EECE 5644 Homework 3

Owen McElhinney

March 2020

Abstract

This homework assignment compares a multi-layer perceptron network to statistical classification using Gaussian Mixtures. The first section will overview the MLP used and the parameter optimization, while section two will focus on the GMM. Both classifiers were tested on the same data set

Contents

1 Multi-layer Perceptron Network

This section will overview the methods and results for tests done using a multi-layer perceptron network. The first section will overview the input data and how it was generated, section two will go over the Multi-layer perceptron network, section three will discuss process used for parameter selection, and section four will review the results of the classifier for varying amounts of input data.

1.1 Data Generation

The test data used for this classifier, as well as the one discussed in Section 2, were drawn from 'ring distributions.' Figure 1 shows an example of 1000 points drawn from this distribution. This assignment will test this distribution with training sets of [100, 500, 1000] points drawn from this set.

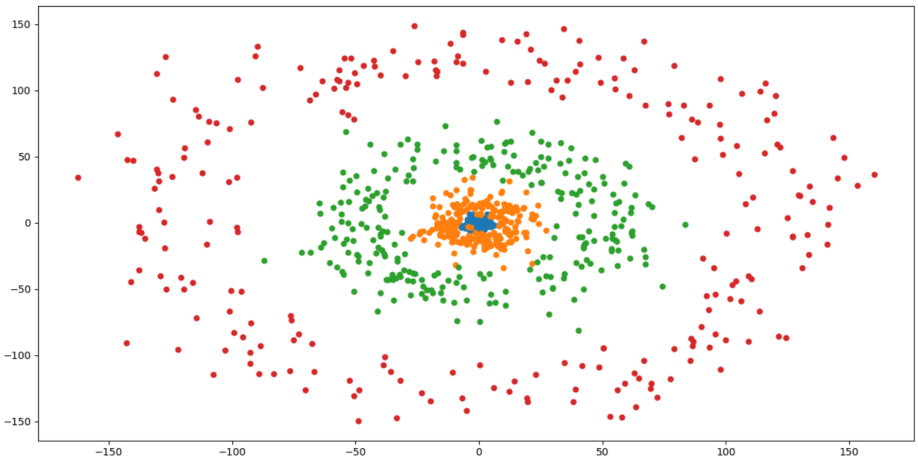

Figure 1: Example of test data with 1000 points drawn.

Each point is calculated as $(x, y) = (r * cos\theta, r * sin\theta)$. The angle θ is distributed uniformly over the interval $[0, 2\pi)$.The radius r of each point is drawn from a Gamma Distribution with shape parameters $k = L^3, \sigma = 2$, where L is the class label. In other words, the classes form progressively larger circles, as is observed in Figure 1. The classes are also chosen to have equal class prior probabilities.

1.2 Multi-layer Perceptron Network

As previously stated, this problem uses a Multi-layer perceptron network as a classifier. The MLP used consisted of two fully-connected layers with a third softmax layer to condition the output.

For this model, we have several parameters that must be optimized. In this problem, we are asked to pick an optimal number of perceptrons in the first layer

as well as an activation function. The next section will overview the method by which these parameters were selected.

1.3 Paramter Selection

The problem required the optimal selection of both the number of perceptrons and and the activation function. The parameters were selected using a 10-fold cross validation method. This means that the training data is segmented in to 10 partitions and the network is trained on each of these 10 segments. For each trained network, it is then validated using the remaining 9 segments as testing data. The overall performance is taken to be the average of these 10 trials. This helps to get a better idea of the variance in accuracy that the network produces.

Per the assignment instructions, the perceptron size was tested from [1,6] and the activation function was tested on a sigmoid vs a soft-RELU function.

