# $K$-means Clustering and Gaussian Mixture Models

ETH Zürich

2–3 April 2020

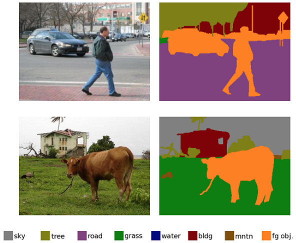
## $K$-means Clustering

## The clustering problem

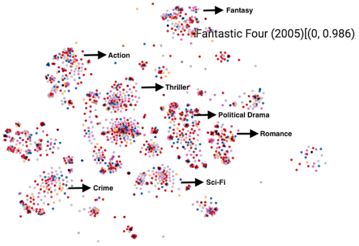
- Also known as *vector quantization*, depending on the application.
- Consider $N$ data points in a $D$-dimensional space, i.e. each data point is a $D$-dimensional vector $\boldsymbol{x}_n$, $n = 1, \ldots, N$.
- Our goal is to partition the data set into $K$ clusters.
- In other words, find $K$ representative vectors (centroids) $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$, one for each cluster, that best fit the data according to some distance metric.

# (Some) applications of clustering

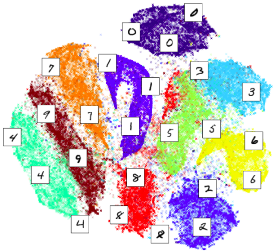
Compression



Semantic segmentation



Topic modeling


Pattern recognition

## $K$-means

One of the many vector quantization algorithms.

- Arguably the most famous and the simplest
- The distance metric is the *squared* Euclidean distance
- **Not** the Euclidean distance, which results in another algorithm ($K$-medoids)

### Objective

Minimize the cost function

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{k,n} \|\boldsymbol{x}_n - \boldsymbol{u}_k\|_2^2$$

- Data points: $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^D$
- Centroids: $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K \in \mathbb{R}^D$
- Assignments: $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N \in \mathbb{R}^K$ (with $z_{k,n} := (\boldsymbol{z}_n)_k$)

## $K$-means constraints

### Hard assignment constraints

Each point $\boldsymbol{x}_n$ is assigned to exactly one cluster:

- $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N \in \{0, 1\}^K$
- $\sum_{k=1}^{K} z_{k,n} = 1, \ \forall n \in \{1, \ldots, N\}$

### In practice

- $K$-means builds a dictionary that maps code words to points and vice versa.
- The assignment matrix $\mathbf{Z}$ can be just implemented as a list of indices.

## Challenges

- The objective function $J$ is **non-convex** and assignments are discrete

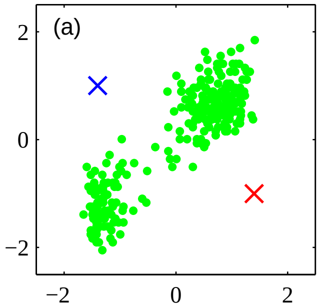
$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{k,n} \|\boldsymbol{x}_n - \boldsymbol{u}_k\|_2^2$$

- Finding the global optimum is NP-Hard
  - Only possible by brute-forcing all assignments
  - Exception: 1D data (dynamic programming solution)

- In practice: local minima are good enough.

## $K$-means algorithm (initialization)

1. Initialize centroids $\boldsymbol{u}_1^{(0)}, \ldots, \boldsymbol{u}_K^{(0)}$ and $t \leftarrow 1$.

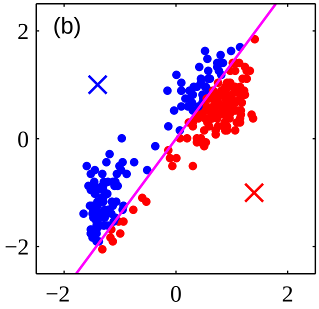


## $K$-means algorithm (E step)

2. **Cluster assignment (assign points to nearest centroid).**

$$k^*(\boldsymbol{x}_n) = \underset{k \in \{1,\ldots,K\}}{\arg\min} \|\boldsymbol{x}_n - \boldsymbol{u}_k^{(t-1)}\|_2^2, \ \forall n \in \{1, \ldots, N\}$$

$$z_{j,n}^{(t)} = \begin{cases} 1 & , \text{ if } j = k^*(\boldsymbol{x}_n) \\ 0 & , \text{ otherwise} \end{cases} \quad , \ \forall n \in \{1, \ldots, N\}$$

