CosRec: 2D Convolutional Neural Networks for Sequential Recommendation

An Yan, Shuo Cheng, Wang-Cheng Kang, Mengting Wan, Julian McAuley
University of California, San Diego
{ayan,scheng,wckang,m5wan,jmcauley}@ucsd.edu

ABSTRACT

Sequential patterns play an important role in building modern recommender systems. To this end, several recommender systems have been built on top of Markov Chains and Recurrent Models (among others). Although these sequential models have proven successful at a range of tasks, they still struggle to uncover complex relationships nested in user purchase histories. In this paper, we argue that modeling pairwise relationships directly leads to an efficient representation of sequential features and captures complex item correlations. Specifically, we propose a 2D convolutional network for sequential recommendation (**CosRec**). It encodes a sequence of items into a three-way tensor; learns local features using 2D convolutional filters; and aggregates high-order interactions in a feedforward manner.

Quantitative results on two public datasets show that our method outperforms both conventional methods and recent sequence-based approaches, achieving state-of-the-art performance on various evaluation metrics.

ACM Reference Format:
An Yan, Shuo Cheng, Wang-Cheng Kang, Mengting Wan, Julian McAuley. 2019. CosRec: 2D Convolutional Neural Networks for Sequential Recommendation. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM'19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3357384.3358113

1 INTRODUCTION

The goal of sequential recommendation is to predict users' future behavior based on their historical action sequences. Different from traditional personalized recommendation algorithms (e.g. Matrix Factorization [10]) which seek to capture users' *global* tastes, sequential models introduce additional behavioral dynamics by taking the *order* of users' historical actions into consideration.

A classic line of work to model such dynamics is based on Markov Chains (MCs), which assumes that a user's next interaction is derived from the preceding few actions only [3, 12]. Recently, many neural network based approaches have achieved success on this task, where users' complete interaction sequences can be incorporated through Recurrent Neural Networks (RNNs) [5] or Convolutional Neural Networks (CNNs) [14]. Note that most existing models operate on *ordered* item representations directly, and thus are constrained by the one-directional *chain*-structure of action sequences. This leads to one advantage that these algorithms are capable of preserving *locally* concentrated dynamics, e.g. as shown in Figure 1a: consecutive purchases of a camera, a memory card, and a camera lens may lead to a strong indication of buying a tripod.

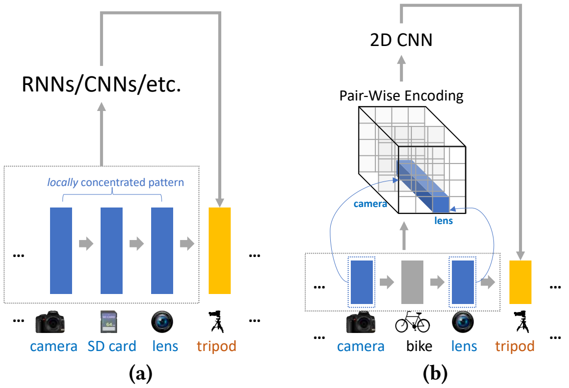


Figure 1: Illustrations of (a) locally concentrated dynamics and how they are preserved in existing models; and (b) an example where 'skip' behavior (bike) exists between two closely related items (camera and lens), and how this pattern is preserved by the proposed framework.

In this paper, we surprisingly find that relaxing the above structure constraint may yield more effective recommendations. Specifically, we propose a 2D CNN-based framework—2D **co**nvolutional networks for **s**equential **rec**ommendation (**CosRec**). In particular, we enable interactions among *nonadjacent* items by introducing a simple but effective pairwise encoding module. As shown in Figure 1b, the 'skip' behavior within item sequences (i.e., the purchase of a bike is less relevant to the context of consuming photography products) may break the locality of the chain-structure but can be easily bypassed through this pairwise encoding. On account of this module, we show that standard 2D convolutional kernels can be applied to solve sequential recommendation problems, where small filters (e.g. 3 × 3) can be successfully incorporated. This also allows us to build an extendable 2D CNN framework, which can be easily adapted to either shallow or deep structures for different tasks.

2 RELATED WORK

Sequential recommendation methods typically seek to capture sequential patterns among previously consumed items, in order to accurately predict the next item. To this end, various models have been adopted, including Markov Chains (MCs) [12], Recurrent Neural Networks (RNNs) [5], Temporal Convolutional Networks (TCNs) [15], and Self Attention [7, 13], among others.

