dlprac1

February 17, 2024

1 Practical 1:- Implement Boston housing price predictionproblem by Linear regression using Deep Neural network. Use Boston House price prediction dataset.

[8]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from sklearn.metrics import accuracy_score
```

2 Loading the inbuilt dataset

[2]:
```
boston = load_boston()
data = pd.DataFrame(boston.data, columns=boston.feature_names)
data['PRICE'] = boston.target
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data.iloc[:, :-1], data.
 ↪iloc[:, -1], test_size=0.2, random_state=42)
```

```
D:\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning:
Function load_boston is deprecated; `load_boston` is deprecated in 1.0 and will
be removed in 1.2.

    The Boston housing prices dataset has an ethical problem. You can refer to
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use of this
    dataset unless the purpose of the code is to study and educate about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::
```

```
        import pandas as pd
        import numpy as np


        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.

  warnings.warn(msg, category=FutureWarning)
```

3 Standardizes the features

[3]:
```python
scaler = StandardScaler()
X_train = scaler.fit_transform(X_traina)
X_test = scaler.transform(X_test)
```

4 Defines a neural network model with three layers (Dense layers with ReLU activation). Compiles the model using mean squared error (mse) as the loss function.

[4]:
```python
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.
 ↪shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])

# Compile the model
model.compile(optimizer='adam', loss='mse')
```

[5]:
```python
# Train the model
history = model.fit(X_train, y_train, epochs=100, batch_size=32,␣
 ↪validation_split=0.2)
```

```
Epoch 1/100
11/11 [==============================] - 4s 31ms/step - loss: 600.6260 -
val_loss: 529.2634
Epoch 2/100
11/11 [==============================] - 0s 13ms/step - loss: 569.4849 -
val_loss: 499.8468
Epoch 3/100
11/11 [==============================] - 0s 14ms/step - loss: 533.6918 -
val_loss: 464.2322
Epoch 4/100
11/11 [==============================] - 0s 14ms/step - loss: 489.2171 -
val_loss: 418.8820
Epoch 5/100
11/11 [==============================] - 0s 14ms/step - loss: 431.5814 -
val_loss: 359.3557
Epoch 6/100
11/11 [==============================] - 0s 12ms/step - loss: 358.8854 -
val_loss: 287.9466
Epoch 7/100
11/11 [==============================] - 0s 10ms/step - loss: 277.8800 -
val_loss: 213.2249
Epoch 8/100
11/11 [==============================] - 0s 7ms/step - loss: 201.1305 -
val_loss: 148.4100
Epoch 9/100
11/11 [==============================] - 0s 8ms/step - loss: 141.7860 -
val_loss: 101.0766
Epoch 10/100
11/11 [==============================] - 0s 9ms/step - loss: 101.5118 -
val_loss: 72.2913
Epoch 11/100
11/11 [==============================] - 0s 8ms/step - loss: 77.2201 - val_loss:
54.2463
Epoch 12/100
11/11 [==============================] - 0s 11ms/step - loss: 61.3402 -
val_loss: 43.5660
Epoch 13/100
11/11 [==============================] - 0s 8ms/step - loss: 49.6721 - val_loss:
36.8523
Epoch 14/100
11/11 [==============================] - 0s 6ms/step - loss: 40.4337 - val_loss:
33.0155
Epoch 15/100
```

```
11/11 [==============================] - 0s 6ms/step - loss: 34.2795 - val_loss:
31.2338
Epoch 16/100
11/11 [==============================] - 0s 7ms/step - loss: 30.1247 - val_loss:
30.0956
Epoch 17/100
11/11 [==============================] - 0s 11ms/step - loss: 27.3277 -
val_loss: 29.5187
Epoch 18/100
11/11 [==============================] - 0s 7ms/step - loss: 25.6215 - val_loss:
29.2508
Epoch 19/100
11/11 [==============================] - 0s 9ms/step - loss: 24.3145 - val_loss:
28.9362
Epoch 20/100
11/11 [==============================] - 0s 19ms/step - loss: 23.3597 -
val_loss: 28.6153
Epoch 21/100
11/11 [==============================] - 0s 21ms/step - loss: 22.7881 -
val_loss: 28.3776
Epoch 22/100
11/11 [==============================] - 0s 14ms/step - loss: 22.0688 -
val_loss: 27.9476
Epoch 23/100
11/11 [==============================] - 0s 13ms/step - loss: 21.4405 -
val_loss: 27.3070
Epoch 24/100
11/11 [==============================] - 0s 13ms/step - loss: 20.8403 -
val_loss: 26.5387
Epoch 25/100
11/11 [==============================] - 0s 14ms/step - loss: 20.0922 -
val_loss: 26.0835
Epoch 26/100
11/11 [==============================] - 0s 13ms/step - loss: 19.5531 -
val_loss: 25.6259
Epoch 27/100
11/11 [==============================] - 0s 13ms/step - loss: 19.1001 -
val_loss: 25.0388
Epoch 28/100
11/11 [==============================] - 0s 8ms/step - loss: 18.8237 - val_loss:
25.3922
Epoch 29/100
11/11 [==============================] - 0s 9ms/step - loss: 18.3050 - val_loss:
25.0191
Epoch 30/100
11/11 [==============================] - 0s 12ms/step - loss: 17.7048 -
val_loss: 24.5823
Epoch 31/100
```

