EECE 5644 HW4

Owen McElhinney

Spring 2020

Abstract

This assignment is assignment

Contents

1 Question 1

Question one explored using a Multi-Layer Perceptron network to find the minimum Mean-Square Error estimate for a regression problem. The estimator guesses the second coordinate of a vector from the first one.

The first part of this section will go over the training data that the model was fit to and how it was generated. The second section will overview the network architecture used in the problem. The third will go over the selection process for the hyper-parameters in the network. Finally, section four will review the results and accuracy of the model.

1.1 Data Generation

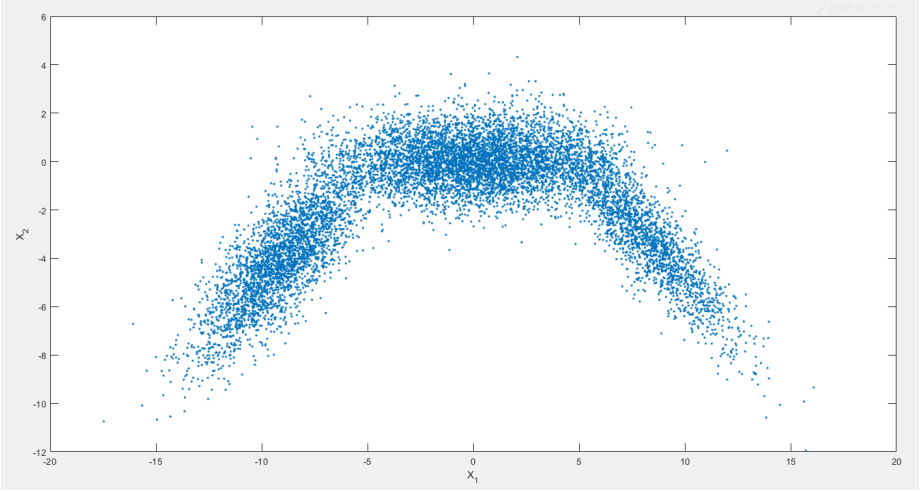

Figure 1: Training data used for Question 1

Pictured in Figure 1 is the data set used to train the model for this question. This data set is generated from three Gaussian distributions. The prior probabilities of each class are [0.3, 0.5, 0.2] from left to right on the figure.

1.2 Network Architecture

For this problem, a 3-Layer Multi-Layer Perceptron (MLP) network was used. The layers were two fully-connected layers of adaptive weights and one non-linear activation function between them. The only activation function tested for this network was a SmoothReLU function. The network was trained using the Mean-Square Error as the cost function. The number of perceptrons in each layer is selected using the method described in the next section.

1.3 Parameter Selection

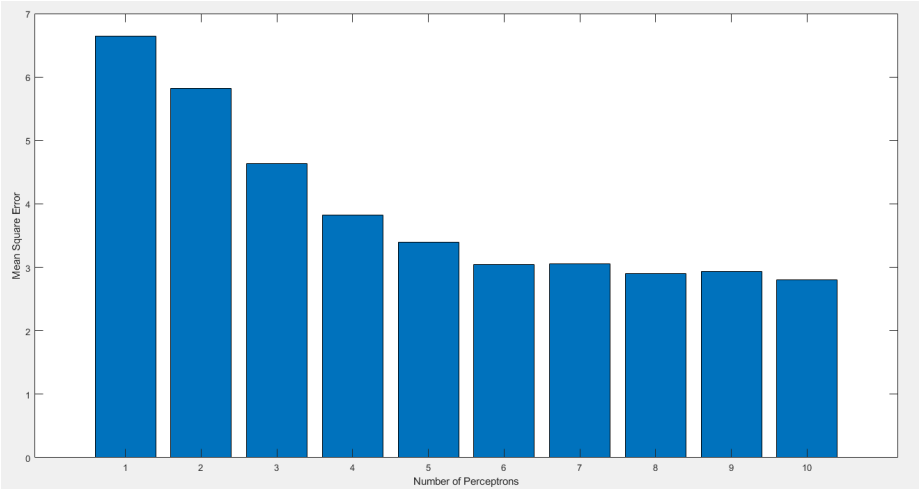

Figure 2: Cross-Validation results for question 1. Shows the mean MSE achieved for each number of layers. Optimal number of layers is chosen to be the one that minimizes the function.

