Answers	Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Subject Teacher
5	5	5	5	20	

Start Date :..... Date of Completion:.....

Group B Deep Learning

Assignment No: 1

Title of the Assignment: Linear regression by using Deep Neural network: Implement Boston housing price. prediction problem by Linear regression using Deep Neural network. Use Boston House price prediction dataset.

Objective of the Assignment: Students should be able to perform Linear regression by using Deep Neural network on Boston House Dataset.

Prerequisite:

1. Basic of programming language
2. Concept of Linear Regression
3. Concept of Deep Neural Network

Contents for Theory:

1. What is Linear Regression
2. Example of Linear Regression
3. Concept of Deep Neural Network
4. How Deep Neural Network Work
5. Code Explanation with Output

What is Linear Regression?

Linear regression is a statistical approach that is commonly used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the variables and uses mathematical methods to estimate the coefficients that best fit the data.

Deep neural networks are a type of machine learning algorithm that are modeled after the structure and function of the human brain. They consist of multiple layers of interconnected neurons that process data and learn from it to make predictions or classifications.

Linear regression using deep neural networks combines the principles of linear regression with the power of deep learning algorithms. In this approach, the input features are passed through one or more layers of neurons to extract features and then a linear regression model is applied to the output of the last layer to make predictions. The weights and biases of the neural network are adjusted during training to optimize the performance of the model.

This approach can be used for a variety of tasks, including predicting numerical values, such as stock prices or housing prices, and classifying data into categories, such as detecting whether an image contains a particular object or not. It is often used in fields such as finance, healthcare, and image recognition.

Example Of Linear Regression

A suitable example of linear regression using deep neural network would be predicting the price of a house based on various features such as the size of the house, the number of bedrooms, the location, and the age of the house.

In this example, the input features would be fed into a deep neural network, consisting of multiple layers of interconnected neurons. The first few layers of the network would learn to extract features from the input data, such as identifying patterns and correlations between the input features.

The output of the last layer would then be passed through a linear regression model, which would use the learned features to predict the price of the house.

During training, the weights and biases of the neural network would be adjusted to minimize the difference between the predicted price and the actual price of the house. This process is known as gradient descent, and it involves iteratively adjusting the model's parameters until the optimal values are reached.

Once the model is trained, it can be used to predict the price of a new house based on its features. This approach can be used in the real estate industry to provide accurate and reliable estimates of house prices, which can help both buyers and sellers make informed decisions.

Concept of Deep Neural Network-

A deep neural network is a type of machine learning algorithm that is modeled after the structure and function of the human brain. It consists of multiple layers of interconnected nodes, or artificial neurons, that process data and learn from it to make predictions or classifications.

Each layer of the network performs a specific type of processing on the data, such as identifying patterns or correlations between features, and passes the results to the next layer. The layers closest to the input are known as the "input layer", while the layers closest to the output are known as the "output layer".

The intermediate layers between the input and output layers are known as "hidden layers". These layers are responsible for extracting increasingly complex features from the input data, and can be deep (i.e., containing many hidden layers) or shallow (i.e., containing only a few hidden layers).

Deep neural networks are trained using a process known as back propagation, which involves adjusting the weights and biases of the nodes based on the error between the predicted output and the actual output. This process is repeated for multiple iterations until the model reaches an optimal level of accuracy.

Deep neural networks are used in a variety of applications, such as image and speech recognition, natural language processing, and recommendation systems. They are capable of learning from vast amounts of data and can automatically extract features from raw data, making them a powerful tool for solving complex problems in a wide range of domains.

How Deep Neural Network Work-

Boston House Price Prediction is a common example used to illustrate how a deep neural network can work for regression tasks. The goal of this task is to predict the price of a house in Boston based on various features such as the number of rooms, crime rate, and accessibility to public transportation.

Here's how a deep neural network can work for Boston House Price Prediction:

1. **Data preprocessing:** The first step is to preprocess the data. This involves normalizing the input features to have a mean of 0 and a standard deviation of 1, which helps the network learn more efficiently. The dataset is then split into training and testing sets.
2. **Model architecture:** A deep neural network is then defined with multiple layers. The first layer is the input layer, which takes in the normalized features. This is followed by several hidden layers, which can be deep or shallow. The last layer is the output layer, which predicts the house price.
3. **Model training:** The model is then trained using the training set. During training, the weights and biases of the nodes are adjusted based on the error between the predicted output and the actual output. This is done using an optimization algorithm such as stochastic gradient descent.
4. **Model evaluation:** Once the model is trained, it is evaluated using the testing set. The

performance of the model is measured using metrics such as mean squared error or mean absolute error.

5. **Model prediction:** Finally, the trained model can be used to make predictions on new data, such as predicting the price of a new house in Boston based on its features.
6. By using a deep neural network for Boston House Price Prediction, we can obtain accurate predictions based on a large set of input features. This approach is scalable and can be used for other regression tasks as well.

Boston House Price Prediction Dataset-

Boston House Price Prediction is a well-known dataset in machine learning and is often used to demonstrate regression analysis techniques. The dataset contains information about 506 houses in Boston, Massachusetts, USA. The goal is to predict the median value of owner-occupied homes in thousands of dollars.

The dataset includes 13 input features, which are:

CRIM: per capita crime rate by town

ZN: proportion of residential land zoned for lots over 25,000 sq.ft.

INDUS: proportion of non-retail business acres per town

CHAS: Charles River dummy variable (1 if tract bounds river; 0 otherwise)

NOX: nitric oxides concentration (parts per 10 million)

RM: average number of rooms per dwelling

AGE: proportion of owner-occupied units built prior to 1940

DIS: weighted distances to five Boston employment centers

RAD: index of accessibility to radial highways

TAX: full-value property-tax rate per $10,000

PTRATIO: pupil-teacher ratio by town

B: 1000(Bk - 0.63)^2 where Bk is the proportion of black people by town

LSTAT: % lower status of the population

The output variable is the median value of owner-occupied homes in thousands of dollars (MEDV).

