Life-Stage Modeling by Customer-Manifold Embedding*

Jing-Wen Yang†, Yang Yu†, Xiao-Peng Zhang‡
† National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
† Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China
‡ Tencent, China
† {yangjw, yuy}@lamda.nju.edu.cn
‡ xpzhang@tencent.com

Abstract

A person experiences different stages throughout the life, causing dramatically varying behavior patterns. In applications such as online-shopping, it has been observed that customer behaviors are largely affected by their stages and are evolving over time. Although this phenomena has been recognized previously, very few studies tried to model the life-stage and make use of it. In this paper, we propose to discover a latent space, called *customer-manifold*, on which a position corresponds to a customer stage. The customer-manifold allows us to train a static prediction model that captures dynamic customer behavior patterns. We further embed the learned customer-manifold into a neural network model as a hidden layer output, resulting in an efficient and accurate customer behavior prediction system. We apply this system to online-shopping recommendation. Experiments in real world data show that taking customer-manifold into account can improve the performance of the recommender system. Moreover, visualization of the customer-manifold space may also be helpful to understand the evolutionary customer behaviors.

1 Introduction

There are various stages through everyone's life. The focuses, concepts and consumption ability of a person can be largely influenced by his/her stages. Some marketing researchers and sociologists have recognized that life stages would have a huge impact on customers' purchasing behaviors [Wells and Gubar, 1966] [Bojanic, 2011]. For example, a woman would buy vitamins during her pregnancy stage, baby food and diapers after baby's birth, and then toys and fairy tale books as baby grows. Obviously, considering life-stage would help us understand customers better, and can help build better service systems such as recommender systems.

Recommender systems aim at helping customers discover most useful information from a massive amount of data. They have been widely adopted in areas such as social networks, entertainment, and e-commerce recent years. Researchers and engineers are always pursuing a better personalized system for great value of both research and business. Most existing recommender systems are based on, e.g., collaborative filtering techniques, but do not take the evolutionary stage information of customers into account.

A few recent studies have tried to improve recommender systems by utilizing the information from the evolutionary custom behaviors. Extended collaborative filtering model [Ding and Li, 2005] [Liu *et al.*, 2010], matrix factorization [Koren, 2010] [Xiong *et al.*, 2010], hidden Markov model [Li *et al.*, 2011], and Gaussian processes [Liu, 2015] were studied for incorporating temporal effects in the models. The time-based functions incorporated have been shown to be able to capture some temporal patterns, but they mostly operate in the customer behavior space where it would be too many different temporal patterns to be well captured.

As an important matter of fact, people in different stages may behave similarly in a current time slice, but have dramatically different future behaviors. For example, one is moving to a new house and another is planing to redecorate his old apartment. They both buy paints, brushes and wallpapers. At this time point, they behave almost the same. But for the few next weeks, the first one purchases furnitures, while the latter one buys a potted plants. Therefore, an approach processing in the customer behavior space would get confused in this situation, meanwhile, if an approach can understand the customer stage, it would distinguish their behaviors easily and make correct recommendations. From the view of this aspect, modeling customer life-stage can be helpful to eliminate the complexity of modeling evolution dynamics. With the information of life-stage, we could have a deeper understanding of customer behavior, thus helping us capture various modes and make the evolution trend predictable.

In this paper, we propose to model customer stages by learning the *customer-manifold* space. The intuition is that the stage of a customer can be reflected by his behavior history. Therefore, we collect customers' daily record to build a novel similarity matrix and employ manifold learning to embed those sequences into a metric space, which we call as the customer-manifold space. Within this space, similar stages are kept aligned, a stage can now be represented by a point and the stage evolution of a customer can be captured

*This research was supported by the NSFC (61375061), JiangsuSF (BK20160066), Foundation for the Author of National Excellent Doctoral Dissertation of China (201451).

as the changing path of the embedding positions. Once a customer's stage is mapping to a point on the manifold, his future behaviors could be predicted according to the path changing patterns. Powerful learners can be applied here to capture the complicated evolution patterns, by learning only a static model in the new space.

To build an efficient and accurate customer modeling system, we embed the customer-manifold into a recurrent neural network model as the hidden layer output. Upon this system, we further introduce a classifier which maps from a customer-manifold position to an item to build a recommender system.

To validate the proposed method, we conduct experiments in a real-world data. Experimental results show that the proposed method can achieve significantly better performance than baseline recommendation approaches. Since the proposed method can also provide a way to visualize the customer behaviors in the system, we demonstrate the manifold space in both 2D and 3D figures to visualize some interesting embedded patterns.

