**Predicting Customers' Behavior in E-commerce using Multilayer Perceptron**

Pei Zhao, Ping Li

**Abstract**⎯ **Predicting the occurrence of commodity purchase is helpful for electricity supplier to achieve a good product recommendation[6]. This paper presents the application of multilayer perceptron for predicting users' purchase behaviors on e-commerce websites. The model is built by learning from user's history data and behaviors at present. The data we use is a four months of data provided by Alibaba, the largest online shopping marketplace in China. Compared to classical three-layer multilayer perceptron model, the model we use here is a deep network with more hidden layers. The performance of our multilayer model is verified to be better than traditional methods do, especially in five-layered multilayer model.**

[[1]](#footnote-2)

**Index Terms⎯ multilayer perceptron, predicting, e-commerce, recommendation.**

# Introduction

In recent years, e-commerce has been rapidly developed, attracting more and more people to participate in it. In order to attract consumers to purchase goods, e-commerce providers and scholars have made various attempts[1],[3], one of which is recommendation[7],[8], a good recommendation make it possible for a person to buy the product. It is a problem of great concern for many electricity providers that how to recommend commodities to users can lead to the purchase[4]. Generally, many machine learning algorithm are used for this purpose, such as logistic regression, collaborative filtering. Certainly, most of them are based on the assumption: there is some relationship between whether to buy a commodity and the user’s history of his behavior. For example, a person clicked a commodity online and put it into his shopping cart a month ago, then that person might buy such goods in the future. Recently, on account of its advantages, multilayer neural network has got more and more attention and application[12],[13]. In this paper, we use a neural network to predict what goods the users will buy in the future.

Our data set is the behavior record about 182,000 user four months provided by Alibaba Group, each record includes a user behavior for a commodity. It is very useful to predict what users will buy in the future, because we can take of the result to recommend appropriate goods to the users, which may prompt him to make a decision when the user hesitant to buy a commodity. It is possible to recommend to a user the commodities which he just need or really like, but he never thought. This recommendation not only promotes the sale of goods, but also increases the user's interest fore-commerce[5].

In this work, we did a survey and analysis of user purchase behaviors in existing data from various angles, then extracted several factors that may affect the user to purchase goods, such as the number of times a user clicks on a commodity, commodity correlation coefficient, merchandise sales trends. Then we put these data into our model for training and test the results. The efficiency of our model has been verified with numerical experiments.

# Method

There are 18, 2881 records in the original data while contain 4 fields, that is brand\_id, user\_id, time, action\_type, show in Table 1. Our goal is through the analysis of historical data to predict what users will buy merchandise in the future. If we see a user and a commodity as a combination, and we can get some information associated, then it can be seen as a classification problem, whether the user will buy the commodity or not under the influence of these factors. Therefore, it is a good choice to solve this problem by using the perception or multilayer perceptron. To this end, one need to train is a perceptron model or a multilayer perceptron, and then enter new data into the model to get a classification result that indicates whether the user buy the commodity.

**2.1 Perceptron**

The perceptron is an algorithm for supervised classification of an input into one of several possible non-binary outputs. It is a type of linear classifier, a classification algorithm that makes its predictions based on a linear predictor function combines a set of weights with the feature vector[14]. The perceptron algorithm is also termed the single-layer perceptron, to distinguish it from a multilayer perceptron. As a linear classifier, the single-layer perceptron is the simplest feed forward neural network.

A features related tuple on a user and a commodity, constitutes the input of the perceptron, as shown in Fig. 1. , j=1,…, d, is associated with a connection weight, or synaptic weight , and the output y, in the simplest case is a weighted sum of the inputs:

 (1)

is the intercept value to make the model more general, it is generally modeled as the weight coming from an extra bias unit , which is always 1. The output of the perceptron can be written as a dot product: , where w= and x= are augmented vectors to include both the bias weight and input.

The output is whether the user eventually purchased the goods, it is set to 1 if the buying behavior occurred, 0 otherwise.

We keep learning the weights w, the parameters of the system, such that correct outputs are generated given the inputs. We use stochastic gradient descent to update the weights, getting the following online update rule:

 (2)

Whereη is the learning factor, is the output and is the actual target value, is the input on the single instance with index t in the regression.

