Deep-Neural-Netwok

June 18, 2021

1 01 *Sequentional Model Strcuture*

[40]:
```
## required lib for the model
```

[41]:
```
from tensorflow.keras.models import Sequential ## keras working over tensor flow
from tensorflow.keras.layers import Dense ## layers
import numpy as np
```

[42]:
```
## creating the mode
```

[43]:
```
model = Sequential() ## default parameter
model.add(Dense(10, activation='sigmoid',input_shape=(3,))) # 10 -> number of␣
 ↪neuron on the hidden layer and input shape is shape of the input 3 neuron
model.add(Dense(1,activation='sigmoid')) ## output layer and number of neuron␣
 ↪is '1'
```

[44]:
```
model.summary() ## remember always there baise neuron on the input layer and␣
 ↪hidden layer
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 10)                40
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 11
=================================================================
Total params: 51
Trainable params: 51
Non-trainable params: 0
_________________________________________________________________
```

2 02 *Train the Model with Dummy matix Data*

[45]:
```
## dummay data
input = np.array([
        [3,3,1],
        [4,3,1],
        [3,1,3],
        [2,3,1]
])

# input shape always be set in the model is number of dim of instance
# in the above input data there are 4 instance and each instance have 3␣
 ↪dimension
# But this is not always be case of the input structure sometime you have␣
 ↪[[2,3],[2,3],[2,3]]

output = np.array([0,0,1,1])
```

[46]:
```
## training the Model
model  =Sequential()
model.add(Dense(10, ## hidden layer neuron
                activation='sigmoid', ## function input
               input_shape = (3,))) ## number of dim of instance
                                    ## Not the number of the instance
model.add(Dense(1,
               activation='sigmoid',))
```

[47]:
```
## model compile
model.compile(loss="binary_crossentropy", # try to reduce the cost function to␣
 ↪make reach near the 1 accuracy
              optimizer = 'sgd', # Sochastic gradient Decent
               metrics = ['accuracy'] # you can use it more
              )
```

[61]:
```
## fiting the Model

model.fit(input[:-1],
          output[:-1],
          epochs=2, ## number of times to train the data
          batch_size=1, ## when data is large enough then need this
          validation_data=(input[-1:],output[-1:])
         )
## in the output accuracy should be one i dont why not give me one accuracy␣
 ↪walla seacrch kary ghy!
```

```
Epoch 1/2
3/3 [==============================] - 0s 19ms/step - loss: 0.5444 - accuracy:
```

```
0.6667 - val_loss: 1.2778 - val_accuracy: 0.0000e+00
Epoch 2/2
3/3 [==============================] - 0s 8ms/step - loss: 0.5428 - accuracy:
0.6667 - val_loss: 1.2808 - val_accuracy: 0.0000e+00
```

[61]:
```
<tensorflow.python.keras.callbacks.History at 0x7efb286b4430>
```

3 03 *Result and Evaluation*

[51]:
```
## Required ib
```

[52]:
```
from tensorflow.keras.models import Sequential ## keras working over tensor flow
from tensorflow.keras.layers import Dense ## layers
import numpy as np
```

[62]:
```
## Dummy data
input  = np.random.rand(10000,3)
output = np.random.randint(2, size=10000) # always less then 2
```

[64]:
```
input.shape
```

[64]:
```
(10000, 3)
```

[65]:
```
output.shape
```

[65]:
```
(10000,)
```

[66]:
```
input
```

[66]:
```
array([[0.64432614, 0.41025333, 0.17986627],
       [0.39338619, 0.44014082, 0.31207345],
       [0.45606186, 0.55358778, 0.38998532],
       …,
       [0.54405043, 0.25634757, 0.79472012],
       [0.93623219, 0.26310953, 0.31209902],
       [0.26518008, 0.28102566, 0.05466528]])
```

[67]:
```
output
```

[67]:
```
array([1, 1, 1, …, 0, 1, 0])
```

[69]:
```
## training the Model
model  =Sequential()
model.add(Dense(10, ## hidden layer neuron
                activation='sigmoid', ## function input
               input_shape = (3,))) ## number of dim of instance
                                    ## Not the number of the instance
```

```
model.add(Dense(1,
                activation='sigmoid',))
```

[71]:
```
model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_10 (Dense)             (None, 10)                40
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 11
=================================================================
Total params: 51
Trainable params: 51
Non-trainable params: 0
_________________________________________________________________
```

[74]:
```
## model compile
model.compile(loss="binary_crossentropy", # try to reduce the cost function to␣
 ↪make reach near the 1 accuracy
              optimizer = 'sgd', # Sochastic gradient Decent
               metrics = ['accuracy'] # you can use it more
              )
```

