Linear and Non-Linear Models for Purchase Prediction

Wenliang Chen, Zhenghua Li, and Min Zhang
School of Computer Science and Technology
Soochow University, Suzhou 215006, China
chenwenliang@gmail.com

ABSTRACT

In this paper, we present our approach for the task of product purchase prediction. In the task, there are a collection of sequences of click events: click sessions. For some of the sessions, there are also buying events. The target of this task is to predict whether a user is going to buy something or not in a session, and if the user is buying, which products (items) the user is going to buy. In our approach, we treat the task as a classification problem and use linear and non-linear models to make the predictions, and then build an ensemble system based on the output of the individual systems. The evaluation results show that our final system is effective on the test data.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithms; Experimentation; Economics

Keywords

Purchase Prediction; Recommender System; Linear Model; Non-Linear Model

1. INTRODUCTION

Recently, recommender system is becoming a popular, useful, yet simple task in web applications. The system recommends some products (items) for users by analyzing history behaviors of users. Many e-commerce businesses use recommender system to serve their clients by listing top-N recommendations.

In the challenge of this year[1], the organizers provide a collection of sequences of click events; click sessions. For some of the sessions, there are also buying events. The data is from YOOCHOOSE[1], which is a recommender service provider. YOOCHOOSE provides top-N recommendations on a user base for different use cases, exploits long tail and last but not least to keep the user entertained. The target of this challenge is to predict whether a user is going to buy something or not in a session, and if the user is buying, which products the user is going to buy. Such information is very important to an e-business as it can indicate not only which products to suggest to the user but also how it can encourage the user to become a buyer.

In this paper, we present our system in the challenge. In our system, we use the models of Maximum Entropy[3, 7] and Factorization Machines[8], and Multi-Layer Perceptronn (Feedforward Neural Networks)[2] to make the predictions individually, and then build a ensemble system based on the output of the individual systems. The evaluation results show that our final system is effective on the test data.

2. TASK

The detailed information of the task can be found in the challenge site[2]. In the evaluation, the system is given a sequence of click events performed by some user during a typical session in an e-commerce website, the goal is to predict whether the user is going to buy something or not, and if he is buying, what would be the items he is going to buy. The task could be divided into two sub-tasks:

1. Is the user going to buy items in this session? Yes|No
2. If yes, what are the items that are going to be bought? A list of bought items

2.1 Data

The data[3] is from YOOCHOOSE containing training and test data files. The training data comprises two different files: yoochoose-clicks.dat for click events and yoochoose-buys.dat for buy events. Table 1 shows the format for each record/line in the files. The test data contains one file: yoochoose-test.dat, which is identically structured as the yoochoose-clicks.dat of the training data.

The Session ID in yoochoose-buys.dat will always exist in the yoochoose-clicks.dat file. In the files, the records with the same Session ID together form the sequence of click events of a certain user during the session. The session could be short (a few minutes) or very long (a few hours), and it could have one click or hundreds of clicks depending on the activity of the user. The statistical information of data is shown in Table 2. From the table, we find that about 30 click evens has one buy event. This will result in the data sparseness problem when training models.

[1]http://www.yoochoose.com/

RecSys '15 Challenge, September 16-20, 2015, Vienna, Austria

DOI: http://dx.doi.org/10.1145/2813448.2813518.

[2]http://2015.recsyschallenge.com/challenge.html
[3]The data is available at http://s3-eu-west-1.amazonaws.com/yc-rdata/yoochoose-data.7z

Click events	
Session ID	the id of the session. In one session there are one or many clicks.
Timestamp	the time when the click occurred.
Item ID	the unique identifier of the item.
Category	the category of the item.
Buy events	
Session ID	the id of the session. In one session there are one or many buying events.
Timestamp	the time when the buy occurred.
Item ID	the unique identifier of item.
Price	the price of the item.
Quantity	how many of this item were bought.