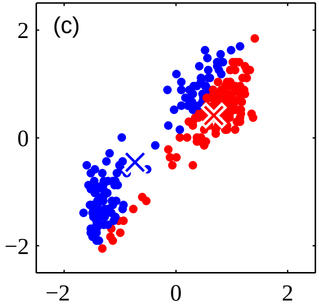


## $K$-means algorithm (M step)

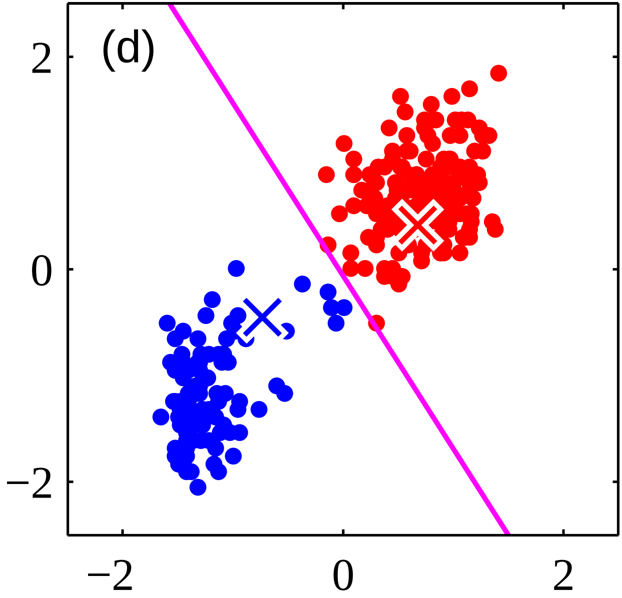
3. **Centroid update.**

$$\boldsymbol{u}_k^{(t)} = \frac{\sum_{n=1}^{N} z_{k,n}^{(t)} \boldsymbol{x}_n}{\sum_{n=1}^{N} z_{k,n}^{(t)}}, \ \forall k \in \{1, \ldots, K\}$$

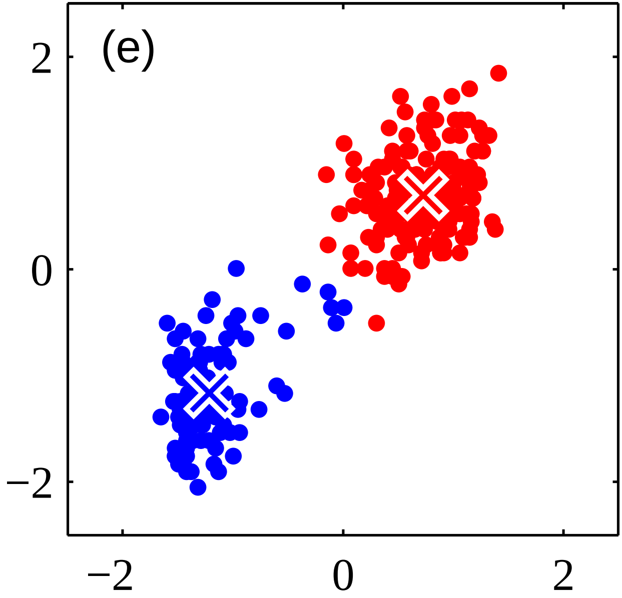
4. If termination condition ($\boldsymbol{u}_k^{(t)} = \boldsymbol{u}_k^{(t-1)}, \ \forall k$) is not met, $t \leftarrow t + 1$ and go to step 2.



