University of Illinois Fall 2018

ECE 544NA: Pattern Recognition
Lecture 17: October 23th

Lecturer: Alexander Schwing Scribe: Jiahao Qian

1 Overview

This lecture introduces Gaussian Mixture Model (GMM). A Gaussian mixture model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities [2]. The goal of this lecture includes:

- Understanding Gaussian mixture models
- Getting to know more details about generative modeling
- Learning the relationship between Gaussian mixture models and k-Means.

The following sections are organized as below: Section 2 provides a recap on previous materials and remind the some important aspects of generative models. Section 3 is a derivation for finding μ,σ in a single Gaussian. However, this approach is problematic. In section 4, we will find μ and σ with multiple Gaussian models and finally achieve the true conclusion. In section 5, an introduction to Gaussian Mixture Model Algorithm will be given. In section 6, the relationship between K-means and GMM model would be explained. In section 7, solutions to quiz questions are provided. In section 8, a quick demo of GMM by scikitlearn is provided.

2 Recap

Recall the equation for linear regression.

$$p\left(y^{(i)} \mid x^{(i)}\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{\sqrt{2\sigma^2}}\left(y^{(i)} - \mathbf{w}^T\phi\left(x^{(i)}\right)\right)^2\right) \tag{1}$$

This is the program for linear regression, which is the most typical discriminative model. Given $x^{(i)}$, we want to find the probability of $y^{(i)}$.

However, in a generative model, we no longer have class labels $y^{(i)}$, what we are interested is to cluster items. Therefore, we would have to determine given a distribution with μ,σ, what is the probability p, such that $x^{(i)}$ belongs to that distribution. Thus, we have the following program for a generative model.

$$p\left(y^{(i)} \mid \mu, \sigma\right) = \mathcal{N}\left(x^{(i)} \mid \mu, \sigma\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{\sqrt{2\sigma^2}}\left(x^{(i)} - \mu\right)^2\right) \tag{2}$$

3 Finding μ,σ in a single Gaussian

In the previous section, we have seen the program for a generative model as

$$p\left(y^{(i)} \mid \mu, \sigma\right) = \mathcal{N}\left(x^{(i)} \mid \mu, \sigma\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{\sqrt{2\sigma^2}}\left(x^{(i)} - \mu\right)^2\right) \tag{3}$$

A very intuitive question for the above program would be given a dataset $\mathcal{D} = \{(x^{(i)})\}$ how to find μ and σ for the distribution that would maximize the overall probability. The answer is that we would minimize the negative log-likelihood for the above program to find the optimum μ and σ.

Taking the negative log-likelihood for the above program we get

$$-\log \prod_{i\in\mathcal{D}} p\left(x^{(i)} \mid \mu, \sigma\right) = \sum_{i\in\mathcal{D}} \left(\frac{1}{\sqrt{2\sigma^2}}\left(x^{(i)} - \mu\right)^2\right) + \frac{||\mathcal{D}||}{2}\log\left(2\pi\sigma^2\right) \tag{4}$$

We then need to minimize the above equation with respect to μ and σ by taking partial derivative, we get

$$\frac{\partial}{\partial\mu} = \frac{1}{\sigma^2}\sum_{i\in\mathcal{D}}\left(x^{(i)} - \mu\right) = 0 \tag{5}$$

