Clustering

Sriram Sankararaman

The instructor gratefully acknowledges Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

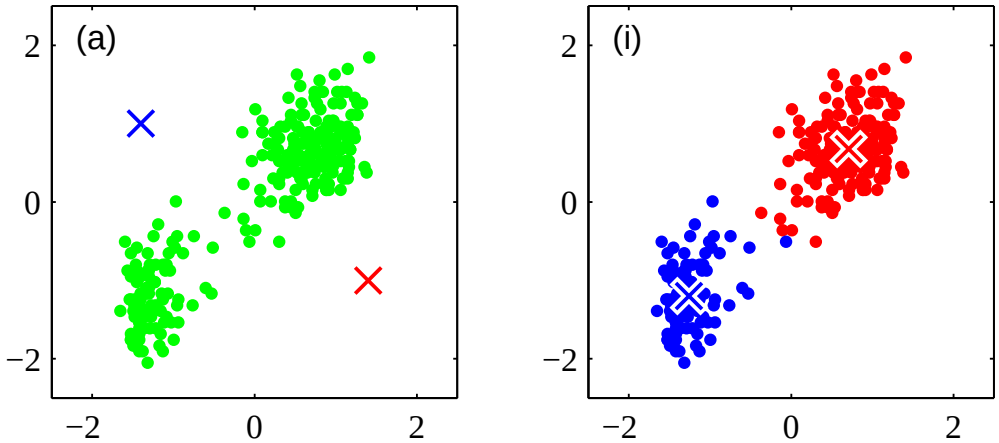
Outline

Clustering

Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ and K, we want to output

- $\{\boldsymbol{\mu}_k\}_{k=1}^K$: prototypes (or centroids) of clusters
- $A(\boldsymbol{x}_n) \in \{1, 2, \ldots, K\}$: the cluster membership, i.e., the cluster ID assigned to $\boldsymbol{x}_n$

Toy Example Cluster data into two clusters.



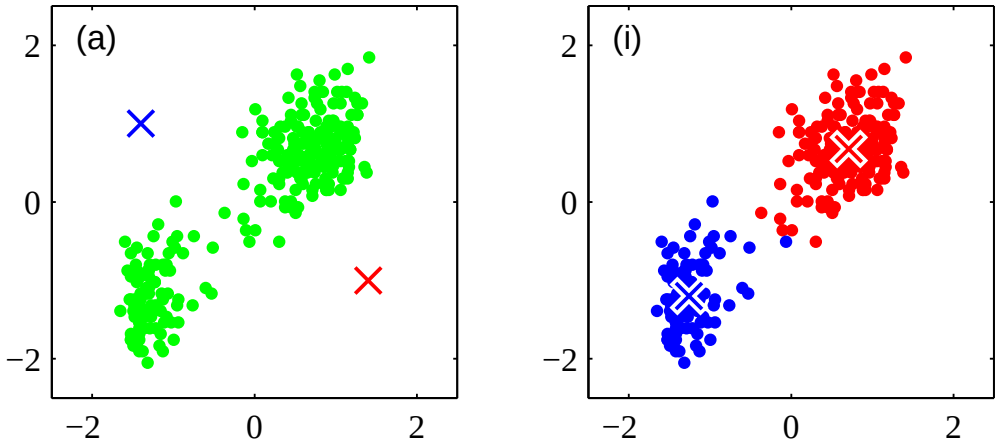
Applications

Clustering

Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ and K, we want to output

- $\{\boldsymbol{\mu}_k\}_{k=1}^K$: prototypes (or centroids) of clusters
- $A(\boldsymbol{x}_n) \in \{1, 2, \ldots, K\}$: the cluster membership, i.e., the cluster ID assigned to $\boldsymbol{x}_n$

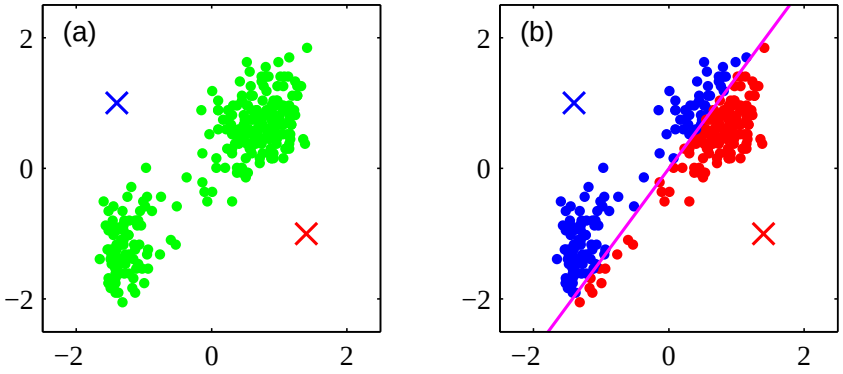
Toy Example Cluster data into two clusters.