To summarize, our main contributions are as follows:

- We propose a simple and general method to learn the static customer-manifold space, which reflects customer stages and can be applied to various areas where customer modeling is in need.
- We embed the customer-manifold into neural networks to build a recommender system that learns to predict the customer behaviors efficiently.
- Experiments on real world data show that capturing life-stage information brings a significant improvement to recommendation precision and some visualization result can be demonstrated.

The rest of the paper is organized as follows. Section 2 introduces the related work, section 3 proposes the method of learning customer-manifold and the recommendation approach using the customer-manifold, section 4 reports the experiment results, and section 5 concludes this paper.

2 Related Work

Since we propose to model customer stages and apply the modeling system to make recommendation. We briefly review some related user modeling and recommender system studies below.

Some user modeling methods try to capture customers' sequential preference. Le *et al.* [2016] incorporated dynamic user-biased emission and context-biased transition for modeling sequential preference. Ferwerda *et al.* [2016] mined the relationships between users' personality and their behaviors, preferences, and needs from user-generated data of social network to infer users' personality traits.

In recommender systems, collaborative filtering (CF) is a classical and arguably the most widely adopted technique. The main idea is that users with similar preferences will have similar opinions on new items. User-Based CF [Resnick *et al.*, 1994] and Item-Based CF [Sarwar *et al.*, 2001] are two well-known collaborative algorithms.

These existing systems usually implicitly assume that the preference of customer is static. It's a reasonable approximation to ignore temporal information for movie ratings like Netflix contest, since the preference for a movie do not change rapidly and dramatically. However, in areas like online shopping, the approximation would lead a relatively large bias for the ignoring of evolution of customer interest.

A few recent studies shared the same observation of our work, which tries to take the evolution of customer behaviors into account. Ding *et al.* [2005] and Liu *et al.* [2010] employed time decay functions to adapt to the changed behaviors quickly. Lathia *et al.* [2009] proposed a method to automatically adjust the neighborhood size of customers based on the updated accuracy, in the framework of user-based CF. Koren *et al.* [2010] tackled temporal effects by using time-related functions to model the customer biases, item biases, and customers' latent factors through matrix factorization methods. Xiong *et al.* [2010] extended the probabilistic matrix factorization method to tensor factorization, where the tensor has a time dimension, with the time factor being specially treated. Liu [2015] treated the evolution of the user's preference as time series which are learned and inferred using GP regression. Liu *et al.* [2014] used the temporal correlations to construct an undirected graph and performed clustering to re-encode the sequence symbol in a better granularity which greatly alleviates the curse of cardinality in tasks of sequential pattern mining. These approaches try to introduce time dimension so that the model can change along time, and have been shown to be able to capture some temporal behavior patterns. But they operated in the customer behavior space, where the temporal patterns would be noising and complex to be captured by time-based models. While we first learn a latent stage space to avoid operating in behavior space directly.

Complex models like deep learning has also received many attentions for recommender systems. However, deep learning methods for temporal recommendation has not yet been extensively studied. Hidas *et al.* [2015] proposed to use Recurrent Neural Networks (RNN) for recommending shopping items to user based on the user's current session history (the framework shows in Figure 1). Powerful as RNN is, customers'

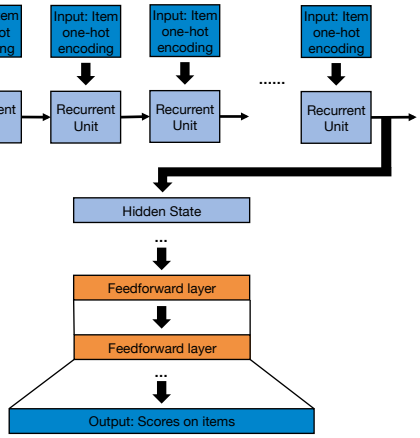

Figure 1: Framework of the RNN for recommendation

stage space is not implicitly modeled, which may affect the performance.

Perhaps the most realted work is [Jiang *et al.*, 2015]. They first noticed to model the life-stage. They applied a maximum entropy semi-Markov model to segment and label user life stages based on the observed purchasing data over time, which makes recommendations conditioned to the predicted life-stage. Although the idea and major models can be applied to various events, domain experts are badly relied on to label stages and handcraft features. While our method can only rely on the history behavior data of customers, and does not restrict the prediction model.

