Assignment 04

Neural Networks and Fuzzy Logic

Submitted to : Dr Samabia Tehseen
Section : BSCS – 7A
Date : January 4, 2021
Submitted by : Abdullah Aslam & Ahsan Goheer
Enrollment no : 01-134172-005 & 01-134172-008

TABLE OF CONTENTS

1 PROBLEM STATEMENT

Neural Networks & Fuzzy Logic

Assignment # 4

Fall 2020

Submission deadline: 28th Dec 2020

An **arrhythmia** is a problem with the rate or rhythm of your heartbeat. It means that your heart beats too quickly, too slowly, or with an irregular pattern. Several tests can help your doctor diagnose an arrhythmia and monitor the effectiveness of your treatment. The most common test used to **diagnose** an **arrhythmia** is an **electrocardiogram** (**EKG** or **ECG**).

Computer -Aided diagnosis of **arrhythmia** is possible through different classification algorithms. UCI Machine learning repository provides the dataset for training and testing of such diagnosis. The aim of this data is to distinguish between the presence and absence of cardiac arrhythmia and to classify it in one of the 16 groups. Class 01 refers to 'normal' ECG classes 02 to 15 refers to different classes of arrhythmia and class 16 refers to the rest of unclassified ones.
This database contains 279 attributes. Below is the link of UCI arrhythmia dataset.

UCI Machine Learning Repository: Arrhythmia Data Set

You need to implement the neural network model for this classification in python. Present the result and its analysis. Show the results with different design choices and discuss the results.

2 INTRODUCTION

Multilayer perceptron was implemented to classify Arrhythmia into 16 classes. The same architecture that was proposed in assignment 03 has been implemented.

2.1 ACCURACY OF TRAINED MODEL ON TEST DATA (UNSEEN TO THE MODEL)

We used two accuracy metrics for the proposed architecture. These are as follows:

Accuracy: 71.74%

Top-K-Categorical Accuracy: 91.30%

3 CODE

3.1 IMPORTING THE LIBRARIES

```
import numpy as np
import pandas as pd
from sklearn.utils import shuffle
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from keras.layers import Dense
from keras.models import Sequential
from keras.utils import to_categorical
```

3.2 LOADING DATASET AS PANDAS DATAFRAME

```
arrhythmia_data=pd.read_csv('./arrhythmia.csv')
arrhythmia_data.info()
```

3.2.1 Information About The Data Frame

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 451 entries, 0 to 450
Columns: 280 entries, 75 to 8
dtypes: float64(120), int64(155), object(5)
memory usage: 986.7+ KB
```

3.3 ASSIGNING NAMES TO COLUMNS

```
column_names=['age','gender','height','weight','qrs_duration','p-r_Interval', ...
,'A_P_V6','A_T_V6','QRSA_V6','QRSTA_V6','Class']
arrhythmia_data.columns=column_names
arrhythmia_data.head()
```

3.3.1 First Five Entries of The Dataset.

	age	gender	height	weight	qrs_duration	p-r_Interval	q-t_interval	t_interval	p_interval	QRS	...	A_Q_V6	A_R_V6	A_S_V6	A_R'_V6	A_S'_V6	A_P_V6
0	56	1	165	64	81	174	401	149	39	25	...	0.0	8.5	0.0	0.0	0.0	0.2
1	54	0	172	95	138	163	386	185	102	96	...	0.0	9.5	-2.4	0.0	0.0	0.3
2	55	0	175	94	100	202	380	179	143	28	...	0.0	12.2	-2.2	0.0	0.0	0.4
3	75	0	190	80	88	181	360	177	103	-16	...	0.0	13.1	-3.6	0.0	0.0	-0.1
4	13	0	169	51	100	167	321	174	91	107	...	-0.6	12.2	-2.8	0.0	0.0	0.9

5 rows × 280 columns

3.4 REPLACING '?' WITH NP.NAN (NOT A NUMBER)

```
arrhythmia_data.replace({'?': np.nan}, regex=False,inplace=True)
null_values=(arrhythmia_data.isna().sum()).sort_values(ascending=False)
null_cols=null_values[null_values!=0].index
arrhythmia_data=arrhythmia_data.astype(float)
print(null_values[null_values!=0])
```

3.4.1 Columns with NaN values.

