Natural Language Processing Assignment 1 Report

Name:Shiv Hansoti (Student-ID:- 1105615)

Abstract— **This document contains methods and steps I used to create Convolution based neural network and the solution is the Non-Linear Regression Model.**

I. INTRODUCTION

For Assignment 1 we were told to create a model using 1D neural network which can predict the housing price and the model must be Non-Linear Regression model. So for that, it is mandatory to understand terms like Linear Regression, Non-Linear Regression, Neural Network.

II. LINEAR REGRESSION

For understanding Non-Linear Regression model it is suggested that one must have knowledge of Linear Regression model. Linear Regression is something which can predict required data based on independent and dependent variables. Dependent variable is on y-axis and independent variable is on x-axis, then a relationship is tried to formed between them. If a independent variable increases and due to it dependent variable increases then there is a positive relation between them. But, if independent variable increases and dependent variable decreases then there is a negative relationship. In Linear Regression, it is tried to fit the line fits all the observations which is also known as Regression Line. Regression line is tried to be framed using certain equations like:- Y=C+C'B where C is Y-intercept, C' is slope and B is X-intercept. (Positive Relationship) Y=C-C'B (Negative Relationship).

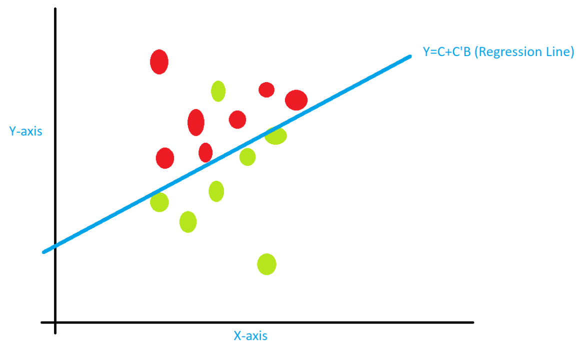


Fig. 1. Linear Regression

*

III. NON-LINEAR REGRESSION

Non-Linear Regression is same as Linear Regression but it contains different Non-Linear features. Mainly there are two different ways to build Non-Linear features that are creating feature manually or using kernel for it.

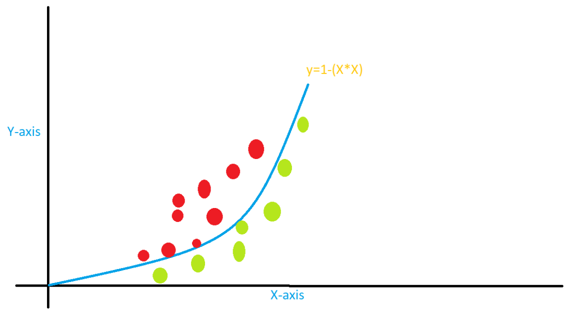


Fig. 2. Non-Linear Regression

IV. NEURAL NETWORK

Neural Network are inspired by brains. There are two different terms Neural and network. Consider Neuron as nothing but a box holding some identical value specifically between 0 and 1. Network is a connections between neurons. There are 3 layers in a network. 1) Input layer, 2) Hidden layer, 3) Output layer.

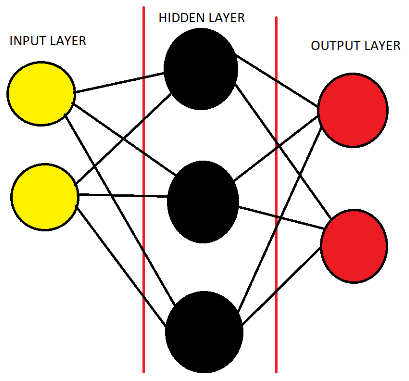


Fig. 3. Neural Network

Neural network works in following steps:-

- Defining a network structure (input node, weights, outputs).
- Training a defined network using certain functions. The process of correcting weights according to the input data is known as Network Training.

- Defining a Loss function for knowing how much data is loss or getting a idea of how module is inaccurate and which factors are needed to tune.
- And based on the designed model, output is obtained and also accuracy can help in determining the correctness in defined model.

A. *Units*

- hi there its a bullet point

V. Flow Chart

Figure 4 shows the flow chart of Neural Network. It describes the working of Artificial Neural Network.

