Life-stage Prediction for Product Recommendation in E-commerce

Team No: 4

Team Members:

Xubin Zhuge (xzhuge)

Yuqi Zhang (yuqi2)

Rui Chang (rchang)

School of Computing

Clemson University

April 19th, 2016

1. Background

Recommender systems have achieved much commercial success and is one of the most important data mining applications. For example, online stores such as Taobao, Amazon, and Walmart.com provide customized recommendations for additional products or services based on a user's historical purchasing records. It's well known that these recommendations are extremely important and have great impact on consumers' online shopping experience. Thus scholars and industry researchers in the data mining community have spent much effort on further improving recommendation performance, exploring all possible approaches.

On the other hand, marketing researchers and sociologists have recognized the importance of life stages on consumer's purchasing behaviors for many years. In our E-commerce system, we also found strong correlation between life cycle stages and consumer purchasing behavior. For example, a woman will buy maternal vitamin during the pregnancy stage, then buy a baby car when her baby is born. A mum will have different purchasing needs during different life stages. Therefore, explicitly modeling and predicting life stages might be very useful for predicting the purchasing timed of products. We expect making recommendations based on this additional information could further improve the recommendation effectiveness. Unfortunately, most of the existing methods do not consider the concept of life stages, which exists in many verticals in E-commerce.

2. Introduction

Although marketing researchers and sociologists have recognized the large impact of life stage on consumer's purchasing behaviors, existing recommender systems have not taken this impact into consideration. Motivated by this, in our project, we introduce the concept of life-stage into E-commerce recommendation systems. The system first predicts the current life stage of a user, then recommends products to the user based on the prediction and our proposed probabilistic recommendation model, which can explicitly incorporate the life-stage information into an E-commerce recommender system. Particularly, in our project, we only focus on Mom-Baby domain.

In our system, there are three major sub tasks. First, we extract features from user's purchasing behavior. Second, we label each user's behavior sequences so that we can tell a user's current life stage. Third, we generate recommendations based on the predicted life-stage. Thus the key components of life-stage based recommendations include how to model and predict life stages and how to give appropriate product recommendation correspondingly.

To process features, we calculate weight of features using TF-IDF model, and we generate feature matrix using training data. More specifically, we utilize the Naïve Bayes model as classifier to predict the label and associated probability of each behavior sequence, and segment the whole sequence. To make recommendation for user based one the predicted life stage, we rank the joint probability of a user

purchasing a product at a specific age, then recommend the products of highest probability.

To evaluate the effectiveness of the proposed life-stage based approach, we conduct extensive experiments. The experimental results on a large dataset demonstrate that the proposed approach can significantly improve the performance in terms of prediction accuracy. The dataset we use can be download from the following website: http://tianchi.aliyun.com/datalab/dataSet.htm?id=3.

3. Feature processing

3.1 Mom-baby domain

Life-stage of a user is a predefined set of phases in life, and each life stage spans a period of time. Changes in life stages may lead to changes in a person's purchasing needs. Examples of life-stage in maternal and child domain include pregnancy, new born- first year, 1 - 2 years old, etc. In our experiments, we mainly focus on the life-stage based recommendation in the mum-baby domain. The candidate life-stage labels/states are shown on Table 1.

In the mom-baby product category, the transition and duration of life stages for a child are deterministic. We only need to find the birthday date, then the whole sequence can be labeled by a baby life stage template with a fixed sequence of life stages and fixed length in each stage. Since most of our mum-baby consumers only have one child per family (due to the one child policy in our country), we assume all our training data are from one child families.

