Non-Linear Regression Model

Vashishtha Master

1100901

Master of Computer Science Department, Lakehead University
vmaster@lakeheadu.edu

Abstract—*This paper focuses on Non-Linear Regression Model which is being implemented using 1D Convolutional Layer and predicted the house value based on other 8 numerical features. For the implementation of purpose, I have used housing dataset from the GitHub link provided. For Scaling purpose, I have used MinMaxScalar. To predict the house value, I have constructed a nonlinear regression model in my CNN I have included multiple convolutional layer and have used RELU as an activation function. By the end of my assignment I had done tuning for hyperparameters in order to achieve high R2 score and low L1 loss. At the completion of my practical I can achieve R2 score of 78.29%.*

1. Introduction

The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.[1]

In this paper we have used CNN which in common is multiple convolutional layers and we have used RELU as an activation function. RELU is a non-linear function so at the end is makes model non-linear. Throughout the paper I would be talking about the step and actions which I took to predict the house value and how I achieved the R2 score.

2. Literature Review

2.1 Dataset

This dataset showed up in a 1997 in paper titled Sparse Spatial Autoregressions by Pace, R. Kelley and Ronald Barry, distributed in the Statistics and Probability Letters diary. They fabricated it utilizing the 1990 California census information. It contains one line for every evaluation square gathering. A square gathering is the littlest land unit for which the U.S. Statistics Bureau distributes test information (a square gathering ordinarily has a populace of 600 to 3,000 individuals). It has following attributes:

- longitude:
- latitude
- housingMedianAge:
- totalRooms:
- totalBedrooms
- population
- households
- medianIncome
- medianHouseValue
- oceanProximity

2.2 Libraries

In order to run my model, I have utilized following libraries:

- **torch**: I have utilized torch library for development and training non-linear regression model
- **pandas:** In my practical I have used pandas for reading data from csv file.
- **Sklearn:** For normalization of data sklearn library is used. For example: MinMaxScaler().
- **Numpy:** It library is used for core scientific computing.

3. Implementation

For implementing our code, we are using google Collaboratory which has in built Jupyter notebook and performs each computation on cloud. By using this technology, it enables us to explore and

perform programming tasks on cloud itself which helps user in keeping track of the work done easily, it provides with higher computational power than local computer so high end tasks can be done quickly.

In my program firstly I have read dataset from github link provided using pandas (shown in Fig.3.1). Then I have displayed first 10 rows of the dataset as stated in the requirement and plotted graph for all features in different subplots. So, once we have the data, we need to perform normalization on our dataset. In my case I have used MinMaxScaler. Then I have performed splitting of dataset for testing and training I went with 70% of data for training and 30% of data for testing at Random state set to 2003. Then I created non-linear regression model for the problem to achieve high R2 score as possible. During my implementation in order to resolve the problem of over/under fitting I have increased the convolutional from 1 to 2 layers and increased the number of epochs to 300.

I have also used zero_grad() in my implementation it clears the old gradients from the last step which would otherwise just accumulate the gradients from all the losses in backward() calls. And loss.backward() computes the derivative of the loss w.r.t. the parameters (or anything requiring gradients) using backpropagation. Thus, my doing this we can handle the problem of Vanishing/ Exploding Gradient. And for optimization I have taken Adam optimizer. The reason behind this is it maintains a per-parameter learning rate that improves performance on problems with sparse gradients which is really use full in NLP and computer vision concepts.

4. Proposed Model

While implementing non-linear regression model below are the hyper-parameters through which I my model received its final R2 score.

- Batch size: 64
- Epochs: 300
- Learning Rate: 1e-3
- Optimizer: Adam
- Kernel size: 1

After running code with above mentioned parameters my model received R2 score for testing of 0.7829 that is **78.29%**

5. Some Important Code

Following are some codes used while my implementation:

- Splitting of Dataset:

```
x_train, x_test, y_train, y_test = t
rain_test_split(X, Y, test_size=0.3,
 random_state=2003)
from sklearn.preprocessing import Mi
nMaxScaler #importing MinMaxScaler f
or scaling purpose
ms = MinMaxScaler()
x_train_np = ms.fit_transform(x_trai
n)
y_train_np = y_train.to_numpy()
x_test_np = ms.transform(x_test)
y_test_np = y_test.to_numpy()
```

- My Model:

```
# Student ID : 1100901, Name: Vashih
stha Master
class _1100901_1dconv_reg(torch.nn.M
odule):
#defining initialization method
  def __init__(self, batch_size, inp
uts, outputs):
#initializing super class and store
parameters
    super(_1100901_1dconv_reg, self)
.__init__()
    self.batch_size = batch_size
    self.inputs = inputs
    self.outputs = outputs
    #defining Convolutional layer ke
rnel size=1
    self.input_layer = Conv1d(inputs
, batch_size, 1)
    #defining Max pooling layer kern
el size=1
    self.max_pooling_layer = MaxPool
1d(1)
```

```
        #another 2 convolutional layers
        self.conv_layer = Conv1d(batch_s
    ize, 128, 1)
        self.conv_layer2 = Conv1d(128, 1
    28, 1)
        #defining flatten layer
        self.flatten_layer = Flatten()
        #defining linear layer
        self.linear_layer = Linear(128,
    64)
        #defining output layer
        self.output_layer = Linear(64, o
    utputs)
```

- Code to remove Vanishing/ Exploding Gradient:

```
#will remove gradient
optimizer.zero_grad()

#will recover it
loss.backward()
```

- Code for optimizer Adam

```
epochs = 300
#defining perfomance measure and opt
imizer
optimizer = Adam(model.parameters(),
 lr=1e-3)
```

6. Results and Screenshots

```
[24] #below is the url for housing dataset (csv file)
url="https://raw.githubusercontent.com/ageron/handson-ml/master/datasets/housing/housing.csv"

dataset=pd.read_csv(url) #reading data from url
```

Fig 1. Reading dataset form the given url.