## *K*-Means: Second E-Step



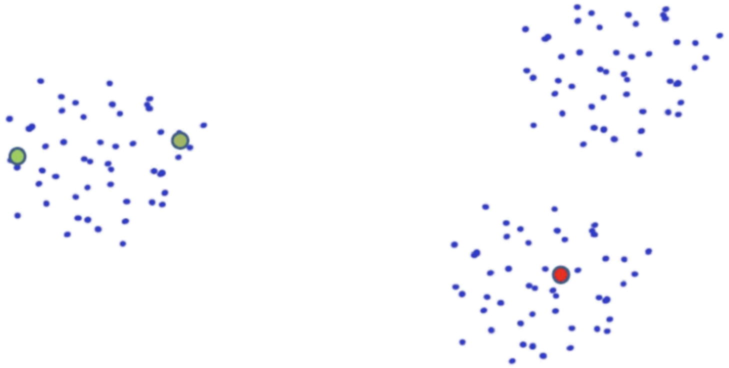
## $K$-Means: Second M-Step



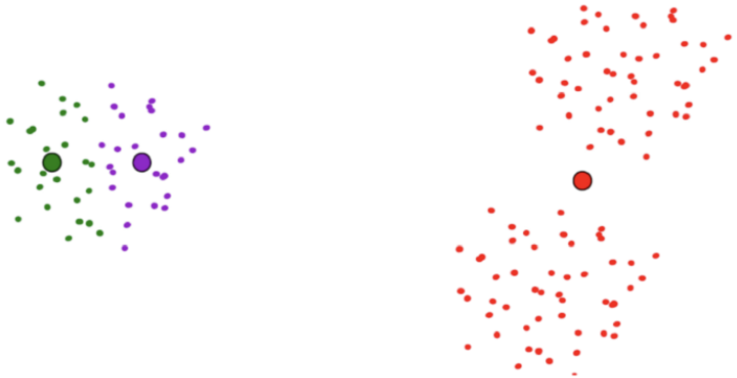
## Practical considerations

- Convergence to local minimum **guaranteed**
- Quadratic convergence rate
  - Equivalent to Newton's method
  - In principle, $J$ can also be optimized via (stochastic) gradient descent
- Computational cost: $\mathcal{O}(nkd)$ per iteration
- Issues: convergence to poor minima (less likely with good initialization), empty clusters. A mitigation is to take the best result out of multiple runs.

## Bad initialization



## Bad initialization



## $K$-Means derivation / convergence proof

### Strategy

Prove that steps **2** (E step) and **3** (M step) always result in a decrease (or no change) of the objective function $J$.

- E step: cluster centroids are fixed, assignments change
- M step: cluster centroids change, assignments are fixed
- What is the minimizer (optimal strategy) of each step?

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{k,n} \|\boldsymbol{x}_n - \boldsymbol{u}_k\|_2^2$$

## $K$-Means derivation / convergence proof

### E step

Objective function $J$ minimized by definition, since we assign each point to the nearest centroid.

$$k^*(\boldsymbol{x}_n) = \underset{k \in \{1,\ldots,K\}}{\arg\min} \|\boldsymbol{x}_n - \boldsymbol{u}_k^{(t-1)}\|_2^2, \ \forall n \in \{1, \ldots, N\}$$

$$z_{j,n}^{(t)} = \begin{cases} 1 & \text{, if } j = k^*(\boldsymbol{x}_n) \\ 0 & \text{, otherwise} \end{cases}, \ \forall n \in \{1, \ldots, N\}$$

## $K$-Means derivation / convergence proof

### M step

Assuming that assignments $z_{k,n}$ are fixed (i.e. constants), how do we find the optimal cluster centroids $\boldsymbol{u}_k$?

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{k,n} \|\boldsymbol{x}_n - \boldsymbol{u}_k\|_2^2$$

**Strategy:** derive gradient w.r.t. centroid $\boldsymbol{u}_k$, set it to zero, and recover closed-form solution.
If we are out of luck (spoiler: we aren't), we could still use gradient descent.

## $K$-Means derivation / convergence proof

M step

$$\nabla_{\boldsymbol{u}_k}\left(\sum_{n=1}^{N}\sum_{k=1}^{K} z_{k,n}\|\boldsymbol{x}_n-\boldsymbol{u}_k\|_2^2\right) = -2\sum_{n=1}^{N} z_{k,n}(\boldsymbol{x}_n-\boldsymbol{u}_k) \overset{!}{=} 0$$

$$\Rightarrow \sum_{n=1}^{N} z_{k,n}\boldsymbol{x}_n - \boldsymbol{u}_k\sum_{n=1}^{N} z_{k,n} = 0 \quad \Rightarrow \quad \boldsymbol{u}_k = \frac{\sum_{n=1}^{N} z_{k,n}\boldsymbol{x}_n}{\sum_{n=1}^{N} z_{k,n}}$$

To show that this is effectively a minimizer, we should also check that the objective function is convex w.r.t. $\boldsymbol{u}_k$ (given $\mathbf{Z}$ constant).

## $K$-Means convergence (final)

- The value of the objective function $J$ can only decrease or stay equal at each step
- Therefore, the algorithm converges
- It must also terminate at some point, since cluster assignments are finite

## $K$-Means as a matrix factorization problem

$K$-Means solves the matrix factorization problem:

$$\min \|\mathbf{X} - \mathbf{UZ}\|_F^2$$

where $\mathbf{Z}$ is an indicator matrix.

Proof strategy

Show that the formulation above is equivalent to the original objective function $J$ under the constraints of $\mathbf{Z}$

$$\|\mathbf{X} - \mathbf{UZ}\|_F^2 \overset{!}{=} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{k,n} \|\boldsymbol{x}_n - \boldsymbol{u}_k\|_2^2 = J$$

## *K*-Means as a matrix factorization problem

Let's expand the matrix norm...

