Nonlinear regression with Deep Learning

Nishchay N. Trivedi
1106924
Department of Computer Science
Lakehead University
Thunder Bay, Canada
February 13, 2020

***Abstract*—Before buying house, it becomes necessary to get housing value. Sometimes it happens that some brokers charge more amount than the original value. So it becomes necessary for the user to get the price of housing. In this assignment, housing value is predicted for housing dataset using nonlinear regression with convolutional neural networks(CNN). The task is performed with two convolution layers followed by relu and maxpooling layer. By setting different parameters I tried to achieve higher performance.**

***Index Terms*—Convolution Neural Network (CNN), Nnlinear regression (NLR), housing dataset**

I. Introduction

Before moving to implementation it is necessary to understand the main terms: Regression and CNN. Regression is a term mainly used in finance, investing etc. to get the value based on the previous outcome. Now linear regression can be categorized in further two terms: 1) Linear Regression and 2) Nonlinear Regression. 1) By applying a linear regression to observed data, linear regression attempts to model the relationship between two variables. .One variable is considered an explanatory variable, and the other variable is considered as dependent. For example, a modeller might want a linear regression model to link the weights of individuals to their heights. 2) Nonlinear regression is a regression in which the variables of dependency or criterion are modelled as a nonlinear function of model parameters and one or more independent variables.

Here the CNN is used with regression to predict value of housing value using relu activation function that makes it nonlinear. To carry out the task different parameters are set based on comparing results such as: learning rate, optimizer, epochs, batch size, kernel etc.

II. Literature review

For the same housing dataset one research has been done R. Kelley Pace, Ronald Barry [1]. In this paper they used spatial autoregression to predict the value in using ordinary least squares (OLS) they checked the result. But in my proposed method, To build a model, I have used NLR with CNN (conv1d) to train and test dataset on the proportion of 70:30. Finally the the perfromance is checked with Root mean square error (RMSE) and R2Score.

Feature	type
longitude	float64
latitude	float64
housing_median_age	float64
total_rooms	float64
total_bedrooms	float64
households	float64
median_income	float64
median_house_value	float64
ocean_proximity	float64

TABLE I
California Dataset.

III. Proposed method

In this section, I will be providing steps taken to complete task. Initially, the dataset is taken of California Housing from github.

A. Dataset

California Housing is used which has 10 columns and 20640 rows. The information of features is given in TABLE 1.

B. Tools

To complete this task python programming language is used with some libraries used to perform machine learning task. The list of library with description is given below:

1) **Pandas**: It is the python's data analytic library. In this task is used to load data as dataframe and used in later part.
2) **Numpy**: The main purpose of the library is to use some mathematical operations. In this task it is used to convert dataframe to numpy arrays that is used later.
3) **Sklearn**: This is the main machine learning libraries. It is used to divide dataset randomly in proportion of 70:30 where 70% of dataset is treated as train and 30% is used as test data to check the performance of results.
4) **Matplotlib**: To plot the features it is used.
5) **Pytorch**: To perform the main task that is to train model using conv1d this library is used. The main loss and R2Score is checked with this library. item **Time**: To calculate the execution of program this library is used.

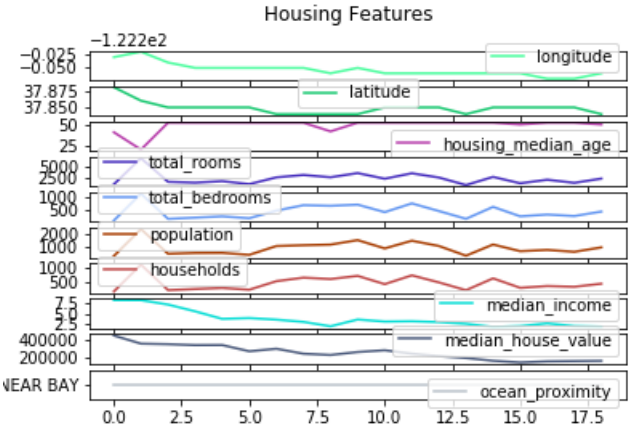

Fig. 1. Plot of features

C. Steps

Initially the program loads data in dataframe using pandas. And the first ten rows of that dataset is shown in figure-1.

