**Department of Electrical and Computer Engineering**

Homework Assignment No. 06:

**HW No. 06: Expectation Maximization (EM) and Hidden Markov Models (HMMs)**

submitted to:

Professor Joseph Picone

ECE 8527: Introduction to Pattern Recognition and Machine Learning

Temple University

College of Engineering

1947 North 12th Street

Philadelphia, Pennsylvania 19122

February 9th, 2023

prepared by:

Gavin Koma // Joe Arthur  
Email: [gavintkoma@temple.edu](mailto:gavintkoma@temple.edu) // [joe.arthur@temple.edu](mailto:joe.arthur@temple.edu)

# Task 1

The goal of the first task was to model each of the given classes as a mixture of multivariate Gaussian distributions and then to estimate the parameters of these distributions using EM. In order to begin, I first plotted the data to ensure that it was similar to the provided graphs. In order to do so, I wrote a simple function which passed the varying datasets to a plot. The code is as follows:

Text

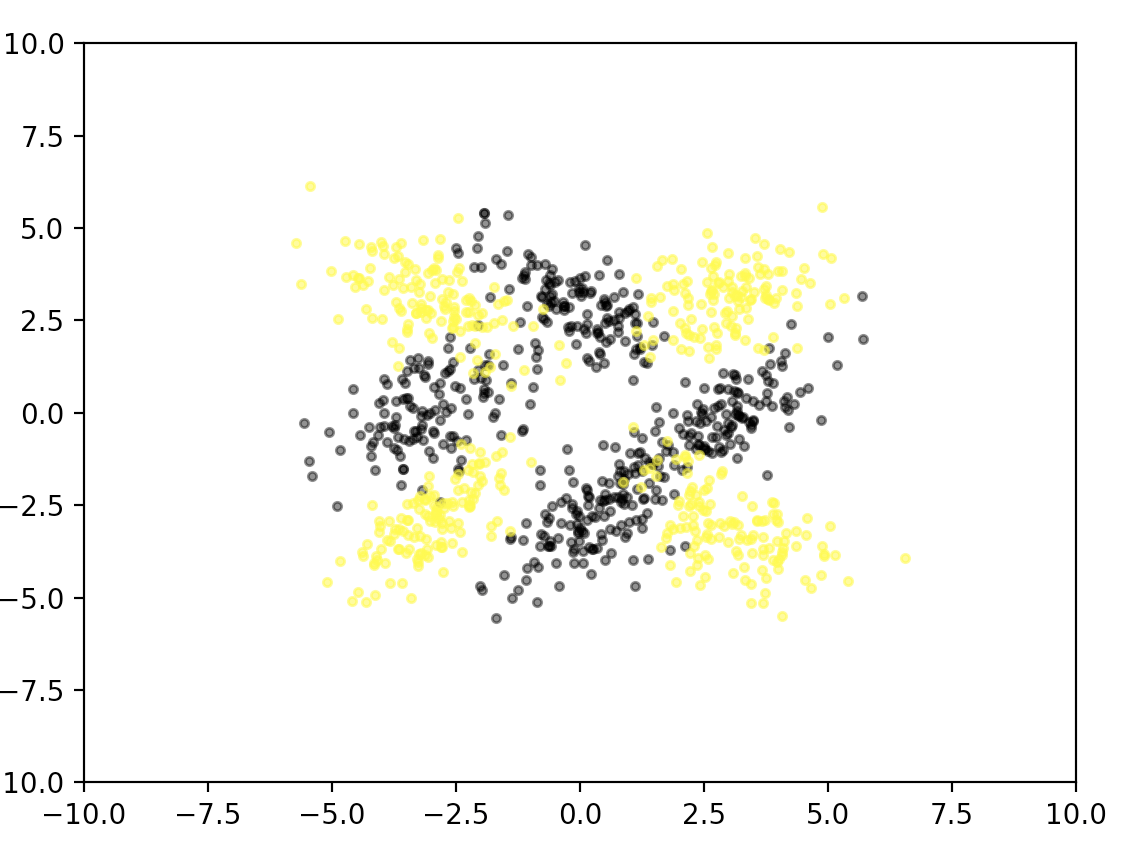
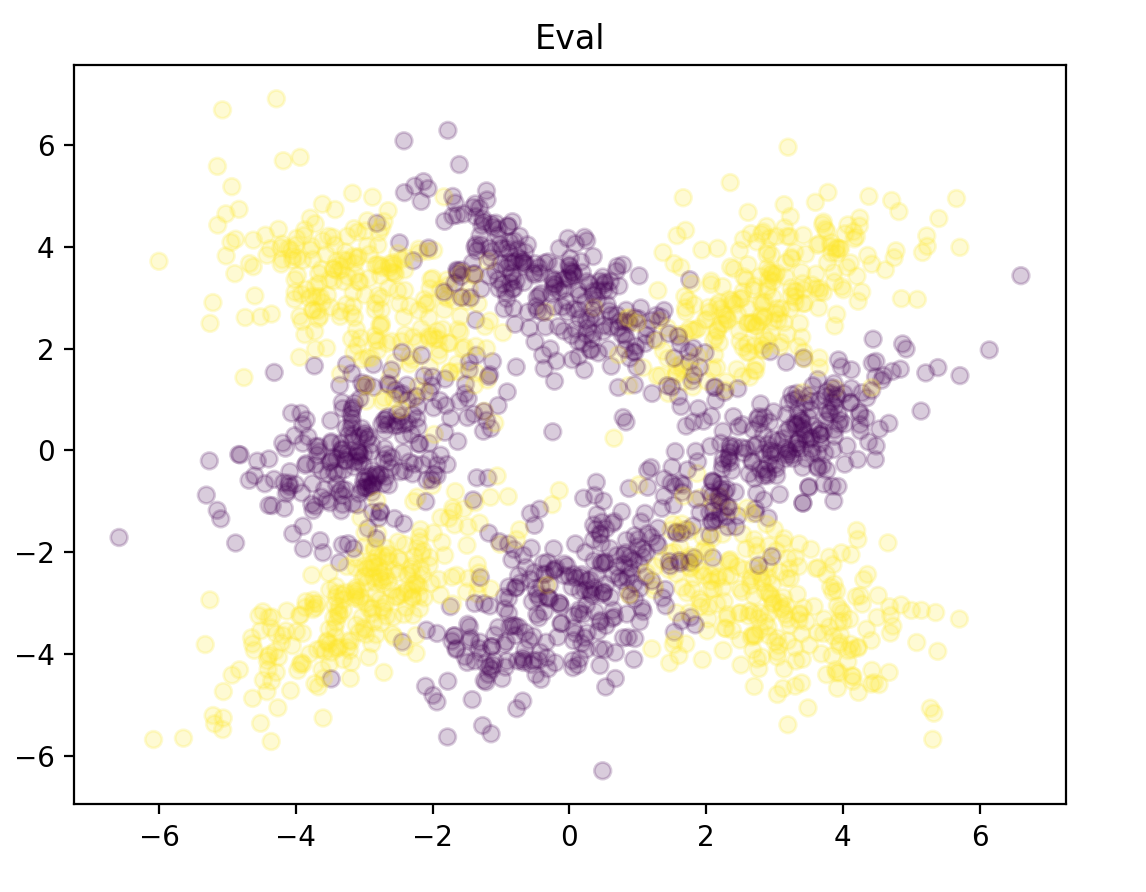
Description automatically generated

