Neural Network

June 15, 2019

1 Neural Network

In [1]:
```
import time
import os, sys
import string

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

from keras.utils import np_utils
from sklearn.preprocessing import LabelEncoder
from keras import Sequential
from keras import layers

import pandas as pd
import numpy as np
```

```
/home/shashwati/anaconda3/envs/py35/lib/python3.5/site-packages/h5py/__init__.py:36: FutureWarn
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

In [2]:
```
def get_data(dataset):
    if os.path.isfile(dataset):
        print("Loading ", dataset, " dataset ...")
        data = pd.read_csv(dataset)
        print("\nDataset loaded successfully\n\n")
        return data
    else:
        print('File not found')
        print('\n\nExiting...')
        sys.exit()
```

In [3]:
```
#The column names are [a, b, c, ..., z, A, B, C, ..., W]
columnNames = list(string.ascii_lowercase)+ list(string.ascii_uppercase)[:23]
```

In [4]:
```
dataset = get_data('./sensIT_train.csv')
label_train = dataset['result']
```

```
        train = dataset[columnNames]

        dataset = get_data('./sensIT_test.csv')
        label_test = dataset['result']
        test = dataset[columnNames]
```

```
Loading  ./sensIT_train.csv  dataset ...

Dataset loaded successfully


Loading  ./sensIT_test.csv  dataset ...

Dataset loaded successfully
```

In [5]:
```
train = np.array(train)
test = np.array(test)
label_test = np.array(label_test)
label_train = np.array(label_train)
```

In [6]:
```
# training the neural network
start_time = time.time()

# preprocessing the test and train data set
encoder = LabelEncoder()
encoder.fit(label_train)
label_train = encoder.transform(label_train)
label_train = np_utils.to_categorical(label_train)

encoder.fit(label_test)
label_test = encoder.transform(label_test)
label_test = np_utils.to_categorical(label_test)

# neural network structure
model = Sequential()
model.add(layers.Dense(30,init = 'uniform', activation = 'relu', input_dim = 49))
model.add(layers.Dense(10,init = 'uniform', activation = 'relu'))
model.add(layers.Dense(3, init = 'uniform', activation = 'softmax'))
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')

model.fit(train, label_train, epochs=15, batch_size=500)
end_time = time.time()
```

```
/home/shashwati/anaconda3/envs/py35/lib/python3.5/site-packages/ipykernel_launcher.py:16: UserW
  app.launch_new_instance()
/home/shashwati/anaconda3/envs/py35/lib/python3.5/site-packages/ipykernel_launcher.py:17: UserW
```

```
/home/shashwati/anaconda3/envs/py35/lib/python3.5/site-packages/ipykernel_launcher.py:18: UserW

Epoch 1/15
78823/78823 [==============================] - 1s 15us/step - loss: 0.9656 - acc: 0.5430
Epoch 2/15
78823/78823 [==============================] - 1s 11us/step - loss: 0.7445 - acc: 0.6513
Epoch 3/15
78823/78823 [==============================] - 1s 10us/step - loss: 0.6942 - acc: 0.6856
Epoch 4/15
78823/78823 [==============================] - 1s 11us/step - loss: 0.6707 - acc: 0.6987
Epoch 5/15
78823/78823 [==============================] - 1s 10us/step - loss: 0.6594 - acc: 0.7055
Epoch 6/15
78823/78823 [==============================] - 1s 11us/step - loss: 0.6527 - acc: 0.7088
Epoch 7/15
78823/78823 [==============================] - 1s 10us/step - loss: 0.6482 - acc: 0.7102
Epoch 8/15
78823/78823 [==============================] - 1s 10us/step - loss: 0.6442 - acc: 0.7127
Epoch 9/15
78823/78823 [==============================] - 1s 10us/step - loss: 0.6414 - acc: 0.7129
Epoch 10/15
78823/78823 [==============================] - 1s 10us/step - loss: 0.6388 - acc: 0.7133
Epoch 11/15
78823/78823 [==============================] - 1s 10us/step - loss: 0.6367 - acc: 0.7147
Epoch 12/15
78823/78823 [==============================] - 1s 10us/step - loss: 0.6351 - acc: 0.7143
Epoch 13/15
78823/78823 [==============================] - 1s 9us/step - loss: 0.6338 - acc: 0.7159
Epoch 14/15
78823/78823 [==============================] - 1s 10us/step - loss: 0.6321 - acc: 0.7163
Epoch 15/15
78823/78823 [==============================] - 1s 10us/step - loss: 0.6312 - acc: 0.7169
```

