University of Colorado **Boulder**

Department of Computer Science

CSCI 5622: Machine Learning

Chenhao Tan

Lecture 19: EM algorithm

Slides adapted from Jordan Boyd-Graber, Chris Ketelsen

Administrivia

- HW4 due, HW5 out
  - Remember that we only count the highest 4 homework scores
- Final project midpoint check in on Wednesday (an experiment!)
  - Midpoint report due on Friday (Nov 15)
  - For the final project, each person will be asked to summarize what everyone in the team did
- Example questions posted on Moodle

Learning Objectives

- Learn about Gaussian mixture models
- Learn about Expectation-Maximization algorithm

Quiz on K-means

Which of the following statements are true?

A. The K-means algorithm is sensitive to outliers. ✓

B. For different initializations, the K-means algorithm will give the same clustering results. ✗

C. The centroids in the K-means algorithm may not be any observed data points. ✓

D. Feature scaling is not important for the K-means algorithm. ✗

Gaussian Mixture Models

Guassian Mixture Models (or GMMs) are a probabilistic generalization of K-Means

In K-Means we made **hard** cluster assignments.

That is, we said $\mathbf{x}_i$ definitely belongs to cluster k

Gaussian Mixture Models

Guassian Mixture Models (or GMMs) are a probabilistic generalization of K-Means

In K-Means we made **hard** cluster assignments.

That is, we said $\mathbf{x}_i$ definitely belongs to cluster k

GMM utilizes **soft** cluster assignments

That is, we'll say $\mathbf{x}_i$ belongs to cluster $k = \{1, \ldots, K\}$ with some probability

We can then estimate that probability for all k and, if need be, assign $\mathbf{x}_i$ to the cluster with the highest probability

Gaussian Mixture Models

The motivation behind GMMs is a generative one

x

Gaussian Mixture Models

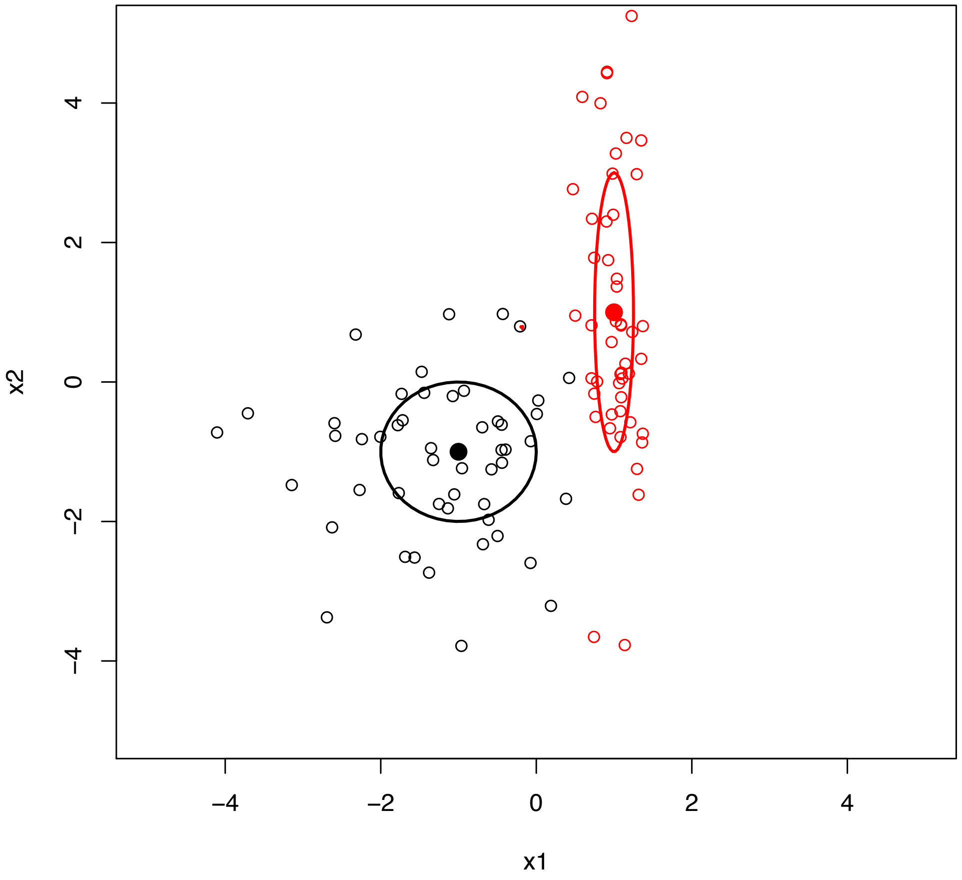
The motivation behind GMMs is a generative one

We have a probabilistic generative story.
We assume each data point is generated in two steps:

1. Cluster assignment, z_i comes from a multinomial distribution (think of rolling a die);
2. Data comes from a Gaussian distribution, $p(\mathbf{x}_i \mid z_i = k) \sim \mathcal{N}(\mu_k, \Sigma_k)$ (given a k, $\mathbf{x}_i$ is multivariate Gaussian).