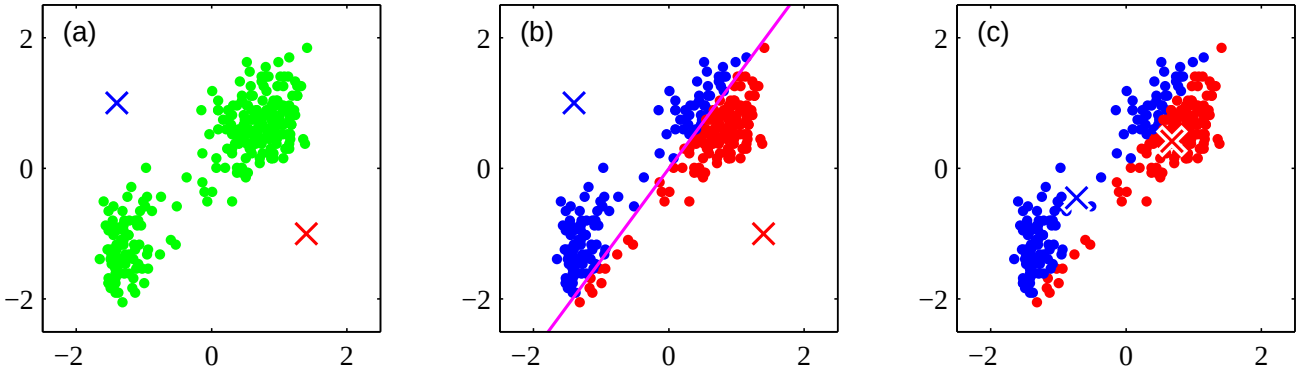
Applications

- Identify communities within social networks
- Find topics in news stories
- Group similiar sequences into gene families