```
In [7]: # predicting
        pred = model.predict(test)

        # Getting the classes
        pred = np.argmax(pred, axis=1)
        label_test = np.argmax(label_test, axis=1)
```

```
In [8]: def metrics(cm, cls, size):
            cm = np.array(cm)
            tp = cm[cls][cls]
            fp = sum(cm[x, cls] for x in range(3))-cm[cls][cls]
            fn = sum(cm[cls, x] for x in range(3))-cm[cls][cls]
            tn = size - tp - fp - fn
```

```
            precision = tp/(tp+fp)
            recall = tp/(tp+fn)
            fmeasure = 2*(precision*recall)/(precision + recall)
            accuracy = (tp + tn)/size

            return precision, recall, fmeasure, accuracy
```

In [9]:
```
# Rows: Actual
# Cols: Predicted
# Classes: 1, 2, 3
cm = confusion_matrix(label_test, pred)
print("Confusion Matrix:\n ")
print(cm)
```

```
Confusion Matrix:

[[3149 1293  161]
 [1331 3036  943]
 [ 663 1276 7853]]
```

In [10]:
```
# Class 1
precision0, recall0, f0, acc0 = metrics(cm, 0, len(test))
print("        Precision Recall F-measure Accuracy")
print("Class 1: ", round(precision0, 3), "  ", round(recall0, 3), \
      " ", round(f0, 3), "   ", round(acc0,3))
```

```
        Precision Recall F-measure Accuracy
Class 1:  0.612    0.684   0.646     0.825
```

In [11]:
```
# Class 2
precision1, recall1, f1, acc1 = metrics(cm, 1, len(test))
print("        Precision Recall F-measure Accuracy")
print("Class 2: ", round(precision1, 3), "  ", round(recall1, 3), \
      " ", round(f1, 3), "   ", round(acc1,3))
```

```
        Precision Recall F-measure Accuracy
Class 2:  0.542    0.572   0.556     0.754
```

In [12]:
```
# Class 3
precision2, recall2, f2, acc2 = metrics(cm, 2, len(test))
print("        Precision Recall F-measure Accuracy")
print("Class 3: ", round(precision2, 3), "  ", round(recall2, 3), \
      " ", round(f2, 3), "   ", round(acc2,3))
```

```
        Precision Recall F-measure Accuracy
Class 3:  0.877    0.802   0.838     0.846
```

```
In [13]: avg_p = (precision0 + precision1 + precision2)/3.0
         avg_r = (recall0 + recall1 + recall2) / 3.0
         avg_f = (f0 + f1 + f2) / 3.0
         avg_a = (acc0 + acc1 + acc2)/ 3.0
         print("        Precision Recall F-measure Accuracy")
         print("Average: ", round(avg_p, 3), "  ", round(avg_r, 3), \
               " ", round(avg_f, 3), "   ", round(avg_a,3))

        Precision Recall F-measure Accuracy
Average:  0.677    0.686   0.68     0.808


In [14]: # Number of instances correctly classified
         acc_score = accuracy_score(pred, label_test)
         print("Accuracy_score: ", round(acc_score, 4))
         print("Training Time: %s secs" % round(end_time - start_time, 3))

Accuracy_score:  0.7124
Training Time: 12.661 secs
```