Convolutional Neural Networks Using Non-Linear Regression Model on Housing dataset

Prashanth Kumar Daram
Student Id: 1116540
Department Of Computer Science
Lakehead University

I. Abstarct

This paper presents implementation of 1-D Convolution based neural networks for predicting better solution. In this paper, one-dimensional convolution neural network predicting median house value using longitude, latitude, housing median age, total number of rooms, total number of bedrooms, population, number of households, and median income. This model uses some of the deep learning-based techniques for generating non-linear regression model. We are using California housing dataset which is in csv form(housing.csv) for predicting the best R2 score. The paper explains about generating the first ten records of dataset and plotting all the samples of dataset using matplotlib. The paper also discusses about methods to load dataset, train the dataset and test the dataset. The goal of this model is to minimize the loss value and generate best R2 score. With the help of machine learning concepts, our model is predicting best values.

keywords used- Kernel Size,batch size, Epoch value, learning rate,neural network, Convolutional , Maxpool, Input, Flatten layers

II. Introduction

Deep learning techniques such as convolution neural networks, recurrent neural networks are becoming popular in artificial intelligence research. Example of convolution neural network is one dimensional convolution neural network. Here I am implementing one dimensional convolution neural network using nonlinear regression model for predicting best solution:

Artificial Neural Networks: We process Data Modeling with the help of artificial neural networks. We use Artificial Neural Networks for performing prediction. The results obtained by this network are very accurate. Representative data is mainly used in artificial neural networks. It is proved that artificial neural networks give greater efficiency and robustness.

Convolution Neural Networks: Convolution Neural Network is an Artificial Neural Network, it is most popularly used for analyzing images. It also can be used in data analysis and classification problems. CNN has ability to detect pattern and makes sense of patterns, pattern detection making CNN most useful.

What makes CNN different form other neural network models: Multilayer perceptrons or other neural networks are fully connected networks. In this network each neuron in one layer is connected to all other neuron in next layer. Whereas, in CNN the neuron is connected to nearest neighbor neuron. CNN makes complex network to be simple. CNN has input layer, output layer and hidden layers which are called as convolution layers. Convolution layer makes CNN most powerful network. The convolution layers contain some filters which helps them to abstract features or patterns. The main advantage of CNN is its architecture. The architecture of CNN is very simple, it contains input layer, convolution layers, pooling, RELU (rectified linear unit layer) acts as activation function ensuring nonlinearity and fully connected layer. The main function of RELU is conversion of non-linear model into linear model. Training and testing of model are simple with CNN. The CNN is used in many applications such as analysing the images, image classification, image and video recognition and in Natural Language Processing.

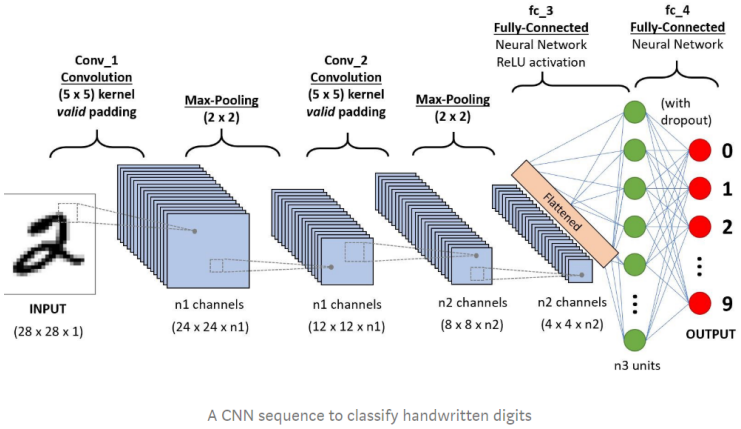

Fig. 1. Example of Convolution Neural Network

In this model I am using One Dimensional CNN (1-D CNN). 1-D CNN helps in providing best solution with help of non-linear regression model. Our model depends on some

978-1-7281-4387-3/19/$31.00 © 2019 IEEE

factors for generating good results. It depends on Batch Size, Learning Rate, Kernel, Number of Layers.

Regression Model: Regression models are used for predicting data by using previous data. It is a process of predicting the values. Example of linear regression model is Estimation of rain fall.

