

**CZ4042 Neural Networks and Deep Learning**

**Project 1 Report**

**Classification and Regression using Neural Networks**

Mittal Madhav (U1822408H)

**Part A : Classification Problem**

**Introduction**

This project aims at building neural networks to classify the Cardiotocography dataset containing measurements of fetal heart rate (FHR) and uterine contraction (UC) features as either “Normal (N)”, “Suspect (S)” or ”Pathological (P)”.

Keras is a very high-level language. The parameters that we choose for training the artificial neural networks can have an immense effect on their training. Thus, in this assignment, we try to deepen our learning of these parameters, and how changing them changes the behavior of the neural network. The goals of this problem are :-

1. Designing a simple feedforward neural network with some specified parameters.
2. Finding out the optimal batch size by training the same neural network on different batch sizes.
3. Finding out the optimal number of hidden neurons by training the same neural network on different numbers of hidden neurons.
4. Finding out the optimal decay parameter by training the same neural network on different values of decay parameters.
5. Implementing 3-layered and 4-layered neural networks and comparing the results.

**METHODS**

**Dataset**

The dataset consists of 23 columns, out of which the first 21 are input attributes and the last 2 columns are output values. Only the last column is used for the classification task, and it contains values of 1,2 and 3 for N, S and P, respectively.

The dataset consists of 2127 rows, in which the first row consists of nan values, and the rest 2126 rows contain cardiotocograms values, which are used to predict the corresponding NSP values.

**Preparing the Dataset**

1. The data is first uploaded using genfromtxt
2. The nan row and the first output columns are first removed from the dataset, as they are not useful for the computations. The second output column’s values are reduced by 1, to ensure the output values of 0,1 and 2, for ease of computation.
3. Then, the dataset is divided into a 70:30 train-test ratio using sklearn’s train\_test\_split() function. The train data is used by the neural network to learn the patterns, , and the test data is used to see how the model performs in the real-world scenario.
4. As it can be seen from the dataset, some of the data features have high values, and some data features have high ranges of values.   
     
   High values of data can result in a model that learns large weight values. This means that the resulting model may have poor performance during learning and may be sensitive to input values - resulting in higher generalization error. A target variable with a large spread of values, in turn, may result in large error gradient values causing weight values to change dramatically, making the learning process unstable.  
     
   Scaling input and output variables is thus a critical step in using neural network models. This is achieved using sklearn’s MinMaxScaler function.

**K-Folds Cross Validation**

Part A of this project uses 5-Fold Cross Validation to train its model.

Several regularization techniques are used to avoid overfitting of data during training. Cross-validation is such a technique that provides a check on how it is performing on a test data. Cross-validation is used to test the generalizability of the model.

The best way to improve the performance of the system without compromising much is to use a small part of the training data itself to validate, as it might give us an idea of the model's ability to predict unseen data. However, selecting the perfectly random training data from the entire dataset, in order to not learn any characteristics particular to that data might prove to be challenging. This is where K-Folds comes in.

Diagram

Description automatically generated

In 5-Folds Cross Validation, the dataset is divided into 5 folds, out of which 4 folds are used for training and 1-fold is used for testing purposes. This is repeated until all folds are used as the test set once. This way, it provides a good idea of the generalization ability of the model, especially when we have limited data and cannot afford to split into test and training data.

This project uses sklearn’s StratifiedKFold function, which ensures a fair and easy division of data into ‘K’ folds.

**Experiment**

**Question 1**

This question requires us to create a simple feedforward neural network with the following layers :-

1. An input layer
2. One hidden layer of 10 neurons, with ReLU activation function
3. An output softmax layer

These layers have the following specifications for parameters :-

1. A learning rate (α) of 0.01
2. A weight decay parameter (β) of
3. A batch size of 32
4. **Using the training dataset to train the model and plot accuracies on training and testing data against training epochs.  
     
     
   Shape

   Description automatically generated**
5. **State the approximate number of epochs where the test error begin to converge**

**A picture containing shape

Description automatically generated  
  
  
Chart, bar chart

Description automatically generated**It is clear from the graph that the sharpest drop in loss takes place before epoch 250, after which the loss starts to stagnate. Although the train loss goes down slowly, but consistently with epochs (even till epoch 5000), the test loss stagnates. Thus, there is little merit in training data after epoch 500 (an additional 300 epochs are added to account of variational errors).

