Machine Learning: Assignment 1

1 Overview

I implemented the Expectation-Maximization (EM) algorithm for two-dimensional Gaussian mixture models. Specifically, the implementation is for data that is generated by exactly two Gaussian functions. The goal of the EM algorithm is to find the optimal values for six parameters (the Gaussian $k \in \{1, 2\}$): $P(k)$, the relative frequency of generating the data, and (μ_k, σ_k^2), the mean and variance for the Gaussian. The EM algorithm contains two steps that iterate until the parameters converge. The first step uses estimates of the parameters to determine the likelihood that a group of data points was generated by one Gaussian versus the other. The second step uses these probabilities to re-estimate all of the parameters.

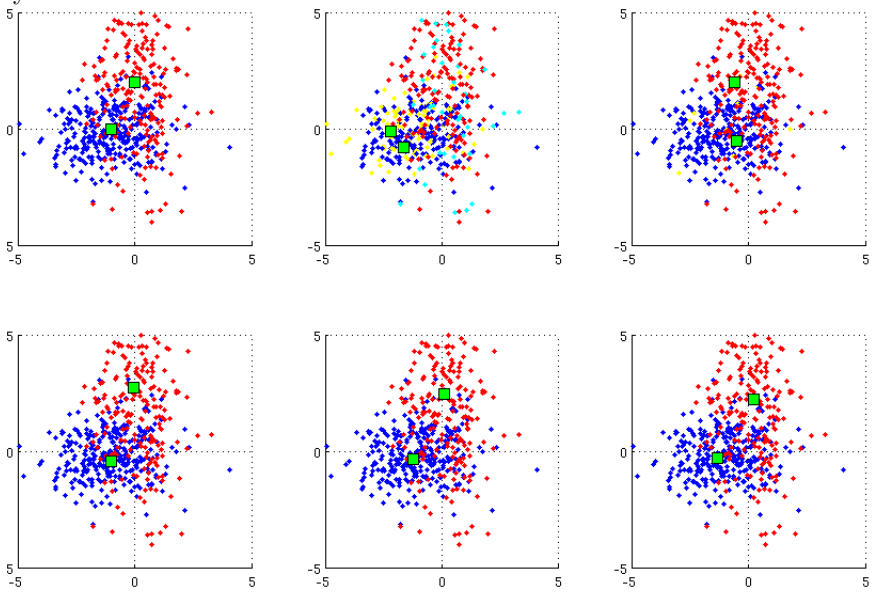
2 Experiments

Each experiment was run for 100 trials, randomly generating new data for each trial. Additionally, for each experiment, I ran the K-Means clustering algorithm on the same data. Each following section describes those results as well. Diagrams were left off the "above and beyond" experiments for brevity.

2.1 Experiment 1: High Overlap

The first experiment I ran evaluated the EM algorithm on a mixture model of two Gaussians that have relatively close means. I chose this setup because I wanted to see how well the algorithm would handle situations where the data points were heavily co-mingled spatially, but with dissimilar variance.

My experimental setup used two predefined Gaussian functions (μ, σ^2): Gaussian A was defined as $(\begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1.9 & 0.4 \\ 0.4 & 1.2 \end{pmatrix})$ and the second was defined as $(\begin{pmatrix} 0 \\ 2 \end{pmatrix}, \begin{pmatrix} 1.0 & 0.1 \\ 0.1 & 5.0 \end{pmatrix})$. Additionally, I set the prior weights P so that 60% of the data would come from Gaussian A and 40% from Gaussian B. For the data, I randomly generated 50 groupings of data points with 10 points in each group. Of course, all of the points in a single group were generated by the same Gaussian.



The average accuracy after 100 trials was an excellent 97.9%. One such test run is exemplified by the following diagrams. The first graph in the figure showed the correct classification along with correct means. The subsequent graphs show iterations 1-5 of the EM algorithm where yellow dots show "blue Gaussian" points incorrectly labelled as "red" and cyan dots show "red Gaussian" points incorrectly labelled as "blue". The algorithm starts by randomly picking two points to serve as the means, using the identity matrix as the variance matrices, as assuming that half the data was generated by each Gaussian. In the first iteration (cell 2), you can see one of the means is quite far from its final position and, as a result, many of the data points are mislabelled. However, after re-estimation, the "red" mean is much closer to its target and more of the points are labelled correctly. By the third iteration, all the points are labelled correctly and for the rest of the algorithm, the means simply make minor adjustments to get closer to the true values.

2.1.1 Comparison to K-Means

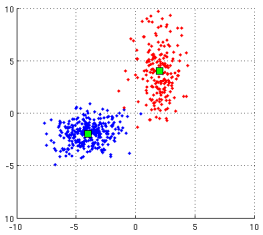
The K-Means algorithm yielded a substantially lower 78.2% accuracy. This difference can be attributed to the fundamental differences between EM and K-Means. I designed this experiment to specifically target this distinction. The K-Means algorithm is very bad at handling overlapping data points. This is because it is only able to classify a point based on its distance from the estimated means. When the data overlaps, there is no clear line that can be drawn to separate those points that are closest to one mean versus those points that are closest to another.