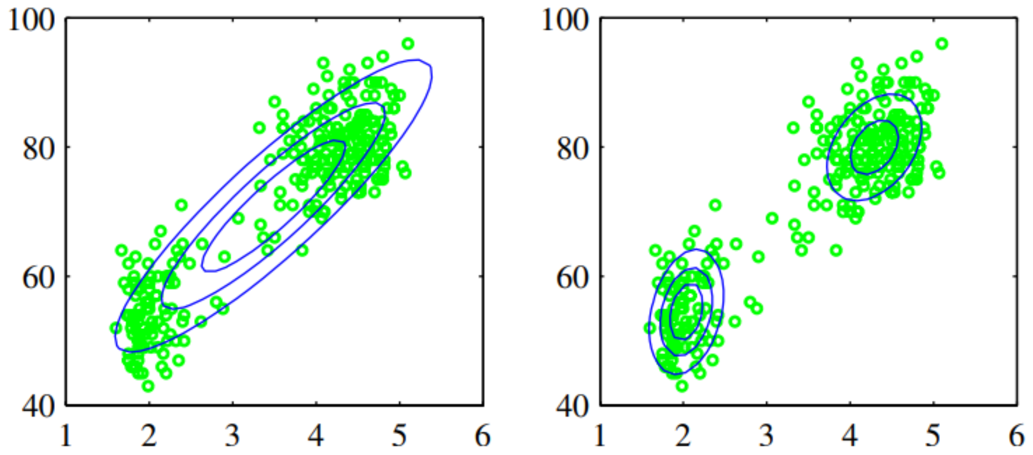
$$\mu = \frac{1}{||\mathcal{D}||}\sum_{i\in\mathcal{D}}\left(x^{(i)}\right) \tag{6}$$

$$\frac{\partial}{\partial\sigma} = -\frac{1}{\sigma^3}\sum_{i\in\mathcal{D}}\left(x^{(i)} - \mu\right)^2 + \frac{||\mathcal{D}||}{\sigma} = 0 \tag{7}$$

$$\sigma^2 = \frac{1}{||\mathcal{D}||}\sum_{i\in\mathcal{D}}\left(x^{(i)} - \mu\right)^2 \tag{8}$$

(Note: The μ and σ we get from one Gaussian distribution fits our intuition and nothing very surprising.)

However, this seems to be too straightforward, and assuming only one Gaussian distribution is actually quite problematic.



As shown in the above diagram, assuming only one Gaussian distribution will get a cluster look like in the left picture. This doesn't meet our expectation which is something similar to the right. The issue here is that single Gaussian isn't flexible and doesn't fit our need for creating multiple clusters.

4 Finding μ and σ in multiple Gaussian

In order to fit multiple Gaussian distributions, we would have to make some modification to our base program.

$$p\left(y^{(i)} \mid \pi, \mu, \sigma\right) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(x^{(i)} \mid \mu_k, \sigma_k\right) \tag{9}$$

We introduce an additional variable pi, which can be understand as the probability that an observation comes from population k (Gaussian with mean μ_k and co-variance σ_k^2). Since an observation must come from the entire dataset and probability cannot be negative, we got the constraint on π that

$$\sum_{k=1}^{K} \pi_k = 1 \ \pi_k \geq 0 \tag{10}$$

Therefore, the modified basic program for our new GMM can be understand as the probability of $x^{(i)}$ is the probability sum of each cluster times a conditional probability π_k.

In order to derive the value for μ,σ, we would like to do the same thing in the previous section on our new basic program. We will minimize the negative log-likelihood for the above program to find the optimum μ and σ by taking derivative and set them to zero.

$$-\log \prod_{i \in \mathcal{D}} p\left(x^{(i)} \mid \pi, \mu, \sigma\right) = -\sum_{i \in \mathcal{D}} \log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}\left(x^{(i)} \mid \mu_k, \sigma_k\right)\right) \tag{11}$$

Unfortunately, at this point we notice that the above equation does not have a closed form solution. Therefore, gradient descent may be useful to minimize the above equation. However, gradient descent may take up too much resources and may take a long time. We would want to use a trick that will allow us to use alternative steps like K-means to minimize our program, which would take less resouces and achieve almost the same result

As a result, we introduce an auxiliary/latent variable called $z_{ik} \in \{0, 1\}$ with $\sum_{k=1}^{K} z_{ik} = 1 \ \forall i$. This means z_{ik} is basically a i by k matrix with every row there is only one entry equals to 1 and the rest equals to 0.