Linear Regression It is on of the regression technique which uses linear function for modeling a data. The linear function models the relationship between dependent value and independent value. There are two categories in linear regression simple linear regression and multiple linear regression. In simpler linear regression only one variable is considered and in multiple linear regression two variables are considered.We can implement linear regression in many applications in reality.

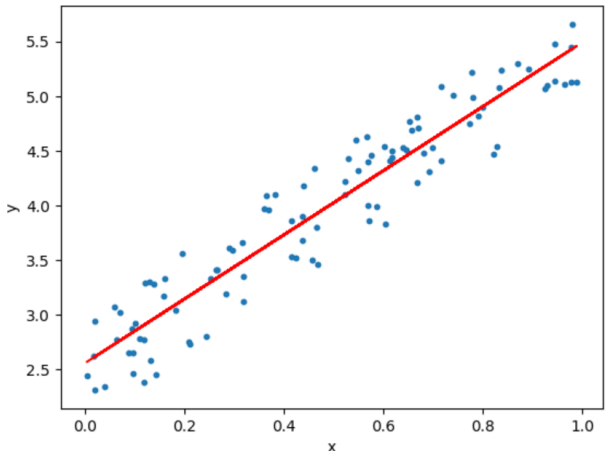

Fig. 2. Example of Linear regression

Non-Linear Regression: Non-linear Regression is one of the regression models which uses non-linear function for modeling a data. The function depends on one or more independent values. Some common models used in non-linear regression are Asymptotic Regression and Growth Model.

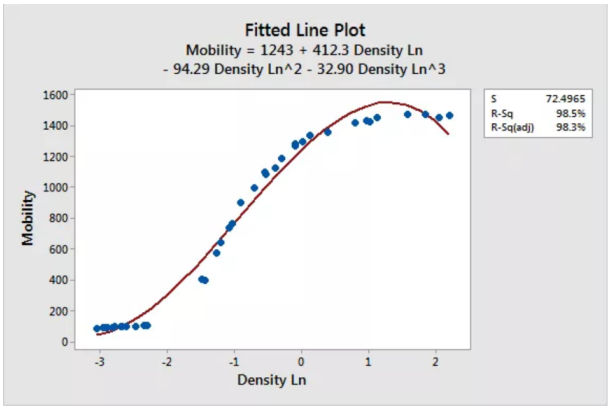

Fig. 3. Example of Non-linear regression

kernel: Kernels are very important in Neural Networks such as convolution neural networks and deep learning. We use kernel to perform statistical analysis. Kernels also called as filters. They perform filters on input. Kernel has two parameters which are height and width. We can customize our kernel size according to our convenience.

Inference Time: It can be stated as amount of time source code taken to run on both CPU and GPU. The inference time is given by formula X+Y. where X is a CPU and Y is a GPU. GPU is better than CPU because it takes less time to run a code compared to CPU.

- Number of hidden layers
- Kernel size
- Activation, pooling functions.

Keras: It is an open source library and it is related to neural network library. It can be treated as one step higher than tensor flow. It is mainly used for deep neural networks.

Literature review: I have choosen the following IEEE paper and completely reviewed and did my analysis part.
Paper title- Bongjin Oh ; Junhyeok Lee. "A case study on scene recognition using an ensemble convolution neural network, IEEE, 2017

The main purpose of the project is to recognize scene images which are based on an grouping of two convolution neural networks. A convolution neural network is used for training huge scene images, and the other convolution neural network is used to extract object details from the image scenes. The lists of the objects are stored according to classes scenes, and used as a reference to decide the top classes during scene image recognition stage.

III. Proposed Model

Main Steps Involved in this Model:

- **Pre-Setup (Importing all the required packages)**
- **Dataset Processing**
- **Network Creation**
- **Training and Testing the model**

Pytroch: Pytroch is an open source machine learning library developed by Facebook. It supports C++, Python and Cuda.

