Implement two neural network (with 3 and 5 layers) regression to predict housing prices and discuss accuracy of one over the other.

Name: Pushkar Khetrapal
Department: *Machine learning*
Organization: *AI tech systems*
City, Country: New Delhi, India
Email:
pushkarkhaterpalpm@gmail.com

Introduction to Dataset

This paper is about implementing neural networks regression to predict housing price. The dataset is taken from http://www.kaggle.com/c/house-prices-advanced-regression-techniques/data. I have implement two Neural Networks with different layers, one is with 3 layers and other is with 5 layers. And I compare accuracy with each other. The dataset contain various features like Sale Price, Lot Area, Street, Lot Shape, Land Slope, Neighbourhood, Foundation, Electrical, Kitchen, Garage Type, Year Sold, Sale Type, Sale Condition etc. The data contain some strings values in some features so we need to convert it with One Hot Encoder and Categorical variables using sklearn library. And then, it should be normalize since neural networks have a lot calculations so greater numbers give greater values which is difficult to handle. We need to normalize independent variables as well as dependent variable which price of house.

3-Layers Neural Network

I implemented 3 layers neural network with First layers have 40 nodes or neurons and activation function is reLU, Second layer have 8 nodes or neurons, having activation function reLU and third layer have 1 node or neuron and activation is reLU. Using tensorflow I optimize the cost function. I have run 1500 epochs with mini batches of 73 and I got 0.000033 cost after 1500 epochs.

5-Layers Neural Network

I also implemented 5 layers neural network with First layers have 40 nodes or neurons and activation function is reLU, Second layer have 20 nodes or neurons, having activation function reLU and third layer have 10 nodes or neurons and activation is reLU. Fourth layer have 10 nodes or neurons, having activation function reLU and Fifth layer have 1 nodes or neurons and activation is reLU. Using tensorflow I optimize the cost function. I have run 1500 epochs with mini batches of 73 and I got 1.967093 cost after 1500 epochs.

Equations:

3-Layers Neural Network

X having 302 different features and 1460 total examples and Y having price of 1460 examples.

$$Z1 = W1 * X + b1$$
$$A1 = relu(Z1)$$
$$Z2 = W2 * A1 + b2$$
$$A2 = relu(Z2)$$
$$Z3 = W3 * A2 + b3$$

Cost function:

$$Cost = SUM([Z3 - Y]^{**}2) / (2^*m)$$
#m = 1460 -> total number of examples given

Dimensions:

X: [302, 1460]
W1: [40, 302]
b1: [40, 1]
A1: Z1: [40, 1460]
W2: [20, 40]
b2: [20, 1]
A2:Z2: [20, 1460]
W3: [1, 20]
b3: [1, 1]
Z3: [1, 1460]
Y: [1, 1460]

5-Layers Neural Network

X having 302 different features and 1460 total examples and Y having price of 1460 examples.

$$Z1 = W1 * X + b1$$
$$A1 = relu(Z1)$$
$$Z2 = W2 * A1 + b2$$
$$A2 = relu(Z2)$$
$$Z3 = W3 * A2 + b3$$
$$A3 = relu(Z3)$$
$$Z4 = W4 * A3 + b4$$
$$A4 = relu(Z4)$$
$$Z5 = W5 * A4 + b5$$

Cost function:

Cost = SUM([Z3 - Y]**2) / (2*m)
#m = 1460 -> total number of examples given

Dimensions:

X: [302, 1460]
W1: [40, 302]
b1: [40, 1]
A1: Z1: [40, 1460]
W2: [20, 40]
b2: [20, 1]
A2:Z2: [20, 1460]
W3: [20, 20]
b3: [20, 1]
Z3: [20, 1460]
W4: [10, 20]
B4: [10, 1]
A4:Z4: [10, 1460]
W5: [1, 10]
B5: [1, 1]
Z5: [1, 1460]
Y: [1, 1460]

Algorithm:

I use tensorflow to implement neural network.