$$\|\mathbf{X} - \mathbf{UZ}\|_F^2 = \sum_{d=1}^{D} \sum_{n=1}^{N} \left( x_{d,n} - \sum_{k=1}^{K} u_{d,k} z_{k,n} \right)^2 \tag{1}$$

$$= \sum_{d=1}^{D} \sum_{j=1}^{N} \left( \sum_{k=1}^{K} z_{k,n} (x_{d,n} - u_{d,k}) \right)^2 \tag{2}$$

$$= \sum_{d=1}^{D} \sum_{j=1}^{N} \sum_{k=1}^{K} z_{k,n}^2 (x_{d,n} - u_{d,k})^2 \tag{3}$$

$$= \sum_{k=1}^{K} \sum_{j=1}^{N} z_{k,n}^2 \sum_{d=1}^{D} (x_{d,n} - u_{d,k})^2 \tag{4}$$

$$= \sum_{k=1}^{K} \sum_{n=1}^{N} z_{k,n} \|\mathbf{x}_n - \mathbf{u}_k\|_2^2 = J \tag{5}$$

In (2) and (3), $z_{k,n}$ is 1 for only one $k$. In (5), $z_{k,n}^2 = z_{k,n}$ (binary).

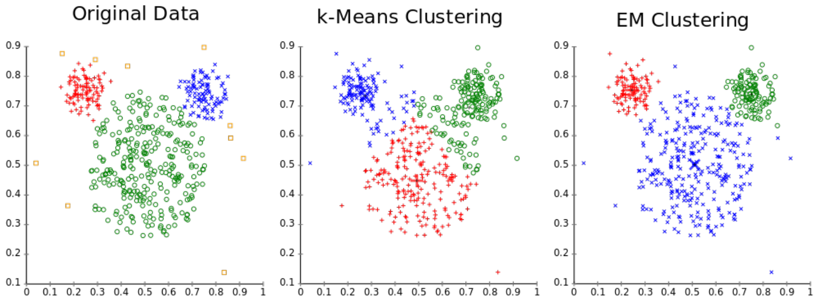
## Gaussian mixture models

## *K*-means vs GMM

*K*-means limitations (or features?):

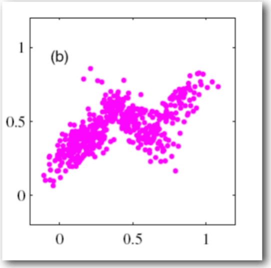
- Hard cluster assignments
- Spherical clusters with equal variance

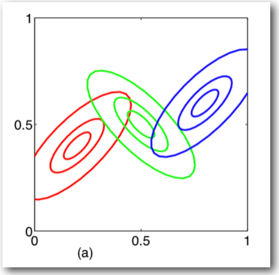
Different cluster analysis results on "mouse" data set:



## What if...

Our data looks like:





## Data vs. Distribution

- Data: input
- Distribution: model assumption

- ML methods usually make some general assumption about the distribution (e.g. a parametric family) then try to obtain ("infer") the specifics from the data available.

- Example:
  - **Modeling step:** Assume a Gaussian distribution as model (parameterized by mean and variance)
  - **Inference Step:** Estimate parameters mean & variance from data.
- Gaussian Mixture Models can be intuitively understood as generative models (you can sample from a GMM)

## Gaussian Mixture Models

Assume data is generated from a weighted mixture of $K$ Gaussian distributions:

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

- Normalization and positivity require: $\pi_k \geq 0, \; \sum_{k=1}^{K} \pi_k = 1$

### Generation Process

- Sample $k$ with probability $\pi_k$.
- Sample $x$ with probability $\mathcal{N}(\boldsymbol{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.

### Mixing Coefficients

The mixing coefficients $(\pi_k)$ can be interpreted as a prior prob.:

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k) p(x \mid k)$$

## Gaussian Distribution (d-D)

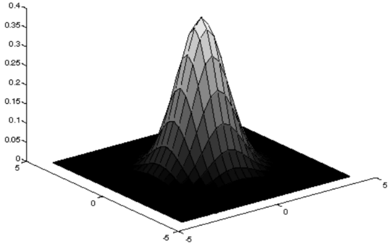
- Random vector $\boldsymbol{X} = (X_1, \ldots, X_d)$ with $\mathcal{X} = \mathbb{R}^d$

- Probability density function

$$p(\boldsymbol{x}) := \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left( -\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right)$$

- $\mathrm{E}[\boldsymbol{X}] = \boldsymbol{\mu}$

- $\Sigma$ is the covariance matrix of $\boldsymbol{X}$ and $|\Sigma|$ is its determinant.



## GMM - Parameters

K mixture components with parameters (for $k = 1, \dots, K$):

- $\boldsymbol{\mu}_k$: mean of the $k$-th component (similar to centroid $\boldsymbol{u}_k$ in $K$-means)
- $\boldsymbol{\Sigma}_k$: covariance matrix of the $k$-th component
- $\pi_k$: mixture weight of the $k$-th component

Maximum likelihood estimation?