Libraries Uesd in this Model

- **Pandas**-Pandas is used for performing analyzing on values and to manipulate data.
- **Sklearn**-Sklearn supports Regression, classification and clustering algorithms. It also supports SVM (Support Vector Machine).
- **Numpy**-It is used for performing mathemathical calculations on multi dimensional arrays.
- **Matplotlib**-Matplotlib is used for ploting mathemathical values in forms of graphs

- **Torch.nn**- We train the neural network using Torch.nn library.
- **Flatten**-Flatten is mainly used to convert a different size grids into a straight one
- **RELU**-RELU (rectified linear unit layer) acts as activation function makes sure nonlinearity and fully connected layer. The main function of RELU is converting nonlinear model into linear model.
- **SGD**-SGD is a sophisticated Gradient Decent. It is optimizer mainly used for optimizing.

The proposed model is based on implementation of one-dimensional convolution neural network using non-linear regression for prediction. Here I am using Google Colab as IDE. Load the California dataset(housing.CSV) into google colab. Once dataset is loaded, we will create a custom model, from scratch, using 1D convolution layers. Next, we divide the dataset in training and testing. In this model we are dividing dataset into 70 percentage for training and 30 percentage for testing. we will create a custom method to train our new model in batches and let it run for a set number of epochs. At last, we will evaluate our model on the testing dataset to see how it performs. Here I will explain clearly in steps.Google Colab is free to use, and it have many advantages. Colab runs the program in less time and generates the output. The program is stored in drive and we can access it when ever we need. The first step is open the google colab and connect to GPU runtime. Second step is downloading the California housing dataset(housing.csv) and upload it. Third step is import all the libraries which we required. We need a pandas library for read our dataset. The sklearn train-test library is imnported from sklearn for training and testing our model. The pandas library is used for manipulating the data. Next, we read the dataset using pandas read library. We use dropna() for removing incomplete entities. head function is used to print rows of dataset. By specifying the value in function, it prints the value number of rows. We are printing 10 rows so I am specifying head (10). Next we plotting the values using matplotlib libraries. I am using plt.style.use(ggplot) and plt.show() functions for ploting the subplots. We will be predict the median-house-value column and the remaining columns will be used for predicting Y. Finally, we split the dataset into training and testing portions. We are splitting the dataset into 70 percentage for training and 30 percentage for testing. After that we converting the datasets to numpy arrays using .to-numpy ().

First, let us import all the required libraries to build our network. So, import torch, Conv1d, MaxPool1d, Flatten, Linear, relu, DataLoader and TensorDataSet libraries. Next, we define our model. To defining our model, we need to define the initialization method and then initialize the super class and store the parameters. Define the input layer which has the parameters like channels, output channels and kernel size, Define the max pooling layer means defining kernel size, defining one Convolution layers, Define the flatten layer, Define the linear layer and finally, Define the output layer. Define a method which helps us to feed the inputs through the model and reshaping the entry so that it can feed the input layer. Next we need to get the output from each layer and run it through the activation function. Finally, we get the output from output layer.

To perform the training on models. Firstly, we need to import the optimizer and the performance measure packages are using. We importing the SGD package for performing optimisation and importing L1Loss package for measuring the performance. Import the R2 score package also. Next we need to define our model by initializing the batch size and setting up the model to use GPU for processing. I provided batch size as 128. Next we create a method which is used for running the batches of data through our models. model-loss() function returns the average L1 Loss and R2 Score. We will be getting the models predictions for the training dataset, models loss and models R2 Score. Next we need to mention the number of epochs to train our model. I have given 300 epochs. Define the performance measure and optimizer. Next convert the training set into torch variables for our models using GPU as floats. The reshape() is used to remove the warning PyTorch outputs. Create a DataLoader to work instances with our batches and then start the loop. It calculates the average loss and R2Score for every epoch and generates the output. Finally we are taking the avg-loss and R2Score.

Over /under fitting issue: In Over fitting, model performs well in train time but not performs good at testing time.To avoid this situation,we reduce number of Epchos. In Under fitting, model not performs well both in testing and training set.To avoid this issue, we need to change the optimizer.

The accuracy which is good in training time reduces during testing time . While I was using SGD optimizer I got a r2 square value of 0.30 which I didnt find accurate. Under fitting is the concept that occurs when algorithm shows low variance and high bias. In order to solve this problem I used a new optimizer called as RMS Prop where I got a r2square value of 0.67

Displaying First Ten Rows of Dataset Frist we need to load a dataset using pandas read command.Next,we need to drop unnecessary data using dropna command. finally we displaying the dataset using head function.

