Regression based Neural Network for House Price Prediction

Sandeep Mehta

415MA5025

National Institute of Technology, Rourkela

under the supervision of

Prof. B.K. Ojha and Prof. S. Chakraverty

Professor

National Institute of Technology, Rourkela

Abstract

.

This paper introduces a new algorithm for the neural network training in order to have efficient learning and training of the network. In this project report, we are studying about the various methods that can be used for house price prediction. We'll be discussing how different regression machine learning methods can be used in the improvement of the neural network and how it helps in making the predictions better.

Contents

1 Introduction

In recent past, artificial neural networks have been used widely in various fields of engineering. In this report, we have taken a California housing dataset, which consist of total six features, i.e., TotalRooms, TotalBedrooms, Population, Households, Median-income, MedianHouseValue. Our main aim to create a model such that it will predict the MedianHouseValue using the other five features as input feature. Several methods such as the keras(python library) neural network, Support vector machines, linear regression, etc. have been proposed earlier to predict the MedianHouseValue. In this report, we aim to initialize the weights using regression based approach and then use those weights for training the neural network model. In this report, we'll be discussing several methods to generate the weights.

2 Preliminaries

Now we will discuss some basic definitions, which are given below.

- **Artificial Neural Network :** Artificial neural network (ANN) is a powerful data modeling tool, which captures and represents complex input/output relationships. The motivation for the development of ANN technology stemmed from the desire to develop an artificial system that could perform "Intelligent" tasks similar to those performed by the human brain. ANNs resemble the human brain as these acquire knowledge through learning and this knowledge is stored within inter-neuron connection strengths known as weights. The advantage and power of ANNs lies in their ability to represent both linear and non-linear relationships and their ability to learn these relationships directly from the data being modeled. ANN models possess some basic properties such as, these models make use of the given set of training patterns, and artificial neural networks can acquire knowledge about the system by learning about its internal parameters. Training patterns are generally described in terms of set of input and output values for the system that is to be identified. We'll be using the idea described in this [1], and using the concepts defined in the above paper for weight initialization.

- **Optimizers :** Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses. Optimization algorithms[2] or strategies are responsible for reducing the losses and to provide the most accurate results possible. Some various types of optimizers are:
  - **Gradient Descent Optimizer** Gradient descent is a first-order optimization algorithm which is dependent on the first order derivative of a loss function. It calculates that which way the weights should be altered so that the function can reach a minima. Through backpropagation, the loss is transferred from one layer to another and the model's parameters also known as weights are modified depending on the losses so that the loss can be minimized.
  - **Stochastic Gradient Descent Optimizer** It's a variant of Gradient Descent. It tries to update the model's parameters more frequently. In this, the model parameters are altered after computation of loss on each training example. So, if the dataset contains 1000 rows SGD will update the model parameters 1000 times in one cycle of dataset instead of one time as in

Gradient Descent.

- **Adam Optimizer:** Adam (Adaptive Moment Estimation)[3] works with momentums of first and second order. The intuition behind the Adam is that we don't want to roll so fast just because we can jump over the minimum, we want to decrease the velocity a little bit for a careful search. In addition to storing an exponentially decaying average of past squared gradients like AdaDelta, Adam also keeps an exponentially decaying average of past gradients

- **Ada Grad Optimizer** One of the disadvantages of all the optimizers explained is that the learning rate is constant for all parameters and for each cycle. This optimizer changes the learning rate. It changes the learning rate η for each parameter and at every time step 't'. It's a type second order optimization algorithm. It works on the derivative of an error function.

• **Scaling:** Data scaling or normalization is a process of making model data in a standard format so that the training is improved, accurate, and faster. The method of scaling data in neural networks is similar to data normalization in any machine learning problem. The two scaler functions that we are going to use in our project are:

- **MinMaxScaler**: [4] Transforms features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.

  The transformation is given by:

$$Xstd = \frac{(X - X.min(axis = 0))}{(X.max(axis = 0) - X.min(axis = 0))}$$

$$Xscaled = Xstd * (max - min) + min$$

where min, max = feature range.

The transformation is calculated as:

$$Xscaled = scale * X + min - X.min(axis = 0) * scale$$

where

$$scale = \frac{(max - min)}{(X.max(axis = 0) - X.min(axis = 0))}$$

– **StandardScaler**: Standardize features by removing the mean and scaling to unit variance

The standard score of a sample x is calculated as:

$$z = \frac{(x - u)}{s}$$

where u is the mean of the training samples or zero if with mean=False, and s is the standard deviation of the training samples or one if with std=False. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using the transform method.

