# ECE276A: Sensing & Estimation in Robotics
# Lecture 5: Unsupervised Learning


Instructor:

Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistants:

Qiaojun Feng: qif007@eng.ucsd.edu
Tianyu Wang: tiw161@eng.ucsd.edu
Ibrahim Akbar: iakbar@eng.ucsd.edu
You-Yi Jau: yjau@eng.ucsd.edu
Harshini Rajachander: hrajacha@eng.ucsd.edu

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

# Gaussian (Mixture) Discriminant Analysis

- A generative model that uses a **Gaussian Mixture** with $J$ components to model $p(\mathbf{x}_i \mid y_i, \omega)$:

$$p(\mathbf{y}, X \mid \omega, \theta) = p(\mathbf{y} \mid \theta) p(X \mid \mathbf{y}, \omega) = p(\mathbf{y} \mid \theta) \prod_{i=1}^{n} p(\mathbf{x}_i \mid y_i, \omega)$$

$$p(\mathbf{y} \mid \theta) := \prod_{i=1}^{n} \prod_{k=1}^{K} \theta_k^{\mathbb{1}\{y_i = k\}} \quad p(\mathbf{x}_i \mid y_i = k, \omega) := \sum_{j=1}^{J} \alpha_{kj} \phi\left(\mathbf{x}_i; \mu_{kj}, \Sigma_{kj}\right)$$

- Training via MLE: $\max_{\theta, \omega} p(\mathbf{y}, X \mid \theta, \omega)$
  - The MLE of $\theta$ can be obtained via the softmax trick and differentiation as we saw for the single-Gaussian discriminant analysis
  - Obtaining MLE estimates for $\omega := \{\alpha_{kj}, \mu_{kj}, \Sigma_{kj}\}$ is no longer straight forward because $\log \sum_{j=1}^{J} \alpha_{kj} \phi\left(\mathbf{x}_i; \mu_{kj}, \Sigma_{kj}\right)$ is not convex/concave
  - Also, need to ensure that $\sum_{j=1}^{J} \alpha_{kj} = 1$, $\forall k$.

# Data Log Likelihood

- $$\log p(\mathbf{y}, X \mid \omega, \theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{y_i = k\} \log \theta_k + \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}\{y_i = k\} \log \left( \sum_{j=1}^{J} \alpha_{kj} \phi\left(\mathbf{x}_i; \mu_{kj}, \Sigma_{kj}\right) \right)$$
- Focus on max wrt $\omega := \{\alpha_{kj}, \mu_{kj}, \Sigma_{kj}\}$; the first term can be ignored
- To simplify notation, let $D_k := \{(\mathbf{x}_i, y_i) \mid y_i = k\} \subseteq D$ and define:

$$\lambda(X, \omega) := \sum_{k=1}^{K} \sum_{\mathbf{x} \in D_k} \log \left( \sum_{j=1}^{J} \alpha_{kj} \phi\left(\mathbf{x}; \mu_{kj}, \Sigma_{kj}\right) \right)$$

# Gaussian Mixtures

- Gaussian Mixtures are well suited for modeling clusters of points:
  - each cluster is assigned a Gaussian
  - the mean is somewhere in the middle of the cluster
  - the covariance measures the cluster spread
- **Sampling** from a Gaussian Mixture:
  - Draw an integer between 1 and $J$ with probability $\alpha_{kj}$
  - Draw a vector $\mathbf{x}$ from the $j$-th Gaussian pdf $\phi(\mathbf{x}; \mu_{kj}, \Sigma_{kj})$
- It is useful to understand the meaning of $q_k(j, \mathbf{x}) := \alpha_{kj}\phi(\mathbf{x}; \mu_{kj}, \Sigma_{kj})$
- Given class $k$, $q_k(j, \mathbf{x})d\mathbf{x}$ is the joint probability of drawing component $j$ and data point $\mathbf{x}$ in a volume $d\mathbf{x}$ around it
- The **membership probability** of data point $\mathbf{x}$ is the conditional probability of having selected component $j$ given $\mathbf{x}$:

$$r_k(j \mid \mathbf{x}) := \frac{q_k(j, \mathbf{x})}{\sum_{l=1}^{J} q_k(l, \mathbf{x})} \qquad \sum_{j=1}^{J} r_k(j \mid \mathbf{x}) = 1$$