[83]:
```
## fiting the Model

model.fit(input[:-3000],
          output[:-3000],
          epochs=2, ## number of times to train the data
          batch_size=64, ## when data is large enough then need this
          validation_data=(input[-3000:],output[-3000:])
         )
## in the output accuracy should be one i dont why not give me one accuracy␣
 ↪walla seacrch kary ghy!
```

```
Epoch 1/2
110/110 [==============================] - 0s 4ms/step - loss: 0.6930 -
accuracy: 0.5026 - val_loss: 0.6937 - val_accuracy: 0.4883
Epoch 2/2
110/110 [==============================] - 0s 4ms/step - loss: 0.6929 -
accuracy: 0.5097 - val_loss: 0.6936 - val_accuracy: 0.4907
```

[83]:
```
<tensorflow.python.keras.callbacks.History at 0x7efb28500c70>
```

[84]:
```
## EVALUATION
```

[85]:
```
model.evaluate(input[-2000:],output[-2000:])
```

```
63/63 [==============================] - 0s 2ms/step - loss: 0.6935 - accuracy:
0.5000
```

[85]:
```
[0.6935240626335144, 0.5]
```

[88]:
```
model.predict(input[-5:])
```

[88]:
```
array([[0.4986413 ],
       [0.5003938 ],
       [0.48364276],
       [0.4814856 ],
       [0.5031678 ]], dtype=float32)
```

[106]:
```
pred_y = model.predict_classes(input[-10:-5])
```

[107]:
```
## check the in the data set
# whether is correct or not
y = output[-5:] #actuall value

pred_y ,y
```

[107]:
```
(array([[1],
        [0],
        [1],
        [0],
        [1]], dtype=int32),
 array([1, 0, 0, 1, 0]))
```

[104]:
```
from sklearn.metrics import precision_score,recall_score,f1_score
```

[108]:
```
precision_score(pred_y,y,average='micro')
```

[108]:
```
0.4
```

[109]:
```
recall_score(pred_y,y,average='micro')
```

[109]:
```
0.4
```

[110]:
```
f1_score(pred_y,y,average='micro')
```

[110]:
```
0.4000000000000001
```

4 04 *Working on Real datasets*

[127]:
```
# requried lib
from tensorflow.keras.models import Sequential ## keras working over tensor flow
from tensorflow.keras.layers import Dense ## layers
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

[224]:
```
df = pd.read_csv("Datasets/DatasetUpdate.csv")
df["Label"] = [1 if x == True  else 0 for x in df["Label"]] # binary␣
 ↪representation
```

[225]:
```
df = df.drop(columns=['Unnamed: 0'])
```

[226]:
```
df
```

[226]:

	Text	Label
0	Actually, they didn't. The whole tragedy was c…	1
1	At your service: Comparison I could've jus…	1
2	So which is it: the action is moral, the actio…	1
3	Interesting how the study was set in Pittsburg…	1
4	Ah, I see. Your reasons are secret reasons. …	1
…	…	…
1995	What do you mean by this? Could we not have th…	0
1996	And the answer is: we don't know. Maybe it cam…	0
1997	And what would make them separate species? How…	0
1998	This decision was not solely based on self, bu…	0
1999	A perfect example of why Christian fundamental…	0

[2000 rows x 2 columns]

[227]:
```
x = df["Text"]
y = df.iloc[:,1]
```

[228]:
```
x
```

[228]:
```
0       Actually, they didn't. The whole tragedy was c…
1       At your service:  Comparison  I could've jus…
2       So which is it: the action is moral, the actio…
3       Interesting how the study was set in Pittsburg…
4       Ah, I see.  Your reasons are secret reasons. …
                              …
1995    What do you mean by this? Could we not have th…
1996    And the answer is: we don't know. Maybe it cam…
1997    And what would make them separate species? How…
1998    This decision was not solely based on self, bu…
```

```
1999    A perfect example of why Christian fundamental…
Name: Text, Length: 2000, dtype: object
```

[229]:
```
y
```

[229]:
```
0       1
1       1
2       1
3       1
4       1
       ..
1995    0
1996    0
1997    0
1998    0
1999    0
Name: Label, Length: 2000, dtype: int64
```

[205]:
```

```

[230]:
```
df["Label"]
```

[230]:
```
0       1
1       1
2       1
3       1
4       1
       ..
1995    0
1996    0
1997    0
1998    0
1999    0
Name: Label, Length: 2000, dtype: int64
```

4.1 4.1 Vectorization the dataset

[232]:
```
tfidf = TfidfVectorizer(max_features=200)