```
J             375
P              22
T               8
Heart rate      1
QRST            1
dtype: int64
```

3.5 PREPROCESS TO REPLACE MISSING VALUES WITH THE MEAN OF THE COLUMN

```
for col in null_cols:
    mean_of_col=(arrhythmia_data[col][arrhythmia_data[col].notnull()]).astype(float).mean()
    arrhythmia_data[col]=arrhythmia_data[col].fillna(mean_of_col,inplace= False)
```

3.6 SHUFFLE TO RANDOMIZE DATA

```
arrhythmia_data = shuffle(arrhythmia_data)
```

3.7 APPLYING TRAIN, VALIDATION & TEST SPLIT WITH RATIOS OF 80, 10, 10 RESPECTIVELY

```
X = arrhythmia_data.iloc[:, :-1].values
y = arrhythmia_data.iloc[:, -1].values

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
x_val, x_test, y_val, y_test = train_test_split(x_test, y_test, test_size=0.5)
```

3.8 CONVERTING NUMERIC CLASSES INTO CATEGORICAL DATA

```
y_train = to_categorical(y_train)
y_val = to_categorical(y_val)
y_test = to_categorical(y_test)
```

3.9 AVOIDING DUMMY VARIABLE TRAP

```
y_train = y_train[:,1:]
y_val = y_val[:,1:]
y_test = y_test[:,1:]
```

3.10 APPLYING MIN-MAX-NORMALIZATION

```
mms_x = MinMaxScaler()
pp_x_train = mms_x.fit_transform(x_train)
pp_x_val = mms_x.transform(x_val)
pp_x_test = mms_x.transform(x_test)
```

```
model=Sequential()
model.add(Dense(128,input_dim=(279),activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(32,activation='relu'))
model.add(Dense(16,activation='softmax'))
```

3.12 SELECTING LOSS FUNCTION, OPTIMIZER AND ACCURACY METHOD

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy','top_k_categorical_accuracy'])
```

3.13 FITTING THE MODEL USING TRAIN AND VALIDATION DATA

```
model.fit(pp_x_train, y_train, epochs=15, batch_size=8, validation_data=(pp_x_val, y_val))
```