Gaussian Mixture Models

The motivation behind GMMs is a generative one



Gaussian Mixture Models

The motivation behind GMMs is a generative one
We'll impose on the data a distribution of the form

$$p(\mathbf{x}_i, z_i) = p(\mathbf{x}_i \mid z_i)\, p(z_i)$$

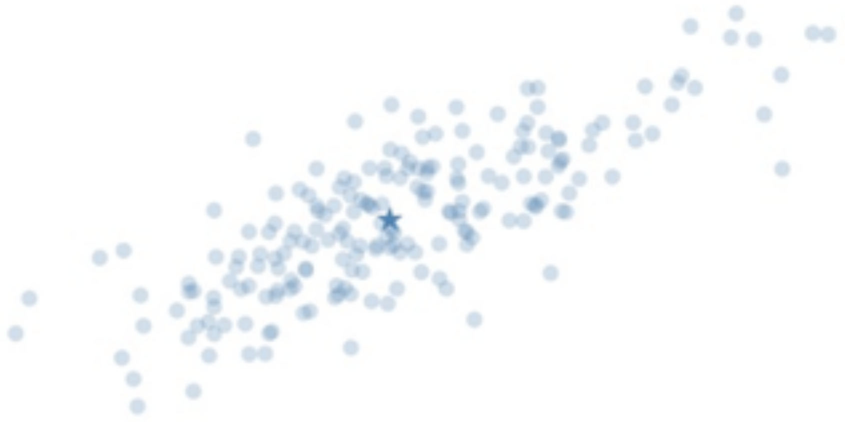
where here z_i is the cluster that $\mathbf{x}_i$ belongs to (though, keep in mind that z_i is a random variable taking on all values in $\{1, \ldots, K\}$
We'll assume:
z_i is multinomial (think rolling a die) $\pi_i,\ i \in \{1, \ldots, K\}$ $\sum_i \pi_i = 1$
$p(\mathbf{x}_i \mid z_i = k) \sim \mathcal{N}(\mu_k, \Sigma_k)$ (given a k, $\mathbf{x}_i$ is Multivariate Gaussian)

Gaussian Mixture Models

$p(\mathbf{x}_i \mid z_i = k) \sim \mathcal{N}(\mu_k, \Sigma_k)$

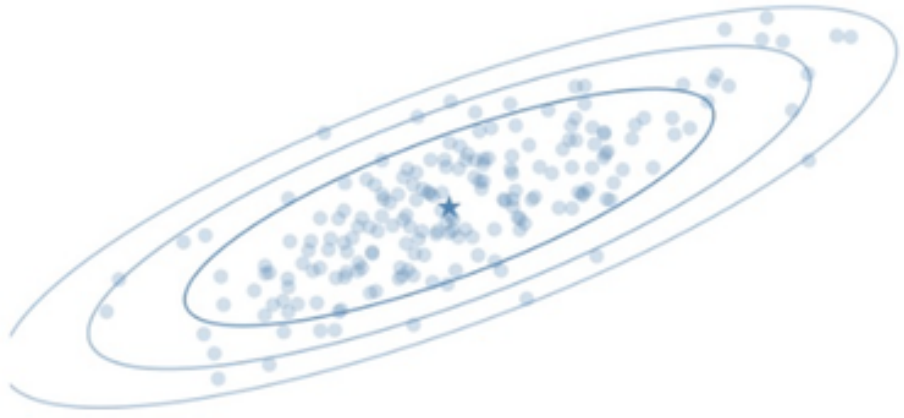
μ_k is a mean vector (just like in K-Means)

Σ_k is a covariance matrix



Gaussian Mixture Models

$p(\mathbf{x}_i \mid z_i = k) \sim \mathcal{N}(\mu_k, \Sigma_k)$
μ_k is a mean vector (just like in K-Means)
Σ_k is a covariance matrix



Gaussian Mixture Models

$p(\mathbf{x}_i \mid z_i = k) \sim \mathcal{N}(\mu_k, \Sigma_k)$
μ_k is a mean vector (just like in K-Means)
Σ_k is a covariance matrix
Density function for $\mathbf{x} \in \mathbb{R}^n$ and cluster k is given by

$$p(\mathbf{x} \mid z_i = k) = \frac{1}{(2\pi)^{n/2}|\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \mu_k)^T \Sigma_k^{-1} (\mathbf{x} - \mu_k)\right\}$$

$$\int_x p(x \mid z_i = k) = 1$$

Gaussian Mixture Models

Can generate data from model by marginalizing over k

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

Gaussian Mixture Models

OK, but we're not trying to generate data
We're trying to cluster data
Our problem is, given our data $\{\mathbf{x}_i\}_{i=1}^{m}$, estimate the parameters in our model so we can say something about the z_i s

