CS 6375 Assignment 2: Artificial neural networks

Names of students in your group:

1. Mounika B (MXB210007)

2. Saketh Dasavathini(SXD190016)

Number of free late days used: 0

Part 2:

Dataset used: “Churn\_Modelling.csv”(https://github.com/bmounikareddy98/Machine-learning-assignments/blob/main/Assignment\_2/Churn\_Modelling.csv)

Note: For the purpose of code execution we have already hosted the dataset on github public repository, hence not required to download it.

Description:

The dataset used is related to banking. We have the independent variables as RowNumber,CustomerId,Surname,CreditScore,Geography,Gender,Age,Tenure,Balance,NumOfProducts,HasCrCard,IsActiveMember,EstimatedSalary. The dependent variable is “Exited(true/false)”. Based on the independent features, we determine whether a person leaves a bank or not. This can also be used to determine the reliability of the person, whether loan can be approved for the customer or not. We have created an artificial neural network model tensorflow library. Below explains in detail about each parts and the steps involved.

Artificial neural network

1. Data preprocessing is performed

Step 1: We have eliminated the first three columns rownumber, customer ID and surname as they are irrelevant to determine whether a person leaves a bank.

Step 2: We have checked the null and duplicate values. There are no null and duplicate records.

Step 3: We have handled the categorical data present in Gender column. Label encoding is done to divide them into categories as the order matters, only label encoding is done, one hot encoding is not performed. One hot encoding is performed on Geography column which has values France, Spain and Germany to handle hierarchy/order of the values present in these columns.

Step 4: Feature scaling is performed to normalize the data present in independent features array(X).

This ends the data pre-processing phase.

1. Building the artificial neural network

Step 1: We have imported Sequential model from Tensorflow’s keras library.

Step 2: We have added an input layer and two hidden layers. We have used different hyperparameters such as :

Activation functions used in hidden layers: relu, tanh in hidden layers

Step 3: We have added an output layer.

Activation function used in output layer: Sigmoid as it must output either 1 or 0.

We have changed the number of neurons each time in the hidden layers and tried different activation functions.

Step 4: We complied the neural network. We used adam optimizer which implements stochastic gradient descent for reducing the loss. We used loss as binary cross entropy as the output is binary(0 or 1) and metrics as accuracy to calculate the accuracy of the model.

Step 5: We have trained the neural network on the training dataset each time with different number of epochs(100,120,150,160,etc) and a batchsize of 32.

Step 6: After the training is done, we have predicted the output on testing dataset and also calculated model’s output on training data as well. Using the confusion matrix, accuracy is calculated.

Step 7: Root mean squared error on the training and testing datasets are calculated.

Note: The steps are repeated with different combinations of hyperparameters and model history is noted. Please find below the plots of model accuracy according to number of epochs and number of neurons used in hidden layers.

**Step 3: Plots between different combinations of hyperparameters and model accuracy**

**Plot between model accuracy and number of epochs when activation function is tanh and number of neurons is 6 for training data**

Chart, line chart

Description automatically generated

**Plot between model accuracy and number of epochs when activation function is tanh and number of neurons is 6 for testing data**

Chart, line chart

Description automatically generated

**Plot between model accuracy and number of epochs when activation function is tanh and number of neurons is 10 for training dataset**

Chart, line chart

Description automatically generated

**Plot between model accuracy and number of epochs when activation function is tanh and number of neurons is 10 for testing dataset**

Chart, line chart

Description automatically generated

**Plot between model accuracy and number of epochs when activation function is relu and number of neurons is 6 for testing dataset**

Chart, line chart

Description automatically generated

**Plot between model accuracy and number of epochs when activation function is relu and number of**

**neurons is 6 for training dataset**

Chart, line chart

Description automatically generated

**Plot between model accuracy and number of epochs when activation function is relu and number of neurons is 10 for training dataset**

Chart, line chart

Description automatically generated

**Plot between model accuracy and number of epochs when activation function is relu and number of neurons is 10 for testing dataset**

Chart, line chart

Description automatically generated

Observations:

How we got below observations:

- We have executed our algorithm with epochs of 100 initially and incremented by 20 each time and used activation functions as tanh and relu in hidden layers. We used the number of neurons as 6 and 10 in hidden layers.