## Local maxima of $\lambda(X, \omega)$

- Maxima of $\sum_{k=1}^{K} \sum_{\mathbf{x} \in D_k} \log \left( \sum_{j=1}^{J} \alpha_{kj} \phi\left(\mathbf{x}; \mu_{kj}, \Sigma_{kj}\right) \right)$ occur at critical points

- $$\begin{aligned} \frac{d}{d\mu_{lm}} \lambda(X, \omega) &= \sum_{\mathbf{x} \in D_l} \frac{\alpha_{lm}}{\sum_{j=1}^{J} \alpha_{lj} \phi\left(\mathbf{x}; \mu_{lj}, \Sigma_{lj}\right)} \frac{d}{d\mu_{lm}} \phi\left(\mathbf{x}; \mu_{lm}, \Sigma_{lm}\right) \\ &= \sum_{\mathbf{x} \in D_l} r_l(m \mid \mathbf{x})(\mu_{lm} - \mathbf{x})^T \Sigma_{lm}^{-1} \end{aligned}$$

- $$\frac{d}{d\Sigma_{lm}} \lambda(X, \omega) = \frac{1}{2} \sum_{\mathbf{x} \in D_l} r_l(m \mid \mathbf{x}) \left( \Sigma_{lm}^{-1} (\mu_{lm} - \mathbf{x})(\mu_{lm} - \mathbf{x})^T \Sigma_{lm}^{-1} - \Sigma_{lm}^{-1} \right)$$

- Use **softmax trick** for $\alpha_{kj}$ to handle simplex constraints

$$\begin{aligned} \frac{d}{d\gamma_{lm}} \lambda(X, \omega) &= \sum_{\mathbf{x} \in D_l} \frac{1}{\sum_{j=1}^{J} \alpha_{lj} \phi\left(\mathbf{x}; \mu_{lj}, \Sigma_{lj}\right)} \sum_{j=1}^{J} \frac{d\alpha_{lj}}{d\gamma_{lm}} \phi\left(\mathbf{x}; \mu_{lj}, \Sigma_{lj}\right) \\ &= \sum_{\mathbf{x} \in D_l} \left( r_l(m \mid \mathbf{x}) - \alpha_{lm} \right) \end{aligned}$$

# Local maxima of $\lambda(X, \omega)$

- Setting the previous derivatives to zero, we obtain:

$$\boxed{\alpha_{kj} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k(j \mid \mathbf{x}_i)}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\}}} \quad \boxed{\mu_{kj} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k(j \mid \mathbf{x}_i) \mathbf{x}_i}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k(j \mid \mathbf{x}_i)}}$$

$$\boxed{\Sigma_{kj} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k(j \mid \mathbf{x}_i)(\mathbf{x}_i - \mu_{kj})(\mathbf{x}_i - \mu_{kj})^T}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k(j \mid \mathbf{x}_i)}}$$

- The mixture weights are equal to the sample mean of the membership probabilities $r_k(j \mid \mathbf{x}_i)$ assuming a uniform distribution over $D_k$
- The latter are the sample mean and covariance of the data, weighted by the membership probabilities
- The three equations are coupled through $r_k(j \mid \mathbf{x})$, which depends on $\omega := \{\alpha_{kj}, \mu_{kj}, \Sigma_{kj}\}$, and hence are hard to solve directly
- **Optimization Idea**:
  - start with a guess $\omega^{(0)}$ and use a descent method
  - iterate between updating $r_k(j \mid \mathbf{x}_i)$ and updating $\omega^{(t)}$

# Clustering

- How do we obtain an initial guess $\omega^{(0)} := \left\{\alpha_{kj}^{(0)}, \mu_{kj}^{(0)}, \Sigma_{kj}^{(0)}\right\}$?
- **Clustering** (or vector quantization) is the task of grouping objects in a way that those in the same group (a **cluster**) are more similar (according to a distance metric) to each other than to those in other groups.
- **Unsupervised Learning**: given an *unlabeled* dataset $D = \{\mathbf{x}_i\}_{i=1}^{n}$, the goal is to partition it into $J$ clusters

