24-787 Homework 9

Due date: 04/18/2021 @ 11:59 pm EST

Note: In case a problem requires programming, it should be programmed in `Python`. In Programming, you should use plain `Python` language, unless otherwise stated. For example, if the intention of a Problem is familiarity with numpy library, it will be clearly noted in that problem to use numpy. Please submit your homework through Gradescope.

Submissions: There are two steps to submitting your assignment on Gradescope:

1. **HW09_Writeup**: Submit a combined pdf file containing the answers to theoretical questions as well as the pdf form of the FILE.ipynb notebooks.

- To produce a pdf of your notebooks, you can first convert each of the .ipynb files to HTML.
- To do this, simply run: `ipython nbconvert -to html FILE.ipynb` for each of the notebooks, where FILE.ipynb is the notebook you want to convert. Then you can convert the HTML files to PDFs with your favorite web browser.
- If an assignment has theoretical and mathematical derivation, scan your handwritten solution and make a PDF file.
- Then concatenate them all together in your favorite PDF viewer/editor. The file name (FILE) for naming should be saved as HW-assignmentnumber-andrew-ID.pdf. For example for assignment 1, my FILE = HW-1-lkara.pdf
- Submit this final PDF on Gradescope, and **make sure to tag the questions correctly!**

2. **HW09_Code**: Submit a ZIP folder containing the FILE.ipynb notebooks for each of the programming questions. The ZIP folder containing your iPython notebook solutions should be named as HW-assignmentnumber-andrew-ID.zip

Q1: Gaussian Mixture Model (GMM) (50 pts)

(a) GMM learning with Expectation Maximization (35 pts)

We can think of **GMMs** as the soft generalization of the **K-Means clustering** algorithm. Like K-Means, GMMs also demand the number of clusters k as an input to the learning algorithm. However, there is a key difference between the two. K-Means can only learn clusters with a circular form. GMMs, on the other hand, can learn clusters with any elliptical shape.

In general, GMMs try to learn each cluster as a different Gaussian distribution. Also, K-Means only allows for an observation to belong to one, and only one cluster. Differently, GMMs give probabilities that relate each example with a given cluster. For each observation, GMMs learn the probabilities of that example to belong to each cluster k. In general, GMMs try to learn each cluster as a different Gaussian distribution. It assumes the data is generated from a limited mixture of Gaussians.

In this problem, you will study how to learn a GMM model with the famous **expectation-maximization (EM)** algorithm on a 1D dataset. The dataset is similar to Fig. 1, which is sampled from 3 Gaussians. The distributions of different Gaussians can overlap.

Since we provide one-dimensional data and the number of clusters k equals 3, GMMs attempt to learn 9 parameters.

- 3 parameters for the means
- 3 parameters for the variances
- 3 scaling parameters

Here, each cluster is represented by an individual Gaussian distribution. For each Gaussian, it learns one mean and one variance parameters from data. The 3 scaling parameters, 1 for each Gaussian, are only used for density estimation. To learn such parameters, GMMs use the EM algorithm to optimize the maximum likelihood. In the process, GMM uses Bayes Theorem to calculate the probability of a given observation x to belong to each clusters k. We can think of GMMs as a weighted sum of the 3 Gaussian distributions.The detailed implementation of EM algorithm for GMMs are shown below.

EM Implementation: before we start running the EM, we need to give initial values for the learnable parameters. We can guess the values for the means and variances, and initialize the weight parameters as 1/k.Then, we can start maximum likelihood optimization using the EM algorithm. EM can be simplified in 2 phases: The **E** (expectation) step and **M** (maximization) step.

- **E Step**: in the E step, we calculate the likelihood of each observation x_i using the estimated parameters using Eqn. 1. For each cluster k, we calculate the probability density function (pdf) of our data using the estimated values for the mean and variance. At this point, these values are mere random guesses.

$$f(x|\mu_k, \sigma_k^2) = \frac{1}{\sqrt{2\pi\sigma_k^2}} exp(-\frac{(x-\mu_k)^2}{2\sigma_k^2}) \tag{1}$$

  Then, we can calculate the likelihood of a given example x_i to belong to the k^{th} cluster with Eqn. 2.

$$b_k = \frac{f(x|\mu_k, \sigma_k^2)\phi_k}{\sum_{k=1}^{K} f(x|\mu_k, \sigma_k^2)\phi_k} \tag{2}$$

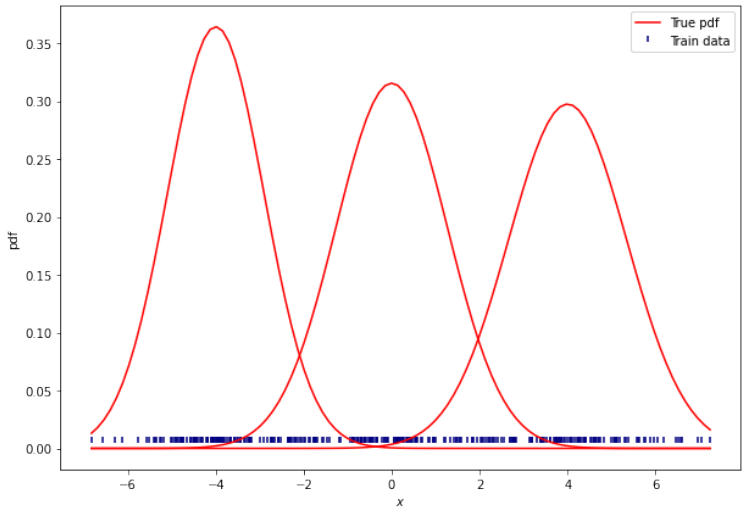

Figure 1: Schematic plots of the 1D dataset used for learning the GMM model.