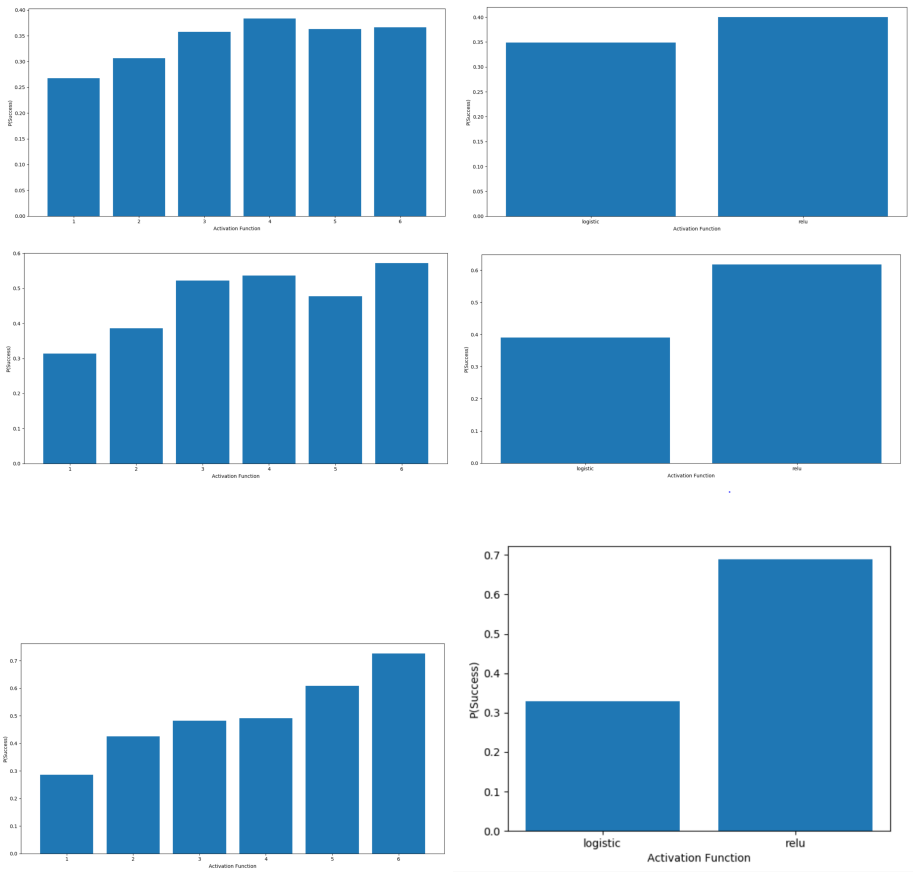

Figure 2: Results for the k-fold validation at all data lengths. Row 1 is results for 100 points, row 2 is 500 points, and row 3 is 1000 points.

Data Points	P(Error)
100	0.3808
500	0.0408
1000	0.0246

Table 1: Probability of error for the classifier under the various test cases

Figure 2 displays the results of this cross-validation for all tested data sets. The y-axis in these plots shows P(success), so the optimum parameter was chosen to be the one that yields the maximum value. For the number of perceptrons this varies slightly from four in the first case to six in the last two. That said, there benefit from adding more perceptrons begins to level off, and the difference from four to six is small. As for the activation function, the soft-RELU is always significantly better than the sigmoid function. The results of this cross-validation were used for the final models.

1.4 Results

The results of this classifier for each test case are tabulated in Table 1 and shown in Figure 3. It can be observed that the majority of the errors in the higher testing point cases lie in the transition regions between classes which is to be expected. The test case with only 100 data points has a lot of errors in the middle where the classes are close together. This shows that without enough data, the network can't accurately make a decision between overlapping cases, but as the input data increased it was able to improve that decision boundary.