3 Life-Stage Modeling by Customer-Manifold

This section introduces the proposed method for learning the *customer-manifold* (CM), which embeds the life-stages of customers. The proposed method mainly consists of two components. The first one is constructing CM, which first splits the customer behavior sequences into evolutionary segments, calculates similarity between segments, and then learns the manifold (as illustrated in Figure 2). The second one as Figure 3 shows is the LSTM embedding, which embeds the CM into a recurrent neural network to build a more accurate and efficient system. The following subsections introduce the method in details.

3.1 Learning Customer-Manifold

Usually, a customer record is in the format of <customer id, item id, time stamp> triple, which is the fundamental element to keep track of customer's behaviors. In other words, a series of these triples form a customer behavior sequence just as in Figure 2 Part A, Customer 1 clicked items `[a,b,c,a,e,f,d,o,q,a,b,w,x,...]` in sequence. Though the data seems quite simple, it contains important information as we assume that behavior data may reflect stages of a customer. We will show how the life-stage is modeled only based on these data.

At first step, we process with the behavior sequence to extract so called evolutionary segments with the purpose that some of them may reflect the stage of customers. We simply split the sequence by day for the intuition that one's stage is unlikely to change suddenly in one day (as Figure 2 Part B).

In order to make segments from the same customer evolve along a path, thus reflecting how his/her stage evolves, we introduce *evolutionary similarity* to calculate similarity between every two segments. Notice that each segment's length is variable, simple Euclidean distance measure is very brittle [Chu *et al.*, 2002], because that an elastic shifting in time axis may be necessary. Dynamic time warping (DTW) distance [Berndt and Clifford, 1994] is widely used in scenario like this. We first recall the DTW distance here.

DTW distance is defined using dynamic programming to evaluate a recurrence [Sakoe and Chiba, 1978]. Given two sequences q and s, the recurrence of DTW distance between them is

$$DTW(i,j) = dist(q_i, s_j) + \min\Big\{DTW(i-1,j-1), \\ DTW(i-1,j),\ DTW(i,j-1)\Big\},$$

where q_i and s_j denote the i-th and the j-th action in segments q and s respectively, the distance $dist(q_i, s_j)$ between two actions is defined in applications(we adopt a simple way to measure it in the experiment).

Given the recurrent definition, the DTW distance between q and s is calculated as $DTW(q,s) = DTW(q.length - 1, s.length - 1)$ (assuming the accessible index is from 0 to $length - 1$).

However, the DTW measures all sequences equally. In our assumption, the stage of a customer shifts smoothly day by day, segments from the same customer should be more similar than from other customers. Therefore, we incorporate customer discrimination information in the formula, and thus defining the evolutionary segments similarity (here the smaller similarity value, the more similar) in Definition 3.1.

Definition 3.1 (Evolutionary Similarity)**.** *Given two segments q and s, the Evolutionary Similarity (ES) between them is defined as*

$$ES(q,s) = (1 + DTW(q,s))^{I(q,s)} - 1\,,$$

where $I(q,s) = 1$ if and only if q and s are from different customers, otherwise, $I(q,s) = \lambda\ (0 < \lambda \leq 1)$

When $\lambda = 1$, the *ES* degrades to the DTW similarity, and the smaller λ the more similar the segments from the same customer. The *ES* combines sequence similarity and customer continuity, which help us not get confused when dealing with two similar segments but succeed from different predecessor stages. For example, let Customer 1 has segments s_1^1 and s_2^1, Customer 2 has segments s_1^2, and s_1^1 is same with s_1^2, then the value of $ES(s_1^1, s_2^1)$ would be smaller than that of $ES(s_1^2, s_2^1)$, since s_1^1 lies in the path of Customer 1.

With the *ES* defined above, we can know how distant a segment is to another segment, as illustrated in Figure 2 Part C. We employ manifold learning methods here to embed the segments in a metric space, in which the stages of the customers are naturally vectorized. The manifold learning has two purposes. The evolutionary similarity can be regarded as a similarity in a latent high dimensional space, in which there warps an intrinsic low-dimensional nonlinear subspace. Manifold learning can discover the intrinsic subspace, in which the global similarity measurement is more effective. Moreover, manifold learning can assign a low-dimensional position to a segment, so that the distance between every two segments is closest to their effective global distance in the latent high dimension space. Therefore, the learned manifold space can be more regular for latter processes. Specifically, inspired by the visualization method in [Liu *et al.*, 2014], we adopt the ISOMAP [Tenenbaum *et al.*, 2000], which estimates the geodesic distances between all pairs of segments on a manifold.