CIKM '19, November 3–7, 2019, Beijing, China
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3358113

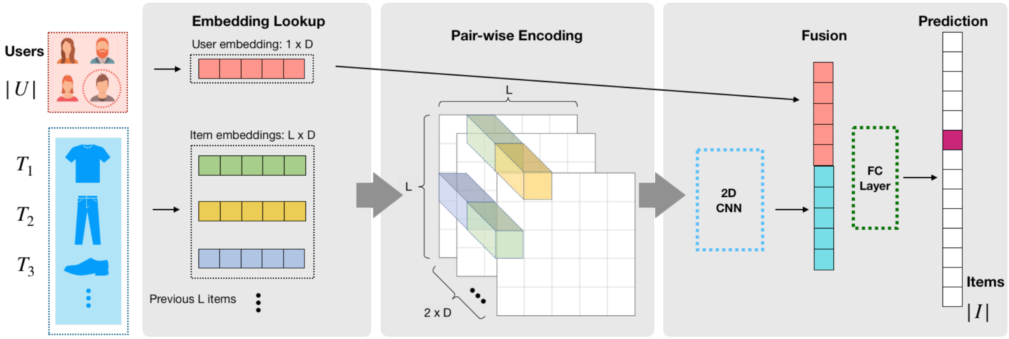


Figure 2: The detailed architecture of the proposed CosRec framework. Previous item embeddings (three examples illustrated here in green, yellow, and blue) are passed into a pairwise encoding module. The output is then fed into a 2D CNN and conditioned on user embeddings to predict the next item.

Caser [14] and **NextItNet** [15] are closer to our work as they also use convolutions. However, **Caser**'s vertical/horizontal convolution and **NextItNet**'s 1D dilated convolution significantly differ from the standard 2D convolution used in our method, due to the different filter shapes.[1] To our knowledge, **CosRec** is the first 2D CNN based approach for next item recommendation.

CNNs have also been adopted for other recommendation tasks. For example, **DVBPR** [6] extracts item embeddings from product images with CNNs, **ConvMF** [8] regularizes item embeddings by document features extracted from CNNs, and **ConvNCF** [4] applies CNNs on the outer product of user and item embeddings to estimate user-item interactions. Our work differs from theirs in that we focus on capturing sequential patterns of L previous visited items.

3 METHOD

We formulate the sequential recommendation problem as follows. Suppose we have a set of users $\mathcal{U}$ and a set of items $\mathcal{I}$. For each user $u \in \mathcal{U}$, given the sequence of previously interacted items $\mathcal{S}^u = (\mathcal{S}^u_1, \dots \mathcal{S}^u_{|\mathcal{S}_u|})$, $\mathcal{S}^u_\cdot \in \mathcal{I}$, we seek to predict the next item to match the user's preferences.

We introduce our **CosRec** framework via three modules: the embedding look-up layer, the pairwise encoding module, and the 2D convolution module. We then compare **CosRec** and other existing CNN-based approaches to illustrate how the technical limitations are addressed in our framework. The detailed network architecture is shown in Figure 2.

3.1 Embedding Look-up Layer

We embed items and users into two matrices $\boldsymbol{E}_{\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}| \times d}$ and $\boldsymbol{E}_{\mathcal{U}} \in \mathbb{R}^{|\mathcal{U}| \times d}$, where d is the latent dimensionality, $\boldsymbol{e}_i$ and $\boldsymbol{e}_u$ denote the i-th and the u-th rows in $\boldsymbol{E}_{\mathcal{I}}, \boldsymbol{E}_{\mathcal{U}}$ respectively. Then for user u at time step t, we retrieve the input embedding matrix $\boldsymbol{E}^L_{(u,t)} \in \mathbb{R}^{L \times d}$ by looking up the previous L items $(\mathcal{S}^u_{t-L}, \dots, \mathcal{S}^u_{t-1})$ in the item embedding matrix $\boldsymbol{E}_{\mathcal{I}}$.

3.2 Pairwise Encoding

We propose an encoding approach to allow flexible pairwise interactions among items. Specifically, we create a three-way tensor $\boldsymbol{T}^L_{(u,t)} \in \mathbb{R}^{L \times L \times 2d}$ on top of the input embeddings $\boldsymbol{E}^L_{(u,t)}$, where the (i,j)-th vector is the concatenated embedding of the item pair (i,j): $[\boldsymbol{e}_i; \boldsymbol{e}_j]$, $i, j \in (\mathcal{S}^u_{t-L}, \dots, \mathcal{S}^u_{t-1})$.

