Predicting California House Prices

April 24, 2024

[1]:
```
import tensorflow as tf
from keras.models import Sequential
import pandas as pd
from keras.layers import Dense, Input
from keras.optimizers import SGD
```

0.1 The Data

Import the cal_housing_clean.csv file with pandas. Separate it into a training (70%) and testing set(30%).

[2]:
```
df=pd.read_csv('/Users/sandinatatu/Desktop/Tensorflow-Bootcamp-master/
  ↪02-TensorFlow-Basics/cal_housing_clean.csv')

df.head()
```

[2]:

	housingMedianAge	totalRooms	totalBedrooms	population	households	\
0	41.0	880.0	129.0	322.0	126.0	
1	21.0	7099.0	1106.0	2401.0	1138.0	
2	52.0	1467.0	190.0	496.0	177.0	
3	52.0	1274.0	235.0	558.0	219.0	
4	52.0	1627.0	280.0	565.0	259.0	

	medianIncome	medianHouseValue
0	8.3252	452600.0
1	8.3014	358500.0
2	7.2574	352100.0
3	5.6431	341300.0
4	3.8462	342200.0

Separate your features and target data (medianHouseValue) into training and testing sets, with 33% reserved for testing.

[3]:
```
X = df.iloc[:, :6]
y=df['medianHouseValue']

import numpy as np
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,␣
↪random_state=42)
```

0.2 Scale the Feature Data

Use sklearn preprocessing to create a MinMaxScaler for the feature data. Fit this scaler only to the training data. Then use it to transform X_test and X_train. Then use the scaled X_test and X_train along with pd.Dataframe to re-create two dataframes of scaled data.

[4]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

0.3 Fit a Densely Connected Neural Network to the Training Data

Construct a Densely Connected Neural Network with 3 hidden layers, each having 6 neurons.

[5]:
```
model = Sequential()
model.add(Input(shape=(6,)))
model.add(Dense(6, activation='relu'))
model.add(Dense(6, activation='relu'))
model.add(Dense(6, activation='relu'))
model.add(Dense(1, activation='linear'))


learning_rate = 0.01
optimizer = SGD(learning_rate=learning_rate)


model.compile(loss='mse',
              optimizer=optimizer)

model.fit(X_train, y_train, epochs=100, batch_size=10, shuffle=True, verbose=0)
```

[5]:
```
<keras.src.callbacks.history.History at 0x15b21e610>
```

0.4 Compute the RMSE on the Test Data

[6]:
```
mse = model.evaluate(X_test, y_test, verbose=0)

print("Root Mean Squared Error on Test Data:", mse**0.5)
```

```
Root Mean Squared Error on Test Data: 115552.42979704062
```