## GMM - Objective

The likelihood of all the data is:

$$p(\boldsymbol{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left( \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

Maximize the log-likelihood of the Gaussian mixture model:

$$L(\boldsymbol{X}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) := \ln p(\boldsymbol{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

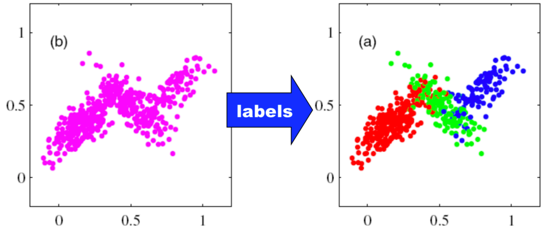
**Log of a sum !**
It is really hard to optimize with respect to $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$. We need to find another method to compute them!

## GMM - Latent Variables

- Let's introduce new variables $z_k$, called **latent variables**, that tell us which points come from which gaussian.
  - Complete data
- For each data point $\mathbf{x}$:

$$z_k = \begin{cases} 1 & \text{if } \mathbf{x} \text{ comes from k-th Gaussian component} \\ 0 & \text{otherwise} \end{cases}$$

- Note: $\sum_{k=1}^{K} z_k = 1$ and $p(z_k = 1) = \pi_k$.

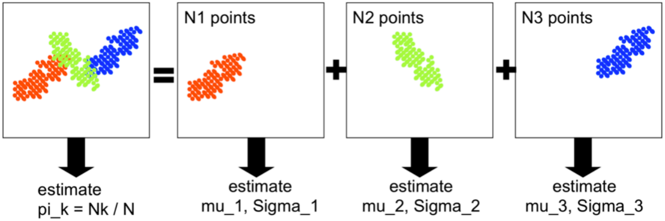


## GMM - Latent Variables

- For each data point we define $\mathbf{z} = (z_1, z_2, \ldots, z_K)$, where $z_i = 0$ for all $i \neq k$, and $z_k = 1$.
- Then the conditional distribution of $\boldsymbol{x}$ given a $\mathbf{z}$ is a Gaussian:

$$p(\boldsymbol{x}|\mathbf{z}) = p(\boldsymbol{x}|z_k = 1) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Given $\mathbf{z}$ for each datapoint, the parameter inference is easy!



## Complete Log-likelihood

- Remember: the likelihood for one data point **x** is:

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

- The **complete** likelihood for one data point **x** is:

$$p(\mathbf{x}, \mathbf{z} \mid \pi_k, \boldsymbol{\mu}, \Sigma) = \prod_{k=1}^{K} [\pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \Sigma_k)]^{z_k}$$

- The complete log-likelihood for the dataset **X** then is:

$$\log\ p(\mathbf{X}, \mathbf{Z} \mid \pi_k, \boldsymbol{\mu}, \Sigma) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \Big( \log \pi_k + \log \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k) \Big)$$

## The EM Algorithm - Key Idea

- The **EM algorithm** proposes instead to look at the expected complete log-likelihood:

$$E_Z\Big[\log p(\mathbf{X}, \mathbf{Z} \mid \pi_k, \boldsymbol{\mu}, \Sigma)\Big] = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \Big( \log \pi_k + \log \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k) \Big)$$

- $\gamma_{nk}$ is the posterior probability of the latent variables.

$$\gamma_{nk} = E(z_{nk}) = p(z_{nk} = 1) = p(z_k = 1 \mid \mathbf{x}_n)$$

- **Remember:** the expectation of a binary variable is the probability that it is equal to 1.

## The EM Algorithm - Lower Bound

- Let's find a lower-bound of the log-likelihood:

$$\log p(\mathbf{X};\theta) = \sum_{n=1}^{N} \log \left[ \sum_{k=1}^{K} \pi_k p_{\theta_k}(\mathbf{x}_n) \right] = \sum_{n=1}^{N} \log \left[ \sum_{k=1}^{K} q_{nk} \frac{\pi_k \ p_{\theta_k}(\mathbf{x}_n)}{q_{nk}} \right]$$
$$\geq \sum_{n=1}^{N} \sum_{k=1}^{K} q_{nk} \left[ \log p_{\theta_k}(\mathbf{x}_n) + \log \pi_k - \log q_{nk} \right] = \mathcal{L}(\mathbf{X};\theta)$$

where $q_{nk}$ is any distribution s.t. $\sum_{k=1}^{K} q_{nk} = 1$.