```
11/11 [==============================] - 0s 9ms/step - loss: 17.3454 - val_loss:
23.8763
Epoch 32/100
11/11 [==============================] - 0s 5ms/step - loss: 16.9519 - val_loss:
23.2500
Epoch 33/100
11/11 [==============================] - 0s 7ms/step - loss: 16.5624 - val_loss:
22.7492
Epoch 34/100
11/11 [==============================] - 0s 9ms/step - loss: 16.2281 - val_loss:
22.4657
Epoch 35/100
11/11 [==============================] - 0s 19ms/step - loss: 15.9576 -
val_loss: 22.2116
Epoch 36/100
11/11 [==============================] - 0s 13ms/step - loss: 15.6528 -
val_loss: 21.7548
Epoch 37/100
11/11 [==============================] - 0s 13ms/step - loss: 15.3520 -
val_loss: 21.5348
Epoch 38/100
11/11 [==============================] - 0s 12ms/step - loss: 15.0695 -
val_loss: 21.2313
Epoch 39/100
11/11 [==============================] - 0s 11ms/step - loss: 14.8552 -
val_loss: 21.1763
Epoch 40/100
11/11 [==============================] - 0s 15ms/step - loss: 14.5744 -
val_loss: 21.3975
Epoch 41/100
11/11 [==============================] - 0s 14ms/step - loss: 14.5133 -
val_loss: 21.0292
Epoch 42/100
11/11 [==============================] - 0s 23ms/step - loss: 14.0946 -
val_loss: 19.8887
Epoch 43/100
11/11 [==============================] - 0s 30ms/step - loss: 13.9110 -
val_loss: 19.7550
Epoch 44/100
11/11 [==============================] - 0s 23ms/step - loss: 13.5938 -
val_loss: 19.5154
Epoch 45/100
11/11 [==============================] - 0s 12ms/step - loss: 13.4258 -
val_loss: 19.0022
Epoch 46/100
11/11 [==============================] - 0s 10ms/step - loss: 13.2388 -
val_loss: 18.8534
Epoch 47/100
```

```
11/11 [==============================] - 0s 9ms/step - loss: 12.9761 - val_loss:
18.8116
Epoch 48/100
11/11 [==============================] - 0s 9ms/step - loss: 12.8209 - val_loss:
18.7257
Epoch 49/100
11/11 [==============================] - 0s 13ms/step - loss: 12.6634 -
val_loss: 18.5245
Epoch 50/100
11/11 [==============================] - 0s 12ms/step - loss: 12.5126 -
val_loss: 18.4571
Epoch 51/100
11/11 [==============================] - 0s 7ms/step - loss: 12.4223 - val_loss:
18.3556
Epoch 52/100
11/11 [==============================] - 0s 11ms/step - loss: 12.5296 -
val_loss: 17.4214
Epoch 53/100
11/11 [==============================] - 0s 11ms/step - loss: 12.2214 -
val_loss: 17.9152
Epoch 54/100
11/11 [==============================] - 0s 21ms/step - loss: 12.0007 -
val_loss: 17.8312
Epoch 55/100
11/11 [==============================] - 0s 9ms/step - loss: 12.0938 - val_loss:
17.3287
Epoch 56/100
11/11 [==============================] - 0s 9ms/step - loss: 12.0284 - val_loss:
16.8679
Epoch 57/100
11/11 [==============================] - 0s 18ms/step - loss: 11.6901 -
val_loss: 16.8250
Epoch 58/100
11/11 [==============================] - 0s 16ms/step - loss: 11.5753 -
val_loss: 16.6181
Epoch 59/100
11/11 [==============================] - 0s 14ms/step - loss: 11.5114 -
val_loss: 16.3434
Epoch 60/100
11/11 [==============================] - 0s 13ms/step - loss: 11.3122 -
val_loss: 16.4792
Epoch 61/100
11/11 [==============================] - 0s 12ms/step - loss: 11.1169 -
val_loss: 16.5548
Epoch 62/100
11/11 [==============================] - 0s 8ms/step - loss: 11.0585 - val_loss:
17.0604
Epoch 63/100
```