The optimal number of perceptrons in the weight layers must be selected. In order to find them, the script uses a 10-fold cross validation technique.

This algorithm divides the training data in to ten equal partitions. A network is trained on each of the partitions and validated on the other nine. The overall performance of the classifier is then taken as the mean of these values. This helps to give a better idea of the range of performance the classifier could see and filter out it getting stuck in a local minima. As such, the reported MSE values reported in Figure 2 are averages.

For this scenario, it was observed that adding perceptrons beyond six per layer yielded diminishing returns. The network converged to a theoretical minimum MSE for this regression case. Ten perceptrons was found to give the minimum by a small margin, and was therefore used in the final model.

1.4 Results

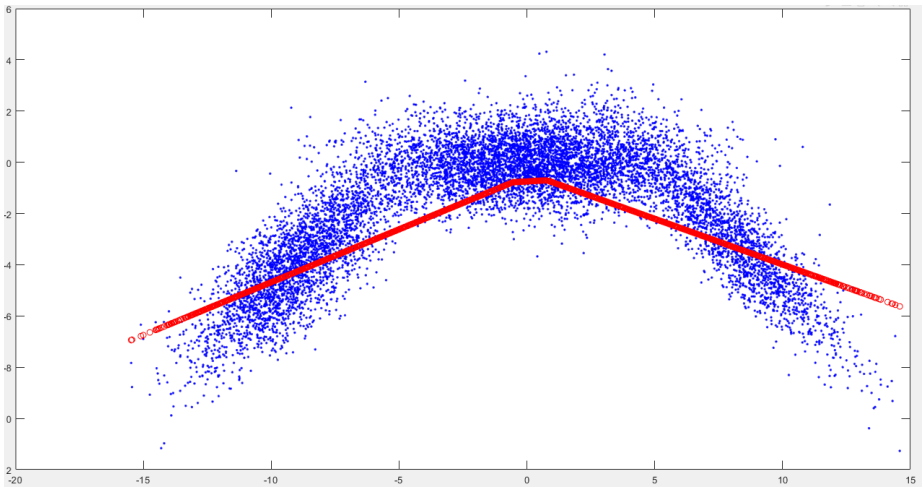

Figure 3: Results for Question 1. The estimates are displayed in red.

The results for the finalized network are shown in Figure 3. The red line shows the converged to minimum-MSE function that the network found. It can be observed that this function is basically a curve going through the mean of each of the three distributions. This aligns with the theoretical solution.

2 Question 2

Question 2 explored using a Gaussian Kernel Support Vector Machine (SVM) for classification. It was tested on the circular ring data set from the third assignment. The SVM had two hyper-parameters which were trained using 10-fold cross validation to maximize the probability of correctly classifying.

The first section will review the data set being tested on and how it was generated. Section two reviews the theory and equations used in the SVM. Section three will review the hyper-parameters for this classifier and how they were selected. Finally, section four will show the results for the final classifier.