$\mu_k, \Sigma_k \quad \forall k \in \{1, \dots K\}$

π_k

Gaussian Mixture Models

OK, but we're not trying to generate data
We're trying to cluster data
Our problem is, given our data $\{\mathbf{x}_i\}_{i=1}^{m}$, estimate the parameters in our model so we can say something about the z_i s
We know we need to estimate μ_k and Σ_k for each k
But we also need to model the Multinomial prior on z
Define $\pi = (\pi_1, \pi_2, \ldots, \pi_K)$ s.t. $\pi_k \geq 0$ and $\sum_{k=1}^{K} \pi_k = 1$
Estimate π_k, μ_k, Σ_k for all k

Gaussian Mixture Models

Suppose we have all the parameters, how do we estimate cluster assignment?

$$P(z|x) = \frac{P(x|z)P(z)}{P(x)} \propto P(x|z)P(z)$$

Gaussian Mixture Models

Suppose we have all the parameters, how do we estimate cluster assignment?
Use the posterior:

$$p(z_i \mid \mathbf{x}_i) \propto p(z_i)p(\mathbf{x}_i \mid z_i = k),$$

just like Naïve Bayes.

Gaussian Mixture Models

It'd be nice if we could do this by Maximum Likelihood Estimation
In that vein, let's define the log-likelihood as

$\log \prod_{i=1}^{m} P(x_i$

$$
\begin{aligned}
\mathcal{L}(\pi, \mu, \Sigma) &= \sum_{i=1}^{m} \log P(\boldsymbol{x}_i \mid \pi, \mu, \Sigma) \\
&= \sum_{i=1}^{m} \log \sum_{k=1}^{k} P(\boldsymbol{x}_i \mid z_i = k, \pi, \mu, \Sigma) P(z_i = k \mid \pi)
\end{aligned}
$$

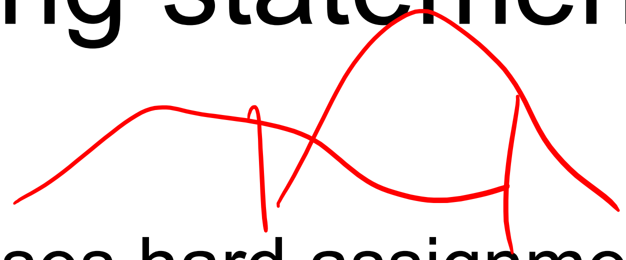
Gaussian Mixture Models

It'd be nice if we could do this by Maximum Likelihood Estimation
In that vein, let's define the log-likelihood as

$$\begin{aligned}
\mathcal{L}(\pi, \mu, \Sigma) &= \textstyle\sum_{i=1}^{m} \log P(\boldsymbol{x}_i \mid \pi, \mu, \Sigma) \\
&= \textstyle\sum_{i=1}^{m} \log \sum_{k=1}^{k} P(\boldsymbol{x}_i \mid z_i = k, \pi, \mu, \Sigma) P(z_i = k \mid \pi)
\end{aligned}$$

It'd be great if we could find MLE estimates in the usual way, by taking derivatives wrt parameters, setting to zero, and solving
However, this is quite hard because of the sum in the log.

Which of the following statements are true?



A. Gaussian Mixture Models uses hard assignment to each cluster. ✗

B. Conditioned on cluster assignment, the distribution of a data point is a Gaussian distribution. ✓ $P(x|z)$

C. *P(x)* is still a Gaussian distribution since it is a mixture of Gaussian distributions. ✗

D. Uniform prior means that a data point is equally likely to be in any cluster a priori. ✓

EM Algorithm

- *z's* correspond to the latent structure that we try to learn in unsupervised learning
- From a modeling perspective, they are usually referred to as latent variables

EM Algorithm

EM Algorithm

Suppose for a sec that we did know the z's

$$\begin{aligned}\mathcal{L}(\pi,\mu,\Sigma) &= \textstyle\sum_{i=1}^m \log P(\boldsymbol{x}_i \mid \pi,\mu,\Sigma)\\ &= \textstyle\sum_{i=1}^m \log P(\boldsymbol{x}_i \mid z_i,\pi,\mu,\Sigma) + \log P(z_i \mid \pi)\end{aligned}$$