By applying this manifold learning process, we can get a low dimensional space, where stage evolution is expected to be well modeled. As Figure 2 Part D shows, customers' evolutionary paths are captured in CM space.

Last but not least, if we set the target dimension d to 2 or 3, we can easily visualize the CM space, leading us to make more interesting and useful observations, so that we can have some insights to the customer behavior evolution.

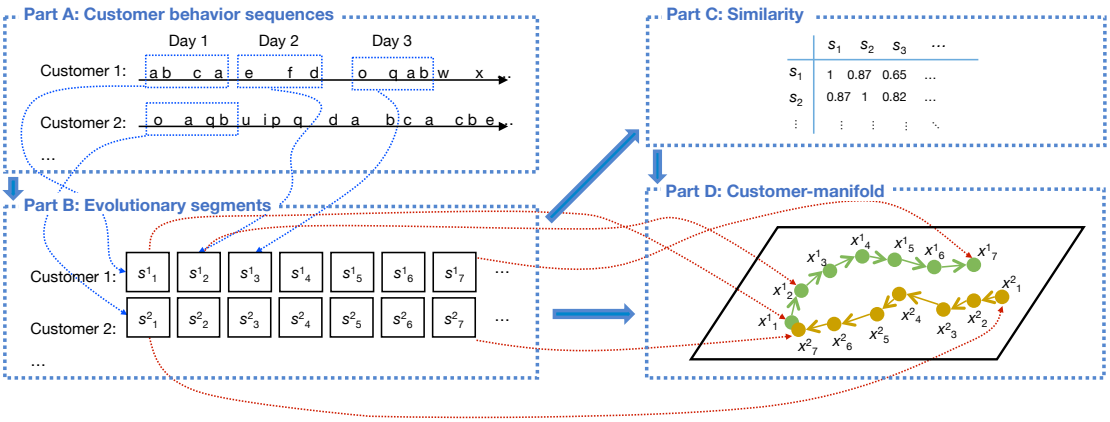

Figure 2: Illustration of the workflow of the customer-manifold learning

3.2 Apply Customer-Manifold to Recommender

From what we have discussed above, customers' life stage is already modeled in the CM and similar stages are kept aligned, it would be possible to capture the stage evolution patterns. We take the recommendation as an example to show how we can take advantage of CM.

As we can see, the CM space can be regarded as a d-dimension feature space. With proper label space design, we can apply various state-of-the-art machine learning methods to train a static prediction model here. For example, we can simply take customers' next behavior in the sequence as supervised label to train a multi-class classifier (as left part of Figure 3 shows, s_2^1 can be used as the supervised label of x_1^1). If we take customers' rating as supervised label, we can train a rating prediction model. In other words, the prediction model and the prediction task are not restricted.

3.3 LSTM Embedding

It is worth noting that there are some time-consuming parts in above steps such as similarity computation and ISOMAP, which make the algorithm face a great performance challenge in real world applications. When there are millions of customers, we can hardly map customers' segments to the customer manifold efficiently.

To make the stage modeling process more scalable and easily trainable, we embed the CM into the powerful RNN. Since RNN is devised to model variable-length sequence data, we just feed the networks with the segments we extracted before and use the constructed manifold space as supervised information to pretrain an embedding layer (as Figure 3 pretrain part shows). In this way, we don't have to pay a lot of time to construct a CM space for big data. We can simply sample some segments to train the network. And it would also be much faster to process fresh segments through this network.

As for recommending, we extract customer's up-to-date evolutionary segment from his/her behavior sequence as one instance (like the s_2^1 in Figure 3). Through the embedding network we trained, this instance can be transformed to the CM space (like the x_2^1 in Figure 3). We introduce a forward

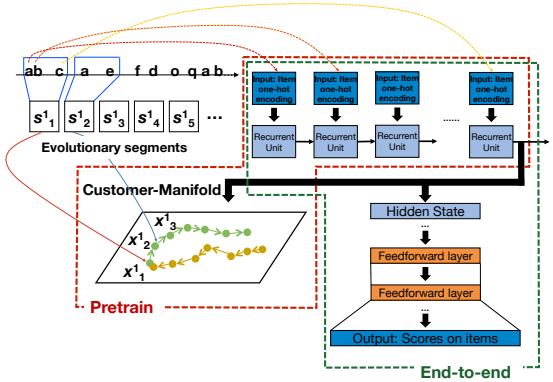

Figure 3: Embedding customer-manifold into a neural network recommender system

neural network as the classifier which maps the position to future behaviors, thus making a recommendation (as showed in the bottom of Figure 3). With the CM and the instances we constructed before, train the whole network is no longer end-to-end (as end-to-end part in Figure 3), but is still straightforward.