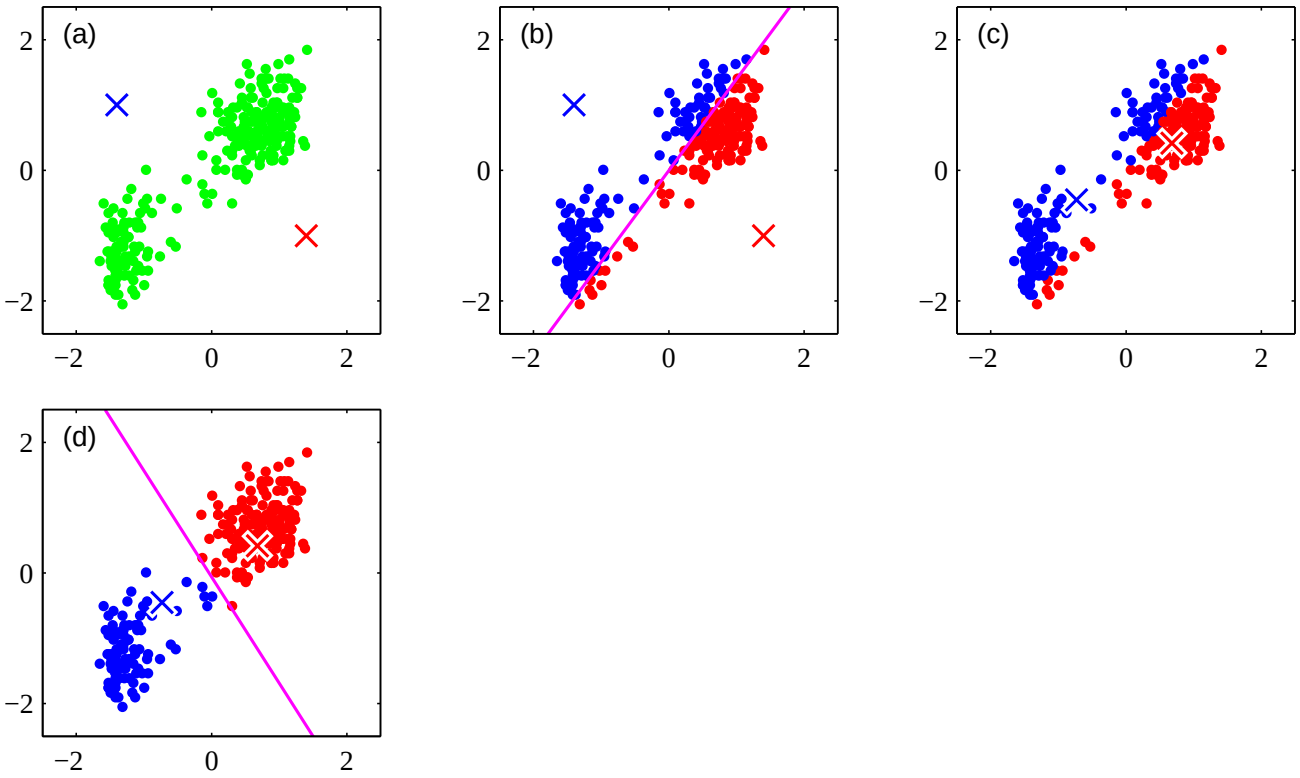
K-means example



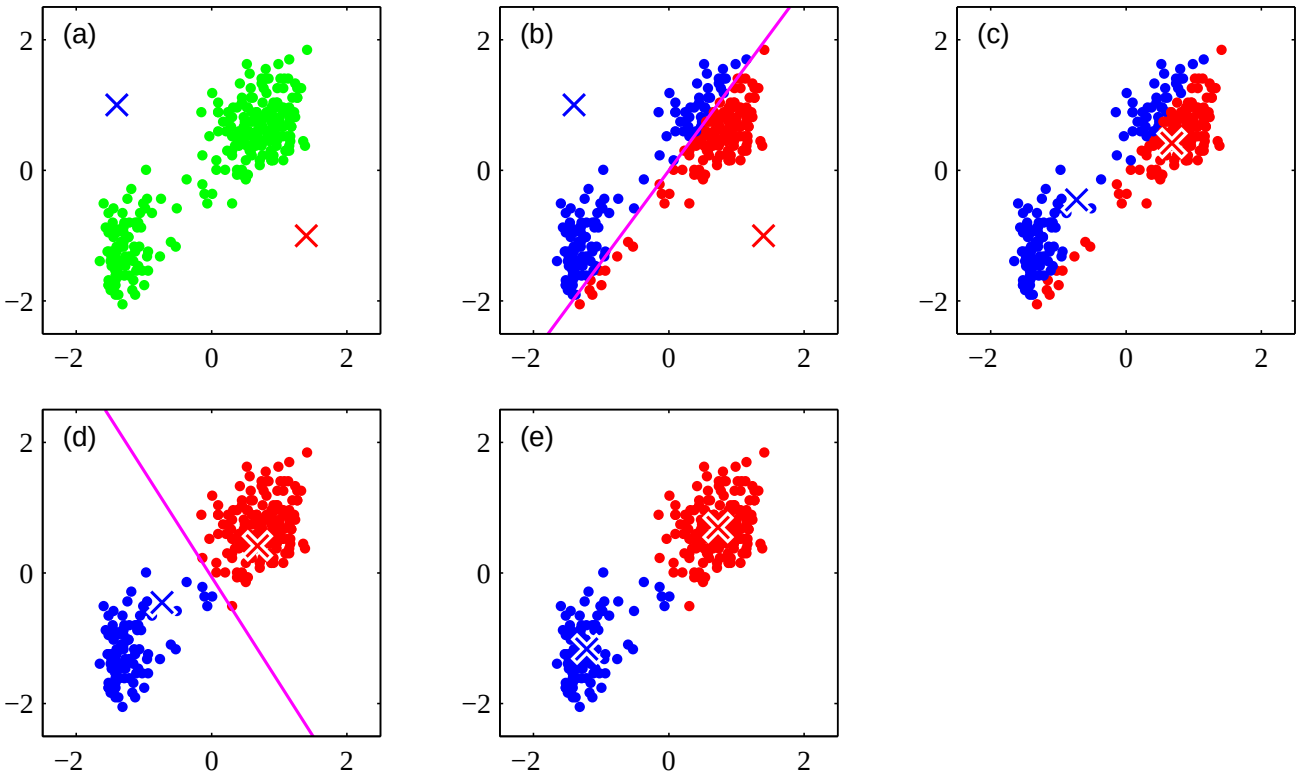
K-means example



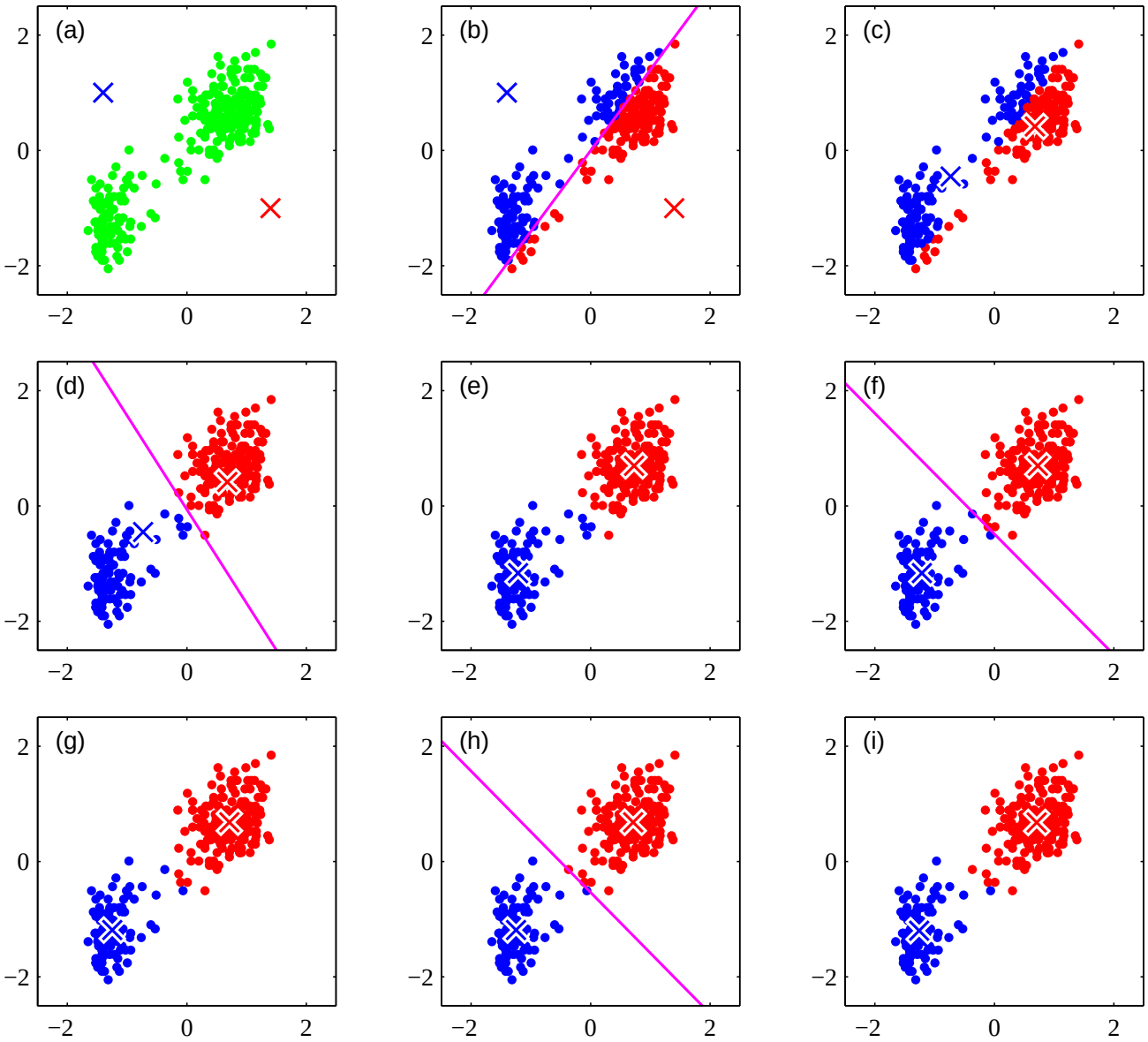
K-means example



K-means example



K-means example



K-means clustering

Intuition Data points assigned to cluster k should be close to $\boldsymbol{\mu}_k$, the prototype.

K-means clustering

Intuition Data points assigned to cluster k should be close to $\boldsymbol{\mu}_k$, the prototype.

Distortion measure (clustering objective function, cost function)

$$J(\{r_{nk}\}, \{\boldsymbol{\mu}_k\}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|_2^2$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if} \quad A(\boldsymbol{x}_n) = k$$

K-means clustering

K-means objective

$$argmin_{\{r_{nk}\},\{\boldsymbol{\mu}_k\}} J(\{r_{nk}\},\{\boldsymbol{\mu}_k\}) = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|_2^2$$

where $r_{nk} \in \{0,1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if} \quad A(\boldsymbol{x}_n) = k$$

- Is a non-convex objective function.
- Minimizing the K-means objective function is NP-hard.

Llyod's algorithm for minimizing the K-means objective

Often simply called the K-means algorithm

Minimize cost function alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ to some values

Llyod's algorithm for minimizing the K-means objective

Often simply called the K-means algorithm

Minimize cost function alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ to some values
- **Step 1** Assume the current value of $\{\boldsymbol{\mu}_k\}$ fixed, minimize J over $\{r_{nk}\}$, which leads to the following cluster assignment rule

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