Table 1: Data format for each record

	Training	Test
#ofSessions	9,249,729	2,312,432
#ofClick Events	33,003,944	8,251,791
#ofBuy Events	1,150,753	N/A
#ofClick Events Per Session	3.571	3.568
#ofBuy Events Per Session	0.124	N/A

Table 2: Statistical information of data

2.2 Evaluation measure

According to the targets, the evaluation is taking into consideration the ability to predict both aspects: whether the sessions end with buying event, and what were the items that have been bought. The evaluation measure is as follows,

$$Score(Sl) = \sum_{\forall s \in Sl} \begin{cases} \frac{|S_b|}{|S|} + \frac{A_s \bigcap B_s}{A_s \bigcup B_s} & \text{if } s \in S_b \\ -\frac{|S_b|}{|S|} & \text{else} \end{cases}$$

where Sl refers to the set of sessions in submitted solution file, S refers to the set of all sessions in the test set, s refers to a session in the test set, S_b refers to the set of sessions in test set which end with buy, A_s refers to the predicted bought items in session s, and B_s refers to the actual bought items in session s.

3. OUR SYSTEMS

3.1 Prediction as classification

In our system, we treat purchase prediction as a classification problem. Figure 1 shows that the task is to predict whether the curItem is going to be bought in one session. For each session, we directly predict if each clicked item is going to be bought. The information of curItem and surrounding items is used as features for the classifiers. We design a set of features to represent the data X and the target label Y has two values: Yes and No. Table 3 lists the feature templates we use in our experiments. In the template of isSim in the table, item-embeddings are trained on the click sequences. Item-embeddings are similar to word-embeddings that represents the words as distributed vectors. The itemIDs are treated as words and the click sequences are as sentences. Traditional LSA and RNN [4, 2] models can be used to learn embeddings but turn out to be slow. Recently, [5] and [6] introduce efficient models to learn high-quality word embeddings from extremely large amounts of raw text. Thus, we use their tool word2vec[4] to train item-embeddings.

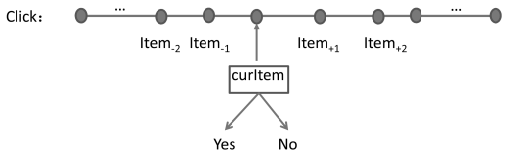

Figure 1: Prediction as classification

We select the combination of feature templates for the final systems by tuning on the development data set.

3.2 Maximum Entropy

Maximum Entropy model has been successfully applied to various classification tasks including text categorization and part-of-speech tagging with the state-of-the-art accuracies [3, 7]. The basic idea behind Maximum Entropy is that one should prefer the uniform models that also satisfy any given constraints. In its most general formulation, the Maximum Entropy model can be used to estimate any probability distribution. Here we are interested in scoring the input examples by the binary classification way. That is, we learn conditional distributions from labeled training data. Specifically, we learn the probability distribution of the classes (positive and negative) given a review content.

With the constraints (features) given the training data, there is a unique distribution that has maximum entropy. In general, a conditional Maximum Entropy model is an log-linear model as follows:

$$p(y|x) = \frac{1}{Z(x)} \exp(\sum_i (\lambda_i f_i(x, y))) \quad (1)$$

where each $f_i(x, y)$ is a feature, y is the a class, λ_i is a parameter to be estimated and $Z(x)$ is simply the normalizing factor to ensure a proper probability:

$$Z(x) = \sum_t \exp(\sum_i (\lambda_i f_i(x, y))) \quad (2)$$

Then, we use feature templates in Table 3 to generate the features for Equation 1.

3.3 Factorization machines

Factorization machines (FM)[8] are a generic approach that can make full use of large-scale features. It combines the generality of feature engineering with the superiority of factorization models in estimating interactions between categories.

The data of the prediction problem in this paper is described by a design matrix $X \in R^{n \times m}$, where n is the number of samples, m is the size of features, x is one sample and y is the prediction target for x. Factorization machines (FM) model all nested interactions between the m input variables in x using factorized interaction parameters. The model is defined as,

$$y(x) = w_0 + \sum_{j=1}^{m} w_j x_j + \sum_{j=1}^{m} \sum_{j'=j+1}^{m} x_j x_{j'} \sum_{f=1}^{k} v_{j,f} v_{j',f} \quad (3)$$

where k is the dimensionality of the factorization and the model parameters w and v to be learned in training data.