4 Experiments

4.1 Experiments Settings

Datasets

The data used in our experiment is provided by Tencent Inc., which consists of customer online shopping history from 7/21/2015 to 2/1/2016 from a real B2C e-commerce system, which servers millions of people everyday. The data set has 4,975,481 records, which includes 102,045 customers and 3,594 items. The data before 1/22/2016 is used for training and the rest is used for testing. We set the supervised labels as the customers' first click items in the next day for the purpose of predicting what customers will click in the next 10 days.

Metric

We adopt the Precision@K to measure the performance of recommendation models, which is the ratio of the successfully predicted test items to the top-K recommendation.

Comparison

To make the result more convincing, we also compare with several baselines. Those include:

- UserKNN [Resnick *et al.*, 1994], known as user-based CF, which finds similar users and recommend according to these users' behaviors.
- ItemKNN [Sarwar *et al.*, 2001], known as item-based CF, which recommends similar items.
- WMRF [Hu *et al.*, 2008], which proposes a regularized least-square optimization with case weights. The case weights can be used to reduce the impact of negative examples.
- AoBPR [Rendle and Freudenthaler, 2014], which is a sample-based method that optimizes the pair-wise ranking between the positive and negative samples..
- SVD++ [Koren, 2008], which is a matrix factorization model integrating implicit feedback.
- RegSVD [Paterek, 2007], which combines several SVD predictors using linear regression.
- KNN(ES). We use the *ES* we defined as similarity metric, and apply KNN to the segments to make recommendation.

Note that, these algorithms are not able to handle sequential data, so we generate a rating-like matrix from the original data. Specifically, a customer's rating for an item equals to how many times he clicked that item. For ItemKNN and UserKNN, we just set the number of neighbors to 10, the similarity is computed using Pearson Correlation.

As for our method (denoted as CM-LSTM), we set the number of neighbors k to 10 (same as ItemKNN and UserKNN). In the evolutionary similarity computing, the λ is 0.5 and $dist$ equals to 0.5 if two items belongs to the same category, otherwise equals to 5. For example, Men's Clothing and Women's Clothing are belong to Clothing, so their $dist$ is 0.5, however Mobile Phones is way different from Men's Clothing, so the $dist$ between them is 5. We adopt only 1 LSTM layer and 2 forward layers. The similarity in KNN(ES) uses the same setting.

4.2 Experimental Results

Recommendation Precision

We first compare the Precision@1 using a same topology network with different parameter configurations to see how the embedded CM will influence the model performance. We random sample 8000 segments to construct the CM, and embed them into three different networks. Three networks differ in the number of hidden units, mini-batch size and training epochs. For example, 100-65-2000 denotes that we train a network with 100 hidden units in mini-batch size of 65 for 2000 epochs.

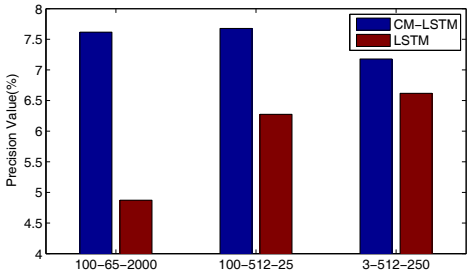

Figure 4: Comparison under different configurations

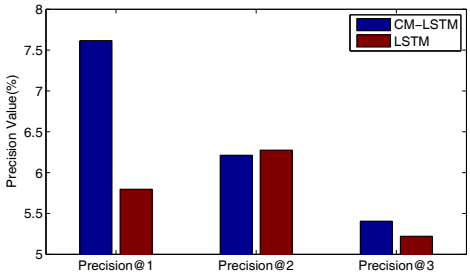

Figure 5: Comparison of recommendation precision

As Figure 4 shows, the original network (denoted as LSTM) is sensitive to the parameters. But with CM embedded, the performance is more accurate and stable. We make a recommendation list of size 3 to compare the recommender precision between the CM-LSTM and LSTM. We set the configuration as the best LSTM model we can train. As the Figure 5 shows, CM-LSTM performs better than the LSTM, not only at the top-1 recommendation, but also at the following positions.

