Comprehensive audience expansion based on end-To-end neural prediction

Jiang, Jinling; Lin, Xiaoming; Yao, Junjie; Lu, Hua

Published in:
CEUR Workshop Proceedings

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Comprehensive Audience Expansion based on End-to-End Neural Prediction

Jinling Jiang  
MiningLamp Technology.  
Beijing, China  
jiangjinling@mininglamp.com

Junjie Yao  
East China Normal University.  
Shanghai, China  
junjie.yao@sei.ecnu.edu.cn

Xiaoming Lin  
MiningLamp Technology.  
Beijing, China  
linxiaoming@mininglamp.com

Hua Lu  
Department of Computer Science, Aalborg University.  
Aalborg, Denmark  
luhua@cs.aau.dk

ABSTRACT
In current online advertising applications, look-alike methods are valuable and commonly used to identify new potential users, tackling the difficulties of audience expansion. However, the demographic information and a variety of user behavior logs are high dimensional, noisy, and increasingly complex, which are challenging to extract suitable user profiles. Usually, rule-based and similarity-based approaches are proposed to profile the users’ interests and expand the audience. However, they are specific and limited in more complex scenarios.

In this paper, we propose a new end-to-end solution, unifying the feature extraction and profile prediction stages. Specifically, we present a neural prediction framework and leverage it with the ACM of the SIGIR 2019 Workshop on eCommerce (SIGIR 2019 eCom) Jinling Jiang, Xiaoming Lin, Junjie Yao, and Hua Lu. 2019. Comprehensive Audience Expansion based on End-to-End Neural Prediction. In Proceedings of the SIGIR 2019 Workshop on eCommerce (SIGIR 2019 eCom), 8 pages.

KEYWORDS
Online Advertising; Audience Expansion; Lookalike Modeling

CCS CONCEPTS
• Information systems → Online Advertising; • Human-centered computing → User Models; • Theory of computation → Computational Advertising theory; • Computing methodologies → Factorization methods;

1 INTRODUCTION
The remarkable growth of online advertisement enables the advertisers to sync up their products according to the fast-changing needs of the consumer. As the development of e-commerce platforms has introduced SMEs (Small and medium-sized enterprises) to enter consumers’ sight, large enterprise advertisers face the crisis of slowing business growth and falling revenue. Therefore, brand advertisers have begun to pay more attention to the contribution of advertising to sales conversion, the actual revenue brought by advertising, requiring advertising agencies and third-party suppliers to provide more refined performance data of advertising effects.

Meanwhile, the emergence of big data technology has subverted the operation model of the entire advertising industry and the traditional way of evaluating advertising effects. By tracking and obtaining user behavior data, a third-party supplier of advertising monitor can analyze the data according to the advertiser needs, not only understanding the communication effects and sales conversion rate generated by the advertisement in time but also predicting the user conversion probability to some extent. Through analysis and modeling on massive data of user behavior, advertisers can accurately reach the target consumer. Therefore, how to better utilize the advertising monitor data in order to optimize ad serving and improve marketing conversion rate has become an important issue.

One of the main challenges in ad serving is how to find the best converting prospects. A typical way is to do audience expansion, that is, to identify and reach new audiences with similar interests to the original target audience. Usually, the methodology used in audience expansion problem is called look-alike modeling. Given a seed user set \( S \) from a universal set \( U \), look-alike models essentially find groups of audiences from \( U - S \) who look and act like the audience in \( S \).

The data flow of audience expansion service is illustrated in Figure 1. The data runs between advertisers and our universal advertising monitor system across different media platforms. The original users come from the advertiser’s CRM System selecting the consumers who recently exercise the purchase actions. Then the users who are tracked by the universal advertising monitor will be matched and treated as "seed" users.

In this paper, we build up a closed-loop data solution for brand advertisers and combines multiple techniques of selecting negative samples and extracting features, as well as machine learning lookalike models to reach the targeted audience. Which greatly enhances the conversion effect of ad serving. Based on the "seed" users and universal user set from advertising monitor, we build a lookalike
Figure 1: Audience Expansion Dataflow

model to predict the probability to be the target audience for all users. Afterward, according to the advertising budget, lookalike model will yield the corresponding number of expanded users to be reached through ad serving system. Finally, the ad serving performance is evaluated by advertiser’s site monitor system that record sales conversion shortly.