Thus, we set EPOCHS = 500

**Question 2**

The main objective of this question is to find the optimal batch size by training the neural network and evaluating the performances for different batch sizes

1. **Plot cross-validation accuracies against the number of epochs for different batch sizes. Limit search space to batch sizes {4,8,16,32,64}. Plot the time taken to train the network for one epoch against different batch sizes  
     
   Chart

   Description automatically generated  
     
     
    Chart, bar chart

   Description automatically generated**
2. **Select the optimal batch size and state reasons for your selection.**There is a trade-off between using smaller and bigger batch sizes in terms of computational speed and validation accuracy. On one hand, using a big batch size leads to faster training, and thus less computational cost. This is because larger the batches, lesser the portions the dataset is divided into, and lesser is the overhead of loading them.   
     
   On the other hand, using bigger batches generally lead to lower validation accuracies. This is because with larger batch sizes, the estimated gradient is close to what the gradient of a full dataset would be. This prevents it from escaping local minima. Smaller batches have noisy gradients, which can help them escape the local minima. However, too small batches may just jump around indefinitely in the presence of noisy data, not being able to learn anything.   
     
   Thus, as we increase the batch size, the training time decreases, and the validation accuracy decreases. We can clearly see this concept realized in the above figures.   
     
    A picture containing icon

   Description automatically generated  
     
   A batch size of 8 has the highest validation accuracy, and only takes 50% of the training time required by the batch size of 4. The reason for this may be explained by the theory written above. Thus, a batch size of 8 is taken for the rest of this experiment.
3. **Plot the train and test accuracies against epochs for the optimal batch size.**

**A picture containing shape

Description automatically generated**

**Question 3**

The main objective of this question is to find the optimal number of hidden layer neurons by training the neural network and evaluating the performances for different number of hidden neurons.

1. **Plot the cross-validation accuracies against the number of epochs for different**

**number of hidden-layer neurons. Limit the search space of number of neurons to {5,10,15,20,25}**

**Shape

Description automatically generated**

1. **Select the optimal number of neurons for the hidden layer. State the rationale for your selection.  
     
    A picture containing bar chart

   Description automatically generated**As the numbers of neurons in a hidden layer increase, the layer’s ability to learn more complex functions increases. Therefore, as we increase the number of neurons from 5 to 10, the validation accuracy increases. However, it can be seen that validation accuracy decreases as we go from 10 to 15 neurons. Increasing the number of hidden units and/or layers may lead to overfitting because it will make it easier for the neural network to memorize the training set but that does not generalize to unseen data. Thus, a hidden neuron layer size of 10 is taken for the rest of this experiment.
2. **Plot the train and test accuracies against epochs with the optimal number of**

**neurons.  
  
  
A picture containing shape

Description automatically generated**

**Question 4**

The main objective of this question is to find the optimal decay parameter by training the neural network and evaluating the performances for different decay parameters.

1. **Plot cross-validation accuracies against the number of epochs for the 3-layer network for different values of decay parameters. Limit the search space of decay parameters to {0, 10−3, 10−6, 10−9, 10−12}**

Shape

Description automatically generated

1. **Select the optimal decay parameter. State the rationale for your selection**  
     
    A picture containing bar chart

   Description automatically generated  
     
     
   By applying L2 regularization on our cost function, we penalize the model for having high weight values and thus prevent the model from overfitting the training data.Here, decay parameter (λ) is the parameter that must be optimized.   
     
   As it can be seen from the graphs, since the λ = gives the highest cross-validation accuracy. Thus, a decay parameter of is taken for the rest of this experiment.
2. **Plot the train and test accuracies against epochs for the optimal decay parameter  
     
    A picture containing shape

   Description automatically generated**

**Question 5**

This question asks us to build a 4-layered neural network, with the following specifications

1. 2 hidden layers with 10 neurons each
2. Batch size of 32
3. A weight decay parameter (β) of
4. **Plot the train and test accuracy of the 4-layer network**   
     
    A picture containing shape

   Description automatically generated
5. **Compare and comment on the performances of the optimal 3-layer and 4-layer**

**Networks**It can be noted that the test accuracy of the 4-layered neural network is more than that of the 3 -layered network we managed to find in Question 4. This is because increasing the number of hidden layers increases the ability of the neural network to learn more complex functions and thus more complex characteristics of the dataset.