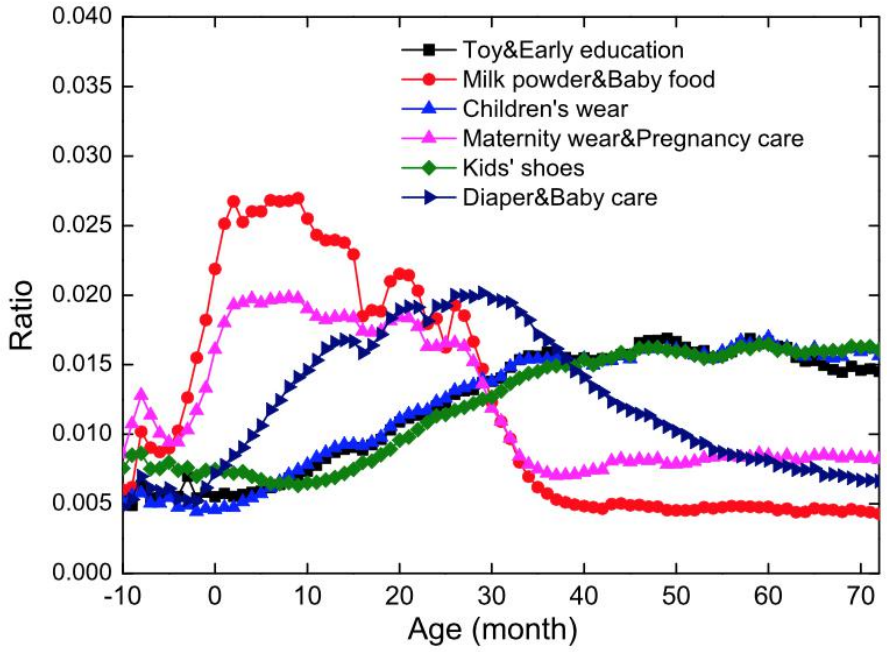
Table 3.1: Life-stage labels for Mom-baby domain

Label	Baby Age Category(life-stage)
0	Pregnant
1	New born – first year
2	1-2 years
3	2-3 years
4	3-4 years

3.2 Features Extraction

We convert each behavior subsequence into a feature vector for training the Naïve Bayes model. In our project, we use two kinds of features:

(1) Category features: Many user actions are related to some items. Given a behavior sequence, we obtain a set of product IDs and could use a bag of IDs as features. All items in Taobao are mapped to a category hierarchy. We expect consumer purchasing behavior patterns at different stages are more obvious on the categorical level. For example, a pregnant woman is more likely to browse or buy products in 'Maternity Cloths' and 'Pregnant Bras' categories. A new parent is likely to browse 'Baby stroller' category. Following figure shows how purchasing ratio of top level categories relates to baby age. We utilize all parent categories and first level categories an item belongs to as features.



(2) Product property features: Taobao is a distributed market space where sellers sell products. Many sellers pro- vide meta data about the products. For example, a seller will label 'Size' as "M" or "L" on children clothes, or label 'Age' as "newborn" or "1-3 years" etc. Each product facet- value pair (i.e. a product property) is associated with a unique feature.

There are many other features such as Queries which stand for users' search activities and product title features which are created by sellers who are not affiliated with Taobao. They also include a lot of information. But considering categories and product property features are the main features which implicit the users' future purchasing possibility, and to reduce the total calculation complexity, we just use these two kind of features. The experiment shows that the results are as good as we predicted.

3.3 TF-IDF model

In order to reduce the influence of popular features, the feature weight is set based TF-IDF model. In information retrieval, TF–IDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining. The TF-IDF value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

TF: Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (the total number of terms in the document) as a way of normalization:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

IDF: Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However, it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

IDF(t) = log10(Total number of documents / Number of documents with term t in it).

In our project, each user stands for a document, each feature stands for a term. For each feature, we can calculate the weight based on the following formula:

$$w_{t,d} = \log(1 + tf_{t,d}) \times \log_{10}(N / df_t)$$

3.4 Generating feature vector and matrix

Considering the temporal effect of features. Whether a user has purchased a diaper 1 month ago or 2 years ago has deferent meanings in terms of predicting the current life stage. To capture the temporal patterns, we divide each consumer behavior time sequence into multiple subsequences with fixed size time windows. A feature vector for each time window is generated.