3 ANN model

A three-layer architecture for ANN is considered here to understand the proposed model of the present problem. Below figure shows the neural network used in the process. The

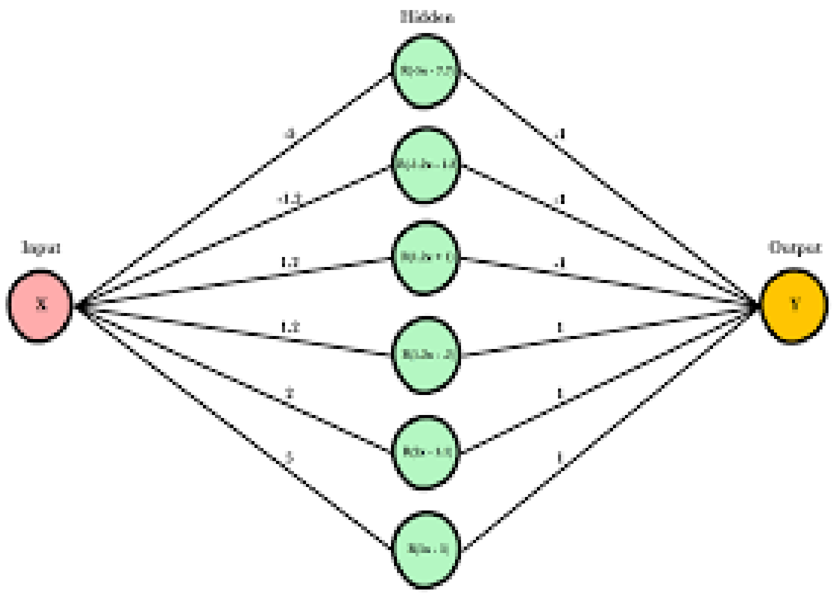

Figure 1: Single input and single output neural network

input layer consists of single input as MedianHouseIncome and the output layer consists of one output in the form of the MedianHouseValue. Various cases of the number of nodes depending upon the proposed parameter of the methodology have been considered in the hidden layer to facilitate a comparative study on the architecture of the network. We can also use multiple input and single output. But initially our prime focus is on using single feature and then going for multiple input feature for output prediction.

4 Dataset Description

We are using California housing dataset, which consist of total six features, i.e., Total-Rooms, TotalBedrooms, Population, Households, Medianincome, MedianHouseValue. Our dataset consist of total 20,460 datapoints. We are dividing the dataset in two parts i.e., training and the testing data. Then we are using the training dataset to create the model and train it, after it's completion we are using the testing dataset for testing purpose. We are using Standard scaler for scaling the dataset, as we have different features whose and the values are not on the same scale, so to avoid the error and make the process easier and efficient. We are scaling on both side i.e., input features and output features, and then using inverse transform to get back the output in its original form.

5 Training and Testing

We are using Adam optimizer for training the dataset, it's basically an optimizing technique using back-propagation for updating the weights and the learning rate for this optimization has been set to 0.001. we are using mean squared loss function for calculating the error and using this error for convering the weights which will help us in getting the desired results. we have used Rectified Unit linear function as our activation function in the model. We are using 40 epochs in all the methods where we have used neural network.

6 Proposed Method

For programming purpose we are using python language and pycharm software. We'll be using keras library[5] which is equipped with tensorflow [6] in the backend, for creating our neural network model. We have tried different numerical method[7] approaches for generating polynomial which can help us in weight initialization to create the model, and then use this model for testing on the dataset and then we are comparing the result using the r2-score, which shows us the goodness of our results.

6.1 Using Support Vector Machines

Here we have used support vector machines to create the model. We are basically using three different kernels and creating three models. Our first kernel is linear, second one is RBF (radial basis function) and the third kernel is polynomial. We are obtaining the results using all the three kernels separately.

6.2 Using Least Square Fitting

Here we have least square fitting method, we are fitting a multivariate equation on the training dataset and then predicting the results using this equation.

6.3 Using Neural Network with automatic weight generation

In this section, we have used neural network consisting of some hidden layers and hidden nodes. Here we have used the weights that are generated by the keras toolbox, and then used those weights for training the model and then used this model for testing.

6.4 Using regression polynomial and neural network

In this method we have used only one feature which is most significant in the output prediction. Here we have used MedianHouseIncome for predicting the MedianHouse-Value. We are first fitting a quadratic polynomial on the dataset, and then used the coefficients in the polynomial for setting the weights of the neural network. We have used quadratic, cubic, biquadratic polynomials for weights initailization.

