Luc Rulinda

Nov. 29th, 2021

Neural Network project 2

ECE 62900- Neural Networks

Project 2

**Task 1**

Timeline

Description automatically generated

Fig 1: Illustration of an example of feature vectors (ESPN) that categorize different players

The task at hand consisted of designing a neural network that classifies various points in two categories using M- dimensional feature vectors. In other words, grouping entry data points between two group categories using various characteristics (feature vectors.) In our case our groups are basketball and football players, and the vector features used are height, weight, and BMI. The data that was used was collected from Sports Encyclopedia of Pro Football and Official NBA basketball Encyclopedia. Below, in table 1 is the data that was used to generate normal distribution of our points.

Table

Description automatically generated

Table 1: Means and standard deviation of data used for normal distribution

The table above corresponds to the two vector feature data used in our project. Figure 2 illustrates the distribution of the 1200 data points that were generated. On its right is figure 3 that illustrates a three-dimensional feature vector.

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

Fig. 2: Two-dimensional F.V plot Fig. 3: Three-dimensional F.V plot

The data points above were used to train our neural network by implementing the use of the “Patternet” function in Matlab. The latter returns a pattern recognition network (feedforward network) that can be trained to classify inputs according to target classes. The Variable Learning Gradient descent, “traingdx” was used as training function because it allows for one to set the learning rate in experimenting with the data. “crossentropy” was used as our performance function.

**Task 1.1**

A picture containing text, clock

Description automatically generated

Fig. 4: Two-layer neural network architecture (30-50 neurons per layer)

We designed a two-layer network and increased the number of neurons in the hidden layer from 5 to 50 with a step size of 5. Figure 4 gives an overview of the architecture of our neural network. The following are graphs that illustrate the classification error of our tested network.

Fig. 5: Classification error across various number of neurons

Fig. 6: Average classification error across various number of neurons

Figure 5 shows the variation of the classification error across five trials and figure 6 the average of the five trials. One thing that is immediately apparent from our classification error graph of our network is the narrow range of classification error obtained. It varies between .0096 and 0.011. This can be attributed to the precision and accuracy of our network; however, it makes it almost impossible to notice any trendline despite averaging numerous trials.

One can also note that the lowest classification error was observed when the number of neurons in 25 (figure 6). Therefore going forward 25 neurons in every layer will be used for optimal results.

**Task 1.2**

We experimented with the use of multiple hidden layers in our network by increasing the number of layers from one to five. Every single layer network that was used contained 25 neurons because it was the optimal number of neurons as discovered from figure 6. Below is the network architecture of our neural network of three, four and five hidden layers.

Diagram

Description automatically generated

Fig. 7: Network architecture of hidden layer network iteration (25 neurons each)

Our network was tested using 10 trials and their performance was illustrated in figure 8.

Fig. 8: Performance across neural network of various hidden layers

Fig. 9: Average performance across neural network of various hidden layers

Figure 8 shows the performance of the average of the trials across the five hidden layers. The performance does not follow a notable relationship with number of hidden layers; however, one notices that the best performance was observed at hidden layer 3.

The performance in the figures above can be compared to those that illustrate the classification error to find any proof of agreement that three is the optimal number of hidden layers for our network.

Fig. 10: Classification error variation across various number of hidden layers

Fig. 11: Average classification error variation across various number of hidden layers

Just like for the performance of our multiple hidden layers, the classification error data has a narrow data range. An observation that can be made with the naked eye is that the classification error is lowest at three hidden layers. Three layers also corresponds to the layer number with the highest performance as observed in figure 9. Figures 9 and 11 compliment each other as there is an indirect relationship that can be observed between the two of them because as the performance increases across number of layers, the classification error seem to decrease.

**Task 1.3 and 1.4**

In our experiment we varied the feature vectors of various dimensions. So far, we have only been using two features for our professional sports players- height and weight. It is time to add a third feature that will help us classify our data points between the football or basketball group category. Below is the network architecture of the 2D and 3D feature vector side by side.

Graphical user interface, application

Description automatically generated Graphical user interface, application

Description automatically generated

Fig. 12: 2-dimensional Network architecture Fig. 13: 3-dimensional Network architecture

One can see that our network has 3 hidden layers with 25 neurorns each. This setup has been the most effective as observed from the performance and classification errors above.