There are 3 steps to generate a feature vector:

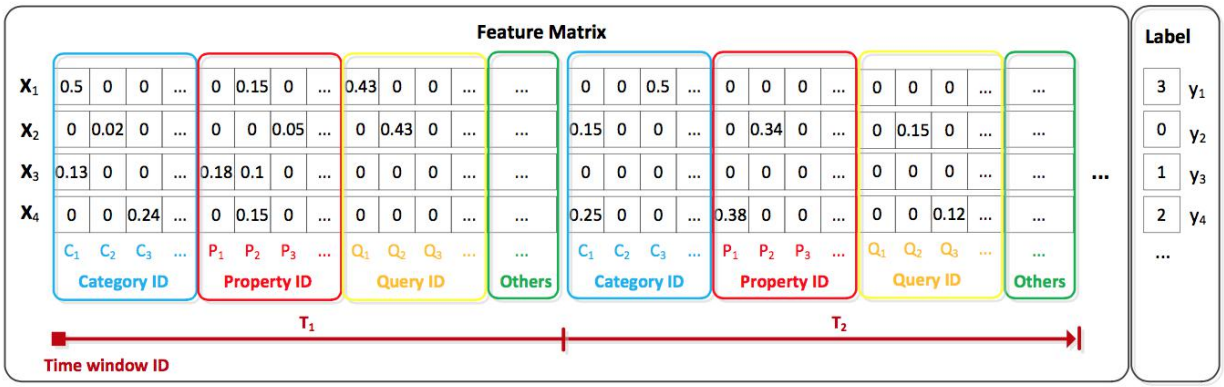
Step1: Choose a time window size. Count the times of each feature appearing in a user's time window

Step2: Calculate weight for each feature utilizing TF-IDF model.

Step3: Normalize feature vector into [0,1].

Once we have feature vectors, we concatenate all the feature vectors into a big feature matrix to present all the activities of all users. following picture illustrates an example of feature matrix, where each row corresponds to a training data point which means a user's information, and each column stands for a feature.

In our training data, we know the baby's birthday. So, we can set label for each user which means each feature vector directly. But for those users who we don't know their baby's birthday information, we utilize this feature matrix to predict their life stage.



4. Bayesian Classification

4.1 Project Analysis

Now we have the feature vector, containing the category id, property id in different life stage of same user. In order to do the product recommendation, we need to have user's age information. However, some user didn't provide their children's birthday, so we need to predict or evaluate a reasonable age label for the client so that we can continue our following work.

We can divide the age into several life stage, each stage has a corresponding label, so we can transform this problem to classification problem. For the classification problem, basically we have data samples, and we need feed this sample to the classifier, training the classifier and when it comes a new test sample we could predict the label using our trained classifier.

In our recommendation system, the training sample is the user's transaction record with the baby's age label, and by feature extract we have talked above, we could get a vector (X1,X2….,Xn) and a label Y for each user entry. Now we need to use this training samples to train a classifier which can predict a good evaluation on the age label if coming entry don't have the age label.

There are lots of classifier can do this problem, such as Bayesian Classification SVM, ANN. Finally, we decide to use Bayesian Classifier because it is Super simple, you're just doing a bunch of counts. If the NB conditional independence assumption actually holds, a Naive Bayes classifier will converge quicker than discriminative models like logistic regression, so you need less training data. And even if the NB assumption doesn't hold, a NB classifier still often does a great job in practice. A good bet if want something fast and easy that performs pretty well. Its main disadvantage is that it can't learn interactions between features (e.g., it can't learn that although you love movies with Brad Pitt and Tom Cruise, you hate movies where they're together). It is really fantastic and appropriate for this problem.

4.2 Bayes rules

Assuming (X1,X2….,Xn) is our feature vector which is extracted from user's purchasing record, and Y represent the life stage label. We can use the Bayes rules to predict the life stage using the Y=y with the maximum probability assuming we know the X1=x1,X2=x2….,Xn=xn, which (x1,x2,..xn) is extract from the property of test sample.

$$P(Y|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y)P(Y)}{P(X_1,\ldots,X_n)}$$

4.3 Calculation of the Conditional probability

In this part, we are talking about how to calculate the conditional probability P(X1,X2,..,Xn|Y) .

4.3.1 By Using Similarity

1) Algorithm

Assuming (X1,X2….,Xn) is our feature vector which is extracted from user's purchasing record, and Y represent the life stage label.

