Predicting Housing Price

Bhargav Lad
Student ID:1117021
Lakehead University,Thunder Bay, ON
bamratbh@lakeheadu.ca

***Abstract*— This report explains the use of neural network based model that uses 1D convolution layer and linear layer to solve regression problem. In this work, a modified version of California housing dataset is used. We test our models' performance using R^2 score.**

Project: :https://github.com/isbhargav/House-price-regression

Keywords*— *linear regression, 1D convolution.

I. Introduction

Regression is a task in which we try to approximate mapping function f(x) from input variables x to output variable y. In case of regression problems the dependent variable y is a continuous variable having real value. In most cases these are often an integer or floating point value that represents quantities such as number of items or price. Here we deal with one such problem of predicting the prices of houses in california based on other variables like population, median income, total number rooms, number of bathrooms ..etc.

II. Data Analysis

A. Data Source

The dataset is obtained from at https:// github.com/ageron/handson-ml/tree/master/datasets/housing. This is a modified version of the original California Housing dataset. Data is in CSV format and provides information like population, median income, total number of rooms, number of bathrooms, price of the house ..etc. In this project we will use meadian_house_price as our dependent variable and longitude, latitude, housing_median_age, total_rooms, total_bedrooms, population, households, median_income as our dependent variables. We won't be using ocean_proximity.

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

Figure 1: Dataset

.

B. Preprocessing

The dataset needs to be preprocessed to handle missing values and remove ocean_proximity column that we won't be using in our project. To handle missing values in our dataset to decide to drop those rows with missing data. We will use pandas to read our data and for the above preprocessing steps.We also need to scale our data in order to feed it into our network. For scaling the data we use sklearn's standard scaler which performs Z-score normalization.

C. Train and Test split

We split out data using sklearn's train_test_split method into 7:3 ratio where 70% samples are in our training dataset and 30% samples are in our testing set. We also set our random seed equal to 2003 for the reproducibility of our experiment.

III. Visualizing The Data

Visualization is a very important aspect of analyzing your dataset. Here we will use seaborn and matplotlib to visualize the features of our dataset.

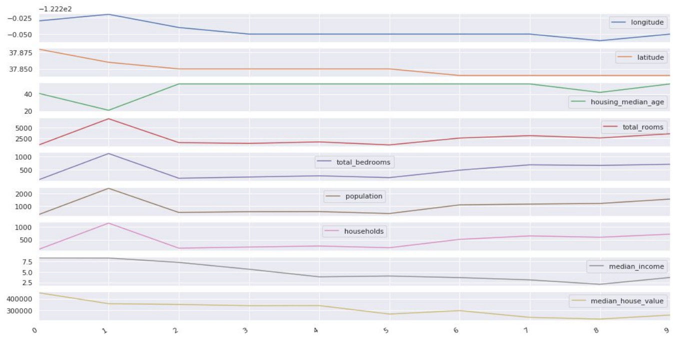

Figure 2: Plot of each feature in dataset for first 10 points.