- In the last step, we used Jensen's inequality

$$\log \left( \sum_k \alpha_k x_k \right) \geq \sum_k \alpha_k \log(x_k)$$

## The EM Algorithm - E-Step

- Given that we cannot directly maximize the log-likelihood, we maximize its lower bound.
- The next step is to find the optimal **q** distribution that maximizes this lower bound
- If this lower bound is the tightest, it represents a good approximation of the log-likelihood
- Next slide: derivation

## The EM Algorithm - E-Step derivation

**Strategy:** compute gradient w.r.t. $\mathbf{q}_k$, set it to 0, and try to find a closed-form solution (as usual).

- This time we also need to add a Lagrange multiplier to enforce $\sum_k \mathbf{q}_k = 1$

Objective (for a single data point $\mathbf{x}$):

$$\max_q \left\{ \sum_{k=1}^{K} q_k \left[\log p_{\theta_k}(\mathbf{x}) + \log \pi_k - \log q_k\right] + \lambda \left( \left( \sum_{k=1}^{K} q_k \right) - 1) \right) \right\}$$

$$\nabla_{q_k} = \log p_{\theta_k}(\mathbf{x}) + \log \pi_k - \log q_k - \frac{q_k}{q_k} + \lambda \overset{!}{=} 0$$

## The EM Algorithm - E-Step derivation

(copied over from previous slide)

$$\nabla_{q_k} = \log p_{\theta_k}(\mathbf{x}) + \log \pi_k - \log q_k - \frac{q}{q} + \lambda \overset{!}{=} 0$$

Let's continue...

$$\log q_k^* = \log p_{\theta_k}(\mathbf{x}) + \log \pi_k - 1 + \lambda \quad \Rightarrow \quad q_k^* = p_{\theta_k}(\mathbf{x})\,\pi_k\,e^{\lambda-1}$$

To get rid of the term with $\lambda$, we take the sum of both sides and exploit the normalization property:

$$\underbrace{\sum_{k=1}^{K} q_k}_{=1} = \sum_{k=1}^{K} p_{\theta_k}(\mathbf{x})\,\pi_k\,e^{\lambda-1} \quad \Rightarrow \quad e^{\lambda-1} = \frac{1}{\sum_{k=1}^{K} p_{\theta_k}(\mathbf{x})\,\pi_k}$$

## The EM Algorithm - E-Step derivation

Finally, you get the result you saw in the lecture (single point $\mathbf{x}$):

$$q_k^* = \frac{\pi_k \, p_{\theta_k}(\mathbf{x})}{\sum_{l=1}^{K} \pi_l \, p_{\theta_l}(\mathbf{x})} = p(z_k = 1 \mid \mathbf{x})$$

Or, for each point $\mathbf{x}_n$:

$$q_{nk}^* = \frac{\pi_k \, p_{\theta_k}(\mathbf{x}_n)}{\sum_{l=1}^{K} \pi_l \, p_{\theta_l}(\mathbf{x}_n)} = p(z_k = 1 \mid \mathbf{x}_n) = \gamma_{nk}$$

- The optimal $q$–distribution is equal to the posterior probability of the latent variables.

## The EM Algorithm - M-Step

- Now we maximize the lower bound w.r.t. the parameters $(\pi_k, \boldsymbol{\mu}, \Sigma)$, given $\gamma_{nk}$ fixed.
- Let's have a closer look at the lower bound with optimal $q$:

$$\mathcal{L}(\mathbf{X};\theta) = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma_{nk}\left[\log p_{\theta_k}(\mathbf{x}_n) + \log \pi_k - \log \gamma_{nk}\right]$$

$$= E_Z\big[\log p(\mathbf{X},\mathbf{Z} \mid \theta)\big] - \underbrace{\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma_{nk}\log\gamma_{nk}}_{\text{constant}}$$

- Hence optimizing the lower bound $\mathcal{L}(\mathbf{X};\theta)$ w.r.t. $\theta$ is equal to maximizing the **expected complete data log-likelihood** (same gradient).

## EM Algorithm: M-Step

**Strategy:** same as before. Compute gradient w.r.t. $\pi_k, \boldsymbol{\mu}, \Sigma$, set them to 0, find closed-form solution (if it exists). Where needed ($\pi_k$), enforce the normalization constraint with a Lagrange multiplier.

$$\log p(\mathbf{X}; \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \Big( \log \pi_k + \log \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k) \Big)$$

Example

Let's try to derive the optimal mixing coefficients $\pi_k$ (next slide)

## Example: optimal mixing coefficients

$$\max_{\pi_k} \left\{ \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \Big( \log \pi_k + \log \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k) \Big) + \lambda \left( \left( \sum_{k=1}^{K} \pi_k \right) - 1 \right) \right\}$$

$$\nabla_{\pi_k} = \sum_{n=1}^{N} \gamma_{nk} \frac{1}{\pi_k} + \lambda \overset{!}{=} 0 \;\Rightarrow\; \frac{1}{\pi_k} \sum_{n=1}^{N} \gamma_{nk} = -\lambda \;\Rightarrow\; \pi_k^* = \frac{\sum_{n=1}^{N} \gamma_{nk}}{-\lambda}$$