To predict the median value of owner-occupied homes, a regression model is trained on the dataset. The model can be a simple linear regression model or a more complex model, such as a deep neural network. After the model is trained, it can be used to predict the median value of owner-occupied homes based on the input features. The model's accuracy can be evaluated using metrics such as mean squared error or

mean absolute error.

Boston House Price Prediction is a example of regression analysis and is often used to teach machine learning concepts. The dataset is also used in research to compare the performance of different regression models.

Source Code with Explanation-

```
# Import necessary libraries
import numpy as np  # For numerical operations
import pandas as pd  # For handling datasets
from sklearn.model_selection import train_test_split  # Splitting data into train & test sets
from sklearn.linear_model import LinearRegression  # Linear Regression Model
from sklearn.preprocessing import StandardScaler  # Standardization of data
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score  # Evaluation metrics

# Importing Keras (for Neural Network)
import keras
from keras.models import Sequential  # To define a sequential model
from keras.layers import Dense  # Fully connected layers

# Importing Google Colab file handling utility
from google.colab import files

# Uploading and Loading Dataset
uploaded = files.upload()  # Opens file upload dialogue in Google Colab
boston = pd.read_csv("boston_house_prices.csv")  # Reads CSV file into a DataFrame

# Selecting Features and Target
# Selecting 3 input features:
# 1. LSTAT (Percentage of lower status population)
# 2. RM (Average number of rooms per dwelling)
# 3. PTRATIO (Pupil-teacher ratio by town)
X = boston[['LSTAT', 'RM', 'PTRATIO']]

# Target variable: House Price
y = boston['PRICE']

# Splitting the Dataset into Training and Testing Sets
# 80% of data used for training, 20% for testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)


# Standardizing the Dataset (Feature Scaling)
# Standardization improves model performance by normalizing feature values
scaler = StandardScaler()  # Initializing StandardScaler
X_train_scaled = scaler.fit_transform(X_train)  # Fit and transform training data
X_test_scaled = scaler.transform(X_test)  # Transform test data using the same scaler

# Linear Regression Model
lr_model = LinearRegression()  # Initializing Linear Regression Model
```

```
lr_model.fit(X_train_scaled, y_train)  # Training the model using scaled training data

# Predicting house prices on test data
y_pred_lr = lr_model.predict(X_test_scaled)

# Evaluating Linear Regression Model
mse_lr = mean_squared_error(y_test, y_pred_lr)  # Mean Squared Error
mae_lr = mean_absolute_error(y_test, y_pred_lr)  # Mean Absolute Error
r2_lr = r2_score(y_test, y_pred_lr)  # R² Score (Model accuracy measure)

# Displaying evaluation metrics
print("Linear Regression Model Evaluation:")
print(f"Mean Squared Error: {mse_lr}")
print(f"Mean Absolute Error: {mae_lr}")
print(f"R2 Score: {r2_lr}")

# Neural Network (ANN) Model
# Creating a Deep Learning Model using Keras Sequential API
model = Sequential([
    Dense(128, activation='relu', input_dim=3),  # Input layer (3 features) & first hidden layer (128 neurons)
    Dense(64, activation='relu'),  # Second hidden layer with 64 neurons
    Dense(32, activation='relu'),  # Third hidden layer with 32 neurons
    Dense(16, activation='relu'),  # Fourth hidden layer with 16 neurons
    Dense(1)  # Output layer (Predicting a single value - House Price)
])

# Compiling the model
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
# Optimizer: Adam (Adaptive Learning Rate Optimization)
# Loss function: Mean Squared Error (MSE) - Suitable for regression problems
# Metric: Mean Absolute Error (MAE) - Helps measure performance

# Training the Neural Network
history = model.fit(X_train_scaled, y_train, epochs=100, validation_split=0.05, verbose=1)
# Training for 100 epochs
# Using 5% of training data as validation set to monitor overfitting
# `verbose=1` displays detailed training progress
#Epoch 1/100
#12/12 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 26ms/step - loss: 547.8306 - mae: 21.6359 -
#val_loss: 445.7750 - val_mae: 20.1572
#Epoch 2/100
#12/12 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 550.6208 - mae: 21.6498 -
#val_loss: 403.5681 - val_mae: 19.1308
#Epoch 3/100
#12/12 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 433.7596 -
# Evaluating the Neural Network Model
y_pred_nn = model.predict(X_test_scaled)  # Predicting house prices on test data
mse_nn, mae_nn = model.evaluate(X_test_scaled, y_test)  # Evaluating model performance

# Displaying Neural Network Evaluation Metrics
print("\nNeural Network Model Evaluation:")
```

```
print(f"Mean Squared Error: {mse_nn}")
print(f"Mean Absolute Error: {mae_nn}")

# House Price Prediction for New Data
new_data = np.array([[0.1, 10.0, 5.0]])
# New input values: LSTAT=0.1, RM=10.0, PTRATIO=5.0

new_data_scaled = scaler.transform(new_data)
 # Applying the same standardization as training data

# Predicting price using trained neural network model
prediction = model.predict(new_data_scaled)

# Displaying the predicted house price
print("\nPredicted House Price:", prediction[0][0])

#Output
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
```

Predicted House Price: 79.24278 thousands dollars.

Conclusion- In this way we can predict the Boston House Price using Deep Neural Network.

Assignment Question

1. What is Linear Regression?
2. What is a Deep Neural Network?
3. What is the concept of standardization?
4. Why split data into train and test?
5. Write Down Application of Deep Neural Network?