Housing

June 7, 2022

0.0.1 COLLINS ABWAO

0.0.2 CIT-223-034/2018

0.0.3 LEARNING AND ADAPTIVE SYSTEM

0.0.4 NEURAL NETWORK IMPLEMENTATION

0.1 Introduction

The notebook illustrates implementation of a regression problem using neural network to predict cost of housing given some features for example Number of rooms.

0.2 Importation of libraries and modules

[1]:
```
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import keras
from keras.layers import Dense, Activation,Dropout
from keras.models import Sequential
import warnings
warnings.filterwarnings('ignore')
```

0.3 Loading dataset from sklearn datatsets

[2]:
```
boston_housing = datasets.load_boston()
```

0.4 Obtaining labels and features

[3]:
```
x,y = boston_housing.data,boston_housing.target
```

[4]:
```
x.shape
```

[4]:
```
(506, 13)
```

[5]:
```
y.shape
```

[5]:
```
(506,)
```

0.5 Splitting the dataset into training and testing

[6]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
```

[7]:
```
x_train.shape,x_test.shape
```

[7]:
```
((404, 13), (102, 13))
```

[8]:
```
#Obtaining number of feature
len(boston_housing.feature_names)
```

[8]:
```
13
```

0.6 Features in dataframe

[9]:
```
boston_data_x = pd.DataFrame(x_train, columns=boston_housing.feature_names)
boston_data_x.head(10)
```

[9]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.03049	55.0	3.78	0.0	0.484	6.874	28.1	6.4654	5.0	370.0	17.6	387.97	4.61
1	0.12083	0.0	2.89	0.0	0.445	8.069	76.0	3.4952	2.0	276.0	18.0	396.90	4.21
2	0.08370	45.0	3.44	0.0	0.437	7.185	38.9	4.5667	5.0	398.0	15.2	396.90	5.39
3	0.13158	0.0	10.01	0.0	0.547	6.176	72.5	2.7301	6.0	432.0	17.8	393.30	12.04
4	1.51902	0.0	19.58	1.0	0.605	8.375	93.9	2.1620	5.0	403.0	14.7	388.45	3.32
5	14.42080	0.0	18.10	0.0	0.740	6.461	93.3	2.0026	24.0	666.0	20.2	27.49	18.05
6	0.10084	0.0	10.01	0.0	0.547	6.715	81.6	2.6775	6.0	432.0	17.8	395.59	10.16
7	9.91655	0.0	18.10	0.0	0.693	5.852	77.8	1.5004	24.0	666.0	20.2	338.16	29.97
8	0.01501	90.0	1.21	1.0	0.401	7.923	24.8	5.8850	1.0	198.0	13.6	395.52	3.16
9	6.71772	0.0	18.10	0.0	0.713	6.749	92.6	2.3236	24.0	666.0	20.2	0.32	17.44

0.7 Label in Dataframe

[10]:
```
boston_data_y = pd.DataFrame(y_train,columns=['PRICE'])
boston_data_y.head(10)
```

[10]:

	PRICE
0	31.2
1	38.7
2	34.9
3	21.2
4	50.0
5	9.6
6	22.8
7	6.3
8	50.0
9	13.4

[11]:
```
boston_housing.keys()
```

[11]:
```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

0.8 Neural Network

0.8.1 Type of model.

- The model created was a simple Artificial neural network model

0.8.2 Layers

- The neural network model is composed of input, hidden and output layer. The model below consists of 5 densely connected layers added to the sequential model. First layer had 128 neurons, second 64, third 32, Fourth 16 and fifth 1.
- The network had a large first layer and following it up with smaller layers which lead to better performance as the first layer can learn a lot of lower-level features that can be fed into a few higher order features in the subsequent layers.

Input Layer

- It accepts the features from the outside world. There is no computation at the input layer. The neurons or nodes therefore pass features to the hidden layer. The model took 13 features as the input dimension

Hidden Layer

- This is where various computation takes place and results are passed to the output layer. It is mainly used for feature extraction and learning.

Output Layer

- The layer brings out the information learned to the outside world. Since problem is regression the output will be of continuous values. One neuron was used since it was not a multiregression problem.

0.8.3 Activation Functions

- They often decide whether a neuron is to be activated or not.They add non-linearity to the output of the neuron
- For model creation Rectified Liniear Unit (relu) was used. It is applied after the dot matrix of input features and weights plus the biases. It gives an output of positive numbers otherwise zero if the output was to be a negative i.e max(0,i). The output layer does not have an activation since this is a regression problem.

0.8.4 Model Compilation

- This is the final step of model creation, the solver or optimizer used to create the model is Adam. It works by adjusting weights and biases based on training data. ### Loss function
- It quantifies the difference between the expected outcome and the output produced by the model. Loss function can be used to derive the gradients which are used to update the weights. The average over all losses constitutes the cost. Loss was calculated using MEAN SQUARE ERROR.

0.8.5 Metrics

- Units used for verification and validation of the model.

[12]:
```
#Model creation
model = Sequential()

model.add(Dense(128,activation  = 'relu',input_dim =len(boston_housing.
 ↪feature_names)))

model.add(Dense(64,activation  = 'relu'))
model.add(Dense(32,activation  = 'relu'))
model.add(Dense(16,activation  = 'relu'))

model.add(Dense(1))

#Model Compilation
model.compile(
    optimizer= 'adam',
    loss = 'mean_squared_error'
)
```

0.9 Model training

[13]:
```
#model training
history = model.fit(x_train,y_train,epochs=250, verbose=0)
```

0.10 Evaluating model using r2 score

[14]:
```
predictions = model.predict(x_test)
```

[15]:
```
score = r2_score(y_test,predictions.reshape(1,-1)[0])
score
```

[15]:
```
0.8657357462202132
```

[16]:
```
pred = model.predict(x_test[30].reshape(1,-1))

pred,y_test[30]
```

[16]:
```
(array([[25.807686]], dtype=float32), 24.1)
```

0.10.1 Actual

[17]:
```
raw_test = pd.DataFrame(x_test[30].reshape(1,-1),columns=boston_housing.
 ↪feature_names)
raw_test.insert(loc=13,column='PRICE',value=y_test[30].reshape(1,-1))
raw_test
```

[17]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.03445	82.5	2.03	0.0	0.415	6.162	38.4	6.27	2.0	348.0	14.7	393.77	7.43	24.1

0.11 Predicted

[18]:
```
pred_test = pd.DataFrame(x_test[30].reshape(1,-1),columns=boston_housing.
 ↪feature_names)
pred_test.insert(loc=13,column='PRICE',value=pred[0])
pred_test
```

[18]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.03445	82.5	2.03	0.0	0.415	6.162	38.4	6.27	2.0	348.0	14.7	393.77	7.43	25.807686

[ ]: