ONLINE VIDEO SESSION PROGRESS PREDICTION USING LOW-RANK MATRIX COMPLETION

Gang Wu*, Viswanathan Swaminathan**, Saayan Mitra** and Ratnesh Kumar*

ABSTRACT
The prediction of online video session progress is useful towards optimizing user experience. We approach the prediction of session progress as a matrix completion problem, and make the prediction by matrix completion based on regularized nuclear norm minimization. We first process a large amount of event tracking logs of user activities in online video sessions, represent them as a partially observed user by video matrix and make the completion using low-rank matrix completion method. Our initial results show improvement from the simple mean baseline. Experiments with Netflix data indicate that better improvements over simple mean methods can be expected by either observing more entries with given users and videos, or including more users and videos while maintaining same sparsity level. Another contribution of this paper is introducing a method to generate sub-matrix with good features and controlled sparsity and size.

Index Terms— Session progress prediction, matrix completion, collaborative filtering, nuclear norm, Netflix challenge

1. INTRODUCTION

1.1. Background
The viewership of online videos, like Youtube, Hulu, etc., has been increasing explosively in recent years. However, user experience in watching online video has not yet been personalized (or optimized? by G) enough, mainly limited by difficulty in predicting user’s interests in videos? by G). Currently most online video websites use rating system to collect feedback from users and make recommend video based on that. A major flaw of such rating system is that it requires user’s manual operations after the session, which normally is short and in a casual situation? by G), and this leads to valid ratings available only for a small fraction of sessions.

Note that in online video session there is always an implicit feedback from the users: session progress, or the fraction of video that been watched in the session. We believe that, compared to rating, session progress is of more practical use (or more useful, better, more practical value? by G) in user experience optimization in the following senses:

• Session progress is a more direct measurement of the video viewership, hence a good prediction can be built into a recommendation engine;

• A good prediction of session progress provides the information of specific part of video going to be watched, which is of significant value in advertisement insertion strategy and resource allocation optimization.

In case of online video sessions, a server can track several features information, like user’s geographic information, local time, video’s genre, etc. Using there features, it is easy to build a simple linear model corresponding to a weighted average of different features. Learning the weight parameters in such a model requires tracking and storing the feature information, which can be a huge burden on the server when the data volume is large. An alternative method without using the feature data is by looking for correlations of historical session progresses across users and videos. To do that, the recorded session progresses can be filled into a user by video session progress matrix $Z_{sp} \in \mathbb{R}^{M \times N}$, in which entry $z_{i,j}$ is session progress of user $i$ watching video $j$. Let $\Omega$ denote the set of indices that corresponded entry of $Z_{sp}$ is observed. See Fig. 1 for example of a partially observed session progress matrix.

The historical data now can be represented as $P_{\Omega}(Z_{sp})$. Prediction of unknown session progress is the problem of completing the partially observed matrix $Z_{sp}$ given $P_{\Omega}(Z_{sp})$.

Fig. 1. Example of a partially observed session progress matrix

<table>
<thead>
<tr>
<th>Video 1</th>
<th>Video 2</th>
<th>Video 3</th>
<th>⋯</th>
<th>Video N</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>13%</td>
<td>?</td>
<td>0%</td>
<td>?</td>
</tr>
<tr>
<td>User 2</td>
<td>?</td>
<td>85%</td>
<td>?</td>
<td>100%</td>
</tr>
<tr>
<td>User 3</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>0%</td>
</tr>
<tr>
<td>⋯</td>
<td>40%</td>
<td>100%</td>
<td>?</td>
<td>17%</td>
</tr>
<tr>
<td>User M</td>
<td>?</td>
<td>?</td>
<td>55%</td>
<td>70%</td>
</tr>
</tbody>
</table>
1.2. Related work

To make the completion of \(Z_{sp}\) meaningful, we assume that matrix \(Z_{sp}\) is approximately low-rank, i.e., there exists a low-rank matrix that can approximate \(Z_{sp}\) with a low level fitting error. A relevant popular problem is the completion of User by Movie rating matrix in Netflix contest, in which among all methods based on single model (without mixture), the low-rank model based method give best results [1].

Significant progresses have been made on the low-rank matrix completion problem in recent years. Srebro et al. [2] studied generalization error bounds for learning low-rank matrices. Candès and Recht [3], Candès and Tao [4], and Keshavan et al. [5] showed theoretically that under certain assumptions the true matrix can be recovered with high accuracy. Mazumder et al. [6] introduced an EM flavored iterative method SOFT-IMPUTE based on regularized nuclear norm minimization. Another similar class of techniques introduced by Srebro et al. [7] is known as maximum margin factorization methods, and use a factor model for the matrix. In this work, we use SOFT-IMPUTE as our low-rank matrix completion method, which achieves improvement from the user mean and video mean. Further experiments indicate better improvements can be achieved when more entries are observed or matrix size grows.

The rest of the paper is organized as follows. In Section 2, we describe our original log data, the approximated measurements of the session progress, and how we structure the data. Section 3 describes regularized nuclear norm minimization for low-rank matrix completion method, and use a factor model for the matrix. In this work, we use SOFT-IMPUTE as our low-rank matrix completion method, which achieves improvement from the user mean and video mean. Further experiments indicate better improvements can be achieved when more entries are observed or matrix size grows.

The rest of the paper is organized as follows. In Section 2, we describe our original log data, the approximated measurements of the session progress, and how we structure the data. Section 3 describes regularized nuclear norm minimization for low-rank matrix completion method, and use a factor model for the matrix. In this work, we use SOFT-IMPUTE as our low-rank matrix completion method, which achieves improvement from the user mean and video mean. Further experiments indicate better improvements can be achieved when more entries are observed or matrix size grows.

2. DATA PREPARATION

2.1. Data description

We start with a massive set of tracking logs from an online video server. Each video is ‘divided’ to multiple segments. The server records the start and complete of each segment in the session. The logs themselves contain over 1 billion events recorded from about 30 million online video sessions. Since we formulate the prediction as a matrix completion problem, we first need to measure the session progresses of existing sessions and fill into user by video matrix. However, there are several limitations in our data that make it difficult. One critical limitation is that, the server doesn’t record when the user actually exits, so in case of user exiting during a segment, we only know during which segment the user exits. This is a common issue in online video sessions, since most sessions end with the user navigating away or closing the browser tab. Hence we have to empirically estimate where the user exits. Another limitation is that, we don’t have the exact duration of each segment of video, which is needed to estimate session progress when the segments have different lengths. Due to that, we estimate segment duration with the timestamps of events recorded from completed sessions, which introduces some error since the user can pause and skip during the segment.

2.2. Session progress measurements

Let \(t_c\) denote the time that user exits the session in video’s timeline. The true session progress is given by,

\[
\text{session progress} = \frac{t_c}{\text{video length}} \times 100\%
\]

As we stated in Section 2.1, we cannot measure the true session progress