2.1 Data Generation

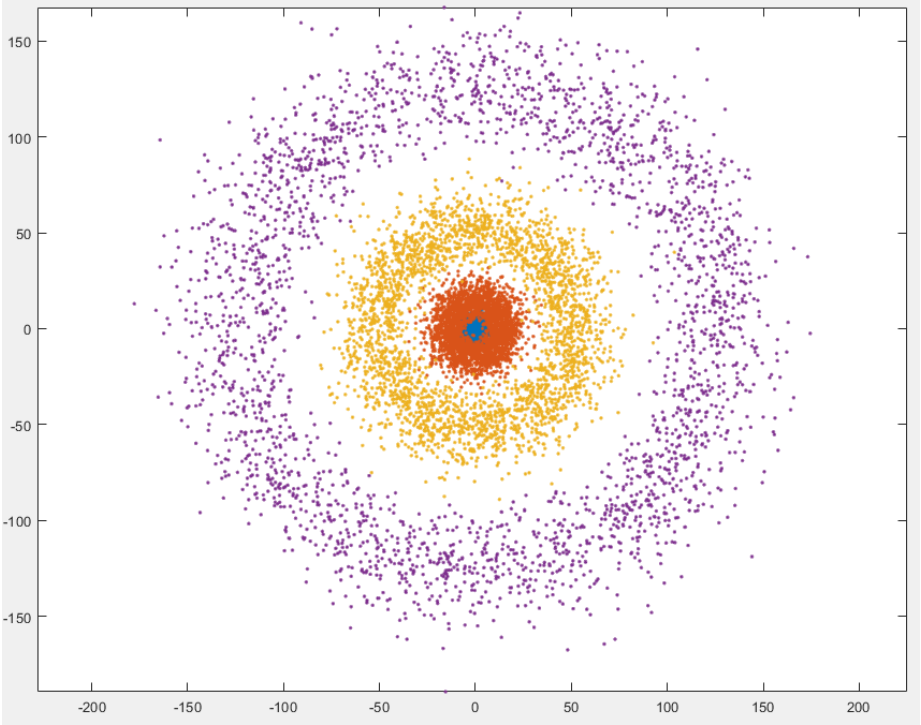

Figure 4: Dataset used for this part of the problem. This is the same data set for assignment 3.

Figure 4 shows the data that this classifier was built for. The data is generated from four class labels. Each point in the set is modeled as (x, y) = (r $\cos(\theta)$, r $\sin(\theta)$). The angle is distributed uniformly from [0, 2π]. The radius is drawn from a Gamma distribution where the scale parameter is given by the class label cubed.

2.2 Support Vector Machine

The classifier used for this question was a support vector machine with a spherically symmetric Gaussian kernel. Support vector machines are a supervised learning technique that uses the data points the data points close to the transition between classes (called Support Vectors) to train decision boundaries. These boundaries are found by solving the dual optimization problem given by

$$maxf(c_1, ..c_n) = \sum C_i - 0.5 \sum \sum y_i c_i K(x, x_j) y_j c_j \tag{1}$$

For this case with circular data, we will use the Gaussian Kernel defined as:

$$K(x, x') = exp(\frac{||x - x'||^2}{2\sigma^2}) \tag{2}$$

This solution uses Matlab's Statistics and Machine Learning toolbox to implement the described functions.

The implementation of these functions requires the selection of two hyper-parameters. The first is the sigma in Equation 1. This parameter controls the smoothness/ sharp features of the classifier. The second parameter is C, the box parameter. This controls how heavily wrong classifications are penalized. The next section will outline how these parameters were chosen.

2.3 Parameter Selection

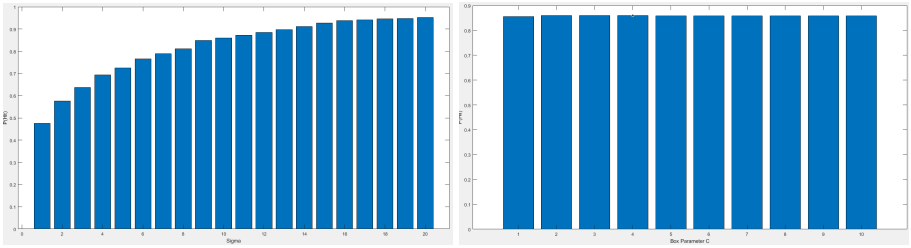

Figure 5: Results of the 10-Fold cross validation for question 2. Results for the Sigma parameter are on the left and results for the box parameter is on the left.

As stated in the previous section, the two hyper-parameters to be tuned are the Sigma and Box parameters. The sigma parameter controls the smoothness of the Gaussian boundaries, and the box parameter determines how errors in the model are penalized.

These parameters were selected using a 10-fold cross validation technique. For more specifics on the implementation of this process, see section 1.3. The important thing to note is that the reported values are averaged over partitions of the entire data-set.

From Figure 5 we observe that the Sigma parameter has much more influence on the accuracy of the classifier than the box parameter. The most important

thing to note is that a poorly chosen sigma can halve the accuracy. The value that gave the maximum from these graphs were chosen for the final model.