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generatedI have included my outputted graphs and attached them below in order to compare them to the provided graphs. One can see that they look the same. From here, I am comfortable with moving forward.

The implementation of Gaussian Mixture Models using python is fairly simple and only requires a few lines of code. However, we also need to show the boundaries that are estimated by this model. To do so, I implemented a similar strategy that I used when making the previous two plots. There are two main goals of using the Gaussian Mixture Model. First, to estimate the parameters of each Gaussian component that is within the Gaussian Mixture. The second is to determine which Gaussian component a data point will belong to.



Text

Description automatically generated

The implementation of the code was incredibly simple. I passed the same previous files that were used to plot normally and instead passed them through a Gaussian Mixture Estimator. Doing so provided me with the following graphs for varying values of n\_components.

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generatedFor n\_components = 1:

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generatedFor n\_components = 2:

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generatedFor n\_components = 4:

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generatedFor n\_components = 8:

# Task 2

Text

Description automatically generatedChart

Description automatically generatedChart, scatter chart

Description automatically generatedThe goal of Task 2 was to implement the same Gaussian Mixture method as previously, but with data that has more than two features. In this example, we are given a dataset with 5 features (making it a little hard to plot without some type of feature transformation). I went through this dataset in the same manner that I had previously. I used a for-loop that allowed me to train a model, pass the data, and output a corresponding plot. However, I felt that this was not entirely descriptive as the data points overlay on top of each other. The code that was used to implement this technique on the 5d data is as follows:

I realized quickly that viewing the boundaries with higher dimensional data would not be a satisfactory option. I did a bit more reading and found another method to assess n\_components and how to ‘judge’ our model. One such analytic technique is the Akaike Information Criterion (AIC). This method allows us to estimate the prediction error and thereby assess the quality of our model given our data. This is fairly easy to implement in python and only required a small bit of list comprehension to loop through our data. The code is as such and the graph that is outputted is attached below:

Graphical user interface, text, application

Description automatically generated

Chart, line chart

Description automatically generated

This graph gives us a good idea of *how many* n\_components we should include in our Gaussian Mixture to minimize probability of error. Here we see that at n\_components of about 8 is probably satisfactory and anymore will not necessarily aid in the prediction rate of our model.

# Task 3

Hidden Markov Models (HMMs) are used to model hidden Markov Processes. They are generally defined by 3 parameters:

1. Initialstate probabilities. A vector that describes the initial probabilities of the system when it is in a particular state.
2. Hidden state transition matrix. The rows of this matrix will correspond to one particular hidden state while the columns for each of the row will contain transition probabilities from one state to a new hidden state.
3. Observable emission probabilities. This is a vector that will describe the emission probabilities for a process given a state.

The article continues to explain HMMs with a simple example regarding moods. It really simplified HMMs in my mind and helped me to gain a better understanding of them. I will continue to summarize this article in the following lines while including lines of my code to not only further my understanding but to also provide myself with materials for studying with at a later date.

The article states that there are three general problems which are typically solved using hidden Markov Models.

1. If we are given a set of observations X and the 3 model parameters, then we should calculate the occurrence probability of the observations of X. We can do this by using a forward algorithm.
2. If we are given a set of observations X and the three model parameters and we want to determine the optimal set of hidden states Z that result in X, then we should use the Viterbi algorithm.
3. Finally, if we are given only a set of observations X and our goal is to determine the optimal set of model parameters. Then we can use the Baum-Welch algorithm to solve this.

To follow along with this blog post, the code was followed word for word and graphs were outputted as the author did as well. The graphs are slightly different in color scheme but show the same data. I am slightly curious as to variations in number output, but I wonder if any data has changed from the git database.

Our first graph shows and identical output and depicts the historic gold prices in USD as well as the change in the corresponding day change. We are better able model the daily change in the gold price and thereby better capture the changing state of the market. One important thing to note from this article is the reason for choosing 3 hidden states: we expect **at the very least** for there to be three different measure of daily changes (low, medium, and high).

Chart

Description automatically generated

We are able to confirm our states by viewing the output below. Here we see three states, given in no particular order. Another important thing to note is found in our transition matrix. The values in the diagonal are considered large as opposed to the smaller values that are outside of the diagonal. This allows us to know that our model prefers to remain in the state that it is already in.

Text

Description automatically generated

The final portion of this task required us to output an image that shows market volatility when it has been modeled using a Gaussian emissions Hidden Markov Model. Our model varies slightly in that our state 1 is blue and state 0 is orange while the author of this paper has state 1 as orange and state 0 as blue. This causes it to look slightly different but the outputted states from previous sections show that our model would most likely start in state 1.

Chart, scatter chart

Description automatically generated

An image of the short code was included below:

Text

Description automatically generated

Text

Description automatically generated

# Conclusion

I genuinely enjoyed this homework assignment. A lot of the work that we’ve done so far has always included applications of the algorithms of some manner and it’s been really helpful for me to be able to visualize these ideas. I think the difference for this assignment was the article that was attached and offered further explanation while looking at the code. Normally, I have to deduce the assumptions included in a library or figure it out through trial and error, but it was really nice to have the code and explanation next to each other. I think it definitely helped to facilitate my understanding of *how* as well as *why* the Gaussian Mixture Model works the way it does.