But both traditional and current look-alike strategies for an advertiser to look for the target audience are mainly based on user demographics. There are two main problems with demographics-based audience segmentation: user demographics (age, gender, and geographical location) itself is not precise as it is estimated via various statistical methods or machine learning models based on a small group of surveyed samples (10-100 thousand); the number of users that are specified by demographics is large, more sophisticated screening is required. Accordingly, the details of user behavior data should be harnessed in machine learning models to target accurate audience segment. At the same time, there are two main problems that need to be solved based on user behavior data modeling: user-generated behavior data through the Internet is generally high-dimensional and sparse; advertisers usually can only provide positive samples, while negative samples need to be carefully picked up from a substantial unlabeled sample set.

Besides, the ecologically closed Internet tycoons (represented by Facebook, Amazon, Tencent, Alibaba and etc.) provide the advertisers the capability to perform audience expansion within their own platforms. However, ad serving data of these platforms are not connected with the advertiser’s CRM (Customer Relationship Management) system. Thus, it is difficult to directly track the real conversion rate. In order to verify that lookalike models based on the user behaviour work better than traditional demographics-based approach regarding the sales conversation rate, we need to integrate data flow during the whole advertising life cycle.

The contributions of this paper can be summarized as follows.

- We have improved the commonly used ad serving mode from demographics-based crowd segmentation to a comprehensive audience expansion framework.
- We propose a lookalike model that has better generalization ability for audience expansion problem.
- We conduct extensive and effective experiments to extract negative samples from unlabeled data.
- We prove the effectiveness of the proposed lookalike models in an online environment.

The rest of the paper is organized as follows. In Section 2, we review the related work on various kinds of look-alike models and illustrate different design philosophy behind them.

Section 3.3 gives out the formal problem statement and specifies the notations used in the paper. We then introduces our proposed lookalike models and Section 3.4 reveals the sampling strategies. The evaluation of the algorithm is presented in Section 4. Finally the conclusion and future work are discussed in Section 5.

2 RELATED WORKS

We briefly review the related literature of look-alike modeling. Generally in online user-targeted advertising areas, look-alike modeling which supports audience expansion system can be categorized in three lines: rule-based, similarity-based and model-based.

**Rule-based approaches** focus on explicit positioning, where users with specific demographic tags (age, gender, geography) or interests are targeted directly for advertiser. The core technical support in the background is user profile mining, which means, the interest tags are inferred from the user behaviour [20][27]. Furthermore, Mangalampalli et al. [17] builds a rule-based associative classifier for campaigns with less conversion; Shen et al. [24] and Liu et al. [14] present detailed in-depth analysis of multiple methods under different considerations(such as similarity, performance, whether or not campaign-agnostic) for online social network advertising. The main disadvantage of rule-based look-alike modeling is that it only captures the high-level features, therefore loses sophisticated details of user behaviour.

**Similarity-based approaches** apply different similarity metrics to solve the problem of look-alike modeling. Naive similarity-based method computes pairwise similarities between and seed user and all the other users in the set while the locality-sensitive hashing (LSH) [25] technique is often applied to decrease the computation complexity of pairwise similarity. In addition, based on Ma et al. [15][16] provide several similarity scoring methods to
We formalize the look-alike modeling as a prediction problem. Advertisers submit a list of customers, which we call seed user set \( S \), as positive samples and there is a universal user set \( U \) existing in advertising monitor platform. Then the problem is transformed into a Positive and Unlabeled learning problem: using a small number of labeled positive samples \( S \) and a large number of unlabeled samples \( U - S \) to derive a prediction classifier. Eventually unlabeled users are scored by the classifier and the target audience set \( T \) is taken out according to advertising requirements. The dataset sizes are typically configured in real business environment as follows: \( ||S|| = 0.1-0.2M \) (Million), \( ||T|| = 10-20M \) and \( ||U|| = 2000-3000M \). Meanwhile, a user is represented by a feature vector which indicates the user’s past behaviour collected by the advertising monitor system. The feature vector always occurs with high-dimension \( D \) and extreme sparsity. \( D \) is usually around 100-300 thousands and only 0.1 percent of the feature vector are non-zero elements.