2.4 Results

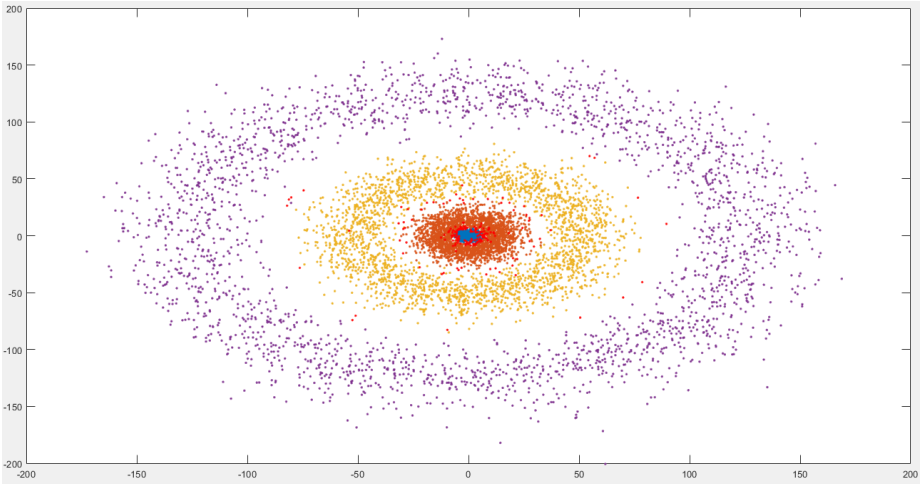

Figure 6: Results for the final classifier. Erroneous classifications are shown in red.

The final classifier was trained on the results from section 2.4. The results of this classifier are shown in Figure 6. The classifier was able to achieve a P(Error) of 0.014.

The errors for this classifier all fall in the transition regions between classes. These were what we allowed from the box parameter. Perhaps the accuracy could be further improved by adding more training data or further complicating the kernel to change the decision boundaries, but a 98 percent accuracy is satisfactory.

3 Question 3

Question 3 explored using Gaussian Mixture Models for image segmentation. This question will take two images, one of a bird and one of a plane and use an extracted feature space to cluster them with Gaussian Models. The first part of the question uses two classes for clustering and the second part uses cross validation to select an optimal number of classes.

The first subsection will discuss the images and the features extracted from them. Subsection two will discuss the Gaussian Mixture Model and the algorithm used to optimize it. Finally, subsection three will show the final image clustering results.

3.1 Image Data

Figure 7: Original Images used for segmentation

The original image are shown in Figure 7. Each pixel in the image has three features for Red, Green, and Blue intensities respectively. For this assignment, we will add two additional features, one for the x value of the pixel and one for the y value of the pixel. This adds spatial proximity to our clustering in addition to color clustering. Each of these features is divided by it's maximum value to normalize them all to be between zero and one.

3.2 Gaussian Mixture Models

As stated in the introduction, this clustering was done using Gaussian Mixtures. This takes the data and fits K Gaussian Distributions to the data. Each of these must find a mean and covariance to fit each cluster.

The optimal clusters were determined using Matlab's gmdistribution function. This uses the Expectation Maximization algorithm to fit a mean and covariance to the data.

The only parameter that must be manually chosen is the number of clusters. For the first part of the problem, a base of two clusters were used. In the second part, the optimal number of clusters must be selected. The process for doing so will be discussed in the following section.

3.3 Parameter Selection

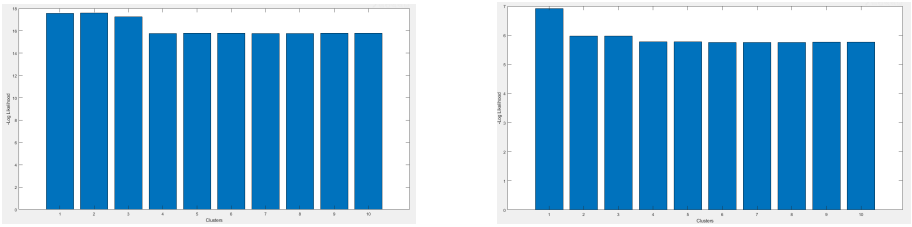

Figure 8: Cross-validation results for the two images. The graphs have number of clusters on the x-axis and negative log likelihood on the y-axis.