Table 1: The description of each field of the original dataset provided by Alibaba Group

| Field | Description | |
| --- | --- | --- |
| The field description | Extraction description |
| User\_id | User mark | Sampling |
| Time | Behavior time | Precision level to the day |
| Action\_type | The user of the brand behavior type | Including 4 kinds of behavior  (click:0 buy:1 collection:2 shopping cart: 3) |
| Brand\_id | The brand of digital | Sampling |

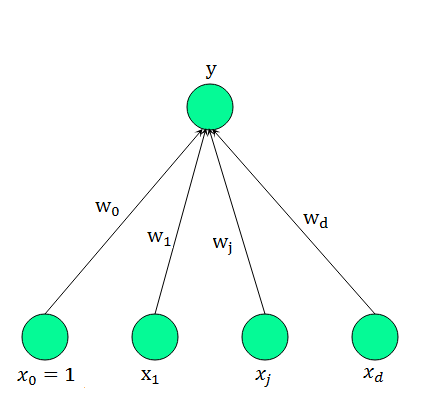


Fig. 1. The structure of a perceptron.*, j=0,…, d* are the input units. is the bias. *y* is the output unit. is the weight of the directed connection from input to the output.

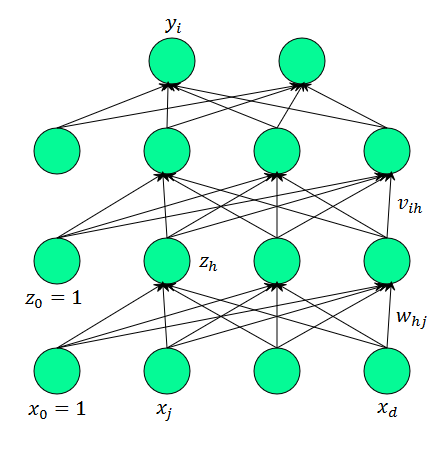


Fig. 2. The structure of a multilayer perceptron. *, j=0,…, d* are the inputs and, are the hidden units where *H* is the dimensionality of this hidden space.is the bias of the hidden layer. *y* is the output unit. are weights in the *hth* layer, and are the weights in the second layer.

**2.2 Multilayer Perceptrons**

A multilayer perceptron is a feed forward artificial neural network model that maps sets of input data onto a set of appropriate outputs. MLP is useful in research in terms of their ability to solve problems stochastically, which often allows one to get approximate solutions for extremely complex problems like fitness approximation[15]. Unlike the perceptron (only used for the problem linearly separable), the multilayer perceptron can approximate nonlinear functions of the input.

As shown in Fig. 2, Input X is fed to the input layer (including the bias); the "activation" propagates in the forward direction, and the values of hidden units are calculated. Each hidden unit is a perceptron by itself and applies the nonlinear sigmoid function to its weighted sum:

 (3)

Other activation functions, such as tangent[9] and rectified linear function[10], can be used as well[2]. The output are perceptrons in the second layer taking the hidden units as their inputs:

 (4)

where there is also a bias unit in each hidden layer, which we denote by , and are the bias weights.

The common method to train the multilayer perceptron is back propagation algorithm (see **Algorithm 1**). Considering the hidden units as inputs and the second layer a perceptron, we update the parameters by the rule: , given the inputs . For the first-layer weights , the chain rule is used to calculate the gradient:

 (5)

Therefore, it looks like the error propagates from the output y back to the inputs.

After a finite iteration, the algorithm will eventually converge, and then we get all the parameters of the model.

# Experiments

Before training a model, there are some preparatory work needs to be done, for example, feature extraction[11], data normalization, and sampling. In this regard, it is the same in perceptron experiment and multilayer perceptron experiment.