Llyod's algorithm for minimizing the K-means objective

Often simply called the K-means algorithm

Minimize cost function alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ to some values
- **Step 1** Assume the current value of $\{\boldsymbol{\mu}_k\}$ fixed, minimize J over $\{r_{nk}\}$, which leads to the following cluster assignment rule

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- **Step 2** Assume the current value of $\{r_{nk}\}$ fixed, minimize J over $\{\boldsymbol{\mu}_k\}$, which leads to the following rule to update the prototypes of the clusters

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

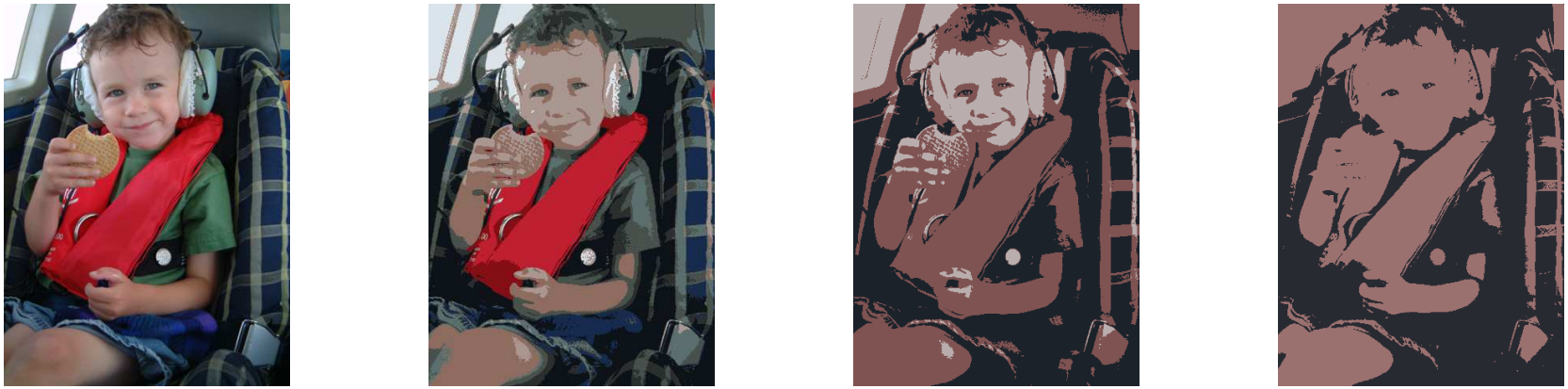
- **Step 3** Stop if the objective function J stays the same or return to Step 1

Remarks

- Prototype $\boldsymbol{\mu}_k$ is the mean of data points assigned to the cluster k, hence 'K-means'
- The procedure reduces J in both Step 1 and Step 2 and thus makes improvements on each iteration

Application: vector quantization

- Replace data point with associated prototype $\boldsymbol{\mu}_k$
- In other words, compress the data points into i) a codebook of all the prototypes; ii) a list of indices to the codebook for the data points
- Lossy compression, especially for small K



Clustering pixels and vector quantizing them. From left to right: Original image, quantized with large K, medium K, and a small K. Details are missing due to the higher compression (smaller K).

Properties of the K-means algorithm

- Does the K-means algorithm converge (*i.e.*, terminate)?
  - Yes.
- How long does it take to converge ?
  - In the worst case, exponential in the number of data points.
  - In practice, very quick.