Using Bayes Theorem, we get the posterior probability of the k^{th} Gaussian to explain the data. That is the likelihood that the observation x_i was generated by k^{th} Gaussian. Note that the parameters ϕ act as our prior beliefs that an example was drawn from one of the Gaussians we are modeling. Since we do not have any additional information to favor a Gaussian over the other, we start by guessing an equal probability that an example would come from each Gaussian. However, at each iteration, we refine our priors until convergence.

- **M Step**: in the maximization, or M step, we re-estimate our learning parameters as follows (Eqn. 3).

$$\mu_k = \frac{\sum b_k x}{\sum b_k}, \quad \sigma_k^2 = \frac{\sum b_k (x - \mu_k)^2}{\sum b_k}, \quad \phi_k = \frac{1}{N} \sum b_k \tag{3}$$

We may repeat these steps until converge. That could be up to a point where parameters' updates are smaller than a given tolerance threshold. At each iteration, we update our parameters so that it resembles the true data distribution.

Based on the above description of the EM algorithm for learning GMMs, you will implement part of the EM algorithm based on some base code. In the code given in the `HW9_Q1a.ipynb` where we have generated the training data and coded some part of the EM algorithm, read the given code carefully and then fill in the code marked with **"Enter your code here"**. You only need to code the **M step** and the **pdf function** of the Gaussian. In order to avoid division by zero error, you could add a small number (such as 1e-8) to the denominators whenever needed.
After finishing the code, run the EM code, and it should automatically generate the plot along the training process if you fill in the code correctly. When submitting your ipynb pdf file, make sure the generated plot is visible. Print out and report your learned means and variances for the 3 clusters after 10 iterations.
Submit the completed code with the automatic generated plots along the EM iterations; submit the means and variances values after 10 EM iterations.

(b) GMM with `sklearn` library (15) pts

Now, with the same generated dataset you used in (a), please run the GMMs again with the `sklearn` in-built libraries with the same number of clusters. Report the learned means and variances for the

3 clusters. Did you get similar values compared to a?
Submit the code, the means and variances values of the learned GMMs.

Q2: Clustering Algorithms (50 pts)

For this question, you are given a dataset → `threeblobs.txt` as shown in Fig. 2. You need to implement GMM, DBSCAN, and KMeans clustering algorithm on the given dataset. You may assume that the number of clusters is known (k=3). **You may use inbuilt scikit functions to implement your code.**
Submission: plot the results, using a different color for each cluster.

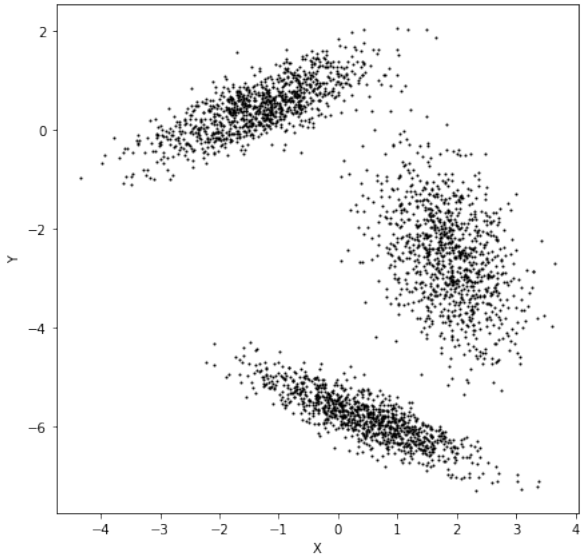

Figure 2: 2D clustering data

I. GMM (20 pts) Cluster the dataset by implementing Gaussian Mixture Model. You may assume that the number of components is known (k=3). (*Refer:* `https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html`)

Produce a color plot showing the clustering results for the data.

II. DBSCAN (20 pts) Cluster each of the dataset using DBSCAN clustering. You will need to tune the parameter values for `eps` to obtain appropriate results. (*Reference:* `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html`)

Produce a color plot showing the clustering results for the data.

III. KMeans (10 pts) Cluster each of the dataset using kmeans. Assume the number of clusters is 3. You can use the Euclidean distance as your distance measure. (*Reference:* `https://scikit-learn.org/stable/modules/clustering.html#k-means`)

Produce a color plot showing the clustering results for the data.