1. **[EXTRA] We now alter the parameters of the 4-layered architecture to match the optimal parameters from the previous questions, and plot the graph  
     
     
    A picture containing shape

   Description automatically generated**   
   As can be seen, the testing accuracy of the optimal 4-layered network (0.911) is greater than the testing accuracy of the non-optimal 4-layered network (0.904). This indicates how important it is to train your network with optimal parameters.

Thus, the model with

* Batch size : 8
* Hidden Layer Neuron Count : 10
* Decay Parameter : 0.001
* Layers :

has the highest test accuracy of 0.9106582999229431

**Part B : Regression Problem**

**Introduction**

This project aims at building neural networks to predict the chances of approval of a student’s application for a Master’s Program, given a list of features like GRE Score, TOEFL Score etc.

The goals of this problem are :-

1. Designing a simple feedforward neural network with some specified parameters, and stopping training automatically after convergence is reached
2. Implementing Recursive feature elimination (RFE) to remove unnecessary features.
3. Implementing multi-layered neural networks, introducing drop-out layers between the layers, and comparing the results.

**Dataset**

The dataset consists of 9 columns, out of which the columns 2-8 are input attributes and the 9th column is the output column. The input columns contain features like different scores, university ratings etc, whereas the output column contains the chances of admission based on these input features.

The dataset consists of 401 rows, in which the first row consists of nan values, and the rest 400 rows contain student’s academic records and ratings, which are used to predict the corresponding chances of admission.

**Experiment**

**Question 1**

This question requires us to create a simple feedforward neural network with the following layers :-

1. An input layer
2. One hidden layer of 10 neurons, with ReLU activation function
3. An output linear layer

These layers have the following specifications for parameters :-

1. A learning rate (α) of 0.001
2. A weight decay parameter (β) of
3. A batch size of 8
4. **Use the train dataset to train the model and plot both the train and test errors against epochs.**

**Shape

Description automatically generated**

**Fig : No of epochs vs MSE for starter model**

1. **State the approximate number of epochs where the test error is minimum and use it to stop training.**It is clear from the graph that the MSE drops the most a in the initial epochs, and then it drops slowly, but consistently. To stop the execution on its own, the Keras callback method called ”EarlyStopping” is used. The MSE is carefully monitored, and a patience level of 200 is chosen. Patience level adds a delay to the trigger in terms of the number of epochs on which we would like to see no improvement.

To find the optimal value of x, we plot the frequency graph of MSE. As the number of epochs increase, the MSE decreases and slowly stagnates. From the frequency graph, we can intuitively find the frequency (and thus the epoch #) at which the MSE starts to stagnate.  
  
Chart, pie chart

Description automatically generated  
  
From the pie chart, it can be seen that from a MSE of 0.003700000001117587 (0.0037 approx.), it starts to stagnate. After 0.0037, the MSE reduces very slowly.

From the training history of the model, we can find the epoch at which the MSE is 0.0037.  
  
Thus, we can either force stop at epoch 1400, or we can find the frequency of 0.0037, and set the patience level such that the execution stops before reaching that point. We have found that the frequency of 0.038 is 200. To stop the execution slightly before 0.037, the patience level is set to 200.  
  
**A picture containing shape

Description automatically generated**

**Fig : EarlyStopping Plot**

As we can see, the training was stopped at 1270 epochs.

1. **Plot the predicted values and target values for any 50 test samples.**

Chart, scatter chart

Description automatically generated

**Fig : Predicted vs actual regression values for starter model**

Given the small dataset we have to train on, the graph shows that the predictions are reasonably close to the actual value, showing that the model performs well.

**Question 2 - Recursive feature elimination (RFE)**

RFE is a feature selection method that removes unnecessary features from the inputs. We remove all the features one-by-one and train the models with 6 features each. We then find the model with the best MSE. Repeat the same steps for removing one more feature from the best 6-feature model and compare the 5-feature model with the 7 and 6-feature models.

We first plot the correlation matrix to see how the features are related to each other and the output.