Once data is loaded in dataframe the the first 18 values of all features are represented as a graph. the graph generated is shown as Fig 2. The code snippet to generate this plot is given as below.

```
axSubPlt = []

fig, subPlotObj = plt.subplots(10)
fig.suptitle('Housing Features')
for i in range(10):
  axSubPlt.append(subPlotObj[i])

name_list = []

for i in dataset:
  name_list.append(str(i))

def random_color():
    rgba=(random.random(),random.random()
    ,random.random())
    return rgba

numbers = [i for i in range(19)]

for i in range(10):
  axSubPlt[i].plot(numbers,
  dataset.iloc[:,i][0:19],
  color = random_color())
  axSubPlt[i].legend([name_list[i]])

plt.savefig('books_read.png')

    return M
```

Once the dataset is loaded, it is divided in train and split dataset. this split is done using train_test_split() of sklearn library.

```
x_train, x_test, y_train, y_test =
train_test_split(X, Y, test_size= 0.3,
random_state = 2003)
```

After that, all required libraries to train model using con1d are imported. that is shown as below.

```
import torch
from torch.nn import Conv1d
from torch.nn import MaxPool1d
from torch.nn import Flatten
from torch.nn import Linear
from torch.nn.functional import relu
from torch.utils.data import DataLoader,
                             TensorDatasetz
from torch.optim import SGD, Adam
from torch.nn import L1Loss

!pip install pytorch-ignite
from ignite.contrib.metrics.regression.r2_score
                        import R2Score
```

After importing all the libraries, the model is designed with conv1d(). The code snippet is show as below. The detailed information of created model is defined after the snippets.

```
class CnnRegressor(torch.nn.Module):
  def __init__(self, batch_size, inputs, outputs):
    super(CnnRegressor, self).__init__()
    self.batch_size = batch_size
    self.inputs = inputs
    self.outputs = outputs
    self.input_layer = Conv1d(inputs, batch_size,
    self.max_pooling_layer = MaxPool1d(1)
    self.conv_layer = Conv1d(batch_size, 64, 1)1)
    self.flatten_layer = Flatten()
    self.linear_layer = Linear(64, 32)
    self.output_layer = Linear(32, outputs)

  def feed(self, input):
    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.max_pooling_layer1(output)
    output = self.flatten_layer(output)
    output = self.linear_layer(output)
    output = self.output_layer(output)
    return output
```

Initially one convolutional layer is defined, which is given to relu layer after that maxpooling is done. Basically, This

Parameters	Value
Learning rate	0.01
Batch size	64
Epochs	500
Hidden layers	2
Optimizer	Adam
Kernel size	1

TABLE II
PARAMETERS VALUE.

is one hidden layer that combined all convolutional, relu and maxpooling layers. after that same second layer is defined and that is followed by flatten layer. Ultimately linear layer and lastly output layer is designed that gives the final result.

The different parameters used are:

- Learning rate
- Batch size
- Epochs
- Hidden layers
- Optimizer
- Kernel size

After, several experiments I ended up with fixed values that gave in better performance. Which is shown in table 2.

The trained model is shown as below:

```
CnnRegressor(
  (input_layer): Conv1d(8, 32, kernel_size=(1,),
  stride=(1,))
  (max_pooling_layer): MaxPool1d(kernel_size=1,
  stride=1, padding=0, dilation=1, ceil_mode=False)
  (conv_layer): Conv1d(32, 64, kernel_size=(1,),
  stride=(1,))
  (flatten_layer): Flatten()
  (linear_layer): Linear(in_features=64,
  out_features=32, bias=True)
  (output_layer): Linear(in_features=32,
  out_features=1, bias=True)
)
```

For every training epoch the different loss and r2score is calculated and plotted in image 3.

IV. RESULTS

During performing some results are tested such as:

- If we increase number of epochs the model tends to overfit. if we increase the epochs from 500 to 2000 it was giving the learning r2score of arond 0.80 and while testing on that model r2score dropped to 0.55
- While increasing the number of hidden layers model tries to extract more features that ends up overfitting and performs poor in testing dataset.
- Out of all 3 optimizers: 1) Adam, 2) SGF and 3) RPROP that I used in experiment Adam gave the best performance.

Testing parameters	score
L1LOSS	45831.83
R2SCORE	0.7018
Average time	400.29 sec

TABLE III
FINAL RESULT

The final result is shown in table - 3

REFERENCES

[1] R. Kelley Pace a, Ronald Barry b, "Sparse spatial autoregressions", STATISTICS PROBABILITY LET'IrlRS, ELSEVIER, 1997

[2] Donald F. Specht, "A General Regression Neural Network", IEEE TRANSACTIONS ON NEURAL NETWORKS. VOL. 2. NO. 6. NOVEMBER 1991

[3] Author links open overlay panelE.P.NahasM.A.HensonD.E.Seborg , "Nonlinear internal model control strategy for neural network models" , Computers Chemical Engineering Volume 16, Issue 12, December 1992, Pages 1039-1057

[4] J.-P. Vila ; V. Wagner ; P. Neveu, "Bayesian nonlinear model selection and neural networks: a conjugate prior approach", IEEE, March 2000