Properties of the K-means algorithm

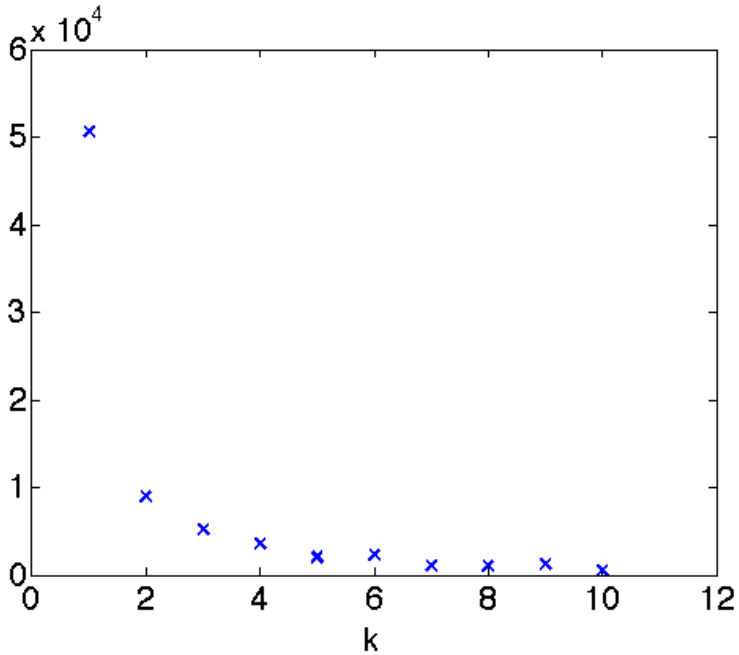
How good is the K-means solution?

- Converges to a local minimum.
- The solution depends on the initialization.
- In practice, run many times with different initializations and pick the best.
- K-means++ is a neat approximation algorithm that has theoretical guarantees on the final value of the objective.
  - Still no guarantee that you will reach the global minimum
  - You are guaranteed to get reasonably close (approximation guarantee on the final value).

Other practical issues

Choosing K

- Increasing K will always decrease the optimal value of the K-means objective.
  - Analogous to overfitting in supervised learning.
- Information criteria that effectively regularize more complex models.



K-medoids

- K-means is sensitive to outliers.
- In some applications we want the prototypes to be one of the points.
- Leads to K-medoids.

K-medoids

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ by randomly selecting K of the N points
- **Step 1** Assume the current value of $\{\boldsymbol{\mu}_k\}$ fixed, assign points to clusters:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- **Step 2** Assume the current value of $\{r_{nk}\}$ fixed, update the prototype of cluster k. In K-medoids, the prototype for a cluster is the data point that is closest to all other data points in the cluster

$$k* = \arg\min_{m:r_{mk}=1} \sum_n r_{nk} \|\boldsymbol{x}_n - \boldsymbol{x}_m\|_2^2$$

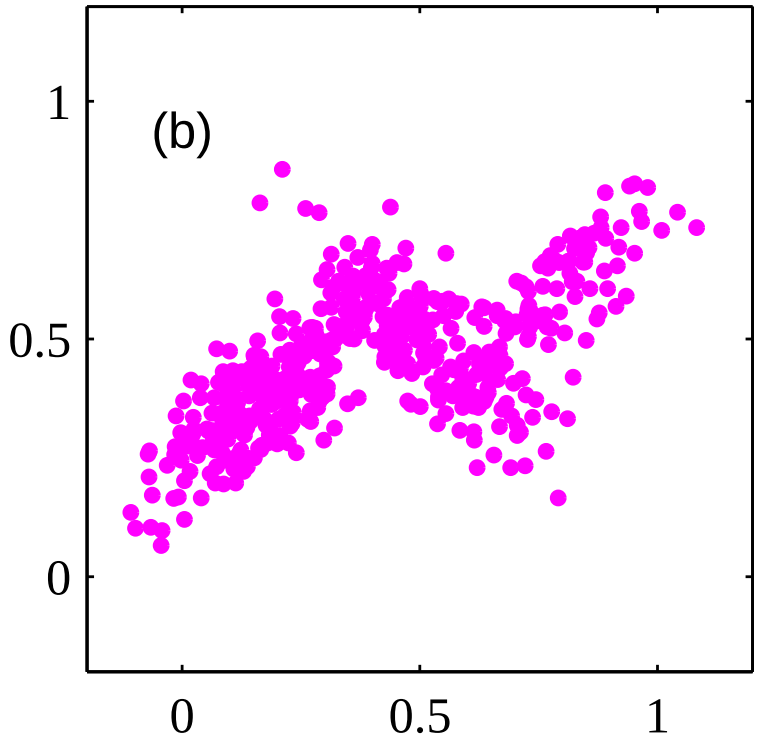
$$\boldsymbol{\mu}_k = \boldsymbol{x}_{k*}$$

- **Step 3** Stop if the objective function J stays the same or return to Step 1

Probabilistic interpretation of clustering?

We can impose a probabilistic interpretation of our intuition that points stay close to their cluster centers

- How can we model $p(\boldsymbol{x})$ to reflect this?