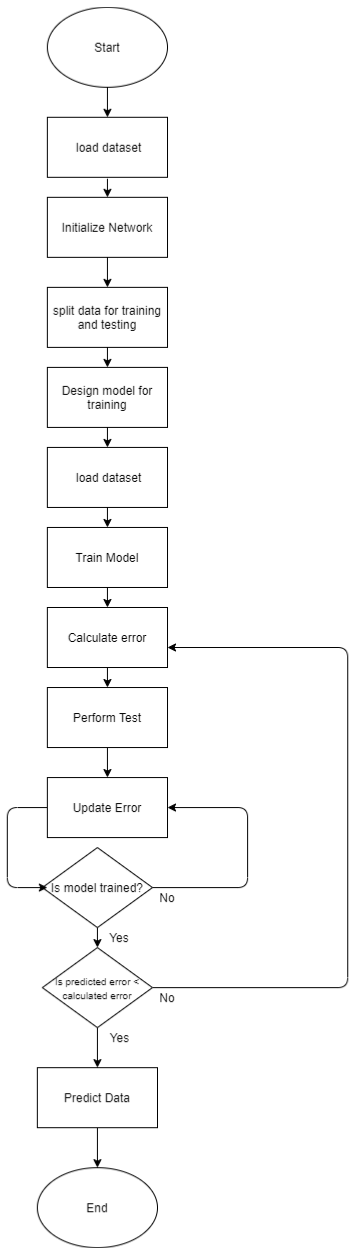


Fig. 4. Flow Chart

VI. Code Description

- First thing first. Import and installing all the required dependencies or libraries required for the program.

- pandas is a free accessible library mainly used for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
- Numpy is a library for playing with large, multi-dimensional arrays and matrices, along with a certain set of mathematical functions to operate on these arrays.
- Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy .

  import pandas as pd
  import numpy as np
  import matplotlib.pyplot as plt
  from sklearn import linear$_{m}odel$
  $fromsklearn.model_{s}electionimporttrain_{t}est_{s}plit$
  $fromsklearn.metricsimportmean_{s}quared_{e}rror, r2_{s}core$

- Retrieving the data-set from the csv file in Google Colab jupyter notebook

- read method is stored in pandas which gets the data from csv file of given path.

  dataset=pd.read$_{c}sv('/content/housing.csv')$

- After reading the data, it is recommended that remove missing values.

  dataset=dataset.dropna()

- The below line of code "Prints the first 10 data or iterates through first 10 rows of data-set from CSV file."

  dataset.head(10)

- Getting the X and Y co-ordinates and further splitting the data-set in ratio of 70:30. 70 percentage data for training and 30 percentage data for testing.

  Y=dataset['median$_{h}ouse_{v}alue']$
  $X = dataset.loc[:,' longitude' :' median_{i}ncome']$

  x$_{t}rain, x_{t}est, y_{t}rain, y_{t}est =$
  $train_{t}est_{s}plit(X, Y, test_{s}ize = 0.3)$

  plotting first 25 dataset (Coloums) of each dataset on graph below shown code is snipet of it.

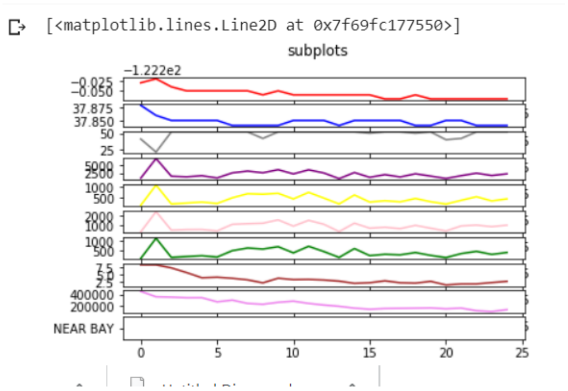
  fig, (ax1, ax2, ax3, ax4, ax5, ax6, ax7, ax8, ax9, ax10) =

plt.subplots(10)

fig.suptitle('subplots')

lst = [i for i in range(25)]

ax1.plot(lst, dataset['longitude'][0:25], color='red',label='longitude')



```
[<matplotlib.lines.Line2D at 0x7f69fc177550>]
```

Fig. 5. First 25

- Network Initialization

- Pytorch was used to initialize a network. It's a Python-based scientific computing package targeted at two sets of audiences: A replacement for NumPy to use the power of GPUs a deep learning research platform that provides maximum flexibility and speed.
- Initially function takes input, batch size and output. After that performing a 1 Dimensional Convolution on Input data.
- After performing 1D convolution we need to perform Max pooling (It is nothing but sampling process on dataset) on layer.

  self.input$_layer = Conv1d(inputs, batch_size, 1)$
  $self.max_pooling_layer = MaxPool1d(1)$
  $self.conv_layer = Conv1d(batch_size, 128, 1)$

- This is layer 1, I used total 2 layers because the data is not linearly separable so a Non-Linear regression neural network model fits best to solve the problem.
- Next step is to flatten the layer, make it Linear and just getting the output.
- defining a feed function
- it is just getting the output of convolution layer, max pooling layer, flatten layer, and linear layer. And applying relu function to remove all the negative values of the matrix and avoid over-fitting problem.

  input = input.reshape((self.batch$_size, self.inputs, 1))$
  $output = relu(self.input_layer(input))$
  $output = self.max_pooling_layer(output)$
  $output = relu(self.conv_layer(output))$
  $output = self.flatten_layer(output)$
  $output = self.linear_layer(output)$
  $output = self.output_layer(output)$

