Machine Learning: Chenhao Tan

University of Colorado Boulder

LECTURE 24

Slides adapted from Jordan Boyd-Graber, Chris Ketelsen

Logistics

- EMPTY!

Learning objectives

- Learn about Gaussian mixture models
- Learn about the high-level intuitions about EM-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.

Outline

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.
What makes a model generative?

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

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

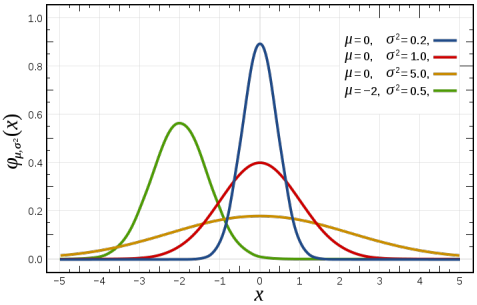
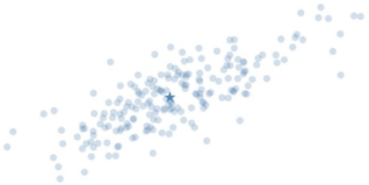


Image from Wikipedia.

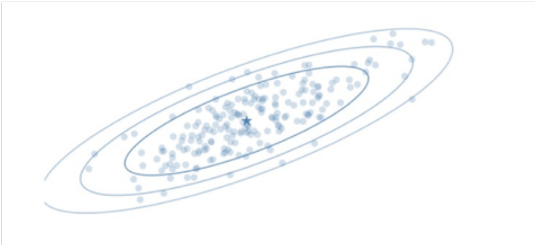
Gaussian mixture models

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

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

Gaussian mixture models

We can generate data from model by marginalizing over k clusters.

Gaussian mixture models

We can generate data from model by marginalizing over k clusters.

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

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.

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.
That is, 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?

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

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

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

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

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

Outline

Expectation minimization

- 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.

Expectation minimization

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

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

Expectation minimization

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

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

Expectation minimization

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

Expectation minimization

OK, but 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.

Expectation minimization

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

Expectation minimization

EM Algorithm:
Each iteration contains two steps, given $\theta_t = (\pi, \mu, \Sigma)$:
(E-step) Compute expectations of latent variables to obtain

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

(M-step) Find $\theta^{(t+1)}$ that maximizes $Q(\theta \mid \theta^{(t)})$

Expectation minimization

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:

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

Expectation minimization

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:

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

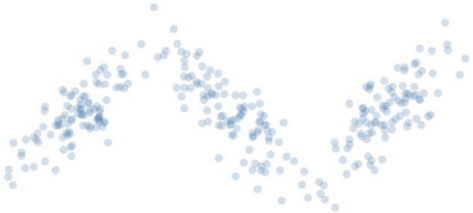
Expectation minimization

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.

Expectation minimization

Example: Consider our toy data set again

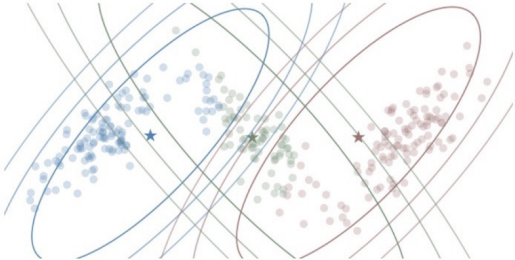
Initial distributions



Expectation minimization

Example: Consider our toy data set again

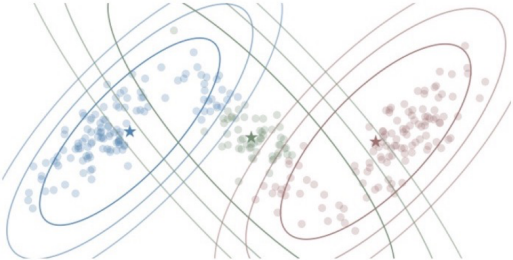
After random initialization of EM algorithm



Expectation minimization

Example: Consider our toy data set again

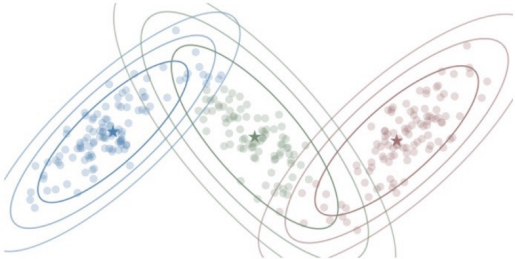
After 1 EM iteration



Expectation minimization

Example: Consider our toy data set again

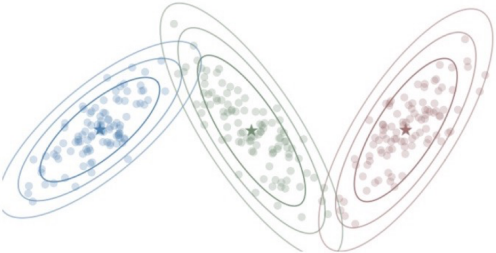
After 3 EM iterations



Expectation minimization

Example: Consider our toy data set again

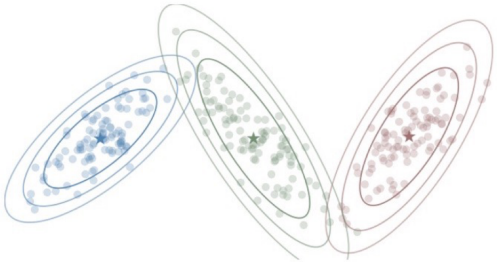
After 6 EM iterations



Expectation minimization

Example: Consider our toy data set again

After 9 EM iterations



Expectation minimization

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 $\sigma^2 \to 0$).

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

Expectation minimization

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.

Bonus: the underlying math

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

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

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