Therefore, it means that we can rewrite all previous equations we had before:

For marginal probability of z_{ik}, we get

$$p\left(z_{ik} = 1\right) = \pi_k \tag{12}$$

When z_{ik} equals to 1, meaning the i^{th} being observed from the k^{th} category is equals to π_k.

$$p\left(z_i\right) = \prod_{k=1}^{K} \pi_k^{z_{ik}} \ , \ z_i = \left[z_{i1}, ..., z_{iK}\right]^T \tag{13}$$

This means the probability of observing z_i is the product of multiplying all π_k.

For conditional probability, we get

$$p\left(x^{(i)} \mid z_{ik} = 1\right) = \mathcal{N}\left(x^{(i)} \mid \mu_k, \sigma_k\right) \tag{14}$$

The probability of an item that belongs to a specific category is the probability of that item in that particular Gaussian distribution.

We can confirm that with the introduction of our new latent variable, nothing has changed by calculating the marginal probability for $x^{(i)}$

$$p\left(x^{(i)} \mid \pi, \mu, \sigma\right) = \sum_{z_i} p\left(x^{(i)} \mid z_i\right) * p\left(z_i\right) \tag{15}$$

Plugging in equation 13 and 14, we get

$$\sum_{z_i} p\left(x^{(i)} \mid z_i\right) * p\left(z_i\right) = \sum_{z_i} \prod_{k=1}^{K} \pi_k^{z_{ik}} \mathcal{N}\left(x^{(i)} \mid \mu_k, \sigma_k\right)^{z_{ik}} = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(x^{(i)} \mid \mu_k, \sigma_k\right) \tag{16}$$

Therefore, we get

$$p\left(x^{(i)} \mid \pi, \mu, \sigma\right) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(x^{(i)} \mid \mu_k, \sigma_k\right) \tag{17}$$

which is the exact same thing as we had in equation 9.

Finally, we could introduce another variable r_{ik} to be our posterior probability.

$$r_{ik} = p\left(z_{ik} = 1 \mid x^{(i)}\right) = \frac{p\left(z_{ik} = 1\right) * p\left(x^{(i)} \mid z_{ik} = 1\right)}{\sum_{\hat{k}=1}^{K} p\left(z_{i\hat{k}} = 1\right) * p\left(x^{(i)} \mid z_{i\hat{k}} = 1\right)} = \frac{\pi_k \mathcal{N}\left(x^{(i)} \mid \mu_k, \sigma_k\right)}{\sum_{\hat{k}=1}^{K} \pi_{\hat{k}} \mathcal{N}\left(x^{(i)} \mid \mu_{\hat{k}}, \sigma_{\hat{k}}\right)} \tag{18}$$

With all those definitions at hand, we can start to minimize the negative log-likelihood of our original program.

$$-\log \prod_{i \in \mathcal{D}} p\left(x^{(i)} \mid \pi, \mu, \sigma\right) = -\sum_{i \in \mathcal{D}} \log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}\left(x^{(i)} \mid \mu_k, \sigma_k\right)\right) \ s.t \ \sum_{k=1}^{K} \pi_k = 1 \tag{19}$$

We will have $N_k = \sum_{i \in \mathcal{D}} r_{ik}$ to denote the effective number of points in cluster k.

Then we can derive μ_k,σ_k,π_k:

For μ_k

$$\frac{\partial}{\partial \mu_k} = -\sum_{i \in \mathcal{D}} r_{ik}\left(-\frac{1}{2\sigma^2}\left(X^{(i)} - \mu_k\right)\right) = 0 \tag{20}$$

Solving the equation we get:

$$\mu_k = \frac{1}{N_k} \sum_{i \in \mathcal{D}} r_{ik} X^{(i)} \tag{21}$$

For σ_k

$$\frac{\partial}{\partial \sigma_k} = \sum_{i \in \mathcal{D}} r_{ik}\left(\frac{1}{\sigma} - \frac{1}{\sigma^2}\left(X^{(i)} - \mu_k\right)^2\right) = 0 \tag{22}$$

Solving the equation we get:

$$\sigma_k = \frac{1}{N_k} \sum_{i \in \mathcal{D}} r_{ik}\left(X^{(i)} - \mu_k\right)^2 \tag{23}$$

