Predicting houses price by deep learning neural network

Zidong Xu

Stony brook institute of Anhui university, HeFei, China, 230039

xu-zidong.co@foxmail.com

Abstract. Deep learning is widely used in various fields, the article proposed a deep learning method to predict house prices through different characteristics of real estate, establish a prediction model, and carry out simulation experiments. First, extracting data from property transactions records, it is difficult to directly input the raw-data into the deep learning model, and there may be overfitting in the model training, so data will be pretreated. Second, Fully-Connected Neural Network is used to model different features' influences to price. The sample data will be randomly divided into a training set and a validation set, of which 70% of the samples are used for building and training the model, and the remaining 30% are used for model accuracy verification. Experimental results show that the model can achieve a high accuracy in predicting houses price. The model can be used as a reference for the evaluation of housing prices.

Keywords: machine learning, artificial neural network, predicting, weight values, back propagation neural network.

1. Introduction

There are lots of achievement on deep learning models like playing Go [1] or imitate human brain [2]. In Go, AI, which is supported by deep learning algorithms to make decisions, has completely surpassed human chess players. AI players represented by AlphaGo have successfully defeated the world champion in the competition [3]. This means that the accuracy of AI decision-making is beyond doubt after training with neural network. This also makes us hope that AI can make more accurate judgments than human decision-makers in other fields, so as to free people from some heavy labor that needs human decision-making in the past. In 1986, Rumelhart proposed a back-propagation algorithm to train artificial neural network automatically.[4] which also be used in this work. Back propagation algorithm, referred to as BP algorithm, is a learning algorithm suitable for multi-layer neural networks. It is based on gradient descent method. The input-output relationship of BP network is essentially a mapping relationship: the function of a BP neural network with n inputs and m outputs is the continuous mapping from n-dimensional Euclidean space to a finite field in m-dimensional Euclidean space, which is highly nonlinear. Its information processing ability comes from the multiple composition of simple nonlinear functions, so it has a strong function reproduction ability [5]. A fully-Connected Neural Network has been used to predict houses price. Input the predicted parameters at the input level, that is, the specific conditions of the property, such as size, construction time, etc. A predicted house price will be given by the model. The traditional method of building a function to predict house prices needs to use a lot of statistical knowledge, and the effect is significant when the sample size is large. It is difficult to give significant and relevant results in a small sample. Comparing to traditional machine learning which calculated by a custom rule function, neural network does well in handling unstructured data [6]. Also, research shows that Artificial Neural Network do better in capturing different features of data [7]. In order to solve the problem that the model operation is too slow in the case of large amounts of data, and to improve the convergence performance of the model, mini-batch has been used to reduce the randomness and calculation amount. Mini batch divides the data into several batches, and updates the parameters by batch. The batch size is the data amount of each batch. In this way, a group of data in a batch jointly determines the direction of the gradient, making it difficult to deviate when descending. On the other hand, because the number of samples in the batch is much smaller than the whole dataset, the calculation is not very large. Professor Mu Li from Carnegie Mellon University, Pittsburgh, PA, USA shows that mini batch can effectively optimize the random gradient descent algorithm [8].

2. Method and tools

2.1. The structure of neural network

The structure of artificial neural network model is illustrated in figure 1. The system is constructed with back-propagation neural network with four layers. The relevant labeled information which includes eleven features corresponding to target house price is used to train the model.

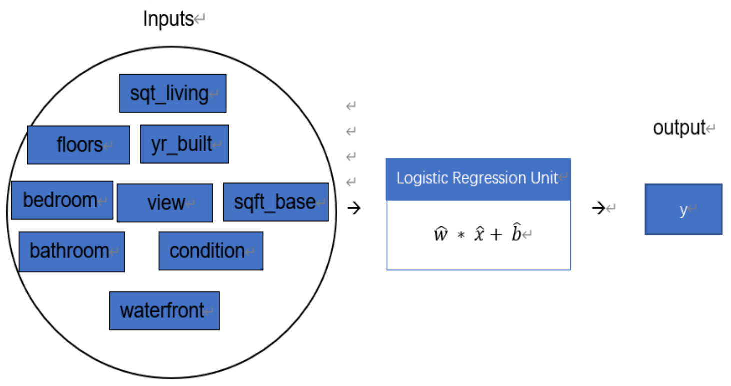

Figure 1. Schematic diagram of model principle (credit: original).