3.13.1 Model Training

```
Epoch 1/15
45/45 [==============================] - 0s 4ms/step - loss: 2.2064 - accuracy: 0.4052 - top_k_categorical_accuracy: 0.7273 - v
al_loss: 1.4742 - val_accuracy: 0.6444 - val_top_k_categorical_accuracy: 0.8667
Epoch 2/15
45/45 [==============================] - 0s 2ms/step - loss: 1.6883 - accuracy: 0.5377 - top_k_categorical_accuracy: 0.8653 - v
al_loss: 1.3692 - val_accuracy: 0.6444 - val_top_k_categorical_accuracy: 0.9111
Epoch 3/15
45/45 [==============================] - 0s 2ms/step - loss: 1.5517 - accuracy: 0.5637 - top_k_categorical_accuracy: 0.9121 - v
al_loss: 1.3980 - val_accuracy: 0.6444 - val_top_k_categorical_accuracy: 0.9111
Epoch 4/15
45/45 [==============================] - 0s 2ms/step - loss: 1.5210 - accuracy: 0.5661 - top_k_categorical_accuracy: 0.8829 - v
al_loss: 1.2524 - val_accuracy: 0.6667 - val_top_k_categorical_accuracy: 0.9111
Epoch 5/15
45/45 [==============================] - 0s 2ms/step - loss: 1.3845 - accuracy: 0.6082 - top_k_categorical_accuracy: 0.8906 - v
al_loss: 1.2583 - val_accuracy: 0.6889 - val_top_k_categorical_accuracy: 0.9333
Epoch 6/15
45/45 [==============================] - 0s 2ms/step - loss: 1.3255 - accuracy: 0.6111 - top_k_categorical_accuracy: 0.9227 - v
al_loss: 1.1930 - val_accuracy: 0.6667 - val_top_k_categorical_accuracy: 0.9333
Epoch 7/15
45/45 [==============================] - 0s 2ms/step - loss: 1.2960 - accuracy: 0.5991 - top_k_categorical_accuracy: 0.9286 - v
al_loss: 1.1539 - val_accuracy: 0.6889 - val_top_k_categorical_accuracy: 0.9111
Epoch 8/15
45/45 [==============================] - 0s 2ms/step - loss: 1.1655 - accuracy: 0.6754 - top_k_categorical_accuracy: 0.9494 - v
al_loss: 1.1736 - val_accuracy: 0.6889 - val_top_k_categorical_accuracy: 0.9333
Epoch 9/15
45/45 [==============================] - 0s 2ms/step - loss: 1.1375 - accuracy: 0.6805 - top_k_categorical_accuracy: 0.9369 - v
al_loss: 1.2693 - val_accuracy: 0.6667 - val_top_k_categorical_accuracy: 0.9556
Epoch 10/15
45/45 [==============================] - 0s 2ms/step - loss: 1.0802 - accuracy: 0.6573 - top_k_categorical_accuracy: 0.9479 - v
al_loss: 1.2340 - val_accuracy: 0.6889 - val_top_k_categorical_accuracy: 0.9556
Epoch 11/15
45/45 [==============================] - 0s 4ms/step - loss: 1.1089 - accuracy: 0.7026 - top_k_categorical_accuracy: 0.9282 - v
al_loss: 1.2165 - val_accuracy: 0.7333 - val_top_k_categorical_accuracy: 0.9556
Epoch 12/15
45/45 [==============================] - 0s 2ms/step - loss: 1.0057 - accuracy: 0.7059 - top_k_categorical_accuracy: 0.9528 - v
al_loss: 1.2113 - val_accuracy: 0.6889 - val_top_k_categorical_accuracy: 0.9778
Epoch 13/15
45/45 [==============================] - 0s 2ms/step - loss: 1.0203 - accuracy: 0.7104 - top_k_categorical_accuracy: 0.9525 - v
al_loss: 1.1955 - val_accuracy: 0.7333 - val_top_k_categorical_accuracy: 0.9778
Epoch 14/15
45/45 [==============================] - 0s 2ms/step - loss: 0.9066 - accuracy: 0.7612 - top_k_categorical_accuracy: 0.9447 - v
al_loss: 1.2944 - val_accuracy: 0.6889 - val_top_k_categorical_accuracy: 0.9556
Epoch 15/15
45/45 [==============================] - 0s 2ms/step - loss: 0.9634 - accuracy: 0.7061 - top_k_categorical_accuracy: 0.9829 - v
al_loss: 1.3963 - val_accuracy: 0.6444 - val_top_k_categorical_accuracy: 0.9556
```

3.14 CALCULATING ACCURACY OF THE TRAINED MODEL ON TEST DATA

```
_, accuracy, top_k_categorical = model.evaluate(pp_x_test, y_test)
print('Accuracy: %.2f & Top K Categorical : %.2f ' % (accuracy*100, top_k_categorical*100 ))
```

3.14.1 Output For Accuracy Metrics

```
2/2 [==============================] - 0s 2ms/step - loss: 1.2954 - accuracy: 0.7174 - top_k_categorical_accuracy: 0.9130
Accuracy: 71.74 & Top K Categorical : 91.30
```

4 JUSTIFICATION FOR SELECTION OF PARAMETERS

4.1 PREPROCESSING

4.1.1 Missing values

The dataset had not been preprocessed and contained missing values in multiple columns. We identified these columns, replaced "?" with nan (not a number) and then used the pandas' *fillna* method to replace these missing values with the mean of that column.

4.1.2 Shuffling the rows

We shuffled the rows to reduce any pattern.

4.1.3 Normalization

Min-max normalization was applied on each column to map them onto a same range. Applying min-max normalization converted all features to have values between 0 and 1.