- Step 1: Firstly, we need to set the one threshold t1,the function of the threshold is if the cosine similarity value is greater that this value, we could view it as the same entry and add the count by 1.

- Step 2: For $P(X1,X2,\ldots,Xn|Y=y_j)$, we calculate cosine similarity between test feature and all sample feature with label equal yj and count the number that bigger than t1,saying count(yj).

- Step 3:Estimate $P(X1,X2,\ldots,Xn|Y=y_j)$, using $count(y_j)/total(y_j)$,which $total(y_j)$ is the count number of entry with the lable = y_j.

- Step 4:predict the lable with the yj which meet the
  MAX ($P(X1,X2,\ldots,Xn|Y=y_j)*P(Y=y_j)$)

2) Result using different threshold

Here shows our Accuracy by using different Threshold, We use crossing verification to evaluate the accuracy, and by compare the life stage label to calculate the error.

Table 4.1: Accuracy for different threshold

Threshold	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
Accuracy	0.34	0.37	0.40	0.56	0.70	0.77	0.83	0.81	0.72

From the result table showing above, we know that, the accuracy become maximum when threshold equals 0.85.

3) Pros and Cons

To our life stage predict module, we have analyze the pros and cons by using similarity method.

- Pros:

Transform the multi-variable conditional probability to the cosine similarity between 2 vector, it is easy to calculate and do not need to fancy with small decimal and the influence of zero probability.

- Cons:

To every test data we need to train again, cannot store the previous training result. if test sample input in a batch mode, it will cost a lot of time compute the cosine similarity.

4.3.2 Naive Bayesian Theory

To overcome the efficiency problem of similarity method state above, here show another method, which we known as Naive Bayesian Theory

1) Naïve Bayes Assumption

- all features are independent given the class label Y;
- Naïve Bayes assumption is almost never true;
- The accuracy of the assumption depends on feature selection, we need select the features that are independent as much as possible;
- Naïve Bayes often performs surprisingly well even when its assumptions do not hold.

2) Algorithm Contents

Assuming (F1,F2,..,Fn) is our feature vector which is extracted from user's purchasing record, and Y represent the life stage label.

A naive Bayes classifier models a joint distribution over a label Y and a set of observed random variables, or features, using the assumption that the full joint distribution can be factored as follows:

$$P(F_1 \ldots F_n, Y) = P(Y) \prod_i P(F_i|Y)$$

To classify a datum, we can find the most probable class given the feature values for each pixel:

$$\begin{aligned} P(y \mid f_1, \ldots, f_m) &= \frac{P(f_1, \ldots, f_m \mid y) P(y)}{P(f_1, \ldots, f_m)} \\ &= \frac{P(y) \prod_{i=1}^{m} P(f_i \mid y)}{P(f_1, \ldots, f_m)} \\ \arg\max_y P(y \mid f_1, \ldots, f_m) &= \arg\max_y \frac{P(y) \prod_{i=1}^{m} P(f_i \mid y)}{P(f_1, \ldots, f_m)} \\ &= \arg\max_y P(y) \prod_{i=1}^{m} P(f_i \mid y) \end{aligned}$$

Which (f1,f2,..,fn) is the feature we extract from the test file.

Because multiplying many probabilities together often results in underflow, we will instead compute log probability which will have the same argument max:

$$\arg\max_y log(P(y \mid f_1, \ldots, f_m) = \arg\max_y (log(P(y)) + \sum_{i=1}^{m} log(P(f_i \mid y)))$$

So we choose maximum target value as our predict value to life stage label.

3) Parameter Estimation

Our naive Bayes model has several parameters to estimate. One parameter is the prior distribution over labels (digits) p(y), We can estimate p(y) directly from the

training data:

$$\hat{P}(y) = \frac{c(y)}{n}$$

where c(y) is the number of training instances with label y and n is the total number of training instances.Thus we can calculate the conditional probability by using:

$$\hat{P}(F_i = f_i|Y = y) = \frac{c(f_i, y)}{\sum_{f_i} c(f_i, y)}$$

4) Smoothing

Your current parameter estimates are unsmoothed, that is, you are using the empirical estimates for the parameters. These estimates are rarely adequate in real systems. Minimally, we need to make sure that no parameter ever receives an estimate of zero, but good smoothing can boost accuracy quite a bit by reducing overfitting.