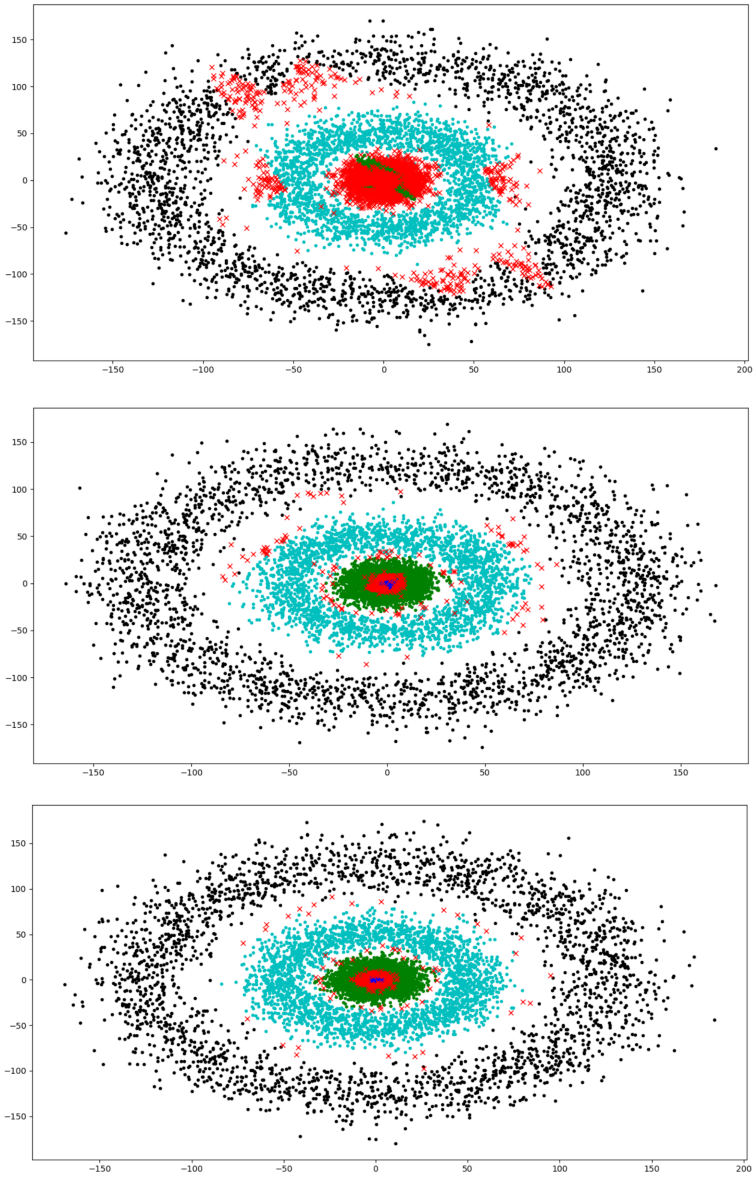

Figure 3: Results for all three cases. Incorrect classifications are marked in red. The figures are for the network trained with 100, 500, and 1000 points in descending order

2 Gaussian Mixtures

The second portion of this assignment designs a classifier using Gaussian Mixture Models (GMMs) for this same data set. Similar to the first part, there is

some freedom in parameter selection for model order that will have to be optimized as well as some general changes to the classification process. Following the structure of the first section, the first part of this will review the input data tested, the second section will overview the assumptions and free parameters in our GMM, the third section will look at how these parameters were selected, and the final section will provide the results for this classifier.

2.1 Data Generation

The input data for this classifier is generated in the same fashion as is outlined in Section 1.1. This classifier will similarly be evaluated on three test cases, being trained on 100, 500, and 1000 training points. Figure 1 shows an example of what this data set will look like.

2.2 Gaussian Mixture Model

A Gaussian Mixture Model is a distribution that assumes data is drawn from a collection of Gaussian distributions, each with their own respective priors. For example, one could have a distribution drawn from two Gaussians with different means, and points being more likely to come from the second distribution than the first.

For this problem, we will assume that each class label is a Gaussian mixture. This means that our overall distribution will have four separate Gaussian mixtures, each comprised of N distributions. To achieve the best results, we will solve a Model Order Selection problem in order to pick the optimum number of components N for each label. The process for this selection is outlined in the following section.

2.3 Parameter Selection

As previously discussed, a model order selection problem will be run for each class label to improve classifier performance. The remainder of this section will discuss the methods used in solving this problem.

The program begins by pulling just the data from a particular class label (L) out of the training. Set L is then partitioned using a K-fold procedure. For the case with 100 data points K was set to five because the data set was too small for ten partitions and a K of 10 was used for the remaining procedures.

Each of these partitions were then used to find a GMM using the Expectation Maximization algorithm. This mixture was then evaluated on the remaining partitions to get an average log-likelihood for the model order. This process was then repeated for model orders from one to six.

At this point, we have the mean negative log-likelihood for each model order taken from a K-fold cross validation procedure. We select the model order for this class label to be the one that minimizes the negative log-likelihood. This process is then repeated for the next class label until each class has it's optimum mixture parameters

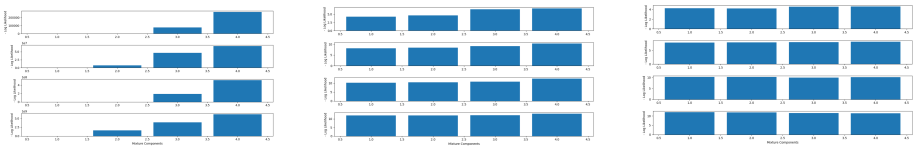

Figure 4: Results of Model Order Selection for each of the Four Classes with 100, 500, and 1000 data points from left to right.

The results of this process are shown in Figure 4. For the 100 point case, a model order of one was selected for all four labels. As more data is added, the difference between model orders becomes smaller, and the larger exterior classes (bottom two labels) tend to prefer three to four components in the mixture. The smaller two labels always pick one centered mixture. From the the smallest radius outward for all three cases, the chosen model orders were [1, 1, 1, 1], [1, 1, 1, 3], and [1, 1, 3, 4] for the 100, 500, and 1000 point cases respectively.