### 3 THE PROPOSED APPROACH

#### 3.1 Problem Statement

We formalize the look-alike modeling as a prediction problem. Advertisers submit a list of customers, which we call seed user set \( S \), as positive samples and there is a universal user set \( U \) existing in advertising monitor platform. Then the problem is transformed into a Positive and Unlabeled learning problem: using a small number of labeled positive samples \( S \) and a large number of unlabeled samples \( U - S \) to derive a prediction classifier. Eventually unlabeled users are scored by the classifier and the target audience set \( T \) is taken out according to advertising requirements. The dataset sizes are typically configured in real business environment as follows: \( ||S|| = 0.1-0.2M \) (Million), \( ||T|| = 10-20M \) and \( ||U|| = 2000-3000M \). Meanwhile, a user is represented by a feature vector which indicates the user’s past behaviour collected by the advertising monitor system. The feature vector always occurs with high-dimension \( D \) and extreme sparsity. \( D \) is usually around 100-300 thousands and only 0.1 percent of the feature vector are non-zero elements.

#### 3.2 Feature Extraction and Analysis

Here we introduce the feature extraction and analysis stages in the lookalike model.

### Table 1: An example of data from advertising monitor system

<table>
<thead>
<tr>
<th>CLICK</th>
<th>Timestamp</th>
<th>USER_ID</th>
<th>SPID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>201809123278</td>
<td>66a7988f</td>
<td>107122831</td>
</tr>
<tr>
<td>0</td>
<td>201809123346</td>
<td>9e664577</td>
<td>107108909</td>
</tr>
<tr>
<td>1</td>
<td>201809123456</td>
<td>9b3f6c94</td>
<td>107106418</td>
</tr>
<tr>
<td>0</td>
<td>201809123787</td>
<td>0043bf4f</td>
<td>107102974</td>
</tr>
<tr>
<td>0</td>
<td>201809132592</td>
<td>1df73293</td>
<td>107108909</td>
</tr>
</tbody>
</table>

Each row of the original data collected by advertising monitor system represents an ad impression. The “CLICK” column is an indicator that shows whether or not the advertisement is clicked by the corresponding user (1 represents CLICK while 0 means the opposite). As shown in Table 1, The main information of an ad impression includes timestamp, user_ id and an spid. The spid refers to the specific information of an advertisement where they are multi-field categorical data [28] which are commonly seen in CTR prediction and recommendation system.

The user behaviour is represented by a high-dimensional sparse feature vector where each feature corresponding to the times an advertisement is clicked or impressions. One typical feature extraction result is shown in Table 2, User “66a7988f” is impressed by spid1 and spid2 both 3 times while he only clicks spid2 once. The user feature vector will be normalized afterwards. The normalization approach is as follows where \( freq \) represents the original frequency and \( norm_freq \) is the frequency after normalization:

\[
norm_freq = \begin{cases} 
1 & \text{if } freq > 0 \\
1 + \exp \left( -\frac{freq}{m} \right) & \text{if } freq = 0
\end{cases}
\] (1)

To this end, every feature value is converted to a number between 0 and 1.

It is noteworthy that the data label is the purchase tag (meaning the corresponding user has purchase action) from CRM system of a particular brand advertiser over a period of time, while features represent the impression and click behaviour for ads of different brands. Unlike the high-dimensional sparse feature transformed by one-hot encoder in CTR prediction task, the original feature space is already sparse and high-dimensional.

The intuitive idea of utilizing spid as feature is that the ads are somehow correlated to the websites highly indicating user interests. That is to say, when an internet user is impressed by an specific ad, the ad itself could describe the user interests to some extend. Moreover, “CLICK” information directly connects user intention. The detailed comparison of different feature extraction methodologies will be incorporated in Section 4.2.

#### 3.3 Comprehensive Modeling

We continue to introduce the lookalike model techniques used in our audience expansion system. Multilayer Perceptron (MLP) is a feedforward neural network consisting of several layers. By adding non-linear activation functions, MLP can fit high-order non-linear features. Figure 2 illustrates a MLP network added by a scale layer.
The process is difficult to converge. Suppose that the feature \( x \) is quite different, the fluctuation of feature values will make training only update \( w \) affect the parameters \( a \) during backpropagation, the partial derivative regarding \( a \) is added. When MLP updates the parameter matrix \( W \), the intermediate result \( X \) need to be feedforward.