# $k$-means Algorithm

- The $k$-**means algorithm** is an iterative clustering algorithm that uses **coordinate descent** to solve the following optimization:

$$\min_{\mu, r} C(\mu, r) := \sum_{i=1}^{n} \sum_{j=1}^{J} r_{ij} \|\mu_j - \mathbf{x}_i\|_2^2$$

  - $\mu_j$ are the cluster centroids
  - $r_{ij} := \mathbb{1}_{\{\mathbf{x}_i \text{ is closest to } \mu_j\}}$ are the cluster membership indicators
- It is common to repeat the algorithm several times with different initialization of $\mu_j$
- Since $k$-means is optimizing $\|\cdot\|_2$, it implicitly makes a spherical assumption on the shape of the clusters.

# $k$-means Algorithm

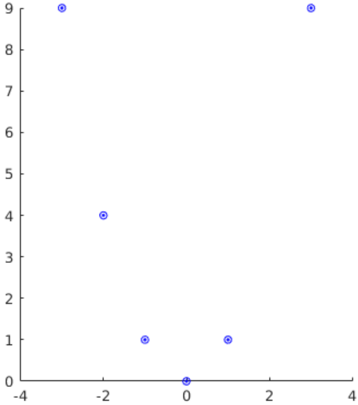
**Algorithm 1** $k$-means clustering

**Input**: unlabeled dataset $D = \{\mathbf{x}_i\}_{i=1}^{n}$, number of clusters $k$
**Output**: cluster centroids $\mu_j$, cluster assignments $\{r_{ij}\}$
**Init**: pick $k$ cluster centroids $\mu_1, \dots, \mu_k$
**repeat**
    # *Assign examples to the nearest centroid*:
        $r_{ij} = 1$, if $j = \arg\min_{l} \|\mu_l - x_i\|_2^2$, and $r_{ij} = 0$, otherwise.
    # *Set each centroid to the mean of the examples assigned to it*:
        $\mu_j = \arg\min_{\mu} C(\mu, r) = \frac{\sum_{i=1}^{n} r_{ij} x_i}{\sum_{i=1}^{n} r_{ij}}$
**until** convergence

# k-means Example

- Consider the following **unlabeled** dataset:

$$X = \begin{bmatrix} -3 & 9 \\ -2 & 4 \\ -1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 3 & 9 \end{bmatrix} \in \mathbb{R}^{n \times d}$$




- Use $k$-means to cluster $X$ into $k = 2$ clusters. Initialize:

$$\mu_1 = \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \qquad \mu_2 = \mathbf{x}_5 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

# k-means Example

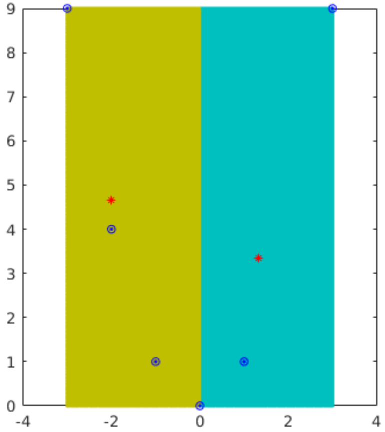
- Assign examples to the nearest centroid, $r_{ij}$:

| $i \backslash j$ | 1 | 2 |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 1 | 0 |
| 3 | 1 | 0 |
| 4 | 0 | 1 |
| 5 | 0 | 1 |
| 6 | 0 | 1 |

- Update the cluster means:

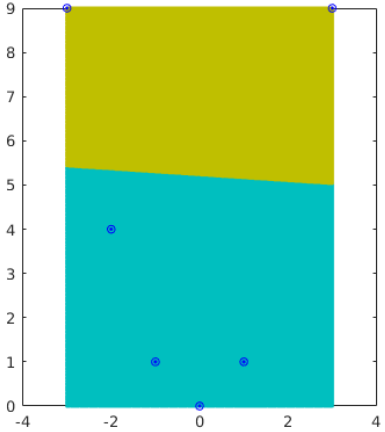
$$\mu_1 = \frac{\sum_{i=1}^{n} r_{i1} x_i}{\sum_{i=1}^{n} r_{i1}} = \begin{bmatrix} -2.00 \\ 4.66 \end{bmatrix}$$