```
11/11 [==============================] - 0s 11ms/step - loss: 11.0184 -
val_loss: 16.9950
Epoch 64/100
11/11 [==============================] - 0s 8ms/step - loss: 10.8373 - val_loss:
16.2817
Epoch 65/100
11/11 [==============================] - 0s 9ms/step - loss: 10.7698 - val_loss:
16.2099
Epoch 66/100
11/11 [==============================] - 0s 13ms/step - loss: 10.6893 -
val_loss: 16.3938
Epoch 67/100
11/11 [==============================] - 0s 19ms/step - loss: 10.5800 -
val_loss: 16.7752
Epoch 68/100
11/11 [==============================] - 0s 11ms/step - loss: 10.5171 -
val_loss: 16.5071
Epoch 69/100
11/11 [==============================] - 0s 11ms/step - loss: 10.3900 -
val_loss: 15.9741
Epoch 70/100
11/11 [==============================] - 0s 7ms/step - loss: 10.3729 - val_loss:
16.0716
Epoch 71/100
11/11 [==============================] - 0s 7ms/step - loss: 10.3326 - val_loss:
16.3042
Epoch 72/100
11/11 [==============================] - 0s 8ms/step - loss: 10.2334 - val_loss:
15.0136
Epoch 73/100
11/11 [==============================] - 0s 7ms/step - loss: 10.1671 - val_loss:
14.9351
Epoch 74/100
11/11 [==============================] - 0s 8ms/step - loss: 10.0402 - val_loss:
14.9469
Epoch 75/100
11/11 [==============================] - 0s 7ms/step - loss: 9.9076 - val_loss:
15.2001
Epoch 76/100
11/11 [==============================] - 0s 8ms/step - loss: 9.9514 - val_loss:
15.8626
Epoch 77/100
11/11 [==============================] - 0s 8ms/step - loss: 9.9200 - val_loss:
15.1977
Epoch 78/100
11/11 [==============================] - 0s 6ms/step - loss: 9.7315 - val_loss:
14.9133
Epoch 79/100
```