Table

Description automatically generated

It should be noted that even though high correlation between input features and output is better for the model, having high inter-feature correlation can be bad for the model as it may give more weight to similar features and cause over-fitting. The best regression models are those in which the predictor variables each correlate highly with the dependent (outcome) variable but correlate at most only minimally with each other. Such a model is often called "low noise" and will be statistically robust.

From the correlation matrix plot, it can be seen that "Research" has the lowest correlation with "Chances of admission". Although Research has lowest correlation with output, it also has lowest correlation with the other features, so removing it might also cause harm, but given its extremely low correlation with the output, it is safe to say that it will be one of the least desired features.

It is thus likely that RFE level 1 will remove Research. It is important to note the performance of different feature is highly dependent upon on the stochastic nature of training, as well as the inter-feature correlation, and thus it is hard to exactly predict the worst feature, as it may be different for every seed value.

**RFE Level 1 – Removing one feature from a 7-feature dataset**

We first iteratively remove one feature each from the original dataset and train the models. We then compare the models in the absence of each feature, and the removal of whichever feature gives us the best model, is removed.

Shape

Description automatically generated

**Fig : Plotting the train MSE's of models without different features on the same plot**

From the figure and code above, it can be concluded that removing Research gives us the least MSE (0.00414), i.e., the most accurate model. As it was stated before, research has a fairly low correlation with the chances of admission, and thus it has the highest chances to be removed.

Thus, we remove Research from the feature set and continue with Level 2 RFE

**RFE Level 2 – Removing one feature from a 6-feature dataset**

We repeat the same process of removing a feature from the dataset acquired after removing the Research column.

Shape

Description automatically generated

**Fig : Plotting the train MSE's of models without different features on the same plot**

As we can see, it is clear that removing CGPA gives us the least MSE (0.0047), i.e., the most accurate model. Thus, at the second level RFE, we remove CGPA from our feature set and continue.

Now that we have completed the RFE, we can compare the performances of the model with 5,6 and 7 features. This comparison will be done final MSE reached by the training dataset.

Icon

Description automatically generated

**Fig : MSE’s of models with 5,6 and 7 features**

As we can see, as we are iterating over the model, and removing features, the training MSE is increasing. This could be attributed to the fact that even though some features have less correlation to output than others, still a correlation of 0.5 may hold some value, and removing those parameters may lead to loss of valuable information.  
  
Thus, we should continue with the original dataset, without removing any features.

**Question 3 – Designing multi-layered neural networks and introducing dropout layers**

Dropout layers help prevent overfitting in a model. During training, some number of layer outputs are randomly ignored or “dropped out.” This has the effect of making the layer be treated-like it has a different number of nodes and connectivity to the prior layer. This helps prevent co-adapting of neurons.

However, it must be noted that dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs.

**Designing 4 layered network**

These layers are defined with the best feature set that was selected from question 2, i.e., the 7-feature dataset. This model uses a learning rate of 0.001 and has 50 hidden layer neurons.

A picture containing shape

Description automatically generated

**Fig : 4-layer neural network training**

A picture containing shape

Description automatically generated

**Fig : 4-layer neural network with dropout layers training**

As expected, the training process has been made noisy due to the dropout layers. It can be noted that the train and test MSE values for a 4-layer network are considerably better when there are no dropout layers involved (0.00435 vs 0.0058).

**Designing 5 layered network with and without dropout layers**

A picture containing shape

Description automatically generated

**Fig : 5-layer neural network training**

A picture containing shape

Description automatically generated

**Fig : 5-layer neural network with dropout layers training**

As expected, the training process has been made noisy due to the dropout layers here as well. It can be noted that the train and test MSE values for a 5-layer network are considerably better when there are no dropout layers involved (0.0044 vs 0.0093).

**Comparing MSE's for different models**Since in Question 2, we got the best MSE with 3-layers, thus we compare this model with the 4 and 5 layered models with no dropout layers involved.

Icon

Description automatically generated

We can see that as number of dense layers increase from 3-4, the model is able to understand more complex (and thus more accurate) relationships, and so the MSE's descrease. However, the test MSE reduces for 5-layered structure. This may be because the increased layers were able to memorise the training set and overfit the data while not being generalizable to the validation set.

Thus, the model with

* Features : None removed
* No dropout layers
* 4 layers

has the lowest test MSE of 0.0044370675