Lab 3

Due: Friday 09/27/2024 @ 11:59pm EST

The purpose of labs is to give you some hands on experience programming the things we've talked about in lecture. This lab will focus on learning (i.e. "fitting") using expectation maximization. In this lab, we will be building a Gaussian Mixture Model: a model where each of the clusters is assumed to be gaussianly distributed. Your job will be to implement the two parts of EM: the E step (where we segment our data), and the M step (where we execute the MLE equations for optimizing our Q function).

Task 0: Setup

Contained with this document are two python files. I first want you to take a look at the file called `two_gaussian_1dgmm.py`. In this file is a class for a Gaussian Mixture Model where there are assumed to only be 2 clusters and that our data is 1 dimensional. Please take a look at this file for hints/help in writing your solution.

Remember from lecture, we basically boiled down EM into a few pieces. In the E-step, we calculate γ_{ij} values: or in other words a single number for an example x_i paired with a cluster c_j. This value γ_{ij} contains $Pr[Z_{ij} = 1|x_i]$, which can be thought of as "after I have observed example x_i, what is the probabilty that it came from cluster c_j?" In lecture we calculated this value using Baye's rule:

$$Pr[Z_{ij} = 1|x_i] = \frac{Pr[x_i|Z_{ij} = 1]Pr[Z_{ij} = 1]}{Pr[x_i]}$$

which we then further reduced to

$$Pr[Z_{ij} = 1|x_i] = \frac{Pr[x_i|Z_{ij} = 1]\pi_j}{\sum_{l=1}^{k} Pr[x_i|Z_{il} = 1]\pi_l}$$

using the total law of probability to rewrite the denominator.

After coming up with our γ_{ij} values, we then looked to find the MLE estimates for π_w, θ_w (for some value of $1 \leq w \leq k$ of the following equation:

$$Q(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{k} \gamma_{ij}(\log(\pi_j) + \log(Pr[x_i|\theta_j]))$$

subject to the following constraint: $\left(\sum_{j=1}^{k} \pi_j\right) = 1$

In lecture we figured out that the prior distributions should be updated using the following MLE equation (we got this by using a lagrangian multiplier, taking derivatives, etc.): $\pi_w^{(MLE)} = \frac{1}{n}\sum_{i=1}^{n} \gamma_{iw}$.

In a **Gaussian** Mixture Model, we assume each cluster is distributed to a gaussian: so we now know what $Pr[x_i|\theta_j]$ should be. θ_j is a placeholder for "the parameters of the j^{th} cluster. Since this cluster is a 1d gaussian, we know that $\theta_j = (\mu_j, \sigma_j^2)^T$. We can also now say:

$$Pr[x_i|\theta_j] = Pr[x_i|\mu_j, \sigma_j^2] = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{\frac{-(x_i-\mu_j)^2}{2\sigma_j^2}}$$

This means that in our Q function above, we can expand the term $\log(Pr[x_i|\theta_j])$:

$$\begin{aligned}\log(Pr[x_i|\theta_j]) &= \log\left(\frac{1}{\sqrt{2\pi\sigma_j^2}}e^{\frac{-(x_i-\mu_j)^2}{2\sigma_j^2}}\right)\\ &= \log\left(\frac{1}{\sqrt{2\pi\sigma_j^2}}\right) + \log\left(e^{\frac{-(x_i-\mu_j)^2}{2\sigma_j^2}}\right)\\ &= -\frac{1}{2}\log(2\pi\sigma_j^2) - \frac{(x_i-\mu_j)^2}{2\sigma_j^2}\\ &= -\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\sigma_j^2) - \frac{(x_i-\mu_j)^2}{2\sigma_j^2}\end{aligned}$$

This means that we can finally write our Q function in an expanded form:

$$Q(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{k}\gamma_{ij}\left(\log(\pi_j) - \frac{(x_i-\mu_j)^2}{2\sigma_j^2} - \frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\sigma_j^2)\right)$$