$\underline{x_i,\ z_i}$

$$\pi_k = \frac{\sum_{i=1}^m I(z_i=k)}{m},$$

$$\bar{\mu}_k = \frac{\sum_{i=1}^m I(z_i=k)x_i}{\sum_{i=1}^m I(z_i=k)}$$

$$\bar{\Sigma}_k$$

$$\max \sum_{i=1}^m \log \pi_{z_i}$$

$$\text{s.t. } \sum_{k=1}^K \pi_k = 1$$

$$\sum_{i=1}^m \log \pi_{z_i} - \lambda \sum_{k=1}^K \pi_k$$

$$\pi_k \propto \sum_{i=1}^m I(z_i=k)$$

EM Algorithm

Suppose for a sec that we did know the z's

$$
\begin{aligned}
\mathcal{L}(\pi, \mu, \Sigma) &= \sum_{i=1}^{m} \log P(\boldsymbol{x}_i \mid \pi, \mu, \Sigma) \\
&= \sum_{i=1}^{m} \log P(\boldsymbol{x}_i \mid z_i, \pi, \mu, \Sigma) + \log P(z_i \mid \pi)
\end{aligned}
$$

The MLE estimates for the parameters are then given by

$$
\begin{aligned}
\pi_k &= \frac{1}{m} \sum_{i=1}^{m} I\{z_i = k\} \\
\mu_k &= \frac{\sum_{i=1}^{m} I\{z_i = k\} \boldsymbol{x}_i}{\sum_{i=1}^{m} I\{z_i = k\}} \\
\Sigma_k &= \frac{\sum_{i=1}^{m} I\{z_i = k\} (\boldsymbol{x}_i - \mu_k)(\boldsymbol{x}_i - \mu_k)^T}{\sum_{i=1}^{m} I\{z_i = k\}}
\end{aligned}
$$

EM Algorithm

OK, but really we don't know the z's. So what should we do?

EM Algorithm

OK, but really we don't know the z's. So what should we do?
Maybe we could iterate?
Estimate the probability that $\mathbf{x}_i$ belongs to each cluster k?
Hold the z's fixed and do the MLE estimate of the parameters?
Sounds a lot like K-Means!
This is the idea behind the EM algorithm

EM Algorithm

- EM stands for Expectation-Maximization
- A classic algorithm in Dempster, Laird, Rubin, 1977
- An iterative method

EM Algorithm

EM Algorithm:
Each iteration contains two steps, given $\theta^{(t)}$:
(E-step) Compute expectations of latent variables to obtain $Q(\theta \mid \theta^{(t)})$;
(M-step) Find $\theta^{(t+1)}$ that maximizes $Q(\theta \mid \theta^{(t)})$

$$Q(\theta \mid \theta^{(t)}) = E_{z \mid x, \theta^{(t)}} \log P(x, z)$$

$$= \sum_{z} P(z \mid x, \theta^{(t)}) \log P(x, z)$$

$$\log P(x \mid z) + \log P(z)$$

EM Algorithm

$Q(\theta|\theta^{(t)}) = \sum_z P(z|x,\theta^{(t)}) \log P(x,z|\theta)$

$$\forall z, \log P(\boldsymbol{x} \mid \theta) = \log P(\boldsymbol{x}, z \mid \theta) - \log P(z \mid \boldsymbol{x}, \theta) \Rightarrow$$

$\forall z,\ P(z|x,\theta^{(t)}) \log P(x|\theta) = P(z|x,\theta^{(t)}) \log P(x,z|\theta) - P(z|x,\theta^{(t)}) \log P(z|x,\theta)$

$\underbrace{\sum_z P(z|x,\theta^{(t)})}_{1} \log P(x|\theta) = \sum_z P(z|x,\theta^{(t)}) \log P(x,z|\theta) - \sum_z P(z|x,\theta^{(t)}) \log P(z|x,\theta)$

$\log P(x|\theta) = Q(\theta|\theta^{(t)}) + H(\theta \,||\, \theta^{(t)})$

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

$\log P(x|\theta^{(t+1)}) - \log P(x|\theta^{(t)}) = Q(\theta^{(t+1)}|\theta^{(t)}) - Q(\theta^{(t)}|\theta^{(t)}) \underbrace{- H(\theta^{(t)}||\theta^{(t)}) + H(\theta^{(t+1)}||\theta^{(t)})}_{\geq 0}$

EM Algorithm

$$
\begin{aligned}
\forall z, \log P(\boldsymbol{x} \mid \theta) &= \log P(\boldsymbol{x}, z \mid \theta) - \log P(z \mid \boldsymbol{x}, \theta) \Rightarrow \\
\log P(\boldsymbol{x} \mid \theta) &= \textstyle\sum_z P(z \mid \boldsymbol{x}, \theta^{(t)}) \log P(\boldsymbol{x}, z \mid \theta) \\
&\quad - \textstyle\sum_z P(z \mid \boldsymbol{x}, \theta^{(t)}) \log P(z \mid \boldsymbol{x}, \theta) \\
Q(\theta \mid \theta^{(t)}) &= \textstyle\sum_z P(z \mid \boldsymbol{x}, \theta^{(t)}) \log P(\boldsymbol{x}, z \mid \theta) \\
H(\theta \mid \theta^{(t)}) &= -\textstyle\sum_z P(z \mid \boldsymbol{x}, \theta^{(t)}) \log P(z|\boldsymbol{x}, \theta) \\
\log P(\boldsymbol{x} \mid \theta) &= Q(\theta \mid \theta^{(t)}) + H(\theta \mid \theta^{(t)}) \geq Q(\theta \mid \theta^{(t)}) \\
\log P(\boldsymbol{x} \mid \theta^{(t)}) &= Q(\theta^{(t)} \mid \theta^{(t)}) + H(\theta^{(t)} \mid \theta^{(t)})
\end{aligned}
$$

