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:

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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

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

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.

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 convolutional networks for sequential recommendation (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., $3 \times 3$) 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.
We propose an encoding approach to allow flexible pairwise interactions among items. Specifically, we create a three-way tensor $T_{(u,t)} \in \mathbb{R}^{L \times d}$ on top of the input embeddings $E^L_{(u,t)}$, where the $(i,j)$-th vector is the concatenated embedding of the item pair $(i,j)$: $[e_i; e_j], i, j \in \{S^u_{t-L}, \ldots, S^u_{t-1}\}$.

Different from previous approaches [14, 15] where convolutional filters directly operate on the input matrix $E^L_{(u,t)}$, we apply convolutional layers on this resulting tensor $T_{(u,t)}$ 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.

### 3.3 2D Convolutions

In order to capture high-level sequential patterns, we feed the above ‘image feature map’ $T^L_{(u,t)}$ 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 $1 \times 1$ 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 $v_{(u,t)} \in \mathbb{R}^d$.

![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.](image)

### Table 1: 2D CNN

<table>
<thead>
<tr>
<th>Layer</th>
<th>Output Size</th>
<th>Kernel Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>$D \times 5 \times 5$</td>
<td>-</td>
</tr>
<tr>
<td>conv1_1</td>
<td>$D_1 \times 5 \times 5$</td>
<td>$1 \times 1$</td>
</tr>
<tr>
<td>conv1_2</td>
<td>$D_2 \times 3 \times 3$</td>
<td>$3 \times 3$</td>
</tr>
<tr>
<td>conv2_1</td>
<td>$D_2 \times 3 \times 3$</td>
<td>$1 \times 1$</td>
</tr>
<tr>
<td>conv2_2</td>
<td>$D_2 \times 1 \times 1$</td>
<td>$3 \times 3$</td>
</tr>
<tr>
<td>FC-D3</td>
<td>$D_3 \times 1 \times 1$</td>
<td>dropout</td>
</tr>
<tr>
<td>FC-[I]</td>
<td>$</td>
<td>I</td>
</tr>
</tbody>
</table>

Layer $D$ 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.

---

1Caser: $h \times d, L \times 1$, CosRec: $h \times h \times d$, NextItNet: $1 \times h \times d$
We show that we adopt the binary cross-entropy loss as the objective function:

$$-\sum_u \sum_t \left( \log(\sigma(y_{u,t}^{(u,t)})) + \sum_{j \in S_u} \log(1 - \sigma(y_{j,t}^{(u,t)})) \right)$$  \hspace{1cm} (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 $y_{u,t}^{(u,t)}$.

### 3.4 Model Training

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

$$-\sum_u \sum_t \left( \log(\sigma(y_{u,t}^{(u,t)})) + \sum_{j \in S_u} \log(1 - \sigma(y_{j,t}^{(u,t)})) \right)$$  \hspace{1cm} (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 $y_{u,t}^{(u,t)}$.

### 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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#users</th>
<th>#items</th>
<th>avg. #act. per user</th>
<th>avg. #act. per item</th>
<th>#actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML-1M</td>
<td>6.0K</td>
<td>3.4K</td>
<td>165.50</td>
<td>292.06</td>
<td>0.993M</td>
</tr>
<tr>
<td>Gowalla</td>
<td>13.1K</td>
<td>14.0K</td>
<td>40.74</td>
<td>38.12</td>
<td>0.533M</td>
</tr>
</tbody>
</table>

**Table 2: Statistics of the datasets.**

In order to capture users’ global preferences, we concatenate the sequential vector $v_{u,t}^{(u,t)}$ with the user embedding $e_u$, project them to an output layer with $|I|$ nodes, and apply a sigmoid function to produce the final probability scores $\sigma(y_{u,t}^{(u,t)}) \in \mathbb{R}^{|I|}$.