As before, to find $\lambda$, let's sum on both sides...

$$\underbrace{\sum_{k=1}^{K} \pi_k}_{=1} = \sum_{k=1}^{K} \frac{\sum_{n=1}^{N} \gamma_{nk}}{-\lambda} \;\Rightarrow\; -\lambda = \underbrace{\sum_{n=1}^{N} \underbrace{\sum_{k=1}^{K} \gamma_{nk}}_{=1}}_{N} \;\Rightarrow\; \pi_k^* = \frac{\sum_{n=1}^{N} \gamma_{nk}}{N}$$

## The EM algorithm - Overview

1. Initialize $\pi_k^{(0)}$, $\boldsymbol{\mu}_k^{(0)}$, $\boldsymbol{\Sigma}_k^{(0)}$ for $k = 1, \dots, K$ and $t \leftarrow 1$.
2. **E-step.** Update latent variables:

$$\gamma_{nk} := \frac{\pi_k^{(t-1)} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k^{(t-1)}, \boldsymbol{\Sigma}_k^{(t-1)})}{\sum_{j=1}^{K} \pi_j^{(t-1)} \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j^{(t-1)}, \boldsymbol{\Sigma}_j^{(t-1)})}$$

3. **M-step.** Update parameters of the clusters:

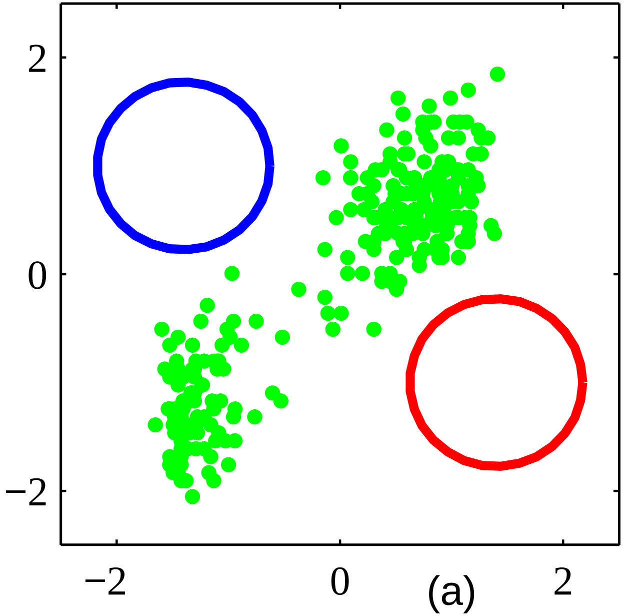
$$\begin{aligned}
\boldsymbol{\mu}_k^{(t)} &:= \frac{\sum_{n=1}^{N} \gamma_{nk} \boldsymbol{x}_n}{\sum_{n=1}^{N} q_{kn}} \\
\boldsymbol{\Sigma}_k^{(t)} &:= \frac{1}{\sum_{n=1}^{N} \gamma_{nk}} \sum_{n=1}^{N} \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})^T \\
\pi_k^{(t)} &:= \frac{1}{N} \sum_{n=1}^{N} \gamma_{nk}
\end{aligned}$$

4. If termination condition is not met, $t := t + 1$ and go to step 2.

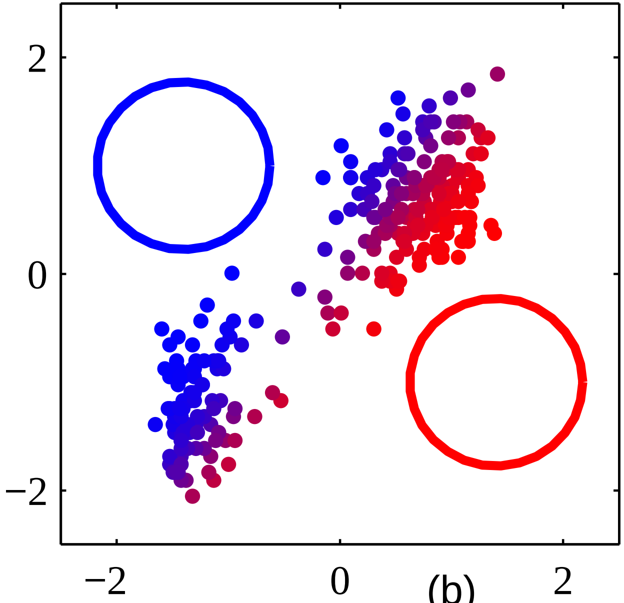
## $K$-means vs. mixture models

- $K$-means is a special case of GMMs!
- $K$-means
  - Hard cluster assignments
  - Spherical clusters with uniform prior
  - Fast runtime (can be used to initialize a mixture model)
- Gaussian mixture models
  - Soft cluster assignments $\leftrightarrow$ probabilities of assignments
  - Each cluster has its own covariance ($\boldsymbol{\Sigma_k}$) and "weight" ($\pi_k$)
  - Slower runtime
- One is not necessarily better than the other. They both have their use cases.