$$\theta = \arg\max_{\theta} Q(\theta|\theta^{(t)})$$

$$
\begin{aligned}
\log P(\boldsymbol{x} \mid \theta) - \log P(\boldsymbol{x} \mid \theta^{(t)}) = Q(\theta \mid \theta^{(t)}) - Q(\theta^{(t)} \mid \theta^{(t)}) \\
+ H(\theta \mid \theta^{(t)}) - H(\theta^{(t)} \mid \theta^{(t)}) \geq 0
\end{aligned}
$$

EM Algorithm

Do until convergence...

(E-step) For each i and k, set

$$T_{ik} = P(z_i = k \mid \boldsymbol{x}_i, \pi, \mu, \Sigma)$$

(M-step) Update the parameters:

$$\begin{aligned}
\pi_k &= \frac{1}{m} \sum_{i=1}^{m} T_{ik} \\
\mu_k &= \frac{\sum_{i=1}^{m} T_{ik} \boldsymbol{x}_i}{\sum_{i=1}^{m} T_{ik}} \\
\Sigma_k &= \frac{\sum_{i=1}^{m} T_{ik} (\boldsymbol{x}_i - \mu_k)(\boldsymbol{x}_i - \mu_k)^T}{\sum_{i=1}^{m} T_{ik}}
\end{aligned}$$

EM Algorithm

Do until convergence...

(E-step) For each i and k, set

$$T_{ik} = \frac{P(\boldsymbol{x}_i \mid z_i = k, \pi, \mu, \Sigma)\pi_k}{\sum_{k'} P(\boldsymbol{x}_i \mid z_i = k', \pi, \mu, \Sigma)\pi_{k'}}$$

(M-step) Update the parameters:

$$\begin{aligned}
\pi_k &= \frac{1}{m}\sum_{i=1}^{m} T_{ik} \\
\mu_k &= \frac{\sum_{i=1}^{m} T_{ik}\boldsymbol{x}_i}{\sum_{i=1}^{m} T_{ik}} \\
\Sigma_k &= \frac{\sum_{i=1}^{m} T_{ik}(\boldsymbol{x}_i - \mu_k)(\boldsymbol{x}_i - \mu_k)^T}{\sum_{i=1}^{m} T_{ik}}
\end{aligned}$$

EM Algorithm

The EM in EM Algorithm stands for Expectation-Maximization

First estimate the Expectation of the z_i's

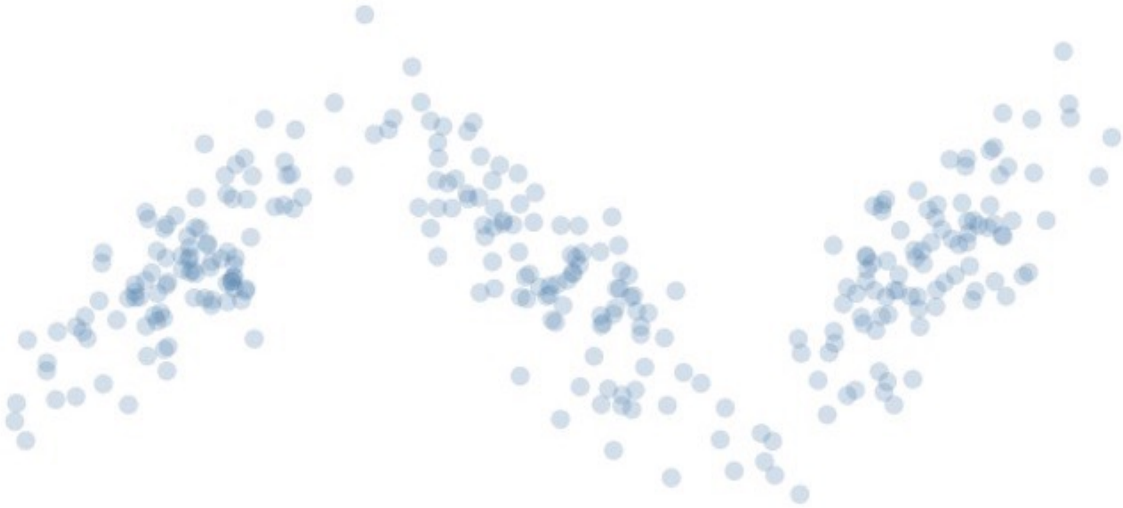
Then Maximize the likelihood of the parameters