6.5 Using Cubic Spline polynomial

In this method, we are using cublic spline polynomial to for weights initialization, here we are generating a cublic spline polynomial between each two points and then using those polynomials coefficients for weight initialization.

7 Results and discussion

The r2 score for all the above methods are provided below in the table:

Dataset	SVR(Linear)	SVR(RBF)	SVR(Polynomial)	Least Square	Neural Network
Test	0.618	0.662	0.667	0.591	0.747
Train	0.541	0.603	0.618	0.593	0.698

The graphs obtained after creating the model[8] and then testing it on the test dataset and then plotting the desired values against the obtained values are provided below:

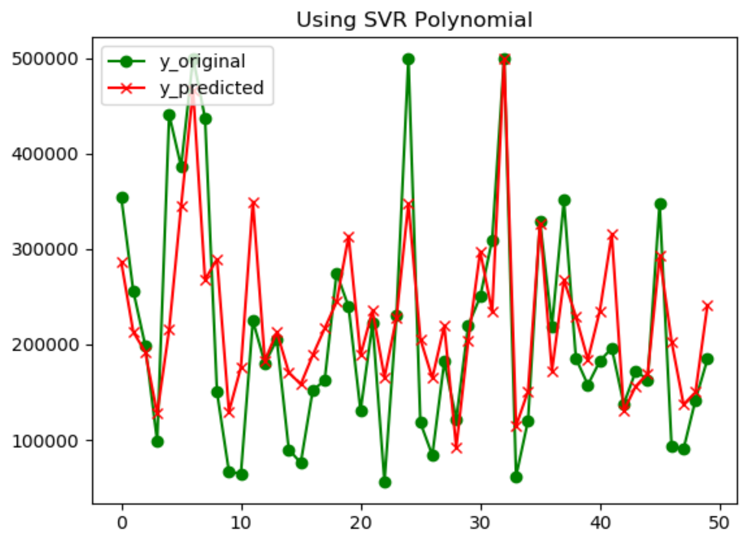

Figure 2: Comparing the predicted output and desired output for test datapoints.

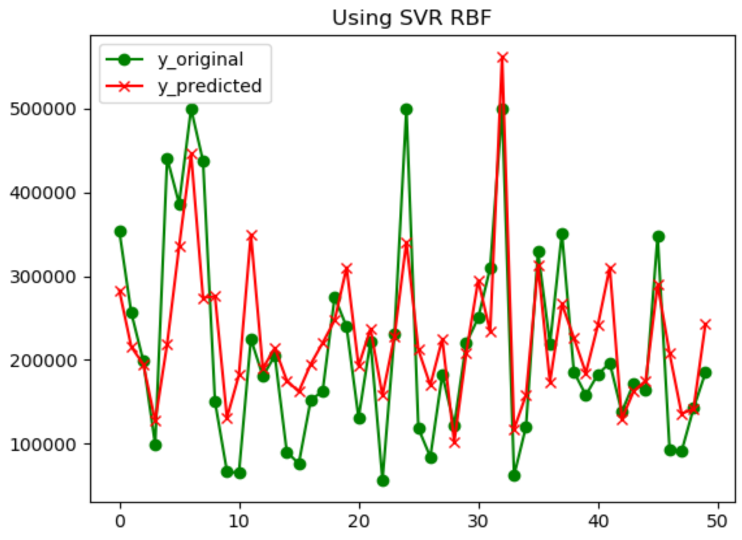

Figure 3: Comparing the predicted output and desired output for test datapoints.

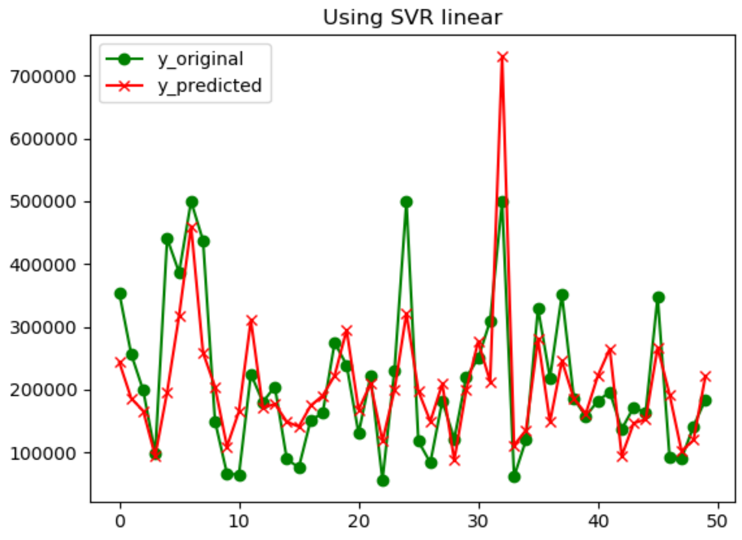

Figure 4: Comparing the predicted output and desired output for test datapoints.