The basic smoothing method we'll use here is Laplace Smoothing which essentially adds k counts to every possible observation value:

$$P(F_i = f_i|Y = y) = \frac{c(F_i=f_i,Y=y)+k}{\sum_{f_i} c(F_i=f_i,Y=y)+k}$$

If k=0 the probabilities are unsmoothed, ask grows larger the probabilities are smoothed more and more. You can use your validation set to determine a good value for k (note: you don't have to smooth P(C)).

5) Numerical Stability

$$\begin{aligned}
\log\left(P(Y|X_1,\ldots,X_n)\right) &= \log\left(\frac{P(X_1,\ldots,X_n|Y)\cdot P(Y)}{P(X_1,\ldots,X_n)}\right) \\
&= \text{constant} + \log\left(\prod_{i=1}^{n} P(X_i|Y)\right) + \log P(Y) \\
&= \text{constant} + \sum_{i=1}^{n} \log P(X_i|Y) + \log P(Y)
\end{aligned}$$

4.4 Final result using Bayesian classification

We show the result of using tow method to calculate the conditional probability. According to the result, we know that by using Naïve Bayes, we can achieve the higher accuracy with the reasonable training and test time.

Table 4.2: Final result

Bayes	Naïve Bayes	Similarity (T=o.8)
Trainning Time(s)	1.5	1.4
Test Time(s)	1.7	12572
Accuracy	0.86	0.83

5. Recommendation Based on Life-stage

To make recommendation, we first studied some popular recommendation approaches. According to our study, there are mainly two kinds of recommendation systems, one is content-based systems, the other is collaborative filtering systems. However, our approach is a probability-based recommendation due to the special case of our problem.

5.1 Content-based systems

Content-based filtering[3] methods are based on a description of the item and a profile of the user's preference. In a content-based recommender system, keywords are used to describe the items and a user profile is built to indicate the type of item this user likes. In other words, these algorithms try to recommend items that are similar to those that a user liked in the past (or is examining in the present). In particular, various candidate items are compared with items previously rated by the user and the best-matching items are recommended.

5.2 Collaborative filtering systems

Collaborative filtering[4] methods are based on collecting and analyzing a large amount of information on users' behaviors, activities or preferences and predicting what users will like based on their similarity to other users. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself. Many algorithms have been used in measuring user similarity or item similarity in recommender systems. Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past.

5.3 Our recommendation method

Our recommendation is more like the collaborative filtering approach, and the specialty for our problem is that when we get the life-stage, we can recommend products just based on that life-stage, and find the most likely products user will buy at that life-stage to make the recommendation. Because our recommendation is based on life-stage, we propose a probability model to estimate the probability of a user purchasing a product at a specific age a for every product.

$$\mathbf{P(p\_product_j, a) = P(a \mid p\_product_j)*P(p\_product_j)}$$

This joint probability can be easily calculated using chain rule based on the statistics of our training data.

$P(p\_product_j)$ is the probability of user purchasing the product j, and we calculate it by using the total counts of purchasing behavior divided by the total counts of purchasing behavior with product j.

$$\mathbf{P(p\_product_j)} = \frac{TotalCount(p\_product_j)}{TotalCount(product)}$$

$P(a \mid p\_product_j)$ is the conditional probability of making the purchase at age a given that the user will purchase product j.

$$\mathbf{P(a \mid p\_product_j)} = \frac{TotalCount(age = a \mid buy\_product_j)}{TotalCount(buy\_product_j)}$$

After calculating the probability of all the products, we can rank all candidate products and choose the products with highest probabiliy to make the recommendation.