Let us look at a simple example to figure out how it works

EM Algorithm

Example: Consider our toy data set again

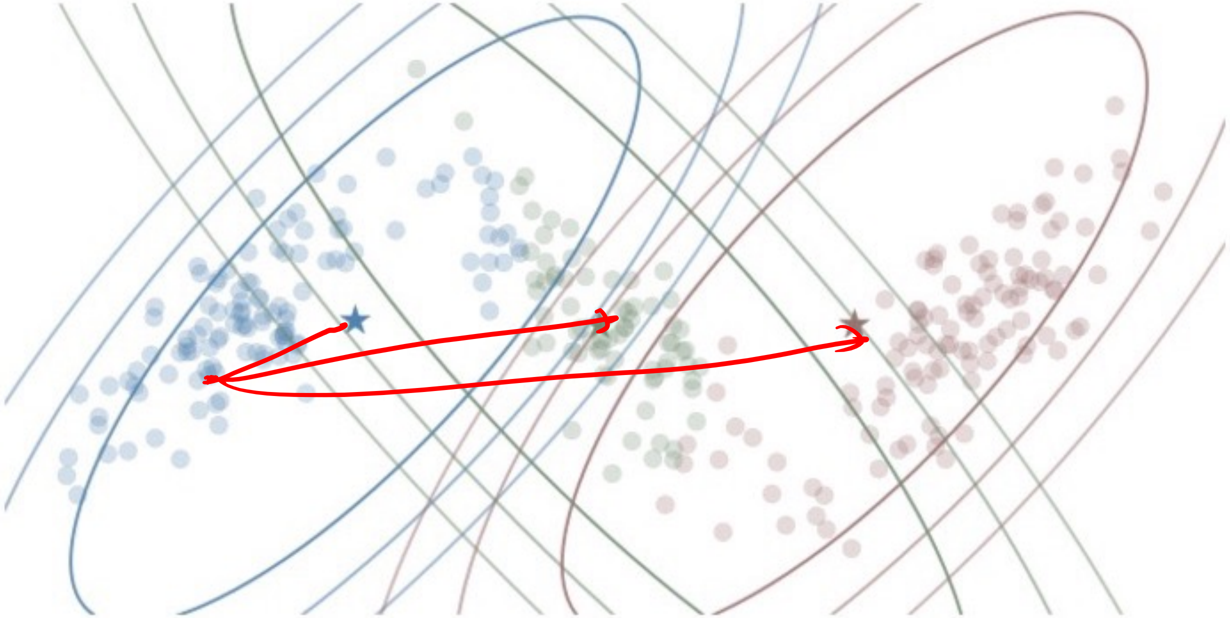
Initial distributions



EM Algorithm

Example: Consider our toy data set again

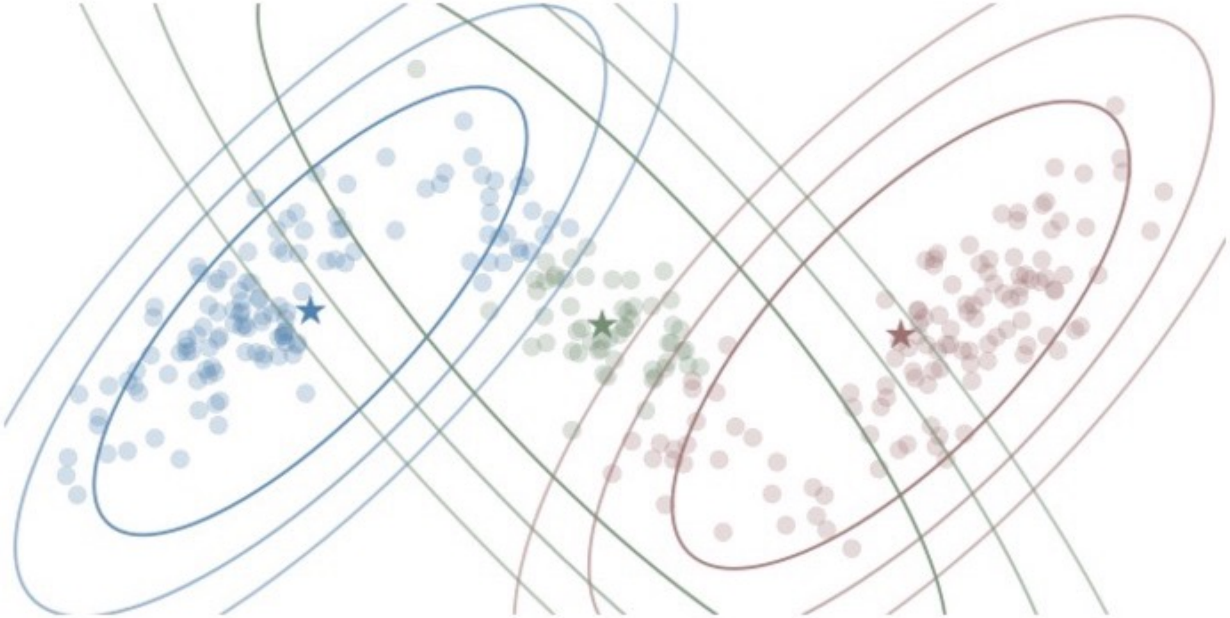
After random initialization of EM algorithm