For π_k

We will use Lagrange multiplier to solve for the answer

$$\sum_{i\in\mathcal{D}} \frac{\mathcal{N}\left(x^{(i)} \mid \mu_k, \sigma_k\right)}{\sum_{\hat{k}=1}^{K} \pi_{\hat{k}} \mathcal{N}\left(x^{(i)} \mid \mu_{\hat{k}}, \sigma_{\hat{k}}\right)} + \lambda = 0 \tag{24}$$

Multiplying both sides with π_k and summation over k, we get:

$$\lambda = -N \tag{25}$$

Multiplying with π_k and rearranging, we get:

$$\pi_k = \frac{N_k}{N} \tag{26}$$

At this point, Note that this is not a closed form solution of the parameters, as they depend on the r_{ik} , which are complex functions of the parameters.

However, we could use a more simple iterative scheme to optimize, which at each individual update, there are analytic solutions.

5 Gaussian Mixture Model Algorithm

Gaussian Mixture Model is very similar to what we have for the K-means algorithm. It will consist of two update steps. Updating the latent variable r_{ik} and then updating the cluster parameters μ_k,σ_k,π_k. An outline of the algorithm will look like below.

Gaussian Mixture Model Algorithm:

- Initialize μ,σ,π
- iterate:
  - E-step : Update

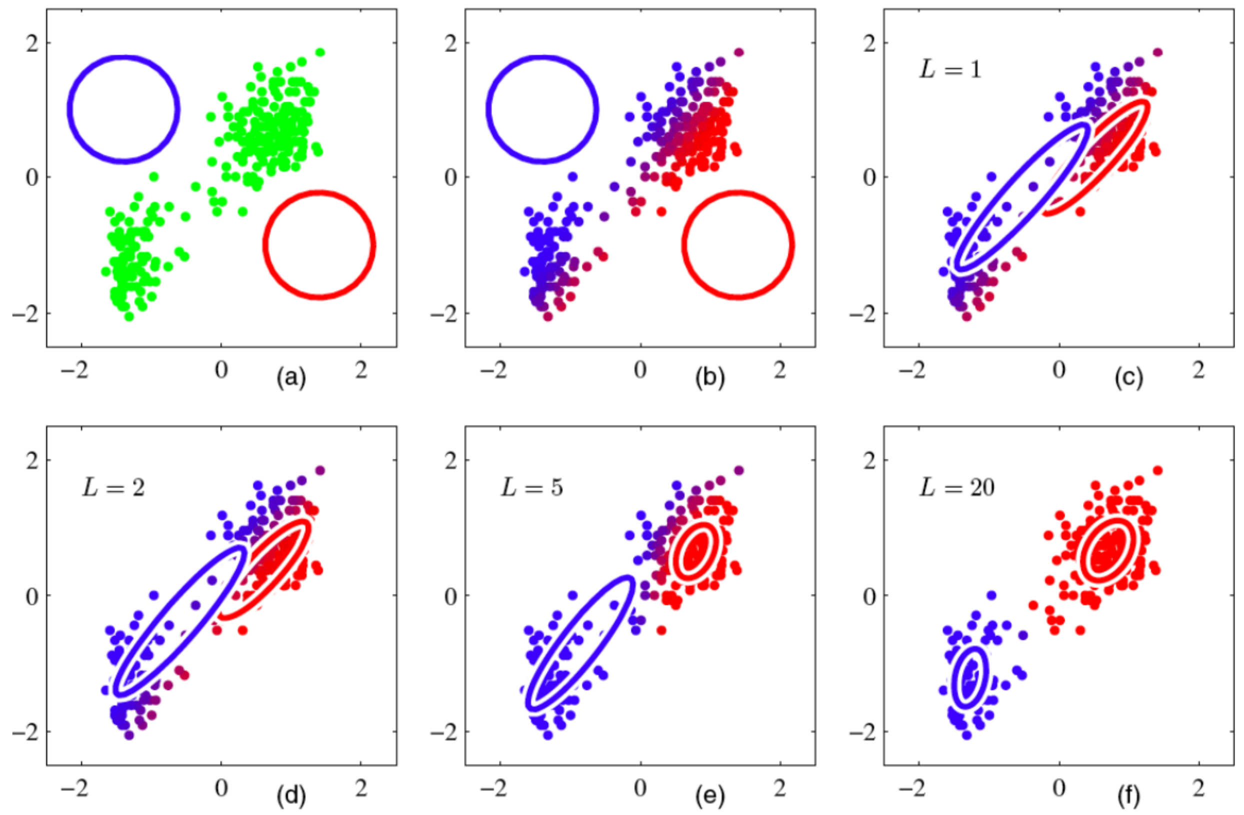
$$r_{ik} = \frac{\pi_k \mathcal{N}\left(x^{(i)} \mid \mu_k, \sigma_k\right)}{\sum_{\hat{k}=1}^{K} \pi_{\hat{k}} \mathcal{N}\left(x^{(i)} \mid \mu_{\hat{k}}, \sigma_{\hat{k}}\right)} \tag{27}$$