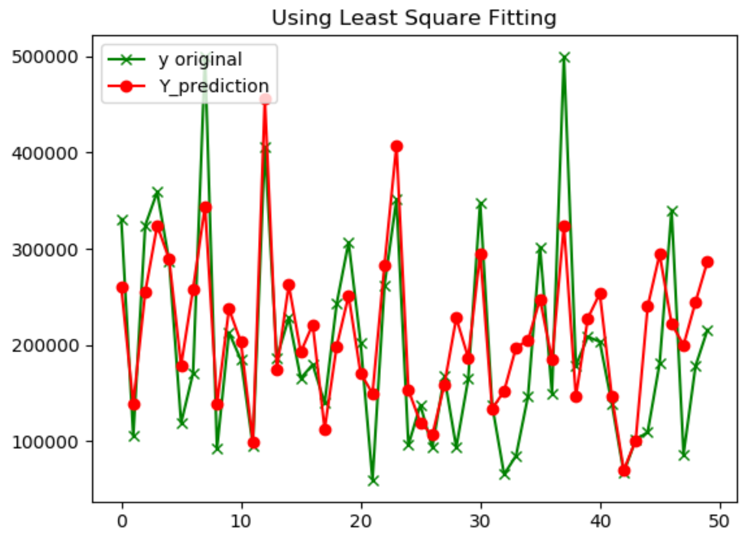

Figure 5: Comparing the predicted output and desired output for test datapoints.

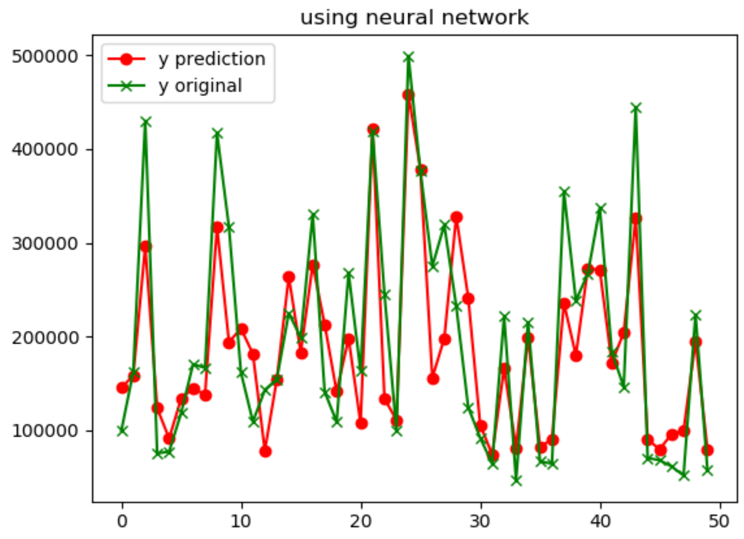

Figure 6: Comparing the predicted output and desired output for test datapoints.

The results for output prediction using single feature input i.e., while using regression for weight initialization are provided below. We have used two methods, the first one is fitting a quadratic polynomial on the training dataset and then use those coefficients for weights initialization and the other one is fitting a cubic polynomial for fitting and then using the coefficient for weight initialization.

Dataset	Random Weight Initialization	Quadratic Polynomial	Cubic Polynomial
Test	0.653	0.663	0.485
Train	0.487	0.492	0.489

The graphs for the above comparisons are provided below;

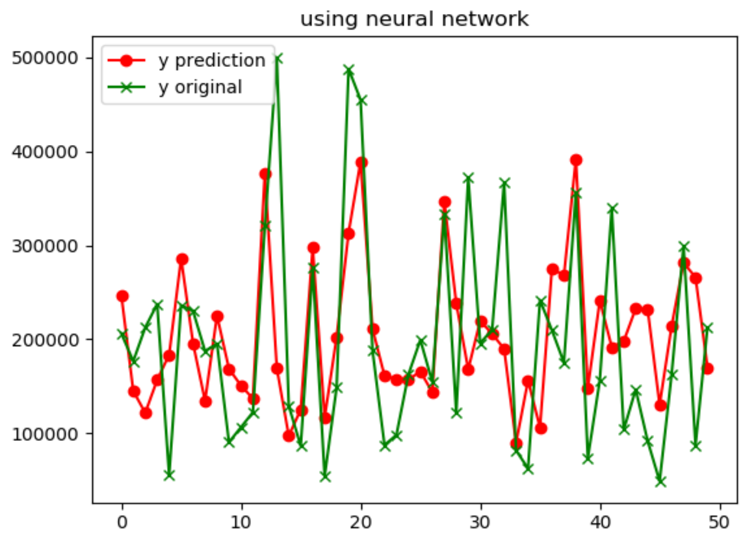

Figure 7: Using the default neural network for predictions.