We compare the performance of our approach with that of baselines and state-of-the-art approaches. Figure 6 shows the precision of different recommendation models with different recommendation list sizes.

From Figure 6, we can see that our proposed method outperforms these baselines. Since other methods cannot handle time-series data, they lose the sequential information at the beginning. On the other hand, customer behavior space is quite complicated, it would be very difficult to precisely capture customer dynamics without considering the stage information. (Note that for algorithms like SVD++, RegSVD, additional hand-crafted features can be added to significantly improve the performance. However we do not provide any implicit data or additional features to these algorithms for the sake of fairness, so the performance of these algorithms may not be the best. But the experimental result already poses their limitations.) We don't compare any more time-related model here, since CM-LSTM already beats LSTM, which is a powerful time-related model.

On the other hand, when the recommendation list is shorter, our method performs better. This is due to that our method aims at predicting customers' next move. Note that the adver-

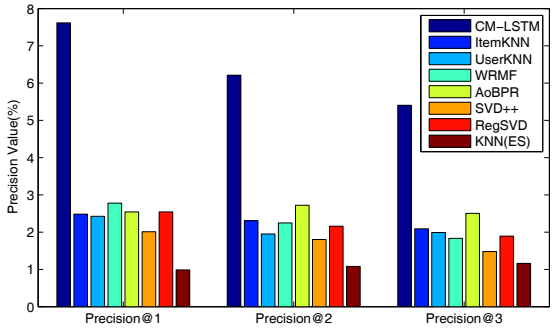

Figure 6: Comparison among different algorithms

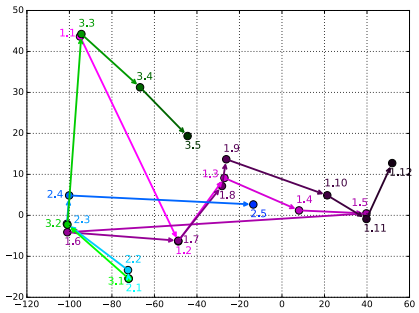

Figure 7: Visualization of 3 customers' evolutionary stages. Customers 1, 2, 3 are indicated by purple, blue and green, respectively. The color goes darker with time evolving. An "$x.y$" number is attached to every point, where x denotes the customer ID, y denotes the evolutionary stage ID

tisement position is always limited, we can not make a long recommendation list to customers, thus an efficient short list is in urgent need.

Time Efficiency

We also record the running time to see how the neural network speeds up the CM method. To embed a new segment by calculate the similarity matrix into a CM would take over 6 seconds, but using the embedded LSTM network only takes less than 1 second on the same machine.

4.3 Visualization

The visualization has two purposes, one is to visualize that a CM is learned, the other is to show that it may help user identify groups of customers. We will take a small example in 2D space to illustrate customers' evolving behaviors and give an overview of hundreds of customers in a 3D way.

As Figure 7 shows, there are three customers' data present in the figure. As we can see, point 1.1 is very close to 3.3, which reflects customer similar stage. If we look into the record, we can find both customer 1 and customer 3 clicked a lot of phone related items and women's clothing at this stage. On 1.2 and 1.7, customer 1 seems to have repeated behaviors. The record shows that he or she viewed many mobile phones, which may indicate he or she is in a stage of buying phones. With almost the same women's clothing browsing record, 1.2 evolved to 1.3 and 1.7 evolved to 1.8, where 1.3 and 1.8 are also very close. However, customer 2 and customer 3 almost share the same evolutionary path in their first four stages, but customer 2's fifth stage is way from customer 3' path, which shows the complexity of stage evolving patterns. Therefore, machine learning is needed here to help us to capture various patterns.

Figure 8 shows an overview of CM, where over 400 random loyal customers (who have long periods of record) are presented and every customer's evolving path is indicated by one color in a 3D space. It can be observed that, first, the stage of the customers are not randomly distributed in the CM space, instead, there are obvious clusters; and the transition lines do not connect all states, instead, there are strong beams indicating frequent transitions.