Factorization machines can use stochastic gradient descent (SGD) and alternating least squares (ALS) optimization as well as Bayesian

[4]https://code.google.com/p/word2vec/

Feature	Description
curItemID	the current Item ID
Prefix(curItemID, i) $i \in \{4, 5, 6\}$	the first i digits of curItemID
IsFirstClick(curItem)	whether the current Item is first clicked in this session
isLastClick(curItem)	whether the current Item is last clicked in this session
ClickNum(curItem)	the click number of the current item. if ClickNum(curItem) > 3, set it as 3
isHotItem(curItem)	whether the current Item has been bought more than 10 times in training data
date	the date of the click occurred
month	the month of the click occurred
day	the day of the click occurred
hour	the hour of the click occurred
isSame(curItem, Item_{+1})	whether the current Item is the same as the next item
isSame(Prefix(curItem, 4), Prefix($\text{Item}_{-1}, 4$))	whether the prefix of the current Item is the same as the previous item
isSame(Prefix(curItem, 6), Prefix($\text{Item}_{-1}, 6$))	whether the prefix of the current Item is the same as the previous item
isSame(Prefix(curItem, 4), Prefix($\text{Item}_{+1}, 4$))	whether the prefix of the current Item is the same as the next item
isSame(Prefix(curItem, 6), Prefix($\text{Item}_{+1}, 6$))	whether the prefix of the current Item is the same as the next item
isSame(curItem, Item_{-1})	whether the current Item is the same as the previous item
Category(Item, i) $i \in [-5, 5]$	the category of the items within window size=5
isSim(curItem, Item_i) $i \in [-2, +2]$ and $i \neq 0$	whether the current item is similar (cos >= 0.8) to the surrounding items based on item-embeddings
isMostClick(curItem)	whether the current Item is most clicked in this session
preClickDiff	the time diff (min) between the current click and the previous click. If the diff is larger than 5 mins, set it as 5
nextClickDiff	the time diff (min) between the current click and the next click. If the diff is larger than 5 mins, set it as 5

Table 3: Feature Templates

inference using Markov Chain Monte Carlo (MCMC) to train the models. In our system, we use MCMC to learn the parameters. We also use feature templates in Table 3 to generate the features for FM.

3.4 Multi-Layer Perceptron

We also use single-hidden-layer Multi-Layer Perceptron (MLP)[2]. First, we learn a non-linear transformation which can projects the input data into a space where it becomes linearly separable. The space is a intermediate layer referred to as a hidden layer. Then, a logistic regression classifier takes the transformed vectors for predicting the finale results. It is possible to use many such hidden layers to transform the input before the classifier.

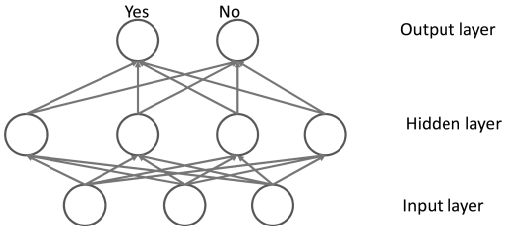

Figure 2: MLP with a single hidden layer

An MLP with a single hidden layer can be represented as Figure 2, where the input layer contain the features defined in Section 3.1 and the output layer has two possible labels (Yes or No).

Given the input x, $h(x) = s(W^1 x + b^1)$ represents the hidden layer, where W^1 is the weight matrix connecting the input vector to the hidden layer and b^1 is a bias vector for the transformation. Here, we use $tanh(a) = (e^a - e^{-a})/(e^a + e^{-a})$ for function s since it is fast when training the parameters.

The prediction is then obtained as:

$$P(y|x) = softmax(W^2 h(x) + b^2) \tag{4}$$

where W^2 is the weight matrix connecting the hidden layer to the output layer and b^2 is a bias vector for the classification. To train an MLP classifier, we learn all parameters of the model by using Stochastic Gradient Descent. In our system, the number of neurons is 100. Finally, the prediction of the model y_{pred} is the class whose probability is maximal:

$$y_{pred} = \text{argmax}_i P(Y = y_i|x) \tag{5}$$

3.5 Ensemble

The ensemble systems usually show powerful ability for open challenges. Thus we also build a ensemble system to generate the final prediction. The Logistic Regression (LR) model is used in our ensemble system, which predicts a binary response (Yes or No) based on the output of base classifiers.

The logistic function $\sigma(t)$ is defined as follows:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}} \tag{6}$$

where $t = \beta_0 + \beta_1 y_1 + \beta_2 y_2 + \cdots + \beta_m y_m$, y_i is the output of the i^{th} classifier. Then the parameters β_j for all j = 0, 1, 2, ..., m are all estimated during training.