matrix_X = tfidf.fit_transform(x)
```

[233]:
```
matrix_X
```

[233]:
```
<2000x200 sparse matrix of type '<class 'numpy.float64'>'
        with 40163 stored elements in Compressed Sparse Row format>
```

[234]:
```
matrix_X.toarray()
```

[234]:
```
array([[0.        , 0.        , 0.23473676, …, 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , …, 0.        , 0.        ,
        0.25011125],
       [0.        , 0.        , 0.        , …, 0.        , 0.1661102 ,
        0.        ],
       …,
       [0.        , 0.33023403, 0.        , …, 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , …, 0.        , 0.13875319,
        0.        ],
       [0.        , 0.        , 0.        , …, 0.        , 0.        ,
        0.        ]])
```

4.2 split the data from training the model

[236]:
```
from sklearn.model_selection import train_test_split
```

[237]:
```
train_X , test_X ,train_y,test_y = train_test_split(matrix_X ,y, shuffle=True␣
↪,train_size=0.8 )
```

4.3 4.2 training the Model

[ ]:
```
model  =Sequential()
model.add(Dense(10, ## hidden layer neuron
                activation='sigmoid', ## function input
               input_shape = (200,))) ## number of dim of instance
                                      ## Not the number of the instance
model.add(Dense(1,
               activation='sigmoid',))
```

4.4 4.3 model compile

[ ]:
```
model.compile(loss="binary_crossentropy", # try to reduce the cost function to␣
↪make reach near the 1 accuracy
              optimizer = 'sgd', # Sochastic gradient Decent
               metrics = ['accuracy'] # you can use it more
              )
```

[240]:
```
model.summary
```

[240]:
```
<bound method Model.summary of
<tensorflow.python.keras.engine.sequential.Sequential object at 0x7efac60387f0>>
```

[241]:
```
train_X.shape
```

[241]: (1600, 200)

4.5 4.4 fiting the Model

[ ]:
```
# ## you will see the error here because he will take array
train_X = train_X.toarray()
test_X = test_X.toarray()
model.fit(train_X,
          train_y,
          epochs=3, ## number of times to train the data
          batch_size=20, ## when data is large enough then need this
          validation_data=(test_X,test_y)
         )
## in the output accuracy should be one i dont why not give me one accuracy␣
 ↪walla seacrch kary ghy!
```

4.6 4.5 Evaluation and results

[288]:
```
from sklearn.metrics import accuracy_score,f1_score,recall_score
```

[289]:
```
model.evaluate(test_X,test_y)
# pred_y = model.predict_classes()
pred_y = np.argmax(model.predict(test_X[:10]), axis=-1)

model.predict([test_X[:10]])

y =  test_y[:10]


print("Preceson Score : ",precision_score(pred_y , y , average='macro'))
print("Recall Score :␣
 ↪",recall_score(pred_y,y,average="macro",zero_division=True))
print("F1 Score : ",f1_score(pred_y, y ,average='macro'))
```

```
Preceson Score :  0.5
Recall Score :  0.65
F1 Score :  0.23076923076923075
```

[259]:

[259]: 0.7

[260]:

[260]: 0.28571428571428575

5 05 Sarcastic Detection using Deep learning neural network

[330]:
```
# requried lib
from tensorflow.keras.models import Sequential ## keras working over tensor flow
from tensorflow.keras.layers import Dense ,Dropout ## layers
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

[331]:
```
df = pd.read_csv("Datasets/DatasetUpdate.csv")
df["Label"] = [1 if x == True  else 0 for x in df["Label"]] # binary␣
 ↪representation

df = df.drop(columns=['Unnamed: 0'])
df
```

[331]:

	Text	Label
0	Actually, they didn't. The whole tragedy was c…	1
1	At your service: Comparison I could've jus…	1
2	So which is it: the action is moral, the actio…	1
3	Interesting how the study was set in Pittsburg…	1
4	Ah, I see. Your reasons are secret reasons. …	1
…	…	…
1995	What do you mean by this? Could we not have th…	0
1996	And the answer is: we don't know. Maybe it cam…	0
1997	And what would make them separate species? How…	0
1998	This decision was not solely based on self, bu…	0
1999	A perfect example of why Christian fundamental…	0

[2000 rows x 2 columns]

[332]:
```
x = df["Text"]
y = df.iloc[:,1]
```

5.1 5.1 Vectorization the dataset

[333]:
```
tfidf = TfidfVectorizer(max_features=200)

matrix_X = tfidf.fit_transform(x)
```

5.2 5.2 split the data from training the model

[334]:
```
from sklearn.model_selection import train_test_split
```

```
train_X , test_X ,train_y,test_y = train_test_split(matrix_X ,y, shuffle=True␣
↪,train_size=0.8 )
```

