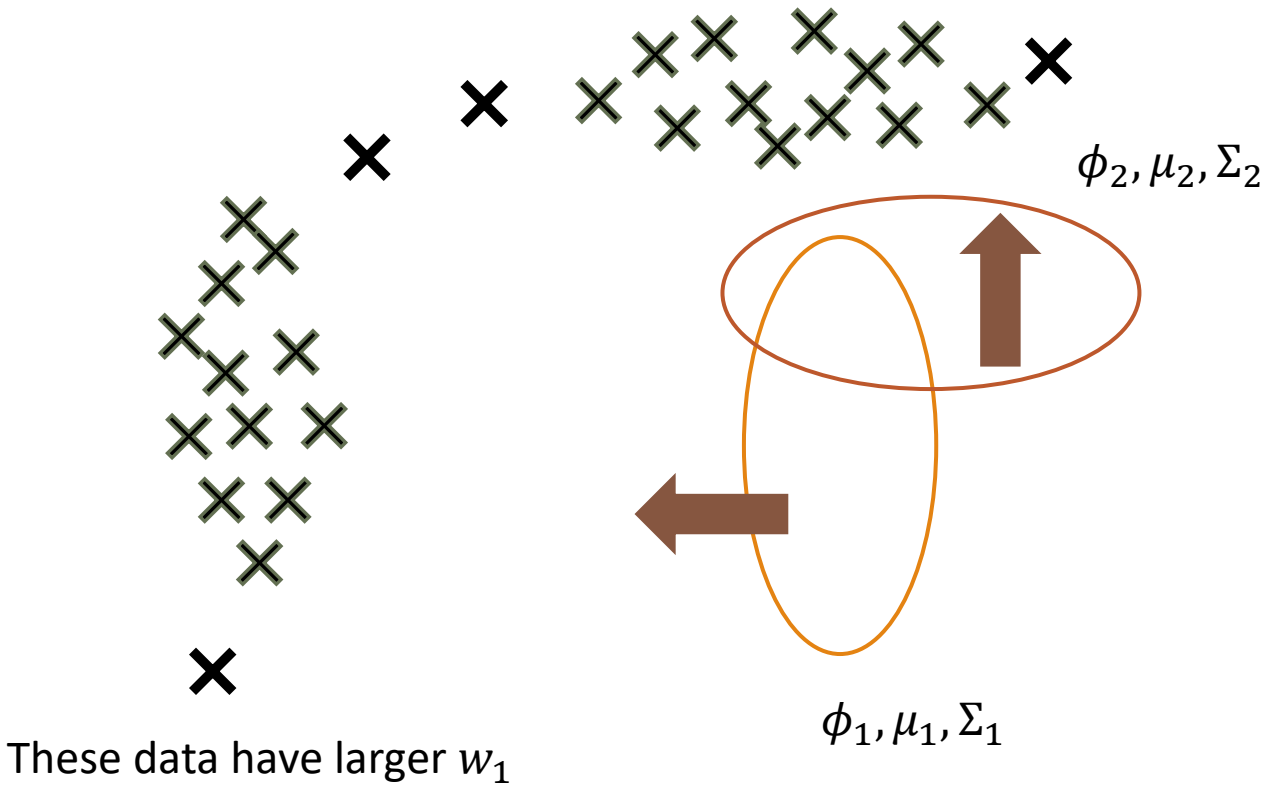
$$w_j^{(i)} := p\left(z^{(i)} = j \middle| x^{(i)}\right) = \frac{p(x^{(i)}|\mu_j, \Sigma_j)\phi_j}{\sum_j p(x^{(i)}|\mu_j, \Sigma_j)\phi_j} \qquad \text{For } j = 1,2, \cdots, k$$

M-step: Update the parameters (as if $w_j^{(i)}$ is correct) by maximizing the likelihood:

$$\mu_j := \frac{\sum_{i=1}^m w_j^{(i)} x^{(i)}}{\sum_{i=1}^m w_j^{(i)}} \qquad \Sigma_j := \frac{\sum_{i=1}^m w_j^{(i)} \left(x^{(i)} - \mu_j\right)\left(x^{(i)} - \mu_j\right)^T}{\sum_{i=1}^m w_j^{(i)}} \qquad \phi_j := \frac{\sum_{i=1}^m w_j^{(i)}}{m} \qquad \text{For } j = 1,2, \cdots, k$$

}

Demo of learning a mixture of 2-D Gaussians



Random initialization

For each $x^{(i)}$, compute

$$w_1^{(i)} := \frac{p(x^{(i)}|\mu_1, \Sigma_1)\phi_1}{p(x^{(i)}|\mu_1, \Sigma_1)\phi_1 + p(x^{(i)}|\mu_2, \Sigma_2)\phi_2}$$

$$w_2^{(i)} := \frac{p(x^{(i)}|\mu_2, \Sigma_2)\phi_2}{p(x^{(i)}|\mu_1, \Sigma_1)\phi_1 + p(x^{(i)}|\mu_2, \Sigma_2)\phi_2}$$

Update:

$$\mu_1 := \frac{\sum_{i=1}^m w_1^{(i)} x^{(i)}}{\sum_{i=1}^m w_1^{(i)}} \qquad \mu_2 := \frac{\sum_{i=1}^m w_2^{(i)} x^{(i)}}{\sum_{i=1}^m w_2^{(i)}}$$

Demo of learning a mixture of 2-D Gaussians (cont.)