```
11/11 [==============================] - 0s 7ms/step - loss: 9.6199 - val_loss:
15.1582
Epoch 80/100
11/11 [==============================] - 0s 8ms/step - loss: 9.5393 - val_loss:
15.1670
Epoch 81/100
11/11 [==============================] - 0s 9ms/step - loss: 9.4597 - val_loss:
14.5944
Epoch 82/100
11/11 [==============================] - 0s 7ms/step - loss: 9.4416 - val_loss:
14.1969
Epoch 83/100
11/11 [==============================] - 0s 8ms/step - loss: 9.3739 - val_loss:
13.8138
Epoch 84/100
11/11 [==============================] - 0s 7ms/step - loss: 9.3774 - val_loss:
13.9418
Epoch 85/100
11/11 [==============================] - 0s 6ms/step - loss: 9.1815 - val_loss:
14.6091
Epoch 86/100
11/11 [==============================] - 0s 7ms/step - loss: 9.2222 - val_loss:
14.6034
Epoch 87/100
11/11 [==============================] - 0s 8ms/step - loss: 9.0925 - val_loss:
14.3042
Epoch 88/100
11/11 [==============================] - 0s 11ms/step - loss: 9.0882 - val_loss:
14.2241
Epoch 89/100
11/11 [==============================] - 0s 16ms/step - loss: 8.9274 - val_loss:
14.2032
Epoch 90/100
11/11 [==============================] - 0s 15ms/step - loss: 8.9384 - val_loss:
14.2421
Epoch 91/100
11/11 [==============================] - 0s 11ms/step - loss: 8.9018 - val_loss:
14.3701
Epoch 92/100
11/11 [==============================] - 0s 7ms/step - loss: 8.8755 - val_loss:
14.0006
Epoch 93/100
11/11 [==============================] - 0s 10ms/step - loss: 9.1686 - val_loss:
13.1046
Epoch 94/100
11/11 [==============================] - 0s 10ms/step - loss: 8.8475 - val_loss:
13.4923
Epoch 95/100
```

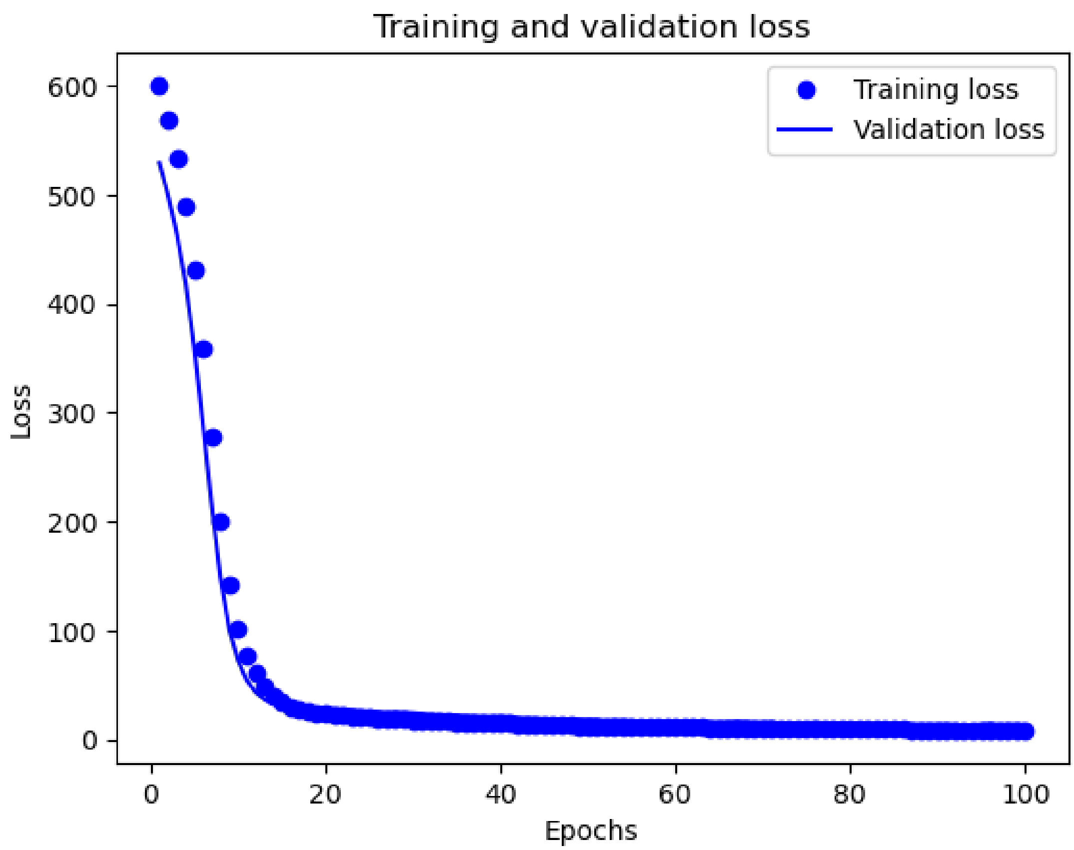
```
11/11 [==============================] - 0s 12ms/step - loss: 8.7029 - val_loss:
14.1478
Epoch 96/100
11/11 [==============================] - 0s 16ms/step - loss: 8.9403 - val_loss:
14.7744
Epoch 97/100
11/11 [==============================] - 0s 14ms/step - loss: 8.7085 - val_loss:
13.9084
Epoch 98/100
11/11 [==============================] - 0s 15ms/step - loss: 8.5641 - val_loss:
14.1686
Epoch 99/100
11/11 [==============================] - 0s 9ms/step - loss: 8.7153 - val_loss:
14.2366
Epoch 100/100
11/11 [==============================] - 0s 8ms/step - loss: 8.4327 - val_loss:
13.7527
```

[6]:
```
# Evaluate the model
model.evaluate(X_test, y_test)
```