On the other hand, EM does much better on the overlapping data. This is because the strength of EM lies in the fact that it is able to incorporate underlying assumptions about how the data was generated. The algorithm as implemented assumes that the data was generated by exactly two Gaussians. Using this knowledge, it can make a better guess as to the likely source of the data because it can look at the trends among the points in a single grouping. The fact that the data is overlapping is only of minimal significance because it only affects how far the points are from the means of the Gaussians.

It is also worth noting that both EM and K-Means seemed to do better on the same data sets and worse on the same data sets indicating that the difficultly of optimizing the parameters might be intrinsic to the data in such as a way that both techniques are either helped or hurt.

2.2 Experiment 2: Increased distance between means

This experiment moved the means to $[-4; -2]$ and $[4; 2]$, respectively, without changing the variances. Putting additional distance between the Gaussians makes the problem easier because the distributions are less similar and, therefore, it is easier to determine from which distribution a particular point was generated. Indeed the EM algorithm takes advantage of this and is able to achieve 100% accuracy on all trial runs.

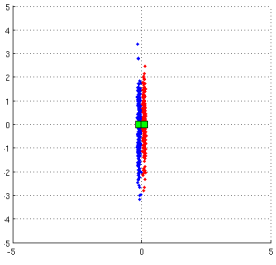


2.2.1 Comparison to K-Means

This experiment was designed to "level the playing field" with K-Means. Since K-Means is appropriate for data that can be grouped by distance, putting more space between the clusters means a clearer distance-based demarcation. As is shown in the diagram, there are very few points in the "overlap" region. As a result, the K-Means algorithm achieves 99.6% average accuracy.

2.3 Experiment 3: Close means, close yet predictable Gaussians

This experiment was used to gauge how well EM performs when the parameters of the Gaussians are nearly identical, but very predictable. The parameters used to generate the data were $(\begin{pmatrix} -0.1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0,001 & 0 \\ 0 & 1 \end{pmatrix})$ and $(\begin{pmatrix} 0.1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0.001 & 0 \\ 0 & 1 \end{pmatrix})$. As is shown in the diagram, this generates two parallel almost-perfectly-vertical bars with almost-overlapping means. The idea here is that even though the points of both Gaussians are very close together spatially, since the variances are so extremely predictable, it should be easy for the EM algorithm to find them. In fact, the accuracy for the EM algorithm was a perfect 100% for all trials.



2.3.1 Comparison to K-Means

Since the K-Means algorithm cannot take the Gaussian trends into consideration, it must cluster using only distance. Since the distances are so close, this is very difficult. As a result, K-Means achieves only 50.1% accuracy.

3 Experiment 4: Fixing a correct answer

This experiment addressed the question of what would happen if one of the Gaussians' parameters were known to the algorithm. For this set I hacked my implementation to hard-code the correct answer for one of the Gaussians. My hypothesis was that performance would increase because there would be more valuable information. However, what I actually saw was accuracy decrease to a much lower 85.9% using the same parameters as Experiment 1. The reason for this drop is that, based on the random start of the second Gaussian's mean, the algorithm can become confused about which set of parameters applied to which points. Since the algorithm doesn't know *a priori* which region of points applies to which Gaussian, it relies on the ability to shift the parameters when necessary. When the non-fixed mean is on the "wrong side" of the fixed one initially, then the algorithm tries in vain to move it to the Gaussian it is actually closest to. This issue could be easily overcome by running a few iterations of the algorithm with the "floating" mean starting on different sides of the fixed one. The trial that resulted in the floating mean moving the most would likely be correct.

Of course, if the correct parameters are fixed for both Gaussians, the accuracy increases; the average in my trials was 99.5%.

4 Code

My code is split among several files, but the script that should be executed to run a test is called `main.m`. At the top of the file you will find a second containing settings that serve as the parameters for the experiment.

4.1 Known Deficiencies

Since this was my first time using Matlab, I struggled a bit to get the hang of vectorization. You will notice that for-loops are prevalent in my code. After running all of my experiments I went back to try to vectorize things, but ended up completely botching the code to the point where it was doing the complete wrong thing: as it iterated, the accuracy goes up and then all of a sudden starts diving.

The code I'm submitting also fails to update the prior weights, P. When I added this part in I again got the "goes up, then crashes and burns" effect. So this version just assumes that the priors are 50/50 and doesn't adjust. It works ok on my 60/40 split, but might not work on something more extreme.

Finally, the code I'm submitting doesn't use the weight estimations when adjusting the means and variances: it simply makes a hard decision as to which cluster the group belongs to and adjusts the parameters based on that. This has the unfortunate side effect that if all the groups are assigned to the same cluster, then the program crashes trying to estimate the parameters. In such a case, I just re-initialize the parameters to random starting points and loop back around. Again, tried to fix $\Rightarrow$ crash and burn.

If you're interested in seeing my attempt to fix these issues, I'd be happy to provide you the code. But I figured you'd want whatever you received to match the output described in this document, so that's why I've included the older-but-still-working version.