The system learns by labeled information firstly gives out prediction by inputting unlabeled information. Logistic regression has been used in this model. Vector X is the input while vector y is the output. First three layers uses linear regression to model neural network as given below:

$$\hat{y} = \hat{w} * \hat{x} + \hat{b} \tag{1}$$

Each feature has different weigh parameter ˆw in this neural network model. After three layers linear regression, a linear rectification function was used as an activation function which shown as below:

$$F(x) = Max(0, wTx + b) \tag{2}$$

The whole dataset is from Kaggle and was compiled by Shree. The data comes from the real estate market. It describes almost all aspects of Melbourne's housing with eleven variables. The data set contains more than 4600 transaction records, enabling the model to predict the final property price after training. The dataset is divided into two sets: training and testing with following codes:

```
Xtrain, Xtest, Ytrain, Ytest = train_test_split(x,y,test_size=0.3)
Xtest = torch.from_numpy(Xtest)
Ytest = torch.from_numpy(Ytest)
```

Then prepare dataset:

```
class DiabetesDataset(Dataset):
    def __init__(self, data,label):
        self.len = data.shape[0]
        self.x_data = torch.from_numpy(data)
        self.y_data = torch.from_numpy(label)

    def __getitem__(self, index):
        return self.x_data[index], self.y_data[index]

    def __len__(self):
        return self.len
```

To date, there is not a standard way to choose the best number of layers in the neural network of a small amount of data fitting, only some rules of thumb can be referred to. For this reason, a typical method is to try different numbers and see which performs better [9]. The more layers, the more inclined the model is to completely simulate the data, including the noise in the data which will reduce the accuracy of the model. S. Tamura, a researcher from Research Laboratories, Nippondenso Company Limited, Aichi, Japan, claimed that a four-layered feedforward network is superior to a three-layered feedforward network in terms of the number of parameters needed for the training data [10]. In view of the article, After many tries, a four layers neural network has been chosen. And to prevent overfitting, the batch size was set as 32, and layer number was set as 4. The model was built with pytorch framework.

```
#mini-batch set
train_dataset = DiabetesDataset(Xtrain,Ytrain)
train_loader = DataLoader(dataset=train_dataset, batch_size=32, shuffle=True, num_workers=1)
```

2.2. *Training and predicting method*

During training, the neural network can assign different weight to different features automatically based on back-propagation algorithm. In pytorch framework, the model used MSEloss function to build loss function and Adaptive Moment estimation to build optimizer. The function of the loss function is to calculate the difference between the forward calculation result of each iteration of the neural network and the true value, so as to guide the next training in the correct direction. In each epoch, the system gives out ˆy prediction, then calculates the loss by given formula:

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2 \tag{3}$$

When loss has been given out, in pytorch framework, backward() function will be called to realize the derivation operation of back propagation.

```
loss = criterion(y_pred, labels)
loss.backward()
```

In this model, the loss function provided by the pytorch framework does not represent the loss result well, a new loss function is reconstructed. Then every epoch train procedure optimized as follows:

Initialize each parameter to 0:

$\mathrm{vdW} = 0;;$

$$\mathrm{sdW} = 0;$$
$$\mathrm{vdb} = 0;$$
$$\mathrm{sdb} = 0;$$

Calculate the gradient db and dW on the current Mini batch:

$$\mathrm{vdW} = \beta1 * \mathrm{VdW} + (1-\beta1) * \mathrm{dW} \tag{4}$$

$$\mathrm{vdb} = \beta1 * \mathrm{Vdb} + (1-\beta1) * \mathrm{db} \tag{5}$$

$$\mathrm{sdW} = \beta1 * \mathrm{sdW} + (1-\beta1) * (\mathrm{dW})^2 \tag{6}$$

$$\mathrm{sdb} = \beta1 * \mathrm{sdb} + (1-\beta1) * (\mathrm{db})^2 \tag{7}$$

Deviation correction of exponential weighted average:

$$v_{dw}^{corrected} = \frac{v_{dW}}{1-\beta_1^t} \tag{8}$$

$$v_{db}^{corrected} = \frac{v_{db}}{1-\beta_1^t} \tag{9}$$

$$s_{dw}^{corrected} = \frac{s_{dW}}{1-\beta_2^t} \tag{10}$$

$$s_{db}^{corrected} = \frac{s_{db}}{1-\beta_2^t} \tag{11}$$

In this way, loss value vs epoch shown as figure 2:

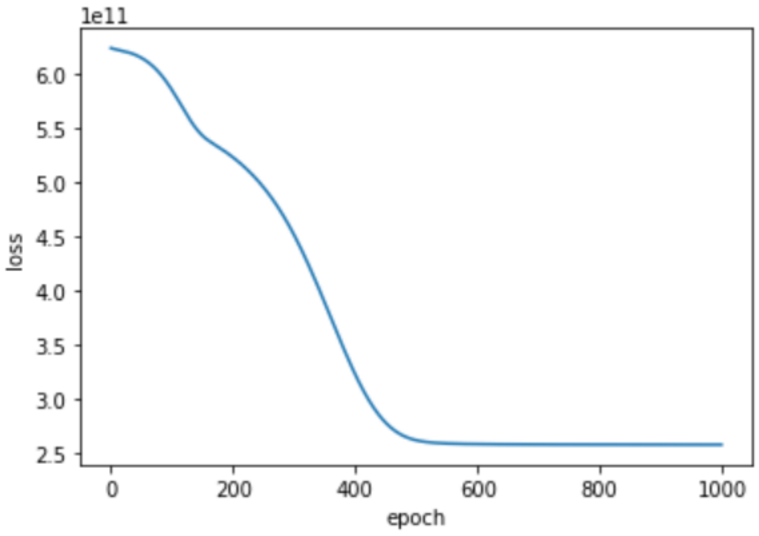

Figure 2. Result of loss (credit: original).