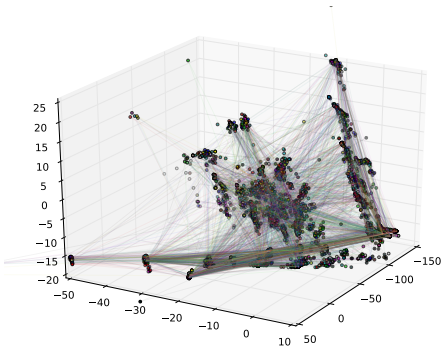

Figure 8: Visualization of stage evolution patterns in 3D

5 Conclusion

In this paper, we propose an approach to model customer lifestage and incorporate it to build a recommender system. Instead of building a model in the behavior space directly, we use only customer behavior history to learn a low dimensional customer-manifold space to align stages. In this static stage space, it would be easier to capture complicated stage dynamics and predict future behavior evolution. Visualization results show that customers' evolving preferences are well embedded and empirical results on a real world data set demonstrate that life stage modeling is effective in the recommendation scenario in comparison with the baselines. From the visualization, we notice that not only the position in the manifold that can affect the future behavior, but also the whole historical path. We leave in the future the study that incorporating the customer path into the recommender system.

References

[Berndt and Clifford, 1994] Donald J. Berndt and James. Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 1th International ACM*

SIGKDD Conference on Knowledge Discovery and Data Mining, pages 359–370, 1994.

[Bojanic, 2011] David C. Bojanic. The impact of age and family life experiences on mexican visitor shopping expenditures. *Tourism Management*, 32(2):406–414, 2011.

[Chu *et al.*, 2002] Selina Chu, Eamonn J. Keogh, David M. Hart, and Michael J. Pazzani. Iterative deepening dynamic time warping for time series. In *Proceedings of the 2nd SIAM International Conference on Data Mining*, pages 195–212, 2002.

[Ding and Li, 2005] Yi Ding and Xue Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 485–492, 2005.

[Ferwerda and Schedl, 2016] Bruce Ferwerda and Markus Schedl. Personality-based user modeling for music recommender systems. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part III*, pages 254–257, 2016.

[Hidasi *et al.*, 2015] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *CoRR*, abs/1511.06939, 2015.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 263–272, 2008.

[Jiang *et al.*, 2015] Peng Jiang, Yadong Zhu, Yi Zhang, and Quan Yuan. Life-stage prediction for product recommendation in e-commerce. In *Proceedings of the 21th International ACM SIGKDD Conference on Knowledge discovery and data mining*, pages 1879–1888, 2015.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th International ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 426–434, 2008.

[Koren, 2010] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[Lathia *et al.*, 2009] Neal Lathia, Stephen Hailes, and Licia Capra. Temporal collaborative filtering with adaptive neighbourhoods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 796–797, 2009.

[Le *et al.*, 2016] Duc-Trong Le, Yuan Fang, and Hady Wirawan Lauw. Modeling sequential preferences with dynamic user and context factors. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II*, pages 145–161, 2016.

[Li *et al.*, 2011] Ruijiang Li, Bin Li, Cheng Jin, Xiangyang Xue, and Xingquan Zhu. Tracking user-preference varying speed in collaborative filtering. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pages 133–138, 2011.

[Liu *et al.*, 2010] Nathan Nan Liu, Min Zhao, Evan Wei Xiang, and Qiang Yang. Online evolutionary collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems*, pages 95–102, 2010.

[Liu *et al.*, 2014] Chuanren Liu, Kai Zhang, Hui Xiong, Geoff Jiang, and Qiang Yang. Temporal skeletonization on sequential data: Patterns, categorization, and visualization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1336–1345, 2014.

[Liu, 2015] Xin Liu. Modeling users' dynamic preference for personalized recommendation. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1785–1791, 2015.

[Paterek, 2007] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD Cup and Workshop*, 2007.

[Rendle and Freudenthaler, 2014] Steffen Rendle and Christoph Freudenthaler. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pages 273–282, 2014.

[Resnick *et al.*, 1994] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[Sakoe and Chiba, 1978] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, pages 43–49, 1978.

[Sarwar *et al.*, 2001] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, pages 285–295, 2001.

[Tenenbaum *et al.*, 2000] Joshua B. Tenenbaum, Vin De Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[Wells and Gubar, 1966] William D. Wells and George Gubar. Life cycle concept in marketing research. *Journal of Marketing Research*, 3(4):355–363, 1966.

[Xiong *et al.*, 2010] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G. Schneider, and Jaime G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the 10th SIAM International Conference on Data Mining*, pages 211–222, 2010.