5.3 5.3 Training Model

[335]:
```
model1 = Sequential()

model1.add(Dense(100,activation='relu',input_shape= (200,)))

model1.add(Dropout(0.15)) ## block some information
model1.add(Dense(80,activation='relu'))

model1.add(Dense(50,activation='relu'))
model1.add(Dropout(0.10)) ## will block some informatin
model1.add(Dense(30,activation='relu'))
model1.add(Dense(1,activation='sigmoid'))
```

[336]:
```
model1.summary()
```

```
Model: "sequential_21"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_57 (Dense)             (None, 100)               20100
_________________________________________________________________
dropout_10 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_58 (Dense)             (None, 80)                8080
_________________________________________________________________
dense_59 (Dense)             (None, 50)                4050
_________________________________________________________________
dropout_11 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_60 (Dense)             (None, 30)                1530
_________________________________________________________________
dense_61 (Dense)             (None, 1)                 31
=================================================================
Total params: 33,791
Trainable params: 33,791
Non-trainable params: 0
_________________________________________________________________
```

5.4 5.4 Model Complile

[337]:
```
# model1.compile(loss="binary_crossentropy", # try to reduce the cost function␣
 ↪to make reach near the 1 accuracy
#               optimizer = 'sgd', # Sochastic gradient Decent
#                metrics = ['accuracy'] # you can use it more
#               )

# chnage with optimmizer function 'rmsprop'
model1.compile(loss="binary_crossentropy", # try to reduce the cost function to␣
 ↪make reach near the 1 accuracy
              optimizer = 'rmsprop', # Sochastic gradient Decent
               metrics = ['accuracy'] # you can use it more
              )
```

5.5 5.5 fitting Model

[338]:
```
# ## you will see the error here because he will take array
train_X = train_X.toarray()
test_X = test_X.toarray()
model1.fit(train_X,
           train_y,
           epochs=10, ## number of times to train the data
           batch_size=64, ## when data is large enough then need this
           validation_data=(test_X,test_y)
          )
## in the output accuracy should be one i dont why not give me one accuracy␣
 ↪walla seacrch kary ghy!
```

```
Epoch 1/10
25/25 [==============================] - 1s 5ms/step - loss: 0.6903 - accuracy:
0.5331 - val_loss: 0.6878 - val_accuracy: 0.5500
Epoch 2/10
25/25 [==============================] - 0s 2ms/step - loss: 0.6760 - accuracy:
0.5844 - val_loss: 0.6757 - val_accuracy: 0.5825
Epoch 3/10
25/25 [==============================] - 0s 2ms/step - loss: 0.6484 - accuracy:
0.6519 - val_loss: 0.6682 - val_accuracy: 0.6075
Epoch 4/10
25/25 [==============================] - 0s 2ms/step - loss: 0.6137 - accuracy:
0.6775 - val_loss: 0.6679 - val_accuracy: 0.5900
Epoch 5/10
25/25 [==============================] - 0s 2ms/step - loss: 0.5859 - accuracy:
0.7150 - val_loss: 0.6739 - val_accuracy: 0.5900
Epoch 6/10
25/25 [==============================] - 0s 2ms/step - loss: 0.5571 - accuracy:
0.7212 - val_loss: 0.6866 - val_accuracy: 0.5925
```

```
Epoch 7/10
25/25 [==============================] - 0s 2ms/step - loss: 0.5298 - accuracy:
0.7600 - val_loss: 0.7026 - val_accuracy: 0.5975
Epoch 8/10
25/25 [==============================] - 0s 2ms/step - loss: 0.4891 - accuracy:
0.7719 - val_loss: 0.7396 - val_accuracy: 0.5850
Epoch 9/10
25/25 [==============================] - 0s 2ms/step - loss: 0.4538 - accuracy:
0.8069 - val_loss: 0.7772 - val_accuracy: 0.5925
Epoch 10/10
25/25 [==============================] - 0s 2ms/step - loss: 0.4224 - accuracy:
0.8213 - val_loss: 0.8096 - val_accuracy: 0.5850
```

[338]:
```
<tensorflow.python.keras.callbacks.History at 0x7efac90bbb80>
```

5.6 5.6 Model Evaluation

[340]:
```
model1.evaluate(test_X,test_y)
# pred_y = model.predict_classes()
pred_y = np.argmax(model.predict(test_X[:10]), axis=-1)

model1.predict([test_X[:10]])

y =  test_y[:10]


print("Preceson Score : ",precision_score(pred_y , y , average='macro'))
print("Recall Score :␣
 ↪",recall_score(pred_y,y,average="macro",zero_division=True))
print("F1 Score : ",f1_score(pred_y, y ,average='macro'))
```

```
13/13 [==============================] - 0s 696us/step - loss: 0.8096 -
accuracy: 0.5850
Preceson Score :  0.5
Recall Score :  0.8
F1 Score :  0.37499999999999994
```

[ ]:
```
```