The result shows that the difference between the forward calculation result and the label converges to about 2.5.

3. Results

Houses price can be predicted by inputting parameters like living space. When relevant parameters are input to neural network, at the same time, the house price is input to the output layer as the supervision signal. To evaluate accuracy of the model, the model used a linear regression accuracy evaluate method:

```
def r2_loss(output, target):
    target_mean = torch.mean(target)
    ss_tot = torch.sum((target - target_mean) ** 2)
    ss_res = torch.sum((target - output) ** 2)
    r2 = 1 - ss_res / ss_tot
    return r2
```

the accuracy of the model vs epoch number is shown as figure 3:

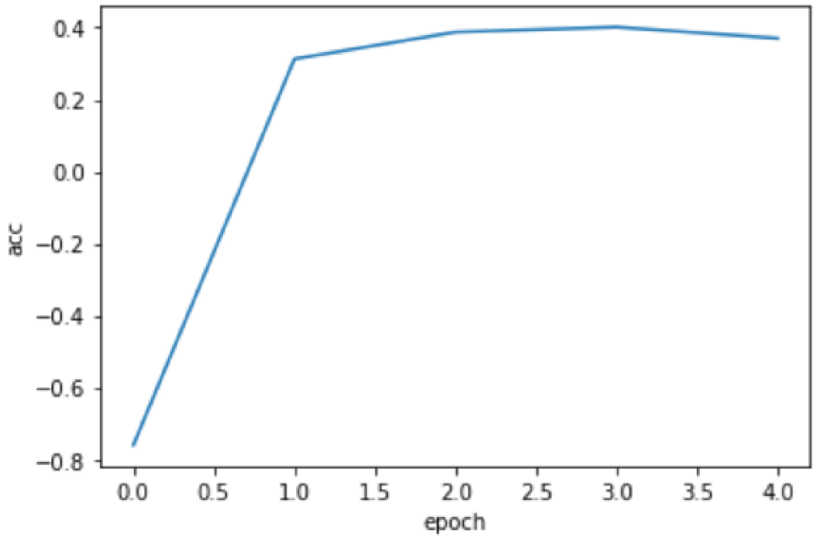

Figure 3. Result of accuracy (credit: original).

According to the accuracy of the model, the prediction of the system is reliable.

4. Comparison with traditional function model

Using R language, the functions related to eleven features are successfully established. The function model with the highest degree of fitting is finally found by continuously reducing the aic value as follows:

$$y\sim x1 + x2 + x3 + x4 + x23 + x22 + x32 + x5 + x6 + x7 + x8 + x9 + x11 + x1{:}x2 + x1{:}x3 + x2{:}x3 + x1{:}x4 + x2{:}x4 + x3{:}x4 + x1{:}x2{:}x3 + x1{:}x2{:}x4 + x1{:}x3{:}x4$$

In R, summary() has been used to evaluate fit:

$ly = lm(y\sim x1 * x2 * x3 + x1 * x2 * x4 + x1 * x3 * x4 + x2 * x3 * x4 + x13 + x23 + x12 + x22 + x32 + x1 * x2 + x1 * x3 + x1 * x4 + x2 * x3 + x2 * x4 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11)$# Fit all variables into the linear model
summary(ly)# View model fit
ly1=step(ly)# Exclude some variables based on AIC value
$ly2 = lm(y\sim x1 + x2 + x3 + x4 + x23 + x22 + x32 + x5 + x6 + x7 + x8 + x9 + x11 + x1{:}x2 + x1{:}x3 + x2{:}x3 + x1{:}x4 + x2{:}x4 + x3{:}x4 + x1{:}x2{:}x3 + x1{:}x2{:}x4 + x1{:}x3{:}x4)$# Refit the model after excluding variables
summary(ly2)

Table 1. Results.