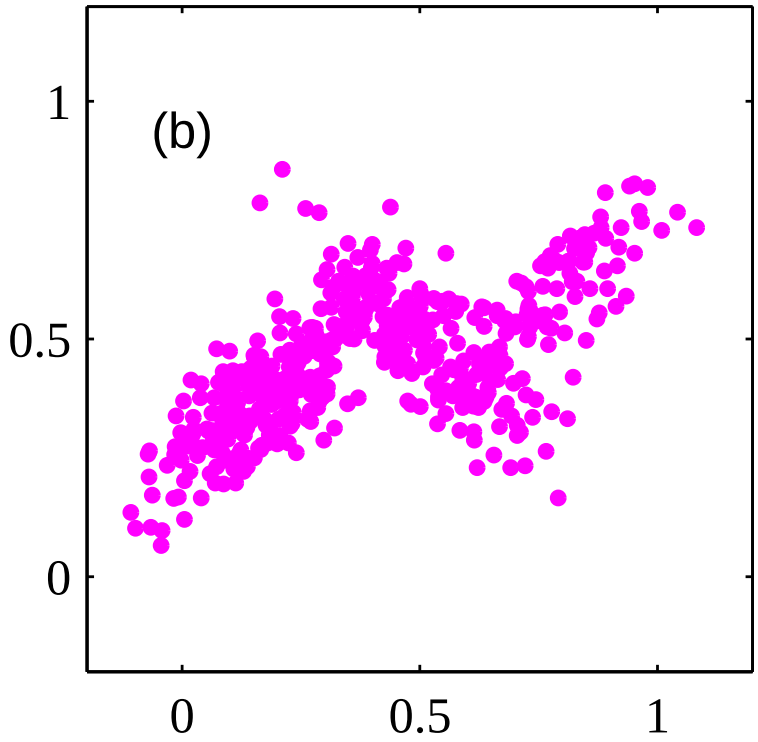


- Data points seem to form 3 clusters

Probabilistic interpretation of clustering?

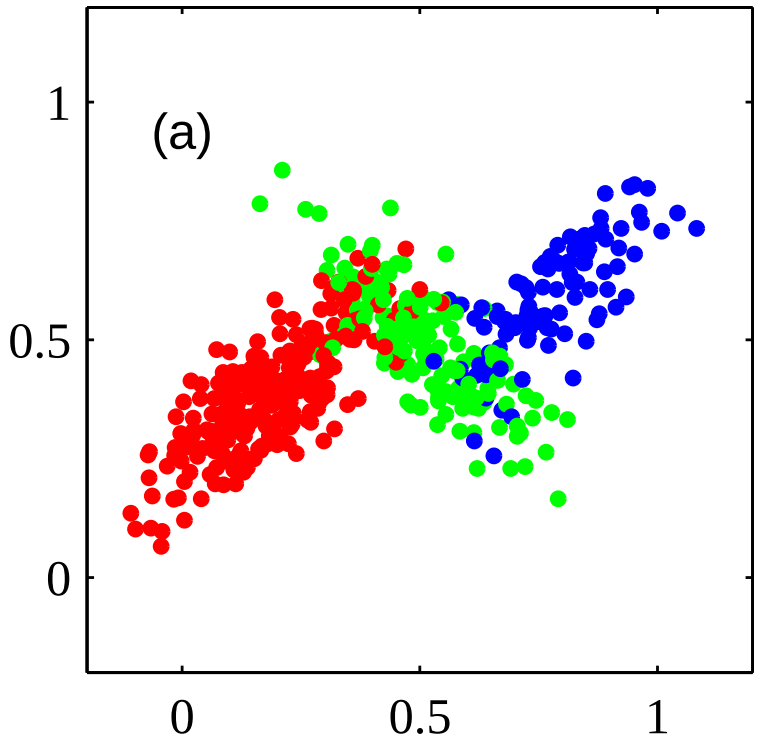
We can impose a probabilistic interpretation of our intuition that points stay close to their cluster centers

- How can we model $p(\boldsymbol{x})$ to reflect this?



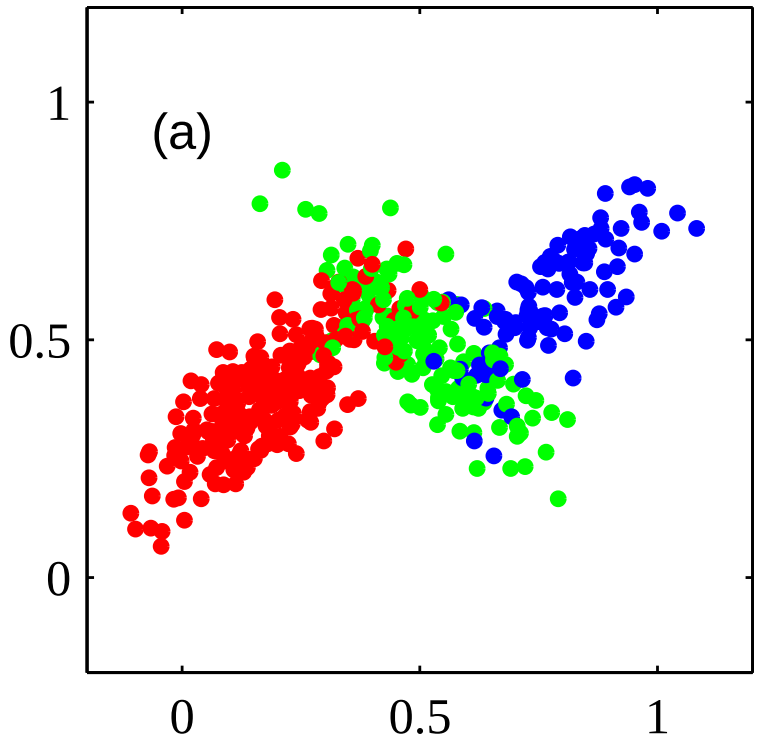
- Data points seem to form 3 clusters
- We cannot model $p(\boldsymbol{x})$ with simple and known distributions
- E.g., the data is not a Guassian b/c we have 3 distinct concentrated regions

Gaussian mixture models: intuition



- We can model *each* region with a distinct distribution
- Common to use Gaussians, i.e., Gaussian mixture models (GMMs) or mixture of Gaussians (MoGs).