5.4 Comparison with the method in the paper

In the paper, for a user u without the age information, they first estimate the age distribution $p_u(a)$ based on the Gaussian mixture model, which is introduced in Section 6. Then rank the products based on:

$$P(p\_product_j)\int p(a \mid p\_product_j)p_u(a)da$$

Where $P(a \mid p\_product_j)$ is the probability of the age of a random user purchasing product j, which they assume follows a Gaussian distribution $N(\mu_j, \sigma_j)$. And they estimate the $P(p\_product_j)$ using a large scale logistic regression model:

$$P(p\_product_j) = \frac{1}{1 + e^{-W^T X}}$$

We think the approach proposed in the paper is not very clear and the computation cost is huge because they should train the logistic regression model. Our method using Bayesian classification to label the user without age information, then all the user can be treated as the same to make a fast recommendation.

6. Gaussian Mixture Model for Multi-kids

6.1 Motivation of Mixture Model

The former analysis is all based on the assumption that each user only has one child corresponding to his or her account. In the paper we focused, a mixture model is proposed to tackle that scenario to meet the facts that more and more people in China

would have more than one child due to the change of one child policy. This motivates us to work on multi-kids life stage modeling and prediction.

Because the dataset we got for users does not contain multiple babies information. Each account just has one birthday information. So we cannot just use the model we proposed above to solve this scenario. However, the paper refers to a Gaussian Mixture Model for this problem.

6.2 Gaussian Mixture Model

A mixture model assumes that a set of observed objects is a mixture of instances from multiple probabilistic clusters, and conceptually each observed object is generated independently.

Assume the probability density function of each cluster follows a 1-d Gaussian distribution. Suppose that there are k clusters.

The probability density function of each cluster are centered at μ_j with standard deviation σ_j, $\theta_j=(\mu_j, \sigma_j)$, we have

$$P(o_i|\Theta_j) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}} \qquad P(o_i|\Theta) = \sum_{j=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}}$$

$$P(\mathbf{O}|\Theta) = \prod_{i=1}^{n}\sum_{j=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}}$$

6.3 The Model of Multi-kids Scenario

For a given user account, we observe a sequence of product purchasing behaviors. For each product j purchased by a user at time tj, we estimate the expected purchasing age of the consumer $a_j = E(a \mid P(age \mid p\_product_j))$ at time t_j based on the age information of all consumers who have purchased the product. Then we can map the expected purchasing age to the projected age of the user at time t by adding the time difference between t and t_j: $a_{j,t} = a_j + (t - t_j)$.

The Gaussian Mixture Model (GMM) is described by this:

$$p(a_{j,t}) = \sum_{c=1}^{k} w_c N(a_{j,t} : \mu_c, \sigma_c^2)$$

where c ∈ [1, K] is the index for each child, K is the total number of children, μ_c and σ_c are the mean and variance of c-th Gaussian component respectively, and w_c is the mixture weight for the component. Given a fixed K and the set of projected age of the user, the model parameter w_c , μ_c , σ_c can be found by the maximum likelihood estimation which can be referred from the PPT of chapter 11 offered by DR. Wang. EM algorithm can be used to calculate the best parameter w_c , μ_c , σ_c . To find the

optimal K, Akaike information criterion (AIC)[1] and Bayesian information criterion (BIC) [1] can be used to estimate the best K. Based on this model, the age of the cth child at time t follows a Gaussian distribute centered on μ_c with variance σ_c.

7. Experiments and Conclusion

7.1 Setup for Experiments

We use a dataset with 10 thousands children birth date information. And 5-fold cross validation is used to test the accuracy of our result. We randomly partition our original dataset into 5 equal subsamples. Of the 5 subsamples, a single subsample is retained as the validation data for testing the model, and the remaining 4 subsamples are used as training data. The cross-validation process is then repeated 5 times, with each of the 5 subsamples used exactly once as the validation data. The 5 results fro the folds can then be averaged to produce a single estimation.

7.2 Effects of Features and Approaches

We first test the effects of different features for child's life-stage prediction. From table 5.1, the result is comparatively better when we use more features than just using the basic category feature for our model. We think that is because property features contain more detailed information of a product.

Our implementation is based on dividing different time window for a user's purchasing behaviors. Referring the original paper, we also test the effects with and without time window. From table 5.2, we can see that the classification accuracy is much worse in the case without time window. That illustrates the huge impact of purchasing behavior in certain time to the life-stage prediction.