Parameters.	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	4.961e+06	6.457e+05	7.683	1.89e-14
x2	1.332e+05	1.051e+05	1.267	0.20538
x3	-1.375e+01	7.500e+01	-0.183	0.85454
x4	-1.901e-01	2.192e+00	-0.087	0.93091
x23	-1.361e+04	5.798e+03	-2.347	0.01897
x22	8.341e+04	4.638e+04	1.798	0.07217
x32	6.059e-02	9.990e-03	6.065	1.43e-09
x5	2.945e+04	1.922e+04	1.532	0.12548
x6	2.414e+05	9.428e+04	2.560	0.01049
x7	4.368e+04	1.092e+04	4.000	6.42e-05
x8	2.973e+04	1.204e+04	2.469	0.01357

Table 1. (continued).

x9	3.903e+01	2.188e+01	1.784	0.07446
x11	-2.559e+03	3.294e+02	-7.767	9.85e-15
x1:x2	-4.354e+04	2.482e+04	-1.754	0.07944
x1:x3	-1.091e+01	2.057e+01	-0.530	0.59581
x3	-5.699e+01	2.933e+01	-1.943	0.05205
x1:x4	1.821e-01	6.556e-01	0.278	0.78119
x2:x4	-2.387e+00	1.806e+00	-1.322	0.18633
x3:x4	2.111e-03	1.184e-03	1.784	0.07456
x1:x2:x3	9.722e+00	6.681e+00	1.455	0.14566
x1:x2:x4	7.502e-01	4.561e-01	1.645	0.10005
x1:x3:x4	-7.730e-04	2.655e-04	-2.911	0.00362

As shown in the above data, only seven parameters passed the significance test, which indicates that most of the parameters in the model do not help the model to make correct predictions. At the same time, by estimating the overall model, we can get the results as shown below:

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 494000 on 4577 degrees of freedom

Multiple R-squared: 0.2359, Adjusted R-squared: 0.2322

F-statistic: 64.24 on 22 and 4577 DF, p-value: < 2.2e-16

According to the statistical results of f, the model is significant as a whole, but the fitting degree of the model to the data is not high, which is nearly twice as bad as the accuracy of the neural network model.

5. Conclusions

In the article, a four-layer artificial neural network is designed and used to predict housing prices. Based on the back propagation algorithm, MSE loss algorithm and adaptive moment estimation, the results of this model are more significant than those of the traditional function building methods. The statistical prediction difference is acceptable and smaller than the model established by traditional functional methods. At the same time, the model has reliable accuracy. When forecasting house prices, the results are close to the target data. However, due to the small data set, the simulation results of the model on housing prices in other regions are not satisfactory. Due to different situations in different regions, it may be difficult to obtain the prediction results suitable for all regions with a single model. It may be a feasible method to establish different models for different regions. In the future, better loss function can be found to fit the model. The model can not only be used to predict housing prices, but also be competent for almost all previous situations predicted by functions as long as it is trained by a small data set. This method saves the tedious process of establishing functions and improves the accuracy of prediction.

References

[1] Silver, David, et al. "Mastering the Game of Go with Deep Neural Networks and Tree Search." Nature, vol. 529, no. 7587, 2016, pp. 484–489., https://doi.org/10.1038/nature16961.

[2] Song, Sen, et al. "Competitive Hebbian Learning through Spike-Timing-Dependent Synaptic Plasticity." Nature Neuroscience, vol. 3, no. 9, Sept. 2000, pp. 919–926., https://doi.org/10.1038/78829.

[3] Silver, David, et al. "Mastering the Game of Go without Human Knowledge." Nature, vol. 550, no. 7676, 2017, pp. 354–359., https://doi.org/10.1038/nature24270.

[4] Rumelhart, David E., et al. "Learning Representations by Back-Propagating Errors." Nature, vol. 323, no. 6088, 1986, pp. 533–536., https://doi.org/10.1038/323533a0.

[5] Goodfellow, Ian, et al. Deep Learning. MITP, 2018.

[6] Arora, Monika, and Vineet Kansal. “Character Level Embedding with Deep Convolutional Neural Network for Text Normalization of Unstructured Data for Twitter Sentiment Analysis.” Social Network Analysis and Mining, vol. 9, no. 1, 2019, https://doi.org/10.1007/s13278-019-0557-y.

[7] Zhong, Botao, et al. “Convolutional Neural Network: Deep Learning-Based Classification of Building Quality Problems.” Advanced Engineering Informatics, vol. 40, 2019, pp. 46–57., https://doi.org/10.1016/j.aei.2019.02.009.

[8] Li, Mu, et al. “Efficient Mini-Batch Training for Stochastic Optimization.” Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, https://doi.org/10.1145/2623330.2623612.

[9] Olson, Matthew, et al. “Modern Neural Networks Generalize on Small Data Sets.” Advances in Neural Information Processing Systems, 1 Jan. 1970, https://papers.nips.cc/paper/7620-modern-neural-networks-generalize-on-small-data-sets.

[10] Tamura, S., and M. Tateishi. “Capabilities of a Four-Layered Feedforward Neural Network: Four Layers versus Three.” IEEE Transactions on Neural Networks, vol. 8, no. 2, Mar. 1997, pp. 251–255., https://doi.org/10.1109/72.557662.