Table 2: All feature extracted from the original data set, they are used as the input of the experiments except the user\_id and the brand\_id

|  |  |
| --- | --- |
| Feature Name | Feature Description |
| User\_id | User mark |
| Brand\_id | Brand mark |
| Click\_count | The times of this user click this brand |
| Buy\_count | The times of this user buy this brand |
| Collection\_count | The times of this user collect this brand |
| ShoppingCar\_count | The times of this user add this brand to the shopping car |
| Click\_day\_count | The number of days this user behavior for this brand |
| Brand\_bought\_repeat | How often this brand is repeat purchased by single user |
| User\_brand\_count | The number of brand this user involved |
| User\_buy\_count | The total number of this user to buy goods |
| User\_action\_count | The total number of this user’s behavior |
| Sell\_trend | The sales trend of this brand |
| Sell\_rank | The brand sales ranking in all brand |
| Brand\_link | The degree of correlation between this brand and all brand purchased by this user in the past |
| User\_link | Probability of similar users who bought this brand |

**3.1 Feature Extraction**

According to user behavior of commodity, we found out many user and commodity tuple in the training data set, and extracted some features related. After screening, we finally chose 13 relatively important features, such as the number of times a user clicks on a commodity, a user's shopping total number, the number of times a commodity is repeat purchase, etc, as showed in Table 2. In order to accurately predict a user's purchase behavior, we combine the correlation coefficient of goods and the user similarity coefficients, which are considered to be quite important factors.

**3.2 Sampling**

We have tens of thousands of pieces of data in the training set. Considering hardware limitation and efficiency requirements, we cannot put all the data used to train the model. Besides, facts prove the need for effective sampling, because the negative sample data set accounted for the majority, that is, the number of the purchase data is very small in a large number of records about users and commodities. Just as we usually like to browse a lot of goods online, but we really decided to buy is one of a very small part.

In order to achieve a balance of positive samples and negative samples, we selected all of the positive samples and twice the number of negative samples while the negative sample is determined through a random sampling.

**3.3 Normalized Processing**

As the difference between the values of various features may be large, it is necessary to normalize the data before training to avoid the inefficient of gradient descent algorithm. In this work, we use the following rule to normalize the dataset:

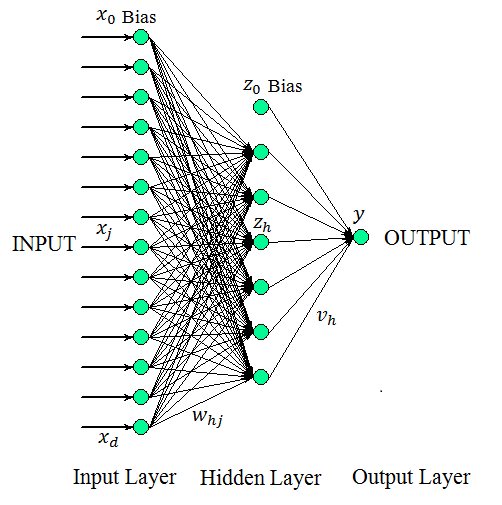
 (6)

**3.4 Training**

Several experiments are designed for perceptron and multilayer perceptron. In this section, the perceptron are shown to successfully classify all the samples which contain more than 2000 samples. By contrast, many experiments for MLP classification have been done for the purpose of achieving best performance. The algorithm is elaborated in details in this section.

We use the data preprocessed in Sec. 3.3 to do experiments, all the features together with a bias compose the input of the perceptron. The model is trained following the steps in Sec. 3.1. Because our training dataset are linearly separable, after a finite number of iterations, the algorithm can ultimately divide all the samples into different categories. We use the model to test the first three months’ data, and compare the results with the fourth month data, the F1 value (A common evaluation criteria in the field of information retrieval, introduced in Sec. 3.5) reaches 7.8%, showing the outperformance of MLP, comparing to most of the existing models on the same dataset, by which the best result is 7.7% (published by Alibaba).

Then we do the multilayer perceptron experiments. A typical MLP model has an input layer, an output layer and more than one hidden layers. In order to obtain better results, besides the commonly used three layer structure multilayer perceptron, we also tried more deep structures. We considered different number of neurons in hidden layer in a large number of tests. In our experiments, we set 14 neurons in the input layer, of which 13 for the input feature values, the other one as a bias. As it is a classification problem with two classes, there are two neurons in the output layer, one for the output, and the other is the bias. In different experiments, the number of the hidden layer is different, also the number of the neurons. Input data shown in Table 2.