  - M-step : Update

$$\mu_k = \frac{1}{N_k} \sum_{i\in\mathcal{D}} r_{ik} X^{(i)} \tag{28}$$

$$\sigma_k = \frac{1}{N_k} \sum_{i\in\mathcal{D}} r_{ik} \left(X^{(i)} - \mu_k\right)^2 \tag{29}$$

$$\pi_k = \frac{N_k}{N} \tag{30}$$



The above diagram shows how GMM algorithm progress overtime. We can see the clusters gradually form as there are more and more iterations.

6 The relationship between K-means and GMM

Similarity to kMeans:

- r_{ik} is an assignment of sample i to cluster k, albeit a soft assignment
- μ_k are the cluster centers

The above definition is relatively informal, let's introduce a more formal similarity.
First we want to fix

$$\sigma_k^2 = \epsilon \ \forall k \tag{31}$$

This means that we are fixing the variance of each cluster.
Then we can transform our posterior probability r_{ik} into

$$r_{ik} = \frac{\pi_k \exp\left(-\frac{1}{2\epsilon}\left(x^{(i)} - \mu_k\right)^2\right)}{\sum_{\hat{k}=1}^{K} \pi_{\hat{k}} \exp\left(-\frac{1}{2\epsilon}\left(x^{(i)} - \mu_{\hat{k}}\right)^2\right)} \tag{32}$$

When $\epsilon \to 0$

- In the denominator the term for which $\left(x^{(i)} - \mu_{\hat{k}}\right)^2$ is smallest goes to zero slowest

- All responsibilities will go to zero except the one for which $\left(x^{(i)} - \mu_{\hat{k}}\right)^2$ is the smallest, which will go to unity
- Responsibilities are hard assignments
- Cost function can be shown to be identical in the limit

In summary, when $\epsilon \to 0$, r_{ik} will approach K-means. On the other hand, when $\epsilon \to 1$, r_{ik} will approach GMM.

7 Quiz answers

- What is the maximum likelihood solution of fitting the mean and variance of a Gaussian?
  - The mean and variance of a single Gaussian will be the same as what we have learned before

$$\mu = \frac{1}{||\mathcal{D}||} \sum_{i \in \mathcal{D}} \left(x^{(i)}\right) \tag{33}$$

$$\sigma^2 = \frac{1}{||\mathcal{D}||} \sum_{i \in \mathcal{D}} \left(x^{(i)} - \mu\right)^2 \tag{34}$$

- Why do we consider mixtures of Gaussians?
  - In short, only using one Gaussian model to represent the different clusters may lack flexibility and not very precise. Therefore, we consider use multiple Gaussian models to represent different clusters.
- How do we find the means, variances and responsibilities of the Gaussian mixture model?
  -