$$\mu_2 = \begin{bmatrix} 1.33 \\ 3.33 \end{bmatrix}$$



# k-means Example

- Repeat until convergence

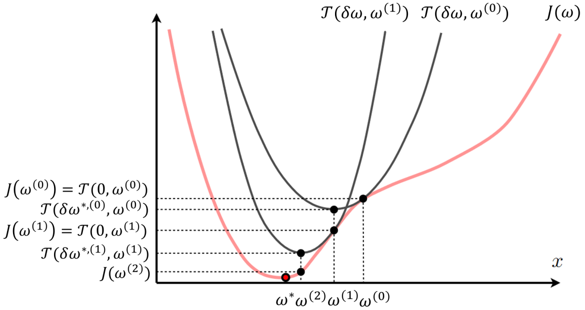


# Expectation Maximization

- Iterative maximization technique based on auxiliary lower bounds
  - Old idea (late 50's) formalized by Dempster, Laird and Rubin in 1977
  - Has two steps: Expectation (E) and Maximization (M)
  - Generalizes $k$-means to probabilistic (soft) cluster assignments
  - Similar to Newton's method but is not restricted to a quadratic approximations of the objective
- Applicable to a wide range of problems:
  - Fitting mixture models
  - Probabilistic latent semantic analysis: produce concepts related to documents and terms (NLP)
  - Learning parts and structure models (vision)
  - Segmentation of layers in video (vision)

# Expectation Maximization

- **Goal**: $\min_\omega J(\omega)$
- $J(\omega)$ is not necessarily convex




- Initialize $\omega^{(0)}$ and iterate:
  E. Construct an auxiliary upper-bound function $\mathcal{T}$ at $\omega^{(t)}$ such that:
  $$J(\omega^{(t)}) = \mathcal{T}(\omega^{(t)}, \omega^{(t)}) \leq \mathcal{T}(\omega, \omega^{(t)})$$
  M. Solve the easier auxiliary minimization to obtain the next point:
  $$\omega^{(t+1)} = \arg\min_\omega \mathcal{T}(\omega, \omega^{(t)})$$
- The properties of $\mathcal{T}$ guarantee that each step gets closer to a local min:
$$J(\omega^{(t)}) = \mathcal{T}(\omega^{(t)}, \omega^{(t)}) \geq \min_\omega \mathcal{T}(\omega, \omega^{(t)}) \geq J(\omega^{(t+1)})$$

# Auxiliary Function

- EM is related to parameter estimation since it can be used to solve:

$$\min_{\omega} J(\omega) \qquad \text{for} \qquad J(\omega) := -\log p(D; \omega)$$

- The above might not be solvable in closed form by setting the gradient $\nabla_{\omega} J(\omega)$ to zero
- EM uses **latent/hidden variables** to construct an auxiliary upper bound $\mathcal{T}(\omega, \omega^{(t)})$ to the negative data log likelihood $J(\omega)$ at a given parameter estimate $\omega^{(t)}$
- The auxiliary upper bound is obtained by applying **Jensen's inequality** to the convex function $-\log(\cdot)$

# Convexity and Jensen's Inequality

- $f : \mathbb{R}^n \to \mathbb{R}$ is **convex** if one of the following holds:
  - $f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}), \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \lambda \in [0,1]$
  - $f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T (\mathbf{y} - \mathbf{x}), \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$
  - $\nabla^2 f(\mathbf{x}) \succeq 0, \quad \forall \mathbf{x} \in \mathbb{R}^n$

- **Jensen's Inequality**: let $Y$ be a random variable and $f : \mathbb{R} \to \mathbb{R}$ be a convex function. Then:

$$f(\mathbb{E}[Y]) \leq \mathbb{E}[f(Y)]$$

- **Example**:
  - $f(x) := -\log(x)$ is convex because $f''(x) = \frac{1}{x^2} > 0$ for $x \in (0, \infty)$
  - Let $Z$ be a discrete random variable with probability mass function $p(z_j) := \mathbb{P}(\{Z = z_j\}) = r_j$ for $j = 1, \ldots, m$
  - Jensen's inequality applied to $f$ and $Y := \frac{Z}{p(Z)}$ shows that:

$$-\log(\mathbb{E}[Y]) = -\log\left(\sum_j r_j \frac{z_j}{r_j}\right) \leq -\sum_j r_j \log\left(\frac{z_j}{r_j}\right) = \mathbb{E}[f(Y)]$$