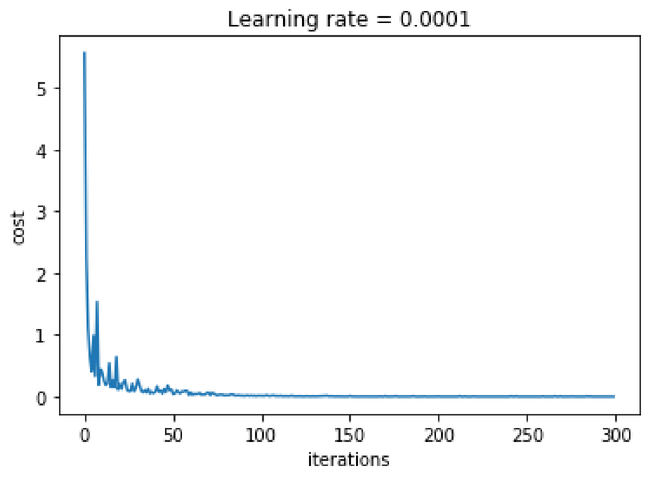
1. Import libraries
2. Import Dataset
3. Refine dataset
   - Removing missing values in dataset
   - Separating string features and numeric features
   - Convert string features into one hot encoder then implement categorical form
   - Normalize the numeric data
   - Concatenate categorical matrix and normalize matrix
   - Normalize the dependent variable i.e., price
4. Import tensorflow
5. Create placeholder of X and Y
6. Initialize parameters
7. Forward propagation
8. Compute cost
9. Initialize optimizer with learning rate = 0.0001
10. Initialize global variable
11. Run the Session
12. For epoch in (1500)
    - Divide dataset in mini batches
    - Run cost and optimizer for very mini batch and add all the cost for every mini batch
13. Retrieve parameters
14. Return parameters
15. End

Outputs:

3-Layers Neural Network

Cost after 100 epochs:

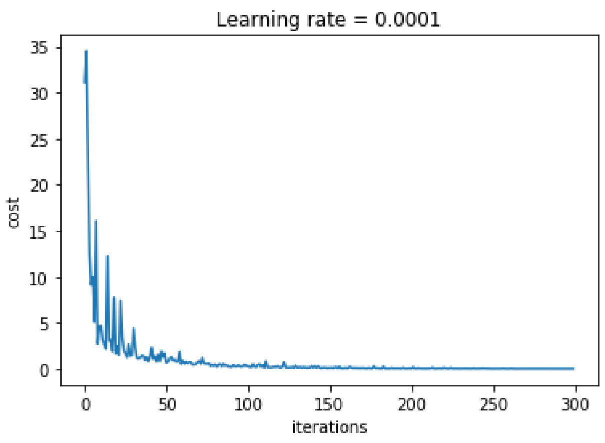
```
Cost after epoch 0: 4.328514
Cost after epoch 100: 0.152285
Cost after epoch 200: 0.048543
Cost after epoch 300: 0.033258
Cost after epoch 400: 0.004840
Cost after epoch 500: 0.002683
Cost after epoch 600: 0.000371
Cost after epoch 700: 0.000148
Cost after epoch 800: 0.000131
Cost after epoch 900: 0.000775
Cost after epoch 1000: 0.000239
Cost after epoch 1100: 0.000038
Cost after epoch 1200: 0.000328
Cost after epoch 1300: 0.000075
Cost after epoch 1400: 0.000047
```



5-Layers Neural Network

Cost after 100 epochs:

```
Cost after epoch 0: 2.870630
Cost after epoch 100: 0.140820
Cost after epoch 200: 0.086662
Cost after epoch 300: 0.036561
Cost after epoch 400: 0.015885
Cost after epoch 500: 0.005783
Cost after epoch 600: 0.003432
Cost after epoch 700: 0.001265
Cost after epoch 800: 0.000609
Cost after epoch 900: 0.000331
Cost after epoch 1000: 0.001646
Cost after epoch 1100: 0.000302
Cost after epoch 1200: 0.000229
Cost after epoch 1300: 0.000279
Cost after epoch 1400: 0.000103
```



Conclusion:

3-Layers Neural Network having **0.9489283958083405** root mean square error.

5-Layers Neural Network having **0.9515096284743161** root mean square error.

Which indicates that both Neural Networks work fine with least error and 3-Layers Neural Network works slightly better.

ACKNOWLEDGMENT

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to our project manager, Mr. Paramkram Raj Bhardwaj and Mr. Rohit Sharma whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project.

REFERENCES

[1] Kaggle dataset of advance housing price. https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

[2] AITS.