Gaussian mixture models: intuition



- We can model *each* region with a distinct distribution
- Common to use Gaussians, i.e., Gaussian mixture models (GMMs) or mixture of Gaussians (MoGs).
- We don't know *cluster assignments* (label) or *parameters* of Gaussians or *mixture components*!
- We need to learn them all from our *unlabeled* data $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$

Gaussian mixture models: formal definition

A Gaussian mixture model has the following density function for $\boldsymbol{x}$

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

- K: the number of Gaussians — they are called (mixture) components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of the k-th component

Gaussian mixture models: formal definition

A Gaussian mixture model has the following density function for $\boldsymbol{x}$

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

- K: the number of Gaussians — they are called (mixture) components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of the k-th component
- ω_k: mixture weights – they represent how much each component contributes to the final distribution. It satisfies two properties:

Gaussian mixture models: formal definition

A Gaussian mixture model has the following density function for $\boldsymbol{x}$

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

- K: the number of Gaussians — they are called (mixture) components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of the k-th component
- ω_k: mixture weights – they represent how much each component contributes to the final distribution. It satisfies two properties:

$$\forall\ k,\ \omega_k > 0, \quad \text{and} \quad \sum_k \omega_k = 1$$

The properties ensure $p(\boldsymbol{x})$ is a properly normalized probability density function.

GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

$$p(\boldsymbol{x}, z) = p(z)p(\boldsymbol{x}|z)$$

where z is a discrete random variable taking values between 1 and K.

GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

$$p(\boldsymbol{x}, z) = p(z)p(\boldsymbol{x}|z)$$

where z is a discrete random variable taking values between 1 and K. Denote

$$\omega_k = p(z = k)$$

Now, assume the conditional distributions are Gaussian distributions

$$p(\boldsymbol{x}|z = k) = N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

$$p(\boldsymbol{x}, z) = p(z)p(\boldsymbol{x}|z)$$

where z is a discrete random variable taking values between 1 and K. Denote

$$\omega_k = p(z = k)$$

Now, assume the conditional distributions are Gaussian distributions

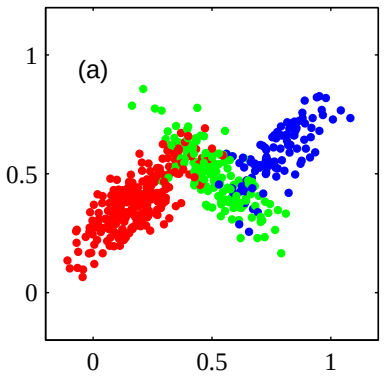
$$p(\boldsymbol{x}|z = k) = N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Then, the marginal distribution of $\boldsymbol{x}$ is

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

Namely, the Gaussian mixture model

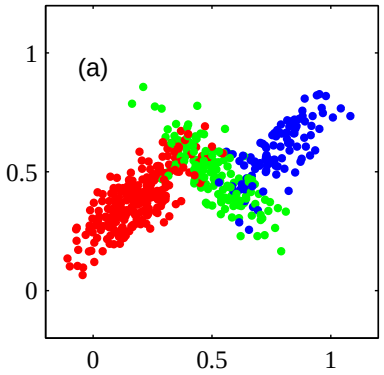
GMMs: example



The conditional distribution between $\boldsymbol{x}$ and z (representing color) are

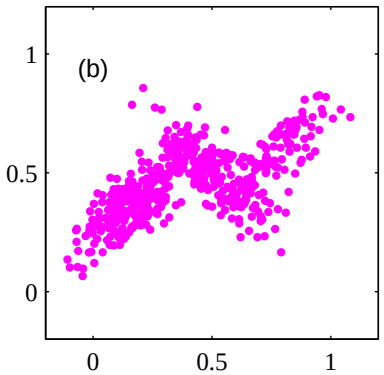
$$p(\boldsymbol{x}|z = red) = N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$
$$p(\boldsymbol{x}|z = blue) = N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$
$$p(\boldsymbol{x}|z = green) = N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

GMMs: example



The conditional distribution between $\boldsymbol{x}$ and z (representing color) are

$$p(\boldsymbol{x}|z = red) = N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$
$$p(\boldsymbol{x}|z = blue) = N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$
$$p(\boldsymbol{x}|z = green) = N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$



The marginal distribution is thus

$$p(\boldsymbol{x}) = p(red)N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + p(blue)N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) + p(green)N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

Parameter estimation for Gaussian mixture models

The parameters in GMMs are $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$. To estimate, consider the simple (and unrealistic) case first.

We have labels z If we assume z is observed for every $\boldsymbol{x}$, then our estimation problem is easier to solve. Our training data is augmented:

$$\mathcal{D}' = \{\boldsymbol{x}_n, z_n\}_{n=1}^N$$

z_n denotes the region where $\boldsymbol{x}_n$ comes from. $\mathcal{D}'$ is the *complete* data and $\mathcal{D}$ the *incomplete* data. How can we learn our parameters?

Parameter estimation for Gaussian mixture models

The parameters in GMMs are $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$. To estimate, consider the simple (and unrealistic) case first.