2.4 Results

Data Points	P(Error)
100	0.0834
500	0.039
1000	0.0283

Table 2: Probability of error for the classifier under the various test cases

The results for this classifier are tabulated in Table 2 and shown in Figure 5. As expected, the classifier improves as more training data is added. All of the errors lie in the boundary between two sets which is where we would expect the classifier to struggle. A larger percentage of the errors appear to happen between the smaller classes, but this is just due to the scaling of the image.

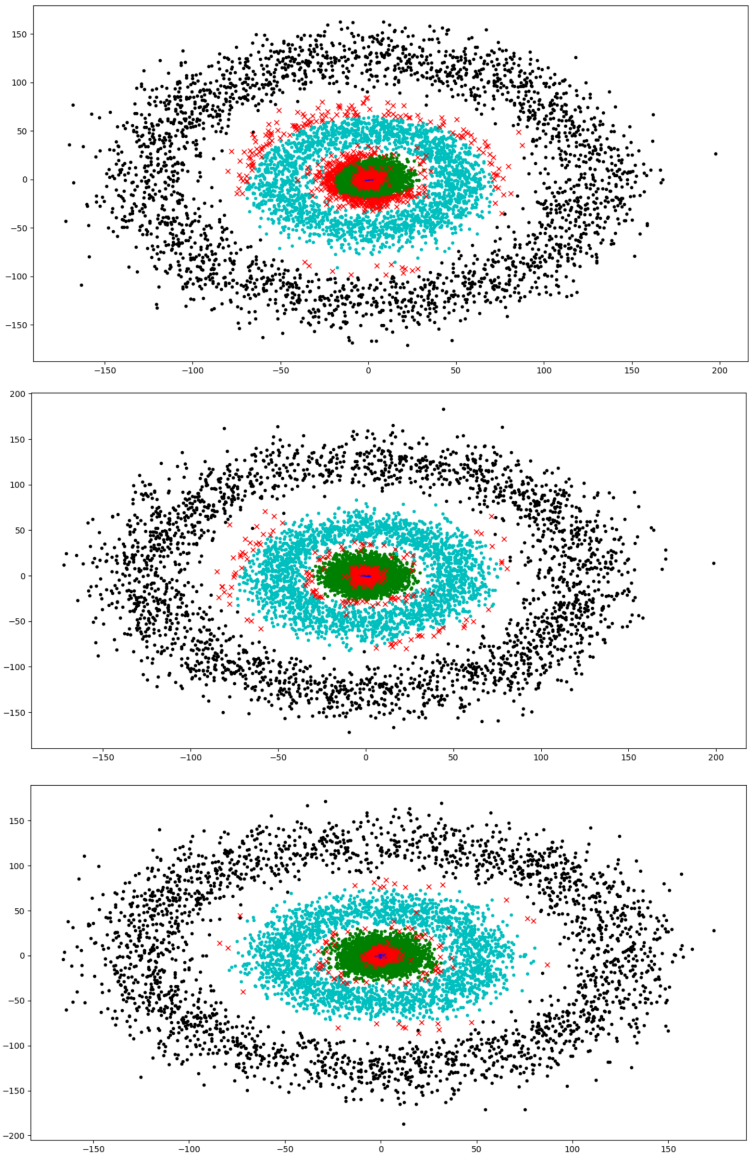

Figure 5: Results for all three cases. Incorrect classifications are marked in red. The figures are for the network trained with 100, 500, and 1000 points in descending order

3 Conclusions

Table 3 shows the results of the classifiers under all test cases for comparison. The Multi-Layer Perceptron struggled greatly in the case with limited input

Data Points	P(Error)	
	MLP	GMM
100	0.3808	0.0834
500	0.0408	0.039
1000	0.0246	0.0283

Table 3: Comparison for the probability of error for the MLP versus the GMM

data, but slightly outperformed the Mixture Models with more training data. While Neural Networks are an amazing tool, this training data dependency is their largest drawback.

It is similarly interesting to note the types of errors for the two classifiers. In the test case with 100 training samples, the GMM still has all it's errors in the regions between classes. On the other hand, the Neural Network has error along entire arcs of a distribution. While it's impossible to truly know why these points were wrongly classified, I suspect that these correspond with gaps in the training data. Entire cones trigger an error because there was no training data from that direction.