#### 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.
- **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

---

**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.$^2$

---

$^2$Code is available at: https://github.com/zzxslp/CosRec
We visualize two example convolutional filters in Figure 3 to show which implies that the model is able to bypass the chain-structure and capture nonadjacent patterns. In this case. In addition, we observe scattered blocks in Figure 3b, to bottom right, which indicates more recent items are attended 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 #1750603. We thank all the reviewers for their suggestions.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metric</th>
<th>PopRec</th>
<th>BPR</th>
<th>FMC</th>
<th>FPMC</th>
<th>GRU4Rec</th>
<th>Caser</th>
<th>CosRec-base</th>
<th>CosRec</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML ~ 1M</td>
<td>MAP</td>
<td>0.0687</td>
<td>0.0913</td>
<td>0.0949</td>
<td>0.1053</td>
<td>0.1440</td>
<td>0.1507</td>
<td>0.1743</td>
<td><strong>0.1883</strong></td>
<td>+25.0%</td>
</tr>
<tr>
<td></td>
<td>Prec@1</td>
<td>0.1280</td>
<td>0.1478</td>
<td>0.1748</td>
<td>0.2022</td>
<td>0.2515</td>
<td>0.2502</td>
<td>0.2892</td>
<td><strong>0.3308</strong></td>
<td>+31.5%</td>
</tr>
<tr>
<td></td>
<td>Prec@5</td>
<td>0.1113</td>
<td>0.1288</td>
<td>0.1505</td>
<td>0.1659</td>
<td>0.2146</td>
<td>0.2175</td>
<td>0.2521</td>
<td><strong>0.2831</strong></td>
<td>+30.2%</td>
</tr>
<tr>
<td></td>
<td>Recall@1</td>
<td>0.0500</td>
<td>0.0070</td>
<td>0.0194</td>
<td>0.0118</td>
<td>0.0153</td>
<td>0.0148</td>
<td>0.0186</td>
<td><strong>0.0202</strong></td>
<td>+32.0%</td>
</tr>
<tr>
<td></td>
<td>Recall@5</td>
<td>0.0213</td>
<td>0.0312</td>
<td>0.0432</td>
<td>0.0468</td>
<td>0.0629</td>
<td>0.0632</td>
<td>0.0771</td>
<td><strong>0.0843</strong></td>
<td>+33.4%</td>
</tr>
<tr>
<td></td>
<td>Recall@10</td>
<td>0.0375</td>
<td>0.0560</td>
<td>0.0722</td>
<td>0.0777</td>
<td>0.1093</td>
<td>0.1121</td>
<td>0.1331</td>
<td><strong>0.1438</strong></td>
<td>+28.3%</td>
</tr>
<tr>
<td>Gowalla</td>
<td>MAP</td>
<td>0.0229</td>
<td>0.0767</td>
<td>0.0711</td>
<td>0.0764</td>
<td>0.0580</td>
<td>0.0928</td>
<td>0.0821</td>
<td><strong>0.0980</strong></td>
<td>+05.6%</td>
</tr>
<tr>
<td></td>
<td>Prec@1</td>
<td>0.0517</td>
<td>0.1640</td>
<td>0.1532</td>
<td>0.1555</td>
<td>0.1050</td>
<td>0.1961</td>
<td>0.1712</td>
<td><strong>0.2135</strong></td>
<td>+08.9%</td>
</tr>
<tr>
<td></td>
<td>Prec@5</td>
<td>0.0362</td>
<td>0.0983</td>
<td>0.0876</td>
<td>0.0936</td>
<td>0.0721</td>
<td>0.1129</td>
<td>0.1012</td>
<td><strong>0.1190</strong></td>
<td>+05.4%</td>
</tr>
<tr>
<td></td>
<td>Recall@1</td>
<td>0.0064</td>
<td>0.0250</td>
<td>0.0234</td>
<td>0.0256</td>
<td>0.0782</td>
<td>0.0571</td>
<td>0.0762</td>
<td><strong>0.0884</strong></td>
<td>+13.0%</td>
</tr>
<tr>
<td></td>
<td>Recall@5</td>
<td>0.0257</td>
<td>0.0743</td>
<td>0.0648</td>
<td>0.0722</td>
<td>0.0529</td>
<td>0.0845</td>
<td>0.0752</td>
<td><strong>0.0890</strong></td>
<td>+05.3%</td>
</tr>
<tr>
<td></td>
<td>Recall@10</td>
<td>0.0402</td>
<td>0.1077</td>
<td>0.0950</td>
<td>0.1059</td>
<td>0.0826</td>
<td>0.1223</td>
<td>0.1107</td>
<td><strong>0.1305</strong></td>
<td>+06.7%</td>
</tr>
</tbody>
</table>

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