4.2 DATA SPLIT

The train-validation-test split was motivated from **[1]**. They have proved that 80,10,10 is the best data split ratio for medical data in machine learning.

4.3 CONVERSION OF NUMERIC CLASSES TO CATEGORICAL MATRIX

The model requires a sparse matrix of categorical variables as it predicts the probability of each class at the end. Therefore, the data was converted to a sparse categorical matrix. This conversion added an extra class for class label 0 which was not a part of the actual data. This column was removed from the sparse matrix.

4.4 ACTIVATION FUNCTION

There are numerous reasons for selecting ReLU as the activation function for hidden layers. **[2]**

1. ReLU is computationally efficient.
2. There is no chance of vanishing gradient.

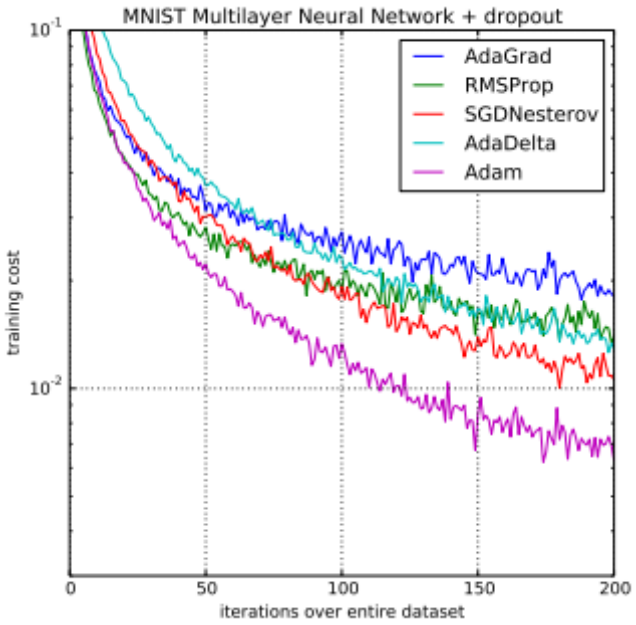
The output layer is predicting classes for each class. Sigmoid function gives probability between 0 and 1 but the sum of these probabilities can exceed 1 in case of a multiclass classification

problem. So, a modified version of sigmoid for multiclass classification (SoftMax) is used in the output layer.

4.5 Selecting loss function, Optimizer and Accuracy for the multi-layer perceptron

Cross entropy error function is used for classification problems. This is a multiclass classification, so we used categorical cross-entropy.

We did not know much about optimization algorithms. [3] states that Adam achieves good results in less amount of time than other algorithms.



1: Graphs of iterations vs loss for different optimization algorithms

"*A metric is a function that is used to judge the performance of your model. Metric functions are similar to loss functions, except that the results from evaluating a metric are not used when training the model.*" [4]

"Computes how often targets are in the top K predictions." [5]

Top-k-categorical-accuracy is used as the metric as is penalizes model on its top K predictions. Default value for K is 5. Note that simple accuracy could have also been used as a metric but the test accuracy on that was low (late 60's). Using top-k-categorical-accuracy resulted in a significant increase in test accuracy.

4.6 Epochs and Batch-Size

Epochs and batch-size are selected empirically by trial and error.

A smaller number of epochs resulted in very small training and validation accuracy. However, a larger number of epochs lead to a large training accuracy but a smaller validation accuracy

(overfitting on training data). This was mitigated by selecting an intermediate value for epochs that did not result in both underfitting and overfitting.

The same thing happened while selecting the batch-size for training. Smaller value resulting in overfitting and larger value led to both underfitting and slow training. Thus, a batch-size of 8 was selected.

5 REFERENCES

[1] https://glassboxmedicine.com/2019/09/15/best-use-of-train-val-test-splits-with-tips-for-medical-data/

[2] https://stats.stackexchange.com/questions/126238/what-are-the-advantages-of-relu-over-sigmoid-function-in-deep-neural-networks

[3] https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[4] https://keras.io/api/metrics/

[5] https://keras.io/api/metrics/accuracy_metrics/#topkcategoricalaccuracy-class