# Auxiliary Function

- Given the current parameter estimate $\omega^{(t)}$, EM introduces a latent random variable $Z$ with pdf $r(z \mid D; \omega^{(t)})$:

$$
\begin{aligned}
J(\omega) = -\log p(D;\omega) &\underset{\text{of prob.}}{\overset{\text{Total law}}{=\!=\!=}} -\log \int p(D, z; \omega) dz \\
&= -\log \int r(z|D;\omega^{(t)}) \frac{p(D,z;\omega)}{r(z|D;\omega^{(t)})} dz \\
&\underset{\text{inequality}}{\overset{\text{Jensen's}}{\leq}} -\int r(z|D;\omega^{(t)}) \log \frac{p(D,z;\omega)}{r(z|D;\omega^{(t)})} dz \\
&\underset{\text{function}}{\overset{\text{Auxiliary}}{=\!=\!=}} \mathcal{T}(\omega, \omega^{(t)})
\end{aligned}
$$

# Auxiliary Function

- Assuming that $-\log p(D, z; \omega)$ is convex in $\omega$, the **auxiliary function** is convex in $\omega$ for a fixed $r$ and convex in $r$ for a fixed $\omega$ (but **not jointly** convex)
- The local minima of $\mathcal{T}(\omega, \omega^{(t)})$ are local minima of $-\log p(D; \omega)$
- The EM algorithm alternates between

(E step) finding the minimum upper bound to $J(\omega)$ at $\omega^{(t)}$ (which is equivalent to determining the pdf of the latent variable $Z$):

$$\omega^{(t)} = \arg\min_{\eta} \mathcal{T}(\omega^{(t)}, \eta)$$

(M step) minimizing the upper bound $\mathcal{T}$ to update the parameters:

$$\omega^{(t+1)} = \arg\min_{\omega} \mathcal{T}(\omega, \omega^{(t)}) = \arg\min_{\omega} -\int r(z|D; \omega^{(t)}) \log \frac{p(D, z; \omega)}{r(z|D; \omega^{(t)})} dz$$

# M Step Details

$$\min_{\omega} \mathcal{T}(\omega, \omega^{(t)}) = \int r(z|D;\omega^{(t)}) \log \frac{p(D,z;\omega)}{r(z|D;\omega^{(t)})} dz$$

$$= \underbrace{h(r(\cdot \mid D;\omega^{(t)}))}_{\substack{\text{Entropy of } r; \\ \text{does not depend on } \omega}} + \underbrace{\int r(z|D;\omega^{(t)}) \log p(D,z;\omega) dz}_{\substack{\text{Weighted MLE where labeled examples} \\ \{(\mathbf{x}_i, y_i, z_i)\} \text{ are weighted by } r(z_i \mid D;\omega^{(t)})}}$$

- **Differential entropy** of a continuous random variable $X$ with pdf $p$:

$$h(X) := -\int p(x) \log p(x) dx$$

- **Kullback-Leibler (KL) divergence** from pdf $p$ to pdf $q$:

$$d_{\mathcal{KL}}(p||q) := \int p(x) \log \frac{p(x)}{q(x)} dx$$

# E Step Details

- Why is $\omega^{(t)} = \arg\min_{\eta} \mathcal{T}(\omega^{(t)}, \eta)$?

$$-\log p(D;\omega^{(t)}) \leq \mathcal{T}(\omega^{(t)}, \eta) = -\int r(z|D;\eta) \log \frac{r(z \mid D;\omega^{(t)}) p(D;\omega^{(t)})}{r(z|D;\eta)} dz$$
$$= -\log p(D;\omega^{(t)}) + d_{\mathcal{KL}}\left(r(\cdot \mid D;\omega^{(t)}) || r(\cdot \mid D;\eta)\right)$$