EM Algorithm

Example: Consider our toy data set again

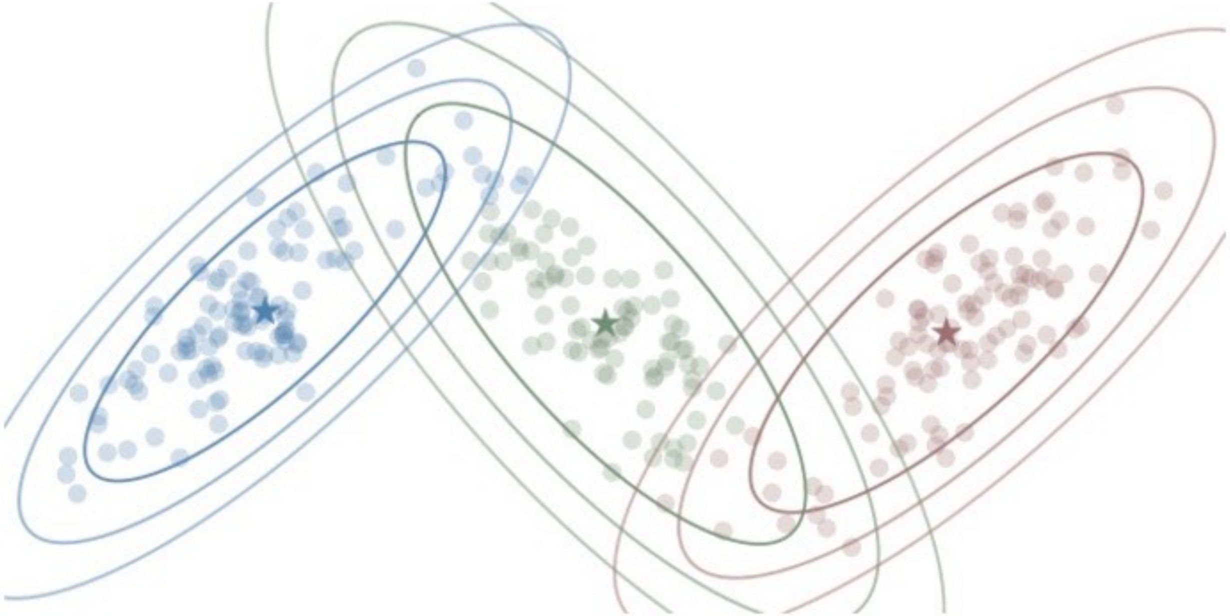
After 1 EM iteration



EM Algorithm

Example: Consider our toy data set again

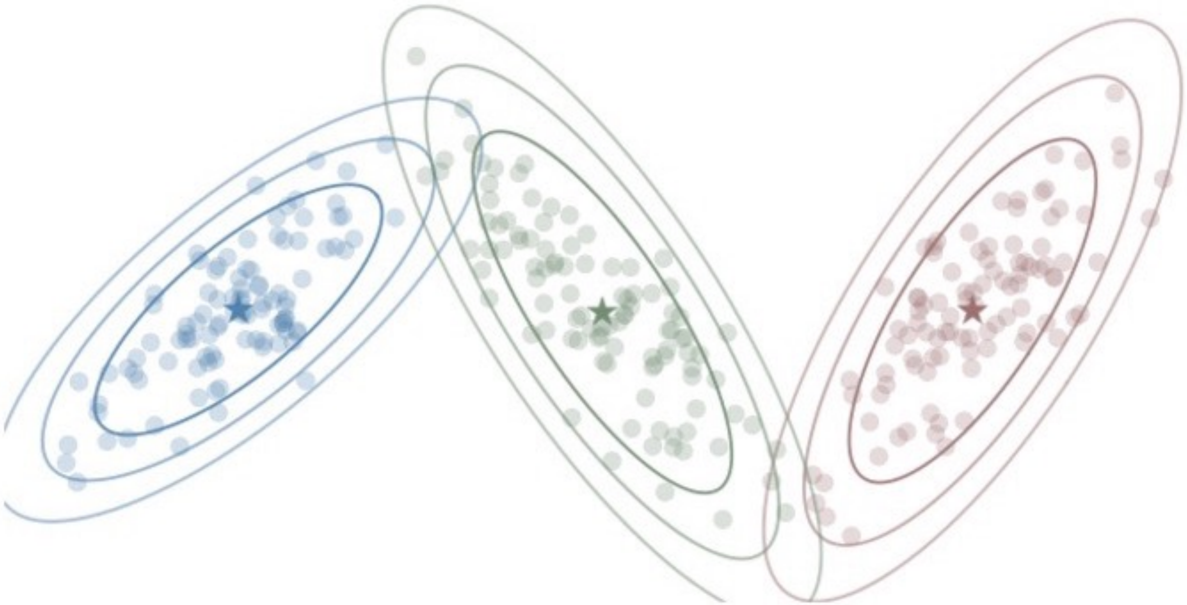
After 3 EM iterations



EM Algorithm

Example: Consider our toy data set again

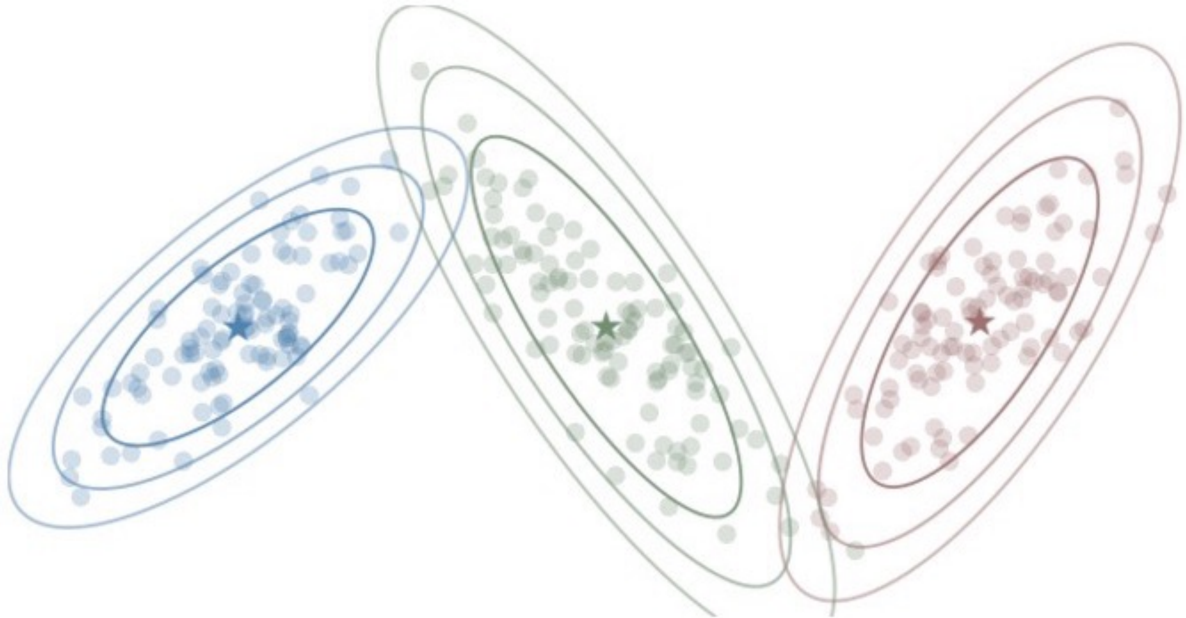
After 6 EM iterations



EM Algorithm

Example: Consider our toy data set again

After 9 EM iterations



GMM and K-means

It turns out that GMM with EM gives you exactly K-Means if you make the assumption that the covariance matrices are diagonal and the variances are known, and use hard assignment in the expectation step (equivalent to set $\sigma^2 \to 0$).

$$\Sigma_k = \begin{bmatrix} \sigma^2 & 0 & 0 & 0 \\ 0 & \sigma^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma^2 \end{bmatrix} \quad \text{for each } k = 1, \ldots, K$$

GMMs and the EM algorithm

- GMMs with the EM Algorithm suffer from some of the same problems as K-Means
  - Doesn't really work with categorical data
  - Usually only converges to a local minimum
  - Have to determine the number of clusters
  - Only generates convex clusters
- But, it also has certain advantages
  - The clusters are allowed different shapes
  - We get a soft partitioning of the data

Which of the following statements are true?

A. The EM algorithm optimizes a lower bound on its objective function, which is the marginal likelihood of the observed data points. ✓

B. The EM algorithm only works for the Gaussian Mixture Models. ✗

C. The EM algorithm is guaranteed to never decrease the value of its objective function on any iteration. ✓

D. The objective function optimized by the EM algorithm can be optimized by gradient descent algorithm which will find the global optimal solution, whereas EM finds its solution more ~~quickly but may return~~ only a locally optimal solution. ✗

https://www.cs.cmu.edu/~epxing/Class/10701-10s/HW/midterm_sol.pdf

Recap

- Gaussian mixture models provides a generative story for clustering
- Expectation-maximization: a general algorithm for mixture models