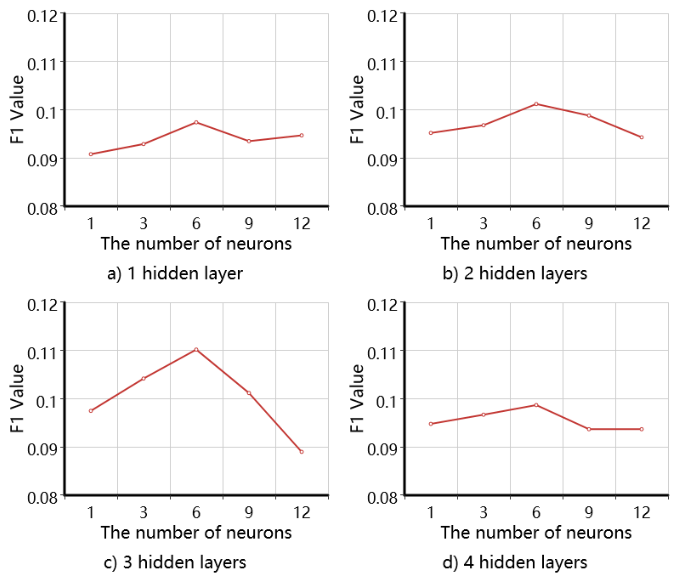


Fig. 3. The structure of the multilayer perceptron with one hidden layer in our experiment.



3**.5 Testing**

To verify the efficiency of MLP, we measure the result of experiment with an evaluation called F-Measure, which can be calculated as follows:

 (7)

Where *P* is Precision, *R* is Recall.

In our experiment, the MLP reached relatively high prediction efficiency, the maximum value of F1 we got is 11.02% for five-layer MLP, which is much higher than the result of single layer perceptron experiment, while the best result in the Alibaba large data race organized is 7.7%. Some experiment results of the experiments are reported in Table 3.

Table 3: The F1 on condition that the mlp have different number of hidden layers and different number of units in each hidden layer in our experiment

| Number Of Hidden Layers | Number Of Units In Each Hidden Layer | | | | |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 6 | 9 | 12 |
| 1 | 0.0908 | 0.0929 | 0.0974 | 0.0935 | 0.0947 |
| 2 | 0.0952 | 0.0968 | 0.1012 | 0.0988 | 0.0943 |
| 3 | 0.0975 | 0.1042 | 0.1102 | 0.1012 | 0.0890 |
| 4 | 0.0948 | 0.0967 | 0.0987 | 0.0937 | 0.0937 |

Fig. 4. The F1 value when the mlp have different number of hidden layers and different number of units in each hidden layer in our experiment.

As shown in Fig. 4, the MLP's performance does not always improve with the increase in the number of hidden layers and the number of units in each hidden layer, it reaches the maximum value when each hidden layer units is about half of the number of units in the input layer, and may decline since then.

Although the performance of our model is much higher than the highest level published by the Alibaba Group, we did several experiments with other algorithms to compare with it, such as Support Vector Machine (SVM)[16], k-Nearest Neighbors (KNN)[17], Naive Bayes[18], Collaborative Filtering[19]. We used the same input data shown in Table 1 and Table 2, and the same criteria (introduced in Sec. 3.5) to measure the results. We found that it is still much higher than the performance of all other models, the Table 4 shows all the comparison results.

Table 4: The experimental results of different model

|  |  |
| --- | --- |
| Models | F1 Value |
| Our model | 11.02% |
| Alibaba top restult | 7.76% |
| Collaborative Filtering | 6.92% |
| KNN | 6.78% |
| SVM | 6.19% |
| Naive Bayes | 5.52% |

# Conclusions

In this paper, we have presented a high performance prediction model for predicting customers' behavior in e-commerce by using deep hidden layers MLP, it is much better than many other algorithms to solve this problem. We investigated the effect of network depth on the overall performance and showed that the deep hidden layers can more efficiently solve the classification problem than the traditional three-layered MLP. We also find that MLP's performance will be better when the number of hidden layer neurons is about half the number of input layer neurons.