If we take derivatives, set them equal to 0, and solve, we end up with the following equations for the MLE estimates of $\mu_w^{(MLE)}$ and $(\sigma_w^2)^{(MLE)}$:

$$\mu_w^{(MLE)} = \frac{\sum_{i=1}^{n}\gamma_{iw}x_i}{\sum_{i=1}^{n}\gamma_{iw}}$$

$$(\sigma_w^2)^{(MLE)} = \frac{\sum_{i=1}^{n}\gamma_{iw}(x_i-\mu_w^{(MLE)})^2}{\sum_{i=1}^{n}\gamma_{iw}}$$

Putting this all together, this means that in our E-step, we need to calculate all of the γ_{ij} values, and in our M-step we need to execute our three MLE equations:

$$\pi_w^{(MLE)} = \frac{1}{n}\sum_{i=1}^{n}\gamma_{iw}$$

$$\mu_w^{(MLE)} = \frac{\sum_{i=1}^{n}\gamma_{iw}x_i}{\sum_{i=1}^{n}\gamma_{iw}}$$

$$(\sigma_w^2)^{(MLE)} = \frac{\sum_{i=1}^{n}\gamma_{iw}(x_i-\mu_w^{(MLE)})^2}{\sum_{i=1}^{n}\gamma_{iw}}$$

That's it! We did all of this work on paper so that we could program just a few equations. The beauty is that these equations are guaranteed to give us monotically-increasing performance if we implement them correctly!

Task 1: `class GMM1D` (100 points)

In the file `gmm1d.py`, you will find a class called `GMM1D`. This class has a few methods in it, but I want you to pay attention to two of them in particular: `estep` and `mstep`. These method implement

the E-step and M-step of an EM iteration respectively, and it is what you will need to finish in this lab. This class assumes that your data is 1-dimensional, but lets the user specify how many clusters (i.e. gaussians) through the constructor. The number of gaussians is stored in a field called `self.num_gaussians`. Your code needs to work for an arbitrary number of gaussians, while the sample file I provided only works for two. Please implement the following:

- `estep`. This method takes in a parameter `X`, which is a numpy array storing a column vector. It should have shape (i.e. dimensionality) of (*num_examples*, 1), where *num_examples* is the number of data points in the dataset. Your E-step method should calculate and return the γ_{ij} values inside a matrix (also a numpy array). The shape of this matrix should be (*num_examples*, *num_gaussians*). Each row in this matrix corresponds to a data point x_i, and is a pmf over the clusters. Remember to calculate your γ_{ij} values using the equation:

$$\gamma_{ij} = Pr[Z_{ij} = 1|x_i] = \frac{Pr[x_i|Z_{ij} = 1]\pi_j}{\sum_{l=1}^{k} Pr[x_i|Z_{il} = 1]\pi_l}$$

  where π_j is the prior probability of picking cluster c_j.

- `mstep`. This method takes two arguments: `X` and `gammas`. `X` contains the column vector data just like in E-step, while `gammas` contains the matrix that is the output of the E-step method. In the M-step method you will need to recalculate the priors as well as the parameters of the gaussians using the following equations:

$$\pi_w^{(MLE)} = \frac{1}{n}\sum_{i=1}^{n}\gamma_{iw}$$

$$\mu_w^{(MLE)} = \frac{\sum_{i=1}^{n}\gamma_{iw}x_i}{\sum_{i=1}^{n}\gamma_{iw}}$$

$$(\sigma_w^2)^{(MLE)} = \frac{\sum_{i=1}^{n}\gamma_{iw}(x_i - \mu_w^{(MLE)})^2}{\sum_{i=1}^{n}\gamma_{iw}}$$

Feel free to run the `gmm1d.py` file, there is some basic testing at the bottom of the file that will be run if you run the file. I would recommend adding some of your own testing (what shape is what, pmfs should sum to 1, etc) to increase your confidence in the correctness of your solution before you submit to the autograder.

Task 4: Submit Your Lab

To complete your lab, please **only turn in the `gmm1d.py` file** on Gradescope. You shouldn't have to worry about zipping it up or anything, just drag and drop it in.