Compared to a standard MLP, Equation 3 reflects that the network updates the network parameter during model training stage. However, some features are often different on different solutions in practice, therefore, the effectiveness of actual models is limited. That is to say, the scale layer during backpropagation can directly change the final influence of each feature on the model.

Generally speaking, for MLP model, matrix \( A \) captures the first-order combinatoric features. In order to learn high-order features, the model need to fit the data by adjusting both the parameters of matrix \( A \) and the hidden layers of MLP. Due to the sparsity of feature space and importance of different features varies, the parameters of matrix \( A \) cannot be very effectively trained. Under such circumstances, the MLP model is easier to overfit. On the other hand, the Scale-MLP model only needs to train the parameters of the scale layer properly for the same purpose. Therefore, Scale-MLP model is much simpler to train in our setting.

Another angel to look at the functionality of the new model is that it adds randomness to the original user feature vector. In other words, if a user is not impressed by some ad, it doesn’t mean that he/she is totally not interested in that ad. Therefore, the scale layer will help to learn a model which has better generalization capability for this task.

3.4 Model Training

3.4.1 The Impact of Sampling Ratio. We evaluate the impact of sampling ratio based on different number of positive and unlabeled samples, seeing unlabeled as negative label. The standard classification algorithm we choose is Logistic Regression. The key metrics need to be taken care are test recall and threshold, meaning positive sample recall on testing data set and the corresponding probability boundary. The number of positive and negative samples in testing data set are 34657 and 72464. The evaluation result is shown in Table 3 shows when ratio of positive and unlabeled reaches 1:2 (the number of positive and negative samples are 69331 and 134584 respectively), the threshold doesn’t change significantly when more unlabeled samples are added. Considering both training efficiency and effectiveness, it is practical to set the sampling ratio of positive:negative as 1:2.

3.4.2 Sampling Techniques. For general classification problem, to determine where the class boundary is, at least some of the negative samples to be close to the positive ones are chosen. Take "active learning" [23] as an example, algorithms will select out those samples that are most indistinguishable from the model for human expert to label. However, look-alike models deal with data without

**Table 2: User behaviour Representation**

<table>
<thead>
<tr>
<th>USER_ID</th>
<th>spid1_click</th>
<th>spid1_impression</th>
<th>spid2_click</th>
<th>spid2_impression</th>
<th>...</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>66a7988f</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>9b3fccc94</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>0043fb4f</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>9e664577</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>1df73293</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 2: Comprehensive Audience Expansion Framework**

Based on it, we proceed the audience expansion with intuitive feature extraction and prediction tasks.

The prediction equation of a standard MLP model is defined as:

\[
y = \text{mlp}(AX + \text{bias}), A \in \mathbb{R}^{k \times n}
\]

After adding a scale layer, the model we call Scale-MLP is updated as:

\[
y = \text{mlp}(A(W \circ X) + \text{bias}), A \in \mathbb{R}^{k \times n}
\]

The model expressibility of Equation 2 and 3 is the same so that there is no difference at model prediction stage. That is to say, the effectiveness of actual models obtained from Scale-MLP and MLP are often different on different datasets.

To be detailed, the essential difference lies in the way backpropagation update the network parameter during model training stage. Compared to a standard MLP, Equation 3 reflects that the network need to adjust the parameters when more unlabeled samples are added. When MLP updates the parameter matrix \( A \) during backpropagation, the partial derivative regarding \( a_{ij} \) is \( x_j \); for Scale-MLP, the partial derivative regarding \( a_{ij} \) is \( w_j x_j \) while regarding \( w_j \) is \( x_j \). In another word, the value of feature \( x_j \) in MLP can directly affect the parameters \( a_{ij}, i=1...k \); for Scale-MLP, feature \( x_j \) can only update \( w_j \).