IV. Network Architecture

The Network architecture used in this experiment has 4 convolution layers that perform 1D convolution operation on the dataset. The first convolution layer has kernel size of 3 and 50 output channels. It performs convolution operation with padding of 1 to maintain the dimensions of the data. Then we perform batch normalization operation on the data. The next 3 convolution operation has kernel size of 3, 3 and 4 respectively. The data is passed through a flatten layer. We then have 4 dense layers with 100, 50, 25 and 12 neurons.. After each convolution layer and dense layer we perform activation using leaky relu. We also perform batch normalization after the first dense layer.The output layer of the network has only one neuron.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv1d-1                [-1, 50, 8]             200
       BatchNorm1d-2                [-1, 50, 8]             100
            Conv1d-3               [-1, 100, 6]          15,100
            Conv1d-4               [-1, 150, 4]          45,150
            Conv1d-5               [-1, 200, 1]         120,200
           Flatten-6                  [-1, 200]               0
            Linear-7                  [-1, 100]          20,100
       BatchNorm1d-8                  [-1, 100]             200
            Linear-9                   [-1, 50]           5,050
           Linear-10                   [-1, 25]           1,275
           Linear-11                   [-1, 12]             312
           Linear-12                    [-1, 1]              13
     CnnRegressor-13                    [-1, 1]               0
================================================================
Total params: 207,700
Trainable params: 207,700
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.02
Params size (MB): 0.79
Estimated Total Size (MB): 0.81
----------------------------------------------------------------
```

Figure 3: Network architecture

V. Experimental Setup

D. Tools and Libraries used

a) Jupyter Notebook : Development environment

b) Pandas : Reading the data

c) Sklearn : Preprocessing and Scaling

d) Pytorch : Tensor library

e) Ignite : For score metrics

E. Loss function and Metrics

For the experiment we optimize our network on L1 loss which is also known as Mean Absolute Error(MAE). MAE absolute error is one of the commonly used loss functions for regression problems. It is defined as the absolute difference between our target and predicted values. For our performance measurement we use coefficient of determination or R^2 score. R^2 score value like between $[-\infty,1]$. Positive R^2 value is desirable.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

Figure 4: L1 Loss

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

Figure 5: R^2 loss

F. Hyperparameters Initialization

To train our model we have used the RMS propagation optimization algorithm. We have used pytorch's implementation of RMS propagation with initial learning rate of 0.0003. We use momentum of 0.3 and weight decay of 3e-5. We also use learning rate scheduler to decrease our learning rate as the number of epochs increases. The learning rate decreased by 10% after 10,25,40,60 and 70 epoch. We set out total number of epochs as 100 and batch size as 1024.

```
Epoch 0/100
 Loss: 96495.48737980769  R2: -0.002901928756498793
Epoch 1/100
 Loss: 50582.44651442308  R2: 0.04698789244682854
Epoch 2/100
 Loss: 41548.72055288462  R2: 0.05602902492411647
Epoch 3/100
 Loss: 38955.53125  R2: 0.057721280504650876
Epoch 4/100
 Loss: 37929.51502403846  R2: 0.058423098924105285
Epoch 5/100
 Loss: 36764.066105769234  R2: 0.059140429108655315
Epoch 6/100
 Loss: 36295.86388221154  R2: 0.059328125492389826
Epoch 7/100
 Loss: 35699.38942307692  R2: 0.060059184059786025
Epoch 8/100
 Loss: 35717.43960336538  R2: 0.05984783534656975
Epoch 9/100
```

Figure 4: Training loss and R2 score after each eopch

VI. Results

No	Results		
	Dataset	***Loss***	***R2***
1	Training Set	24995.04	0.067
2	Testing Set	34573.53	0.000126

Figure 6: Results

VII. Conclusion

In this project, we used a modified version of California housing dataset to perform regression task using 1D convolution neural network. In the above experiment we were able to show how you can train you convolution neural net using tabular data. We looked at how we can deal with the problem of vanishing/exploding gradient problem by normalizing our data and also using batch normalization in our network. For the overfitting problem we use drop out and L2 regularization in our network.

The prediction accuracy can be improved by tuning both the algorithm and the data for specific applications. Although this model has low accuracy as a prediction model, it provides a preliminary framework for further analyses.

Acknowledgment

This project is part of Assignment 1 of the Natural Language Processing course. The author would like to thank Dr Akhilan and GAs of the course for their valuable support.

References

[1] Paszke, Adam and Gross, Sam and Chintala, Soumith and Chanan, Gregory and Yang, Edward and DeVito, Zachary and Lin, Zeming and Desmaison, Alban and Antiga, Luca and Lerer, Adam, “Automatic differentiation in PyTorch”,2017

[2] Pedregosa F, Varoquaux, Ga"el, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. Journal of machine learning research. 2011;12(Oct):2825–30.

[3] McKinney W, others. Data structures for statistical computing in python. In: Proceedings of the 9th Python in Science Conference. 2010. p. 51–6.