We have labels z If we assume z is observed for every $\boldsymbol{x}$, then our estimation problem is easier to solve. Our training data is augmented:

$$\mathcal{D}' = \{\boldsymbol{x}_n, z_n\}_{n=1}^N$$

z_n denotes the region where $\boldsymbol{x}_n$ comes from. $\mathcal{D}'$ is the *complete* data and $\mathcal{D}$ the *incomplete* data. How can we learn our parameters?

Given $\mathcal{D}'$, the maximum likelihood estimation of the $\boldsymbol{\theta}$ is given by

$$\boldsymbol{\theta} = \arg\max \log P(\mathcal{D}') = \sum_n \log p(\boldsymbol{x}_n, z_n)$$

Parameter estimation for GMMs: complete data

The *complete* likelihood is decomposable

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_n \log p(z_n)p(\boldsymbol{x}_n|z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n)p(\boldsymbol{x}_n|z_n)$$

where we have grouped data by its values z_n. Let us introduce a binary variable $\gamma_{nk} \in \{0, 1\}$ to indicate whether $z_n = k$. We then have

Parameter estimation for GMMs: complete data

The *complete* likelihood is decomposable

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_n \log p(z_n)p(\boldsymbol{x}_n|z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n)p(\boldsymbol{x}_n|z_n)$$

where we have grouped data by its values z_n. Let us introduce a binary variable $\gamma_{nk} \in \{0, 1\}$ to indicate whether $z_n = k$. We then have

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \log p(z = k)p(\boldsymbol{x}_n|z = k)$$

We use a "dummy" variable z to denote all the possible values cluster assignment values for $\boldsymbol{x}_n$

$\mathcal{D}'$ specifies this value in the complete data setting

Parameter estimation for GMMs: complete data

From our previous discussion, we have

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \left[\log \omega_k + \log N(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]$$

Regrouping, we have

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \log \omega_k + \sum_k \left\{ \sum_n \gamma_{nk} \log N(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Parameter estimation for GMMs: complete data

From our previous discussion, we have

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \left[\log \omega_k + \log N(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]$$

Regrouping, we have

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \log \omega_k + \sum_k \left\{ \sum_n \gamma_{nk} \log N(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

The term inside the braces depends on k-th component's parameters. It can be shown that the MLE is:

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

What's the intuition?

Intuition

Since γ_{nk} is binary, the previous solution is nothing but

- For ω_k: count the number of data points whose z_n is k and divide by the total number of data points (note that $\sum_k \sum_n \gamma_{nk} = N$)
- For $\boldsymbol{\mu}_k$: get all the data points whose z_n is k, compute their mean
- For $\boldsymbol{\Sigma}_k$: get all the data points whose z_n is k, compute their covariance matrix

This intuition is going to help us to develop an algorithm for estimating $\boldsymbol{\theta}$ when we do not know z_n (incomplete data).

Parameter estimation for GMMs: incomplete data

When z_n is not given, we can guess it via the posterior probability

$$p(z_n = k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^{K} p(\boldsymbol{x}_n|z_n = k')p(z_n = k')}$$

Parameter estimation for GMMs: incomplete data

When z_n is not given, we can guess it via the posterior probability

$$p(z_n = k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^{K} p(\boldsymbol{x}_n|z_n = k')p(z_n = k')}$$

To compute the posterior probability, we need to know the parameters $\boldsymbol{\theta}$!

Let's pretend we know the value of the parameters so we can compute the posterior probability.

How is that going to help us?

Estimation with soft γ_{nk}

We define $\gamma_{nk} = p(z_n = k|\boldsymbol{x}_n)$

Estimation with soft γ_{nk}

We define $\gamma_{nk} = p(z_n = k|\boldsymbol{x}_n)$

- Recall that γ_{nk} should be binary
- Now it's a "soft" assignment of $\boldsymbol{x}_n$ to k-th component
- Each $\boldsymbol{x}_n$ is assigned to a component fractionally according to $p(z_n = k|\boldsymbol{x}_n)$

Estimation with soft γ_{nk}

We define $\gamma_{nk} = p(z_n = k|\boldsymbol{x}_n)$

- Recall that γ_{nk} should be binary
- Now it's a "soft" assignment of $\boldsymbol{x}_n$ to k-th component
- Each $\boldsymbol{x}_n$ is assigned to a component fractionally according to $p(z_n = k|\boldsymbol{x}_n)$

We now get the same expression for the MLE as before!

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

But remember, we're 'cheating' by using $\boldsymbol{\theta}$ to compute γ_{nk}!

Iterative procedure

We can alternate between estimating γ_{nk} and using the estimated γ_{nk} to compute the parameters (same idea as with K-means!)

- Step 0: initialize $\boldsymbol{\theta}$ with some values (random or otherwise)
- Step 1: compute γ_{nk} using the current $\boldsymbol{\theta}$
- Step 2: update $\boldsymbol{\theta}$ using the just computed γ_{nk}
- Step 3: go back to Step 1

Questions:

- Is this procedure reasonable, i.e., are we optimizing a sensible criteria?
- Will this procedure converge?

The answers lie in the *EM algorithm* — a powerful procedure for model estimation with unknown data.

Summary

Clustering

- Group similar instances
- K-means
  - Minimize a cost function that measures the sum of squared distances from the cluster prototypes.
  - Iterative algorithm for minimizing the cost function.
- Variants: K-medoids
- Probabilistic interpretation of K-means: Gaussian Mixture Model
- Can define a number of mixture models for other kinds of data.