Assuming that the influence of different features on the model is quite different, the fluctuation of feature values will make training process difficult to converge. Suppose that the feature \( x_j \) has little effect for the target, when the values of \( x_j \) drifts, it will cause training difficulty unless the absolute value of parameters \( a_{ij}, i=1...k \) are all small; on the other side, as long as the absolute value of the only affected parameter \( w_j \) in Scale-MLP model is small, the influence of the feature on the target can be made smaller. To conclude, adding the scale layer and updating the parameters of the scale layer during backpropagation can directly change the final influence of each feature on the model.

As another angel to look at the functionality of the new model is that it adds randomness to the original user feature vector. In other words, if a user is not interested by some ad, it doesn’t mean that he/she is totally not interested in that ad. Therefore, the scale layer will help to learn a model which has better generalization capability for this task.

For general classification problem, to determine where the class boundary is, at least some of the negative samples to be close to the positive ones are chosen. Take "active learning" [23] as an example, algorithms will select out those samples that are most indistinguishable from the model for human expert to label. However, look-alike models deal with data without
labelled negative samples, hence the goal of sampling is to pick out a reliable set of negative users.

Besides randomly selecting negative samples and directly apply standard classifier to the PU learning problem, we compare the effectiveness of three other sampling techniques: spy, pre-train and bootstrap sampling. The "Spy" [13] [12] and "Pre-Train" sampling strategies are so-called "two-step" approach [8] where the general idea is described as follows: the first step is to identify a subset of unlabeled samples that can be reliably labelled as negative, then positive and negative samples are used to train a standard classifier to the PU learning problem, we compare the effectiveness of three other sampling techniques: spy, pre-train and bootstrap sampling. The algorithm details are depicted in Algorithm 3. Here we set the number of iterations T and for each iteration, a standard classifier responsible for predicting U is trained on bootstrapped sample set U′ and positive sample set P. The final predicted probability equals to the average score of T iterations.

### Algorithm 3: Bootstrap Sampling

**Input:** Positive Sample Set P, Unlabeled Sample Set U
**Output:** Negative Sample Set N with size k

1. Bootstrap a subset U′ from U;
2. Train a classifier M on P and U′;
3. Predict U − U′ using classifier M;
4. Record the classifying scores;
5. Average the classifying scores of all iterations;
6. Select a subset N of k samples with least average scores;
7. Return N;

Table 4 shows the experimental result of different sampling approaches. The sampling parameter represents the percentage of unlabeled samples picked out as negative and threshold indicates the corresponding probability boundary. From the result table it can be seen that when spy and bootstrap approaches sample half size of the unlabeled data, it still guarantees almost the same level of recall on testing data while regarding pre-train sampling approach, the recall on test data is much lower. On the sampling efficiency, spy approach can only run one iteration compared to the other two which need converge after several rounds. Therefore, it is both efficient and effective to utilize spy sampling approach in our setting.

**Logistic Regression:** Logistic Regression (LR) is probably the most widely used baseline model. Suppose there are n features \{x_1, x_2, ..., x_n\} and x_j is either 0 or 1, consider an LR model without a regularization term:

\[
y = \text{bias} + \beta^T X
\]  

(4)

where \( \beta \) is the coefficient vector. This simple linear model misses the crucial feature crosses, therefore, the Degree-2 Polynomial (Poly2) model is always provided to ease the problem.

\[
y = \text{bias} + \beta^T X + W X^T
\]  

(5)

where W is a symmetric parameter matrix with the elements on the diagonal are all equal to 0.
Table 4: The Impact of Sampling Approach