Different from previous approaches [14, 15] where convolutional filters directly operate on the input matrix $\boldsymbol{E}^L_{(u,t)}$, we apply convolutional layers on this resulting tensor $\boldsymbol{T}^L_{(u,t)}$ so that intricate patterns (e.g. Figure 1b) can be captured. Note that the encoded tensor has the same shape of an 'image feature map' in standard CNN models for computer vision tasks. Therefore, a wide range of CNN-like architectures can be borrowed and easily adapted in our context through this pairwise encoding.

Layer	Output Size	Kernel Size
input	$D \times 5 \times 5$	-
conv1_1	$D_1 \times 5 \times 5$	1×1
conv1_2	$D_1 \times 3 \times 3$	3×3
conv2_1	$D_2 \times 3 \times 3$	1×1
conv2_2	$D_2 \times 1 \times 1$	3×3
FC-D_3	$D_3 \times 1 \times 1$	dropout
FC-$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{I}\rvert \times 1 \times 1$	sigmoid

Table 1: 2D CNN. D is the dimension of input item embeddings. D_1, D_2, D_3 are the latent dimensions of each layer. FC means Fully Connected layer.

3.3 2D Convolutions

In order to capture high-level sequential patterns, we feed the above 'image feature map' $\boldsymbol{T}^L_{(u,t)}$ to a 2D convolutional neural network. We use a light-weight CNN, following suggestions from classical CNN architecture designs.

We provide an illustrative network example in Table 1. Here each block consists of two convolutional layers: the first layer uses 1×1 kernels to enrich the feature representations; the second layer, with a kernel size of 3, aggregates sequential features and extracts more complex relations as the network gets deeper. Each convolutional layer is followed by a batch normalization and a rectified linear unit (ReLU) activation. After these two convolutional blocks, we apply a fully-connected layer with dropout and thus obtain the final sequential feature vector $\boldsymbol{v}_{(u,t)} \in \mathbb{R}^d$.

[1] **Caser**: $h \times d$, $L \times 1$, **CosRec**: $h \times h \times d$, **NextItNet**:$1 \times h \times d$

Dataset	#users	#items	avg. #act. per user	avg. #act. per item	#actions
ML-1M	6.0K	3.4K	165.50	292.06	0.993M
Gowalla	13.1K	14.0K	40.74	38.12	0.533M

Table 2: Statistics of the datasets.

In order to capture users' global preferences, we concatenate the sequential vector $\boldsymbol{v}_{(u,t)}$ with the user embedding $\boldsymbol{e}_u$, project them to an output layer with $|\mathcal{I}|$ nodes, and apply a sigmoid function to produce the final probability scores $\sigma(\boldsymbol{y}^{(u,t)}) \in R^{|\mathcal{I}|}$.

3.4 Model Training

We adopt the binary cross-entropy loss as the objective function:

$$-\sum_u \sum_t \left(log\left(\sigma(y^{(u,t)}_{\mathcal{S}^u_t})\right) + \sum_{j \notin \mathcal{S}_u} log\left(1 - \sigma(y^{(u,t)}_j)\right)\right) \qquad (1)$$

The network is optimized via the **Adam** Optimizer [9], a variant of Stochastic Gradient Descent (SGD) with adaptive moment estimation. In each iteration, we randomly sample N negative samples (j) for each target item $\mathcal{S}^u_t$.

3.5 Comparison with Existing CNN-based Approaches

We show that **CosRec** addresses the limitations of existing CNN-based approaches, particularly **Caser** [14] via the following aspects:

- In **Caser**, each user's action sequence is embedded as a matrix, and two types of convolutional filters are applied on top of these embeddings horizontally and vertically. One limitation of such an approach is that it could perform poorly in the presence of noisy or irrelevant interactions as shown in Figure 1b. We address this problem by encoding each action sequence into high-dimensional pairwise representations and applying convolution layers afterwards, so that the above irrelevant actions can be easily skipped.
- Another drawback of **Caser** is the use of vertical filters. It aims to produce a weighted sum of all previous items, while it only performs summations along each dimension and there are no channel-wise interactions, which may lack representational power. This weighted sum also results in a shallow network structure that is suited only for one layer, leading to problems when modeling long-range dependencies or large-scale data streams where a deeper architecture is needed. Our method with 2D kernels naturally brings channel-wise interactions among vectors, along with flexibility to adapt the network to either shallow or deep structures for different tasks, by applying padding operations or changing the kernel size.