- When minimizing the upper bound $\mathcal{T}(\omega^{(t)}, \eta)$ with respect to $\eta$, we are maximizing the similarity between $r(\cdot \mid D;\eta)$ and $r(\cdot \mid D;\omega^{(t)})$
- Choosing $\eta = \omega^{(t)}$ makes the upper bound $\mathcal{T}(\omega, \omega^{(t)})$ **tight**, i.e., it touches the negative log-likelihood function at $\omega^{(t)}$:

$$\mathcal{T}(\omega^{(t)}, \omega^{(t)}) = -\int r(z \mid D;\omega^{(t)}) \log p(D;\omega^{(t)}) dz$$
$$= -\log p(D;\omega^{(t)}) = J(\omega^{(t)})$$

# Auxiliary Function for the GM Log Likelihood

- **Latent variable**: soft cluster assignment $Z$ with pdf $r_k(\cdot \mid \mathbf{x}; \omega^{(t)})$
- Upper-bound the negative Gaussian Mixture log likelihood via Jensen's inequality:

$$
\begin{aligned}
-\lambda(X, \omega) &:= -\sum_{k=1}^{K} \sum_{\mathbf{x} \in D_k} \log \left( \sum_{j=1}^{J} q_k(j, \mathbf{x}; \omega) \right) \\
&\leq -\sum_{k=1}^{K} \sum_{\mathbf{x} \in D_k} \sum_{j=1}^{J} r_k(j \mid \mathbf{x}; \omega^{(t)}) \log \frac{q_k(j, \mathbf{x}; \omega)}{r_k(j \mid \mathbf{x}; \omega^{(t)})} =: \mathcal{T}(\omega, \omega^{(t)})
\end{aligned}
$$

- A theoretical construction only since we already know that the minimum of $\mathcal{T}(\omega^{(t)}, \eta)$ with respect to $\eta$ occurs at $\omega^{(t)}$

# Gaussian Mixture MLE via EM (summary)

- Start with initial guess $\omega^{(t)} := \left\{\alpha_{kj}^{(t)}, \mu_{kj}^{(t)}, \Sigma_{kj}^{(t)}\right\}$ for $t = 0$, $k = 1, \ldots, K$, $j = 1, \ldots, J$ and iterate:

(E step)

$$\boxed{r_k^{(t)}(j \mid \mathbf{x}_i) = \frac{\alpha_{kj}^{(t)} \phi\left(\mathbf{x}_i; \mu_{kj}^{(t)}, \Sigma_{kj}^{(t)}\right)}{\sum_{l=1}^{J} \alpha_{kl}^{(t)} \phi\left(\mathbf{x}_i; \mu_{kl}^{(t)}, \Sigma_{kl}^{(t)}\right)}}$$

(M step)

$$\boxed{\alpha_{kj}^{(t+1)} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i)}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\}}}$$

$$\boxed{\mu_{kj}^{(t+1)} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i) \mathbf{x}_i}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i)}}$$

$$\boxed{\Sigma_{kj}^{(t+1)} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i) \left(\mathbf{x}_i - \mu_{kj}^{(t+1)}\right)\left(\mathbf{x}_i - \mu_{kj}^{(t+1)}\right)^T}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i)}}$$

# Gaussian Mixture MLE via EM (comments)

- Sometimes the data is not enough to estimate all these parameters:
  - Fix the weights $\alpha_{kj} = \frac{1}{J}$
  - Fix diagonal $\Sigma_{kj} = \mathbf{diag}\left([\sigma_{kj1}^2, \ldots, \sigma_{kjn}^2]^T\right)$ or spherical $\Sigma_{kj} = \sigma_{kj}^2 I_n$
  - Estimate a **diagonal covariance**:

$$\Sigma_{kj}^{(t+1)} = \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i) \mathbf{diag}\left(\mathbf{x}_i - \mu_{kj}^{(t+1)}\right)^2}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i)}$$

  - Estimate a **spherical covariance**:

$$\sigma_{kj}^{2,(t+1)} = \frac{1}{d} \frac{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i) \left\|\mathbf{x}_i - \mu_{kj}^{(t+1)}\right\|^2}{\sum_{i=1}^{n} \mathbb{1}\{y_i = k\} r_k^{(t)}(j \mid \mathbf{x}_i)}, \qquad \mathbf{x}_i \in \mathbb{R}^d$$

- How should we initialize $\omega^{(0)}$? Use $k$-**means++**!

- If $\sigma_{kj} \to 0$, the GM component assignments of EM become hard and EM works like $k$-means.