```
4/4 [==============================] - 0s 4ms/step - loss: 12.2175
```

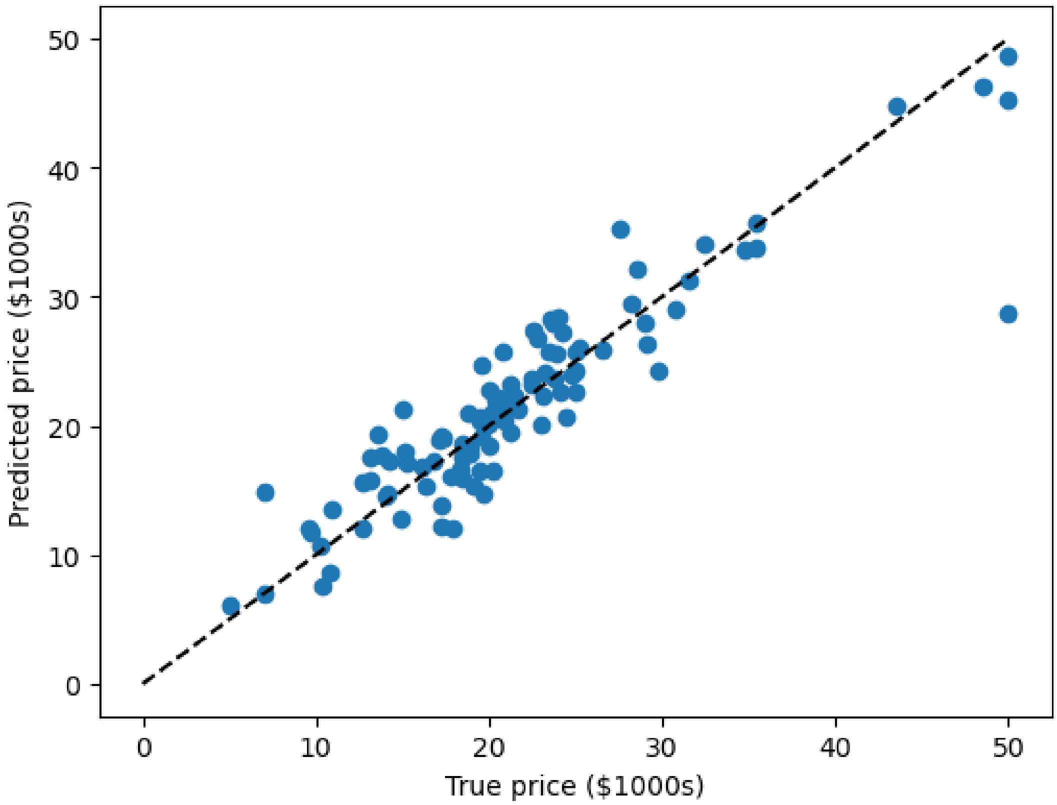
[6]: 12.21752643585205

[9]:
```
# Visualize the training history
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



[10]:
```
# Visualize the linear regression graph
y_pred = model.predict(X_test)

plt.scatter(y_test, y_pred)
plt.plot([0, 50], [0, 50], '--k')
plt.axis('tight')
plt.xlabel('True price ($1000s)')
plt.ylabel('Predicted price ($1000s)')
plt.show()
```

```
4/4 [==============================] - 1s 6ms/step
```



[14]:
```
X_train.shape
```

[14]: (404, 13)

[21]:
```
X_train[0]
```

[21]:
```
array([ 1.28770177, -0.50032012,  1.03323679, -0.27808871,  0.48925206,
       -1.42806858,  1.02801516, -0.80217296,  1.70689143,  1.57843444,
        0.84534281, -0.07433689,  1.75350503])
```

5 Checking for Custom Input

[24]:
```
custom_input = np.array([[0.2, 0.0, 10.0, 0.0, 0.5, 6.0, 70.0, 3.0, 4.0, 400.0,
 ↪17.0, 360.0, 15.0]])
scaled_custom_input = scaler.transform(custom_input)
```

```
D:\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have
valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

[25]:
```
custom_prediction = model.predict(scaled_custom_input)
print(f"Custom Input Prediction: {custom_prediction[0][0]}")
```

```
1/1 [==============================] - 2s 2s/step
Custom Input Prediction: 18.05543327331543
```

[ ]:
```
```