4 EXPERIMENTS

4.1 Datasets and Experimental Setup

4.1.1 Datasets. Following the protocol used to evaluate **Caser** [14], we evaluate our method on two standard benchmark datasets, **MovieLens** and **Gowalla**. The statistics of the datasets are shown in Table 2.

- **MovieLens** [2]: A widely used benchmark dataset for evaluating collaborative filtering algorithms. We use the MovieLens-1M (**ML-1M**) version in our experiments.
- **Gowalla** [1]: A location-based social networking website where users share their locations by checking-in, labeled with time stamps.

We follow the same preprocessing procedure as in **Caser** [14]: we treat the presence of a review or rating as implicit feedback (i.e., the user interacted with the item) and use timestamps to determine the sequence order of actions, and discard users and items with fewer than 5 and 15 actions for **ML-1M** and **Gowalla** respectively. We hold the first 80% of actions in each user's sequence for training and validation, and the remaining 20% actions as the test set for evaluating model performance.

4.1.2 Evaluation metrics. We report the evaluated results by three popular top-N metrics, namely Mean Average Precision (**MAP**), **Precision@N** and **Recall@N**. Here N is set to 1, 5, and 10.

4.1.3 Implementation details. We use 2 convolution blocks, each consisting of 2 layers. The latent dimension d is chosen from {10, 20, 30, 50, 100}, and we use 50 and 100 for **ML-1M** and **Gowalla** respectively. The Markov order L is 5. We predict the next $T = 3$ items at once. The learning rate is 0.001, with a batch size of 512, a negative sampling rate of 3 and a dropout rate of 0.5. All experiments are implemented using PyTorch.[2]

4.2 Performance Comparison

To show the effectiveness of our method, we compare it with a number of popular baselines.

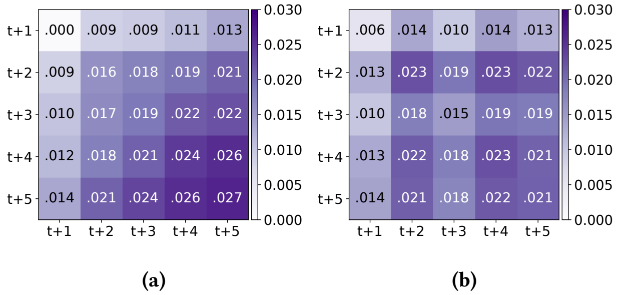
- **PopRec**: A simple baseline that ranks items according to their popularity.
- **Bayesian Personalized Ranking (BPR)** [11]: A classic non-sequential method for learning personalized rankings.
- **Factorized Markov Chains (FMC)** [12]: A first-order Markov Chain method which generates recommendations depending only on the last visited item.
- **Factorized Personalized Markov Chains (FPMC) [12]:** A combination of FMC and MF, so that short-term item transition patterns as well as users' global preferences can be captured.
- **GRU4Rec** [5]: A state-of-the-art model which uses an RNN to capture sequential dependencies and make predictions.
- **Convolutional Sequence Embeddings (Caser)** [14]: A recently proposed CNN-based method which captures high-order Markov Chains by applying convolutional operations on the embedding matrix of the previous L items.
- **CosRec-base**: In order to evaluate the effectiveness of the 2D CNN module, we create a baseline version of **CosRec** which uses a multilayer perceptron (MLP) instead of a 2D CNN on the pairwise encodings.

Experimental results are summarized in Table 3. Among the baselines, sequential models (e.g. **Caser**) outperform non-sequential models (e.g. **BPR**), confirming the importance of considering sequential information. **CosRec** outperforms **FMC**/**FPMC**, since