- We calculated the training and testing accuracy and it is as below for the best combination of number of neurons in hidden layers=10, activation function=tanh and number of epochs=150

- On training data, we got accuracy of 87.3625, the RMSE is 0.224831264163957

- On testing data, we got the accuracy of 85.85, the RMSE is 0.123805100816595

Please find the below recordings of different hyperparameters combinations.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.No | Hyperparameters | Number of epochs | Training accuracy | Test accuaracy | Training error(rmse) | Testing error(rmse) |
| 1 | Input layer-> first hidden layer : number of hidden layer neurons :6, activation='relu' | 100 | 86.5 | 86.45 | 0.168637304 | 0.160958241 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :6, activation='relu' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |
| 2 | Input layer-> first hidden layer : number of hidden layer neurons :6, activation='tanh' | 100 | 86.975 | 86.55 | 0.197888955 | 0.167150432 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :6, activation='tanh' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |
| 3 | Input layer-> first hidden layer : number of hidden layer neurons :6, activation='tanh' | 120 | 87.05 | 86.35 | 0.202507636 | 0.154766051 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :6, activation='tanh' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |
| 4 | Input layer-> first hidden layer : number of hidden layer neurons :6, activation='tanh' | 140 | 87.1625 | 86.8 | 0.202507636 | 0.154766051 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :6, activation='tanh' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |
| 5 | Input layer-> first hidden layer : number of hidden layer neurons :6, activation='tanh' | 150 | 87.0875 | 86 | 0.204816977 | 0.133093386 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :6, activation='tanh' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |
| 6 | Input layer-> first hidden layer : number of hidden layer neurons :10, activation='tanh' | 100 | 87.3 | 86.1 | 0.204816977 | 0.133093386 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :10, activation='tanh' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |
| 7 | Input layer-> first hidden layer : number of hidden layer neurons :10, activation='tanh' | 120 | 87.0625 | 86 | 0.204816977 | 0.133093386 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :10, activation='tanh' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |
| 8 | Input layer-> first hidden layer : number of hidden layer neurons :10, activation='tanh' | 150 | 87.3625 | 85.85 | 0.224831264 | 0.123805101 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :10, activation='tanh' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |
| 9 | Input layer-> first hidden layer : number of hidden layer neurons :6, activation='relu' | 120 | 86.2 | 85.7 | 0.18480269 | 0.139285576 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :6, activation='relu' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |
| 10 | Input layer-> first hidden layer : number of hidden layer neurons :6, activation='relu' | 140 | 86.925 | 86.25 | 0.18480269 | 0.139285576 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :6, activation='relu' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |
| 11 | Input layer-> first hidden layer : number of hidden layer neurons :6, activation='relu' | 160 | 87.05 | 86.6 | 0.18480269 | 0.139285576 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :6, activation='relu' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |
| 12 | Input layer-> first hidden layer : number of hidden layer neurons :10, activation='relu' | 100 | 84.85 | 84.45 | 0.067026308 | 0.037114439 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :10, activation='relu' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |
| 13 | Input layer-> first hidden layer : number of hidden layer neurons :10, activation='relu' | 120 | 84.5625 | 83.4 | 0.049321362 | 0.253918495 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :10, activation='relu' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |
| 14 | Input layer-> first hidden layer : number of hidden layer neurons :8, activation='relu' | 120 | 84.8 | 83.6 | 0.171716425 | 0.126901196 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :8, activation='relu' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |
| 15 | Input layer-> first hidden layer : number of hidden layer neurons :6, activation='relu' | 120 | 86.55 | 85.9 | 0.171716425 | 0.126901196 |
| first hidden layer-> second hidden layer : number of hidden layer neurons :6, activation='relu' |
| second hidden layer-> output layer : number of output layer neurons :1, activation='sigmoid' |

Which activation function performed best?

We are satisfied with our values. The model is trained with different combination of hyperparameters and the values are recorded. From these recordings we selected the best activation function and number of epochs and number of hidden layer neurons. **The best activation function in hidden layers is tanh and number of epochs are 150 and number of hidden layer neurons is 10**. The accuracy is 87.3625 on training dataset and 85.85 on testing dataset. The reason tanh performed well compared to relu because the architecture of our neural network contains only two hidden layers and tanh is better when the neural network is simple. Relu will work better when the neural network is deep and complex because activation is efficient and sparse in that scenario.