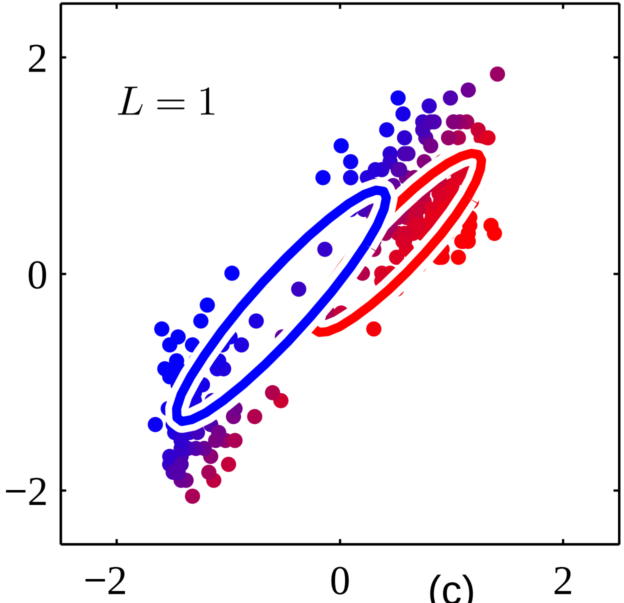
## GMM: Initial configuration



## GMM: First E-Step

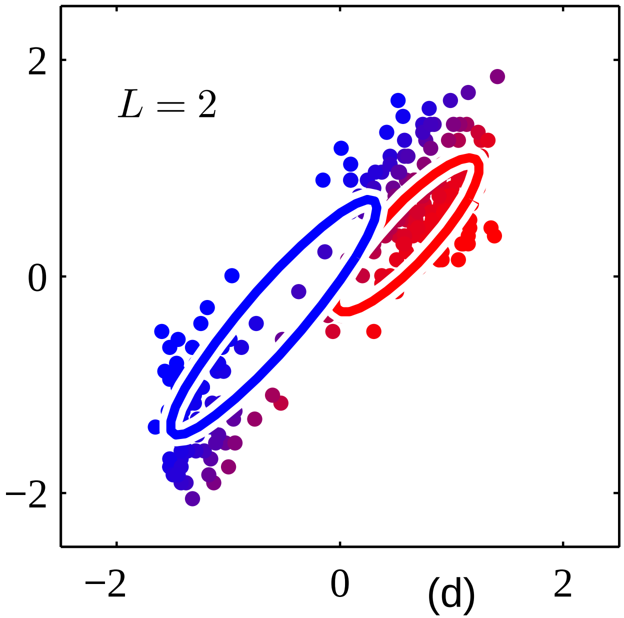


## GMM: First M-Step

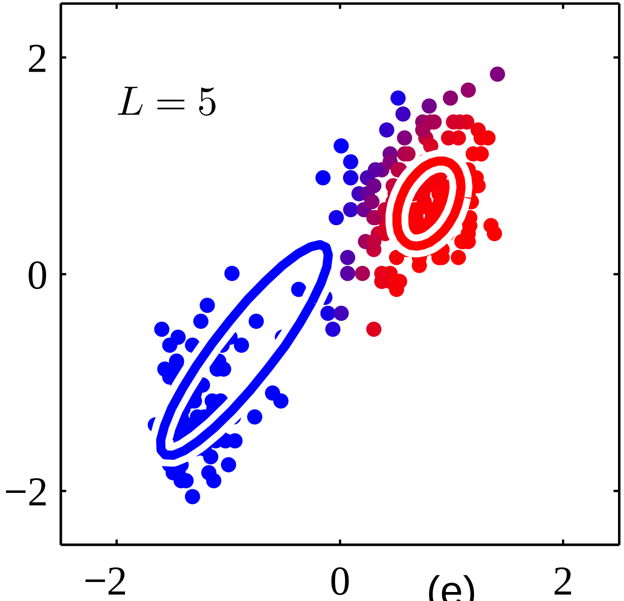


(c)

## GMM: Two EM cycles



## GMM: Five EM cycles



(e)