[2] Code is available at: https://github.com/zzxslp/CosRec

Dataset	Metric	PopRec	BPR	FMC	FPMC	GRU4Rec	Caser	CosRec-base	CosRec	Improvement
ML − 1M	MAP	0.0687	0.0913	0.0949	0.1053	0.1440	0.1507	0.1743	**0.1883**	+25.0%
	Prec@1	0.1280	0.1478	0.1748	0.2022	0.2515	0.2502	0.2892	**0.3308**	+31.5%
	Prec@5	0.1113	0.1288	0.1505	0.1659	0.2146	0.2175	0.2521	**0.2831**	+30.2%
	Prec@10	0.1011	0.1193	0.1317	0.1460	0.1916	0.1991	0.2256	**0.2493**	+25.2%
	Recall@1	0.0050	0.0070	0.0104	0.0118	0.0153	0.0148	0.0186	**0.0202**	+32.0%
	Recall@5	0.0213	0.0312	0.0432	0.0468	0.0629	0.0632	0.0771	**0.0843**	+33.4%
	Recall@10	0.0375	0.0560	0.0722	0.0777	0.1093	0.1121	0.1331	**0.1438**	+28.3%
Gowalla	MAP	0.0229	0.0767	0.0711	0.0764	0.0580	0.0928	0.0821	**0.0980**	+05.6%
	Prec@1	0.0517	0.1640	0.1532	0.1555	0.1050	0.1961	0.1712	**0.2135**	+08.9%
	Prec@5	0.0362	0.0983	0.0876	0.0936	0.0721	0.1129	0.1012	**0.1190**	+05.4%
	Prec@10	0.0281	0.0726	0.0657	0.0698	0.0782	0.0571	0.0762	**0.0884**	+13.0%
	Recall@1	0.0064	0.0250	0.0234	0.0256	0.0155	0.0310	0.0265	**0.0337**	+08.7%
	Recall@5	0.0257	0.0743	0.0648	0.0722	0.0529	0.0845	0.0752	**0.0890**	+05.3%
	Recall@10	0.0402	0.1077	0.0950	0.1059	0.0826	0.1223	0.1107	**0.1305**	+06.7%

Table 3: Performance comparison with state-of-the-art approaches on all datasets.



(a)

(b)

Figure 3: A visualization of two convolutional filters with kernel size 5, trained on ML-1M. Darker colors indicate higher values. The value in grid (i,j) corresponds to the weight for the item pair (i,j)'s pairwise encoding.

they only model the first-order Markov chain while **CosRec** captures high-order relations. Overall, our method outperforms all baselines on both datasets by a significant margin. The performance improvements on **ML-1M** are particularly significant (**25.0%** improvement in terms of **MAP**), presumably due to the fact that **ML-1M** is a relatively dense dataset with rich sequential signals. Note on the **ML-1M** dataset, even our baseline version (MLP) outperforms existing state-of-the-art methods, which validates the effectiveness of the pairwise encoding for capturing more intricate patterns.

4.3 Visualization

We visualize two example convolutional filters in Figure 3 to show that our proposed framework is not only capable of modeling the 'recency' and the 'locality' (as in existing models), but flexible to capture more complex patterns such as the 'skip' behavior. Here each filter serves as a weighted sum of our pairwise encoding. We see a clear trend in Figure 3a that weights increase from top left to bottom right, which indicates more recent items are attended in this case. In addition, we observe scattered blocks in Figure 3b, which implies that the model is able to bypass the chain-structure and capture nonadjacent patterns.

5 CONCLUSION

In this paper, we proposed a novel 2D CNN framework, **CosRec** for sequential recommendation. The model encodes a sequence of item embeddings into pairwise representations and leverages a 2D CNN to extract sequential features. We perform experiments on two real-world datasets, where our method significantly outperforms recent sequential approaches, showing its effectiveness as a generic model for sequential recommendation tasks.

Acknowledgements. This work is partly supported by NSF #1750063. We thank all the reviewers for their suggestions.

REFERENCES

[1] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *KDD*.

[2] F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (TIIS)* (2016).

[3] R. He, W.-C. Kang, and J. McAuley. 2017. Translation-based Recommendation. In *RecSys*.

[4] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer Product-based Neural Collaborative Filtering. In *IJCAI*.

[5] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. In *ICLR*.

[6] Wang-Cheng Kang, Chen Fang, Zhaowen Wang, and Julian McAuley. 2017. Visually-Aware Fashion Recommendation and Design with Generative Image Models. In *ICDM*.

[7] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*.

[8] Dong Hyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In *RecSys*.

[9] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[10] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix factorization techniques for recommender systems. *IEEE Computer* (2009).

[11] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[12] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*.

[13] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. *arXiv preprint arXiv:1904.06690* (2019).

[14] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*.

[15] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *WSDM*.