<table>
<thead>
<tr>
<th>approach</th>
<th>sampling parameter</th>
<th>train loss</th>
<th>test accuracy</th>
<th>test auc</th>
<th>test recall</th>
<th>threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.9</td>
<td>0.4250</td>
<td>0.775</td>
<td>0.847</td>
<td>0.766</td>
<td>0.633</td>
</tr>
<tr>
<td>Random</td>
<td>0.5</td>
<td>0.4150</td>
<td>0.753</td>
<td>0.843</td>
<td>0.676</td>
<td>0.612</td>
</tr>
<tr>
<td>Spy</td>
<td>0.95</td>
<td>0.4138</td>
<td>0.775</td>
<td>0.847</td>
<td>0.767</td>
<td>0.640</td>
</tr>
<tr>
<td>Spy</td>
<td>0.9</td>
<td>0.4141</td>
<td>0.776</td>
<td>0.847</td>
<td>0.763</td>
<td>0.635</td>
</tr>
<tr>
<td>Spy</td>
<td>0.5</td>
<td>0.3660</td>
<td>0.775</td>
<td>0.845</td>
<td>0.768</td>
<td>0.607</td>
</tr>
<tr>
<td>Pre-Train</td>
<td>0.95</td>
<td>0.3677</td>
<td>0.775</td>
<td>0.845</td>
<td>0.771</td>
<td>0.624</td>
</tr>
<tr>
<td>Pre-Train</td>
<td>0.9</td>
<td>0.3829</td>
<td>0.776</td>
<td>0.846</td>
<td>0.771</td>
<td>0.632</td>
</tr>
<tr>
<td>Pre-Train</td>
<td>0.5</td>
<td>0.4382</td>
<td>0.702</td>
<td>0.839</td>
<td>0.632</td>
<td>0.628</td>
</tr>
<tr>
<td>Booststrap</td>
<td>0.95</td>
<td>0.4127</td>
<td>0.775</td>
<td>0.847</td>
<td>0.768</td>
<td>0.638</td>
</tr>
<tr>
<td>Booststrap</td>
<td>0.9</td>
<td>0.4153</td>
<td>0.776</td>
<td>0.847</td>
<td>0.763</td>
<td>0.635</td>
</tr>
<tr>
<td>Booststrap</td>
<td>0.5</td>
<td>0.3976</td>
<td>0.775</td>
<td>0.845</td>
<td>0.766</td>
<td>0.640</td>
</tr>
</tbody>
</table>

Factorization Machine

In order to extract feature crosses while reducing the influence of high-dimensional sparse features, Rendle [22] proposes Factorization Machines to overcome the drawbacks of LR. Regarding LR model, the number of parameters in matrix $W$ need to be learned is $\frac{n(n-1)}{2}$. When $n$ is 100,000, the number of parameters is tens of billions. At the same time, when training the model using gradient descent optimization, the parameter $w_{ij}$ can only be trained when $x_i$ and $x_j$ are both not zero, therefore there is a high demand on both the number of training samples and memory space at training phrase. As a result, for high-dimensional sparse features, the parameter matrix $W$ is almost impossible to train.

To overcome this problem, we will decompose $W$ into $VV^T$ where each $v_i$ in $V = (v_1, v_2, ..., v_n)^T$ can be seen as a latent k-dimensional factor of original feature. The Degree-2 FM model equation is defined as:

$$ y = bias + \beta^T X + XV^TX^T, V \in \mathbb{R}^{n \times k} \quad (6) $$

At this time, the number of parameters need to be estimated is $n \cdot k$ and easier to train even under sparsity setting as FM model break the independence of the interaction parameters by factorizing them.

4 EXPERIMENTS

4.1 Setup

Regarding the model implementation, we use MXNet on a stand-alone 1080TI GPU to compare different model effects and figure out model parameters. When predicting the universal user pool consisting of nearly 2.5 billion users, we used distributed MXNet on a 80-cores hadoop cluster to re-train the model and it took nearly 4 hours to finish the prediction of all users.

4.2 The Impact of Feature Engineering

Table 5 shows the impact of different feature engineering approaches. In this table, Time Slice indicates the strategy of calculating the user behaviour by time slice (None: no time slice; day: slice by day; holiday: slice by holiday and weekday; month: slice by month). For example, if we extract features of user activities by month, one typical feature could be that one specific user is impressed by an ad of “Maybelline” 5 times in July. In general, only activities happening in last three months are to be extracted. Click means whether we distinguish between click action from impression. The experimental results based on LR model (training data volume: 428484; testing data volume: 107121; positive and negative ratio is 1:2) show that if the features are calculated by month and click action is separated from impression, the AUC value will reach 0.8465 in testing phrase which is the best among all settings. Therefore, this feature engineering strategy will be applied in various model methodologies afterwards.