Table 7.1: Effects of features for child life-stage prediction

Features	Accuracy
Basic(category)	74.3%
Basic+Property	87.6%

Table 7.2: Effects of approaches for child life-stage prediction

Approach	Accuracy
Basic+Property (without time window)	58.3%
Basic+Property (with time window)	87.6%

7.3 Analysis of Temporal Effect

Temporal patterns play very significant roles in age stage prediction. As we mentioned above, we divide each consumer behavior time sequence into multiple time windows. We try to find the best time window size and window number for our model.

Figure 5.1 shows the detailed result of the effect of time window size to the performance.

As we increase the window size, the prediction accuracy first increase significantly, then does not change very much, at last decrease. The optimal accuracy is located around 40 to 80 days, and we choose the middle of them 60 as our best window size.

From our analysis, when the time window is too small, the activities in a window are very limited, it is impossible to get a good result. However, when the window size is too big, as we can assume just one window, then the power of temporal features is too weak for the prediction which we can see from above analysis on effects of approaches for life-stage prediction.

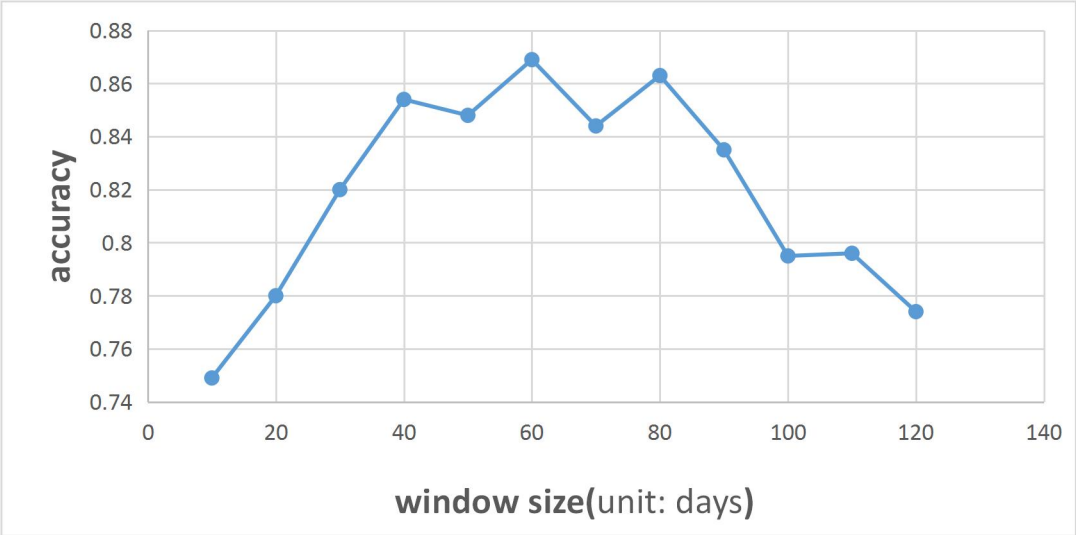


Figure 7.1: The analysis of window size

Given a fixed window size 60, we also test the effects of window number to accuracy. We adjust the window number from 1 to 15 to say how it affects the performance.

As the window number increase, the accuracy increase accordingly before hitting a plateau. This indicates a longer purchasing behavior history gives more accurate life-stage prediction.