As previously stated, the only parameters that must be given to this script is the number of clusters for the final models. This number of clusters was chosen using 10-fold cross validation using negative log-likelihood as our measure of fit. As such, the number of clusters that yielded the minimum value was selected. Figure 8 shows that there is very little benefit to using more than four clusters for this problem. The log-likelihood flattens out beyond 4 or five clusters.

3.4 Results

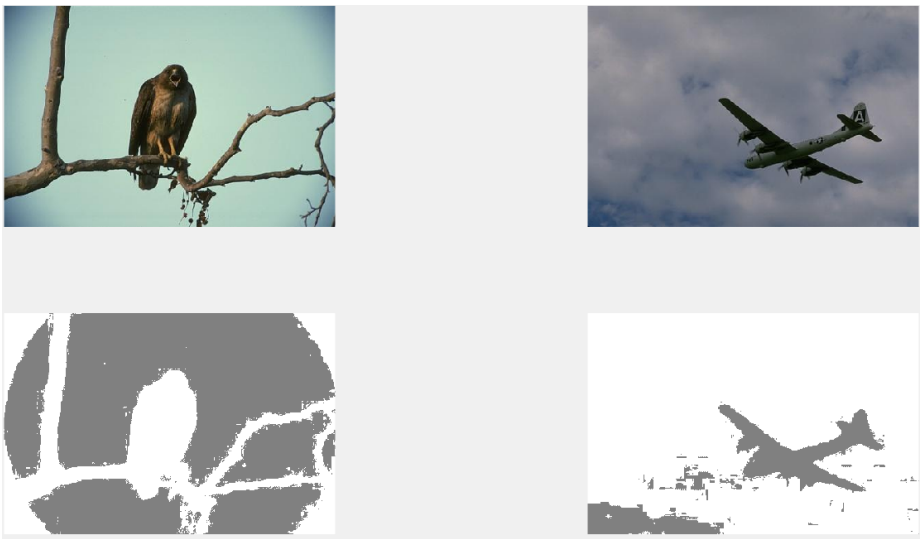

Figure 9: Results for the first part of the question with two clusters.

The results of this clustering procedure are shown in Figure 9 and Figure 10. The first shows the results of the image segmentation with two clusters and the second shows results using the optimal number of clusters from the procedure

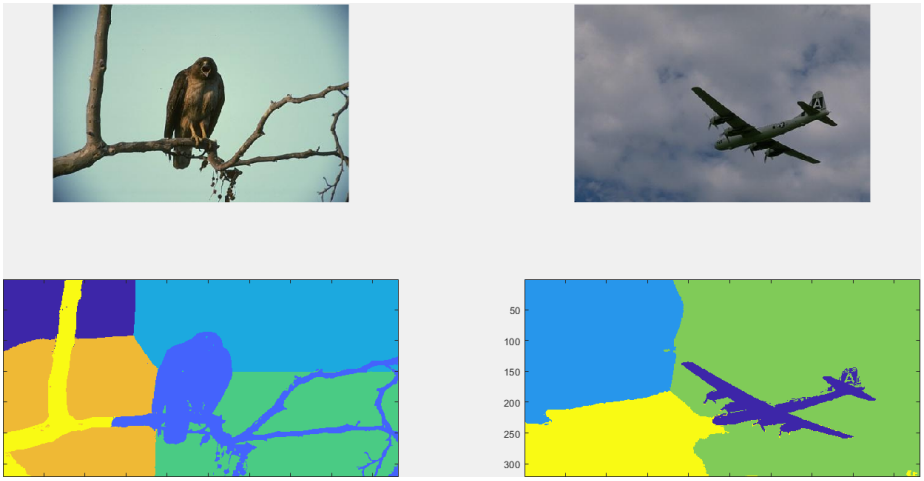

Figure 10: Results for the second part of the problem where number of clusters was chosen from 10-Fold cross val.

described in Section 3.3. The usefulness of the spatial features is immediately apparent from the results, as the output clusters are neatly grouped and well connected

4 Appendix

All of the code used to create this document can be found at
https://github.com/o-mcelhin/eece-5644/tree/master/Homework_4