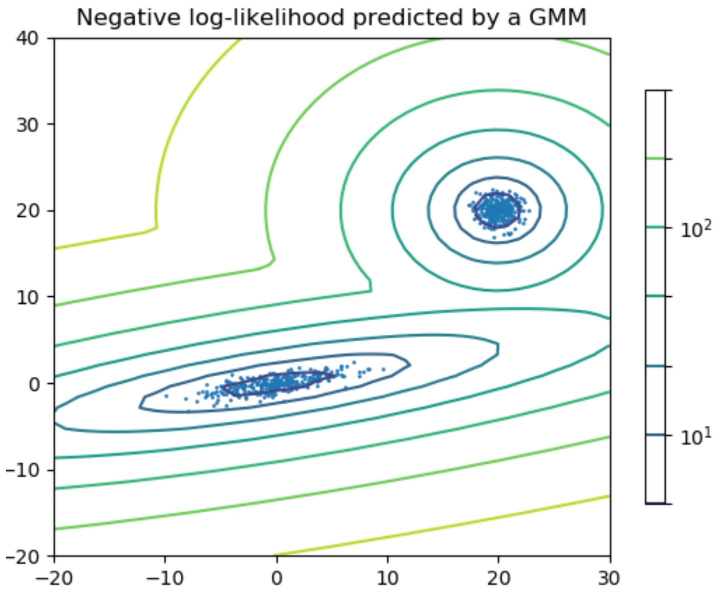
$$\mu_k = \frac{1}{N_k} \sum_{i \in \mathcal{D}} r_{ik} X^{(i)} \tag{35}$$

$$\sigma_k = \frac{1}{N_k} \sum_{i \in \mathcal{D}} r_{ik} \left(X^{(i)} - \mu_k\right)^2 \tag{36}$$

$$r_{ik} = \frac{\pi_k \mathcal{N}\left(x^{(i)} \mid \mu_k, \sigma_k\right)}{\sum_{\hat{k}=1}^{K} \pi_{\hat{k}} \mathcal{N}\left(x^{(i)} \mid \mu_{\hat{k}}, \sigma_{\hat{k}}\right)} \tag{37}$$

8 Scikitlearn GMM

Plot the density estimation of a mixture of two Gaussians. Data is generated from two Gaussians with different centers and covariance matrices. [1]



```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
from sklearn import mixture

n_samples = 300

# generate random sample, two components
np.random.seed(0)

# generate spherical data centered on (20, 20)
shifted_gaussian = np.random.randn(n_samples, 2) + np.array([20, 20])

# generate zero centered stretched Gaussian data
C = np.array([[0., -0.7], [3.5, .7]])
stretched_gaussian = np.dot(np.random.randn(n_samples, 2), C)

# concatenate the two datasets into the final training set
X_train = np.vstack([shifted_gaussian, stretched_gaussian])

# fit a Gaussian Mixture Model with two components
clf = mixture.GaussianMixture(n_components=2, covariance_type='full')
clf.fit(X_train)

# display predicted scores by the model as a contour plot
x = np.linspace(-20., 30.)
y = np.linspace(-20., 40.)
X, Y = np.meshgrid(x, y)
XX = np.array([X.ravel(), Y.ravel()]).T
Z = -clf.score_samples(XX)
Z = Z.reshape(X.shape)

CS = plt.contour(X, Y, Z, norm=LogNorm(vmin=1.0, vmax=1000.0),
                 levels=np.logspace(0, 3, 10))
CB = plt.colorbar(CS, shrink=0.8, extend='both')
plt.scatter(X_train[:, 0], X_train[:, 1], .8)

plt.title('Negative log-likelihood predicted by a GMM')
plt.axis('tight')
plt.show()
```

References

[1] Density estimation for a gaussian mixture. `http://scikit-learn.org/stable/auto_examples/mixture/plot_gmm_pdf.html#sphx-glr-auto-examples-mixture-plot-gmm-pdf-py`, Nov. 2018.

[2] D. Reynolds. *Gaussian mixture models.* Encyclopedia of biometrics, 2015.