<table>
<thead>
<tr>
<th>Feature Size</th>
<th>Time Slice</th>
<th>Click</th>
<th>Train AUC</th>
<th>Test AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>144009</td>
<td>None</td>
<td>True</td>
<td>0.8721</td>
<td>0.8447</td>
</tr>
<tr>
<td>94932</td>
<td>None</td>
<td>False</td>
<td>0.8689</td>
<td>0.8443</td>
</tr>
<tr>
<td>249406</td>
<td>by holiday</td>
<td>True</td>
<td>0.8808</td>
<td>0.8445</td>
</tr>
<tr>
<td>196184</td>
<td>by month</td>
<td>True</td>
<td>0.8780</td>
<td>0.8465</td>
</tr>
<tr>
<td>133605</td>
<td>by month</td>
<td>False</td>
<td>0.8761</td>
<td>0.8461</td>
</tr>
</tbody>
</table>

4.3 Model Performance

In this section, the performance comparison of various models is introduced. The hyper-parameters configured in different models are listed at Table 6. In this table, BN-MLP is a multi-layer perceptron with a batch normalization layer after each hidden layer; Scale-BN-MLP adds a scale layer before BN-MLP; lr and wd represent learning rate and L2 regularization parameter respectively.

Table 6: Hyper-parameter Setting

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>lr=1e-4, wd=1e-6</td>
</tr>
<tr>
<td>FM</td>
<td>lr=1e-4, wd=3e-5, k=6</td>
</tr>
<tr>
<td>MLP</td>
<td>lr=1e-4, wd=3e-5</td>
</tr>
<tr>
<td>BN-MLP</td>
<td>lr=1e-4, wd=3e-5</td>
</tr>
<tr>
<td>Scale-MLP</td>
<td>lr=1e-4, wd=3e-5</td>
</tr>
<tr>
<td>Scale-BN-MLP</td>
<td>lr=1e-4, wd=3e-5</td>
</tr>
</tbody>
</table>
From the experiment results in Figure 3, we can see that the effect of the multi-layer perceptron is better than that of LR and FM, and adding the batch normalization layer and the scale layer can both improve the model performance and convergence speed of the model. Therefore, Scale-BN-MLP outperforms other models regarding $AUC$ value during training phrase. Meanwhile, the convergence speed of Scale-BN-MLP (4 epochs) is the fastest one among all models, requiring early stopping to yield the optimal model in
practice. The result confirms the derivation in section 3.3. Figure 4 shows different learning rates for Scale-BN-MLP model in training and testing data set, the convergence speed performs well when learning rate equals to 0.0001(1e-4).

4.4 Online Effectiveness Evaluation

Regarding effectiveness evaluation in a real closed-loop business setting, we corporate with a brand advertiser and a third-party advertising monitor supplier in order to conduct the online experiments. The final experiment results are shown at Table 7. There are several important business metrics like Impression UV, Purchaser Rate, ATV (Average Transaction Value), CPO (Cost Per Order), CPA (Cost Per Action) and Incremental ROI listed in this table. All indicators of our model perform far better than traditional demographic-based approaches.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we showed an data application architect to utilize advertisement monitor data in audience expansion system for brand advertisers, compared to traditional ad serving based on demographics, the lookalike model in our application focuses on analysing user behaviour. Regarding the way of picking up the negative samples from unlabeled data, we compared four sampling techniques and the impact of different sampling ratios in order to figure out the best setting. Meanwhile, to overcome the sparsity and high dimension of feature space, we proposed Scale-MLP, a modified MLP by adding a scale layer, although the training AUC is lower than other traditional learning strategies, however, it gains performance improvement when generalizing the model to testing data while the efficiency of Scale-MLP is comparable to other approaches. Lastly we prove that the lookalike model outperforms traditional ad serving mechanisms in real business environment.

Several directions exist for future research. The rich information contained in the advertisement could be harnessed to investigate more sophisticated look-alike models. For example, we could incorporate advertising information including advertiser, brand and product in order to explore more detailed feature interactions. For different advertisers’ campaign, adaptive user feature representation also need to be taken into consideration. Meanwhile, CTR prediction task will be a challenging and interesting problem under the setting of growing diversity in targeting users and cross-media advertising platforms. CTR prediction results could be utilized for the purpose of omni-channel uniform budget allocation to effectively enhance ROI by matching brands/products with different media platforms.

REFERENCES


[26] JMLR Linear SVM.