Next we are going to plot the each feature of the dataset on the subplots using matplotlib.To perform this iam using ggplot and plot.show() function.

Training parameters

The following are the training parameters that are used in Non-Linear Regression Model: $W_c = Numberofweightsofthe Convolutionallayer.$
$K = KernelsizeusedinCNN.$
$N = NumberOfKeras.$
$C = Numberofchannelsoftheinputimage.$

Here are the first five rows of the dataset:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462
5	-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.0368
6	-122.25	37.84	52.0	2535.0	489.0	1094.0	514.0	3.6591
7	-122.25	37.84	52.0	3104.0	687.0	1157.0	647.0	3.1200
8	-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.0804
9	-122.25	37.84	52.0	3549.0	707.0	1551.0	714.0	3.6912

Fig. 4. The first ten rows of data set

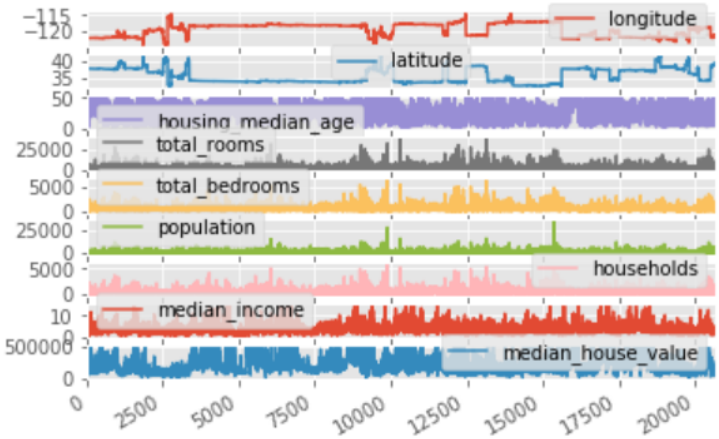

Fig. 5. Plotted graph of each features of data set

```
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,1)
    self.max_pooling_layer=MaxPool1d(1)
    self.conv_layer=Conv1d(batch_size,64,1)
    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.flatten_layer(output)
    output=self.linear_layer(output)
    output=self.output_layer(output)
    return output
```

Fig. 6. defining model and creating network

```
epochs = 300
optimizer=torch.optim.RMSprop(model.parameters(),lr=1e-3)

inputs=torch.from_numpy(x_train_np).cuda().float()
outputs=torch.from_numpy(y_train_np.reshape(y_train_np.shape[0],1)).cuda().float()

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

for epoch in range(epochs):
  avg_loss, avg_r2_score = model_loss(model,loader,train=True, optimizer=optimizer)
  print("Epoch" +str(epoch+1)+ ":\n\tLoss = "+ str(avg_loss)+"\n\tR^2 SCore = " +str(avg_r2_score))
```

Fig. 7. defining model and creating network

Kernal size used : 1
Batch size= 128
Learning rate= 1e-3
Epoch size= 300

IV. Dataset used

The data set is extracted from github and below is the following link where u can download it
https://github.com/ageron/handson-ml/tree/master/datasets/housing

V. Appendix

We defining the number of convolution layers while training the model. First input is went through convolution layer.Next, the output of CNN Layer acts as input to max pooling layer.Next output of max pooling layer passes as input to linear layer. Finally, we get output from output layer.

Below is code for training the model

2.The following code is used to test the data

VI. Conclusion

My model generated best solution using convolution neural networks for given problem. My model minimized loss and generated best R2 score.

The model's L1 loss is: **47500.086269946805**
The model's R2 score is:0.6813309267048147

VII. References

[1] Fatih Ertam, Galip Aydin.**Data classification with deep learning using Tensorflow** IEEE, 2017

[2]A. Krizhevsky, I. Sutskever, G. Hinton**"ImageNet classification with deep convolutional neural networks", Advances in Neural Information Processing Systems, IEE 2012.**

[3]C. Szegedy, W. Liu, Y. Jia et al.**"Going Deeper with Convolutions", IEEE 2014.**

[4]J. Redmon, S. Divvala, R. Girshick.**"A. Farhadi, "You Only Look Once: Unified Real-Time Object Detection",IEEE 2015**