- Now, it's time for training a model
- Defining a loss function and getting a R2 score for it.]
  performance=L1Loss()
  score$_metric = R2Score()$

- Finally adding an optimizer to optimize the model's functionality. I found 3 optimizers were good for classification that are ADAM, Rprop and SGD. But according to variations in data-set I settled down with Adam Optimizer. I was getting R2$_score of 0.5523 using ADAM and using SGD it was 0.5277.$
- I used epoch value to 637 as it was threshold value for it according to me. Model suffered problem of over fitting on Increasing number of epochs

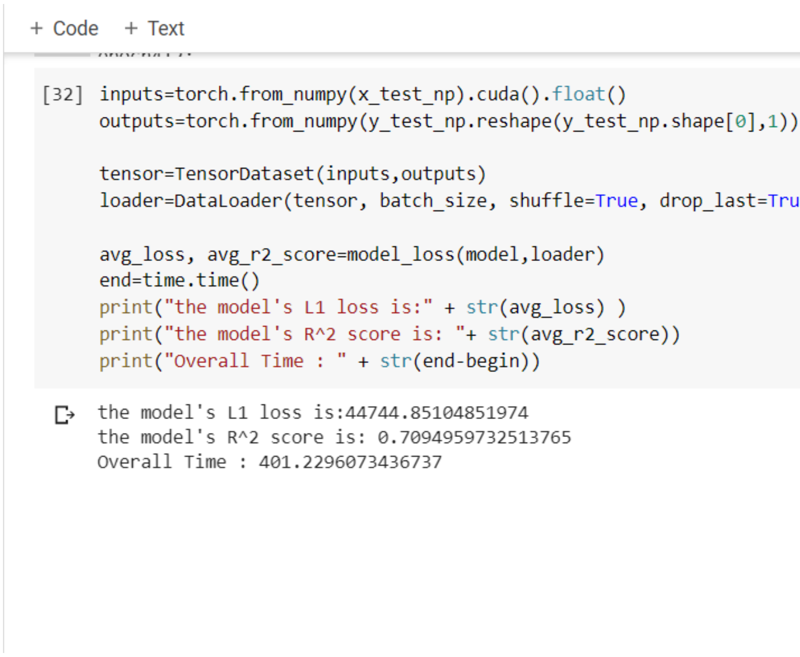
  for epoch in range(epochs):

  avg$_loss, avg_r2_score =$

  model$_loss(model, loader, train = True, optimizer = optimizer)$

  print("epoch" + str(epoch+1)+":= "+ str(avg$_loss) + "^2Score = " + str(avg_r2_score))$

- And at the end using time function to calculate the amount of time to train the model and get output.



```
[32] inputs=torch.from_numpy(x_test_np).cuda().float()
     outputs=torch.from_numpy(y_test_np.reshape(y_test_np.shape[0],1))

     tensor=TensorDataset(inputs,outputs)
     loader=DataLoader(tensor, batch_size, shuffle=True, drop_last=Tru

     avg_loss, avg_r2_score=model_loss(model,loader)
     end=time.time()
     print("the model's L1 loss is:" + str(avg_loss) )
     print("the model's R^2 score is: "+ str(avg_r2_score))
     print("Overall Time : " + str(end-begin))
```

```
the model's L1 loss is:44744.85104851974
the model's R^2 score is: 0.7094959732513765
Overall Time : 401.2296073436737
```

Fig. 6. Output

- After training the model when I tested it,i got the model's

L1 loss is:44744.85104851974
the model's R$^2 scoreis$: 0.7094959732513765
$OverallTime$: 401.2296073436737

VII. Conclusion

Hence, it could be found that 1D-Convolution Neural Network and a Non-linear regression model could be helpful in predicting the median house value and furthermore it could be easily trained with the online resources available for experimental purposes.Neural networks are preferred for predicting and recognising pattern mainly because of learning only from samples and previous data. Neural networks can handle noise. On the other hand, it is difficult to decide what a neural network learned from the data, and possible prediction error as well. However, neural networks used for predicting data which has identical patterns.

.

References

[1] M. Daneshdoost , M. Lotfalian , G. Bumroonggit , J.P. Ngoy,Neural network with fuzzy set-based classification for short-term load forecasting,IEEE transaction,2009

[2] CHAN Man-Chung, WONG Chi-Cheong, LAM Chi-Chung,Financial Time Series Forecasting by Neural Network Using Conjugate Gradient Learning Algorithm and Multiple Linear Regression Weight Initialization, Economic and finance. 2000

[3] T.W.S. Chow ; C.T. Leung, Neural network based short-term load forecasting using weather compensation,IEEE transaction and power Systems, 1996.