# References

1. Sarwar, Badrul, et al. "Analysis of recommendation algorithms for e-commerce." Proceedings of the 2nd ACM conference on Electronic commerce. ACM, 2000.
2. Zhu, Zhenyao, et al. "Multi-view perceptron: a deep model for learning face identity and view representations." Advances in Neural Information Processing Systems. 2014.
3. Schafer, J. Ben, Joseph Konstan, and John Riedl. "Recommender systems in e-commerce." Proceedings of the 1st ACM conference on Electronic commerce. ACM, 1999.
4. Sarwar, Badrul, et al. "Analysis of recommendation algorithms for e-commerce." Proceedings of the 2nd ACM conference on Electronic commerce. ACM, 2000.
5. D Harrison McKnight, Norman L. Chervany. "What trust means in e-commerce customer relationships: an interdisciplinary conceptual typology." International journal of electronic commerce 6.2 (2001): 35-59.
6. Schafer, J. Ben, Joseph A. Konstan, and John Riedl. "E-commerce recommendation applications." Applications of Data Mining to Electronic Commerce. Springer US, 2001. 115-153.
7. Schwab, Ingo, Wolfgang Pohl, and Ivan Koychev. "Learning to recommend from positive evidence." Proceedings of the 5th international conference on Intelligent user interfaces. ACM, 2000.
8. Ma, Hao, Irwin King, and Michael R. Lyu. "Learning to recommend with social trust ensemble." Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval. ACM, 2009.
9. T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. TPAMI, 28:2037–2041, 2006.
10. A. Asthana, T. K. Marks, M. J. Jones, K. H. Tieu, and M. Rohith. Fully automatic poseinvariant face recognition via 3d pose normalization. In ICCV, 2011.
11. Ruck, Dennis W., Steven K. Rogers, and Matthew Kabrisky. "Feature selection using a multilayer perceptron." Journal of Neural Network Computing 2.2 (1990): 40-48.
12. Parlos, Alexander G., Kil To Chong, and Amir F. Atiya. "Application of the recurrent multilayer perceptron in modeling complex process dynamics." Neural Networks, IEEE Transactions on 5.2 (1994): 255-266.
13. Yan, Hongmei, et al. "A multilayer perceptron-based medical decision support system for heart disease diagnosis." Expert Systems with Applications 30.2 (2006): 272-281.
14. F. Rosenblatt. "The perceptron, A perceiving and recognizing automation." Cornell Aeronautical Laboratory Report No. 85-460-1, Jan.1957.
15. Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961.
16. Suykens, Johan AK, and Joos Vandewalle. "Least squares support vector machine classifiers." Neural processing letters 9.3 (1999): 293-300.
17. Fukunaga, Keinosuke, and Patrenahalli M. Narendra. "A branch and bound algorithm for computing k-nearest neighbors." Computers, IEEE Transactions on 100.7 (1975): 750-753.
18. McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naive bayes text classification." AAAI-98 workshop on learning for text categorization. Vol. 752. 1998.
19. Sarwar, Badrul, et al. "Item-based collaborative filtering recommendation algorithms." Proceedings of the 10th international conference on World Wide Web. ACM, 2001.



**Pei Zhao** was born in Henan, China in 1989. He received the B.S. degrees from Henan University, Henan, China in 2013, is currently pursuing the M.S. degree with Southwest Petroleum University, Chengdu. His research interests include social network analysis, machine learning. He joined Prof. Tao Zhou and Dr. Jun-Lin Zhou’s Hire Big Data Company as a Data Analyst.

**Ping Li** received the Ph.D. degree in Computer software and theory from the University of Electronic Science and Technology of China (UESTC), Chengdu, in 2009. From 2009 to 2010, she was invited to The Hong Kong Polytechnic University, Hong Kong, China and awarded the postdoctoral. Since 2011, she has been with the Southwest Petroleum University, where she is presently a full associate professor and graduate supervisor with the School of computer science. Her research interests include social network analysis, machine learning, natural language processing.



1. Manuscript received ??????, 20??; revised ???????, 20??.

   P. Zhao is with School of Computer Science, Southwest Petroleum University, Chengdu, 610500, China (e-mail: pei.zhao@hirebigdata.cn).

   P. Li is with School of Computer Science, Southwest Petroleum University, Chengdu, 610500, China (e-mail: dping.li@gmail.com). [↑](#footnote-ref-2)