Chart, line chart

Description automatically generated Graphical user interface

Description automatically generated with low confidence

Fig. 14: Best validation performance of 2D F.V Fig. 15: Best validation performance of 3D F.V

Chart, waterfall chart

Description automatically generated Chart, waterfall chart

Description automatically generated

Fig. 16: Error histogram of 2D F.V network Fig.17: Error histogram of 3D F.V network

Graphical user interface, application, table

Description automatically generated Application, table

Description automatically generated

Fig. 18: Confusion matrix of 2D F.V network Fig.19: Confusion matrix of 3D F.V network

Fig.20: Performance of both 2D and 3D feature vectors across five trials

Fig.21: Classification error of both 2D and 3D feature vectors across five trials

From the error histograms ( figures 16 and 17) one can tell that the two dimensional neural network has more errors. By comparing the performance plots ( figures 14 and 15) we obsered that the 2D neural network reaches the best validation performance of 0.022305 at epoch 171, while the 3D feature vector reaches its best validation perfomance of 0.0093105 at epoch 102. From this comparison the two dimensional neural network presents a better performance.

Performance and classification error data in both the two dimensional and three dimensional feature vectors were collected and illustrated in figures 20 and 21. The two dimensional feature vector network presented a better performance, while the 3D network shows lower classification error. The latter implies that the 3 dimensional feature vector has more accuracy than the 2 dimensional one. It makes sense that the accuracy would increase as the number of dimensions of our feature vector increases because the network has more information to consider for classification.

**Task 1.5**

We kept the three hidden layer neural network with 25 neurons each for our experiment with the learning rate.

Fig.22 : Neural network performance across various learning rates

Fig.23: performance across various learning rates

Fig.24 : Neural network classification across various learning rates

Fig.25 : Neural network average classification across various learning rates

We were unable to observe a specific trend in both average performance and classfication error graphs. This is due to the high accuracy of our neural network. To observe a higher error fluctuation and a considerable trend in our performance would require to modify the average and standard deviations of our feature vectors (height, weight and BMI). However, that would be tampering with our initial data that we obtained from reliable sources. Next time this data will have to be modified in order to illustrate a trend in the error and performance.

**Task 2**

In order to approximate the function g(p) = 100 + cos(pi\*p/12) + sin(pi\*p/5)), we designed our neural network to have a single input value (p) .We trained our network using the *traingd* (gradient descent) training function provided by the deep network toolbox in Matlab. For the performance function we used the Mean Square Error (MSE) function to measure the network performance.

**Task 2.1**

A picture containing text, clock

Description automatically generated

Fig. 26: Network architecture of two-layer network with 50 neurons each

Figure 1 shows the network architecture of the two-layer network with 50 neurons. Figure two below shows the average approximation error obtained from three trials.

Fig. 27: Average approximation error across three trials

One observation that can be made from the approximation error results is that the error increases as the number of neurons in the hidden layers increases

**Task 2.2**

Diagram

Description automatically generated

Figure 28: Network architecture of a 5 hidden layer network with 5 neurons each.

Figure three shows the network architecture of neural network with 5 hidden layers. Figure 29 is the approximation error obtained from testing across 5 trials.

Fig.29 : Average approximation error across various number of hidden layers

Fig.30: Average approximation error of the five trials

By taking the average of the percentage approximation error of five trials we obtained figure 30 that illustrates the trend across the hidden layers in our experiment. As the number of hidden layers increases the approximation error decreases, therefore our neural network got more accurate as the hidden layers were increased. There is no sign of overfitting yet. Our network behaves the best when there are five hidden layers.

**Task 2.3**

Previously our neural network behaved best with 5 hidden layers (figure 5) and 35 neurons in each hidden layer(figure 27), therefore this optimal setup should be used in experimenting with various learning rates. Figure 31 represents the network architecture of our experiment.

Graphical user interface, application

Description automatically generated

Figure 31: Network architecture for the learning architecture.

Our gradient descent training function in matlab, “traingd”, allowed us to vary our learning rate and below (figure 32) were the approximation error data that resulted. The learning rate range we experimented with is 0.01 to 0.5, in three trials.

Fig. 32: Average percentage approximation error across various learning rates (0.01- 0.5)

Fig.33 Average approximation error across learning rate 0.01- 0.1

Given that the approximation error results in figure 32 presented a lot of fluctuations, we narrowed down our learning to 0.1 in figure 33 to get a better understanding of the behavior of our neural network. In figure 8 one notices an increase in the error as the learning rate increases, especially after the learning of 0.05. This can be attributed to the fact that the neural network is not as diligent in training and testing the network as the learning rate increases. Therefore, an increase in the approximation error was to be expected.