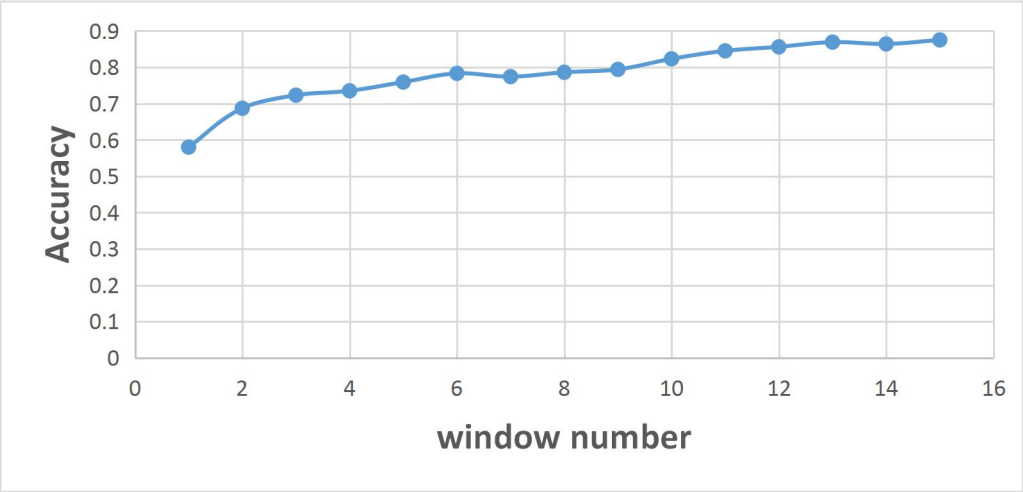


Figure 7.2: The analysis of window number

8. Future Work

Firstly, for improving the accuracy of our model, we will train our model with a larger dataset. Currently we just use 10 thousands user's birthday information for training and testing which may have not cover the cases of real purchasing process. Secondly, we will study how to establish a Guassian Mixture Model for multi-kids scenario which is proposed in the paper but without a thorough description. We can also apply on our model to other categories such as wedding and home remodeling.

9. References

[1] Peng Jiang, Yadong Zhu, Yi Zhang, Quan Yuan. Life-stage Prediction for Product Recommendation in E-commerce. *In Proceedings of the 21th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD'15:1879–1888, 2015. ACM.

[2] H. Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on,* 19(6):716–723, Dec 1974.

[3] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.

[4] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE,* 7(1):76–80, Jan 2003.

[5] Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. *In Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 591–598, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

Appendix

Implementation

We use C++ to implement the system , the source code can be accessed at github repository : https://github.com/yuqi0822/Recommend-System.git

Here shows some key codes：

Predict_Similarity：

```
void predict_Similarity(){
        unordered_map<string,Line*> matrix_test =    transformFeatureVector();
        unordered_map<string,Line*> matrix_sample = f->getMatrix();
        int lableNum = util.getLabelNum();
        string head = "UsereId,Label";
        fIO.setNameW(resultFile);
        fIO.writeLine(head);                //write the head of the file
        for(auto line:matrix_test){
            stringstream str("");
            str<<line.first<<",";
            vector<double> f = (line.second)->getFeature();
            int maxlable = 0;
            double max = 0;
            for(int l=0;l<lableNum;l++){
                map<int,double> result(probLable.begin(),probLable.end());
                int total = lableMatrix[l].size();
                int count = 0;
                for(auto line:lableMatrix[l]){
                    if(isSimilarity(f,(line.second)->getFeature())){
                        count++;
                    }
                }
                double pcon = (double)count / total;
                result[l] *= pcon;
                if(max<result[l]) {max=result[l];maxlable = l;}
            }
            str<<maxlable;
            fIO.writeLine(str.str());
        }
    }
```

Predict_NaiveBayes：

```
void predict_NaiveBayes() {
        unordered_map<string,Line*> matrix_test = transformFeatureVector();
        unordered_map<string,Line*> matrix_sample = f->getMatrix();
        int lableNum = util.getLabelNum();
        int featureNum = util.getFeatureDim();
        string head = "UsereId,Label";
        fIO.setNameW(resultFile);
        fIO.writeLine(head); //write the head of the file
        for(auto line:matrix_test) {
            stringstream str("");
            str<<line.first<<",";
            vector<double> f = (line.second)->getFeature();
            int maxlable = 0;
            double max = 0;
            for(int l=0;l<lableNum;l++) {
                map<int,double> result(probLable.begin(),probLable.end());
                //for(int l=0;l<lableNum;l++){cout<<result[l]<<endl;}
                for(int j=0;j<featureNum;j++){
                    result[l] *= log2(2+condProb[l][j]*f[j]);
                }
                if(max<result[l]){max=result[l];maxlable = l;}
            }
            str<<maxlable;
            fIO.writeLine(str.str());
        }
}
```