Here are the first 10 rows of the dataset:

	longitude	latitude	...	median_house_value	ocean_proximity
0	-122.23	37.88	...	452600.0	NEAR BAY
1	-122.22	37.86	...	358500.0	NEAR BAY
2	-122.24	37.85	...	352100.0	NEAR BAY
3	-122.25	37.85	...	341300.0	NEAR BAY
4	-122.25	37.85	...	342200.0	NEAR BAY
5	-122.25	37.85	...	269700.0	NEAR BAY
6	-122.25	37.84	...	299200.0	NEAR BAY
7	-122.25	37.84	...	241400.0	NEAR BAY
8	-122.26	37.84	...	226700.0	NEAR BAY
9	-122.25	37.84	...	261100.0	NEAR BAY

Fig 2. Displaying First 10 rows of dataset

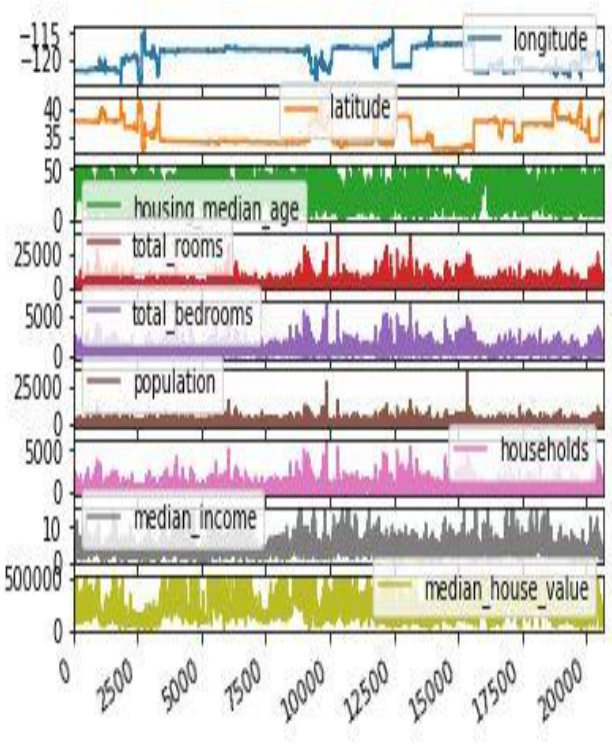

Fig 3. Plotting all the features on different subplot

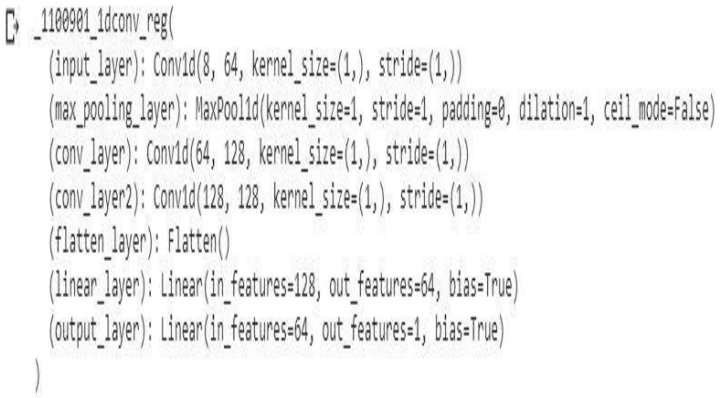

Fig 4. My proposed model

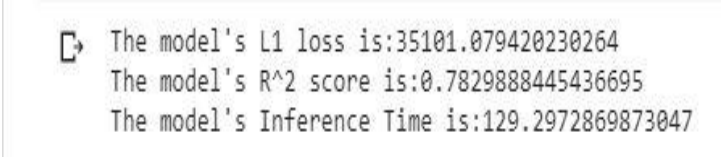

Fig 5. Performance Output of my model

7. Conclusion

This assignment has helped and improved my knowledge about developing a CNN for my needs, I have also learned to work in colab as this was my first project in colab, Working with Colob was really nice experience. with okay score of my model I think I can improve the accuracy of this model by adding some more layer in future and one of the issues deciding on learning rate and batch size which in thinking can be only done by trying again and again. Besides all these I got to learn more in depth about different optimizer, normalization technique so that we can improve our model's accuracy.

8. Acknowledgment

I would like to acknowledge that the efforts which were made by Professor Dr. Thangarajah Akilan was remarkable not only he introduced the topic in detail it has created huge interest for this subject in my mind. Also the work done by Andrew and Purandeep who were always available for help made by progress of learning the concept really easy.

9. Link for my GitHub Repo

https://github.com/vashishtha18/nlp_assignment1

References

[1] Ian Goodfellow and Yoshua Bengio and Aaron Courville (2016). *Deep Learning*. MIT Press. p. 326.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[3] K. Elissa, "Title of paper if known," unpublished.

[4] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.