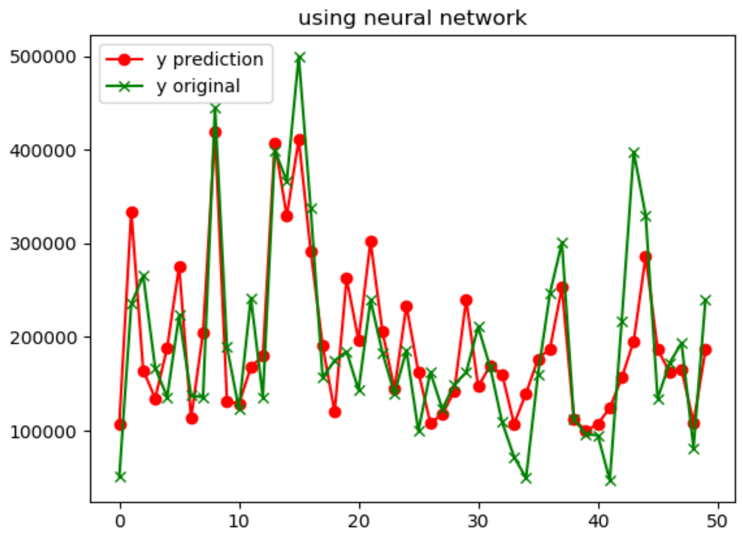

Figure 8: Using quadratic polynomial for regression.

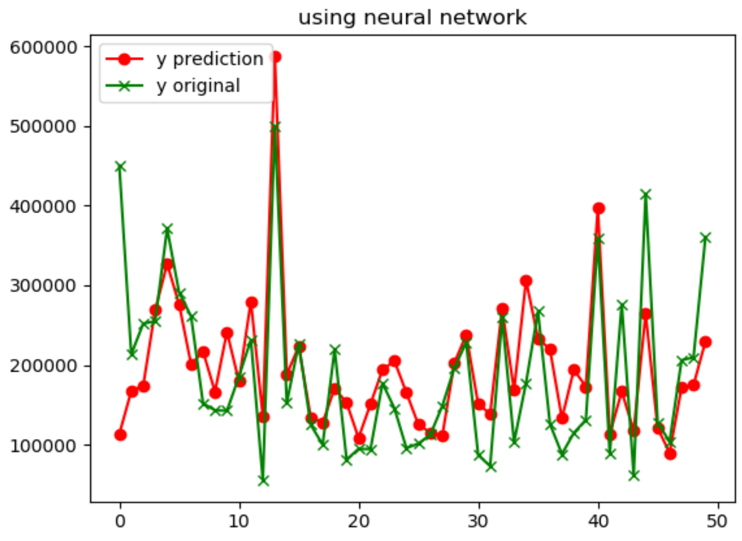

Figure 9: Using the third degree polynomial for regression.

8 Conclusion

We are still looking for some other ways which can help us in the improvement of the result. It is visible from the results that the predictions made by using neural network are better than other methods i.e,, SVR, Linear Regression. In future, we'll be working with other types of polynomial which can help us generate good coefficients for weight initialization.

References

[1] S. Chakraverty, V. Singh, and R. Sharma, "Regression based weight generation algorithm in neural network for estimation of frequencies of vibrating plates," *Computer methods in applied mechanics and engineering*, vol. 195, no. 33-36, pp. 4194–4202, 2006.

[2] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* " O'Reilly Media, Inc.", 2017.

[3] A. Tato and R. Nkambou, "Improving adam optimizer," 2018.

[4] R. D. Martin, V. J. Yohai, and R. H. Zamar, "Min-max bias robust regression," *The Annals of Statistics*, pp. 1608–1630, 1989.

[5] F. Chollet *et al.*, "Keras." https://keras.io, 2015.

[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga,

S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[7] E. Isaacson and H. B. Keller, *Analysis of numerical methods.* Courier Corporation, 2012.

[8] M. N. Mitchell *et al.*, *Interpreting and visualizing regression models using Stata*, vol. 5. Stata Press College Station, TX, 2012.