4. EVALUATION

To prevent the overfitting problem, we use a development set to build our systems. Because there is not official development data set, we select some sessions (sessionID % 10 == 6) from the

training data. The number of negative (NoBuy) samples is much larger than the one of positive samples. This causes the imbalance problem that will result in the worse performance. Thus, we randomly remove the negative samples and keep the ratio to 2:1 (negative:positive) in the training data.

To know how far our systems is behind the perfect system, we define score-ratio (SR) as follows,

$$SR = 100 \times \frac{Score(Sl)}{MaxScore} \quad (7)$$

where MaxScore is the score a perfect system can achieve.

4.1 Systems

We build several systems for this task. The systems are as follows:

- NaiveSystem: a rule-based system. The detail information can be found at a blog[5].
- MaxEnt: a Maximum-Entropy-based system as described in Section 3.2. MaxEnt[6] is used to build the system.
- FM: a Factorization machines-based system as described in Section 3.3. libFM[7] is used to build the system.
- MLP: a system based on Multi-Layer Perceptron as described in Section 3.4.
- Ensemble: a ensemble system based on the output of MaxEnt and FM. According to the results on the development set, we do not use MLP for the ensemble system. We build the ensemble system on the development set.

4.2 Tuning parameters

We only tune the feature combinations for the MaxEnt system on the development set. For FM and MLP, we use the chosen features directly. The results are shown in Table 4. From the table, we find that even with Base Features the MaxEnt system performs much better than the NaiveSystem. The new added features can further improve the system: 1) The category information is quite useful for prediction; 2) The most clicked item in one session is tend to be bought; 3) The time gap between two clicks is important for predicting buy event; 4) The similarity information between the current item and the previous/next items is also useful. Since the ensemble system is trained on the development set, we do not evaluate it in this section.

4.3 Final results

Finally, we run our systems on the test set and the results are listed in Table 4. The results show that MaxEnt and FM outperform the NaiveSystem while MLP is a little worse than the NaiveSystem. The ensemble system is built on MaxEnt and FM and achieves better scores than the individual systems. Because we use MLP to build the prediction system in the last month of the challenge period, we have not enough time to tune the MLP system. This might be the reason that the MLP system has not shown great power on this evaluation. We will continue to investigate the effect of MLP in near future.

[5]http://playwithnlp.blogspot.com/2014/12/how-to-build-naive-very-naive-system.html
[6]http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html
[7]http://libfm.org/

System	Score(dev)	SR(dev)	Score(test)
NaiveSystem	12361.6	18.52	33780.1
MaxEnt			
Base Features	18881.4	28.29	
+Category	20343.8	30.48	
+isSim	20622.6	30.89	
+mostClick	21208.4	31.77	
+ClickDiff	21715.1	32.53	52511.4
FM	17241.9	27.31	42520.6
MLP	12531.4	18.77	32662.8
MaxEnt+FM	N/A	N/A	53341.5

Table 4: Results on the dev and test data

5. CONCLUSION

In this paper, we have present our purchase prediction system which submitted the results to the challenge site. We design a set of features and use three different classifiers to build the systems. Then, we build an ensemble system as our final system. The evaluation results show that the final system is effective on the test data.

As for the deep learning techniques, the item-embeddings is used for calculating the similarity between items which is useful in feature representation. However, the MLP model has not shown the effective on the test data. We plan to investigate how to make MLP work in this task in future work.

6. ACKNOWLEDGMENTS

We were supported by the National Natural Science Foundation of China (Grant No. 61203314, 61373095, and 61273319). This work was also partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

7. REFERENCES

[1] D. Ben-Shimon, A. Tsikinovsky, M. Friedman, B. Shapira, L. Rokach, and J. Hoerle. Recsys challenge 2015 and the yoochoose dataset. In *the 9th ACM conference on Recommender systems*. ACM, 2015.

[2] Y. Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[3] A. L. Berger, S. A. D. Pietra, and V. J. D. Pietra. A maximum entropy approach to natural language processing. *Journal of Computational Linguistics*, 22(1):39–71, 1996.

[4] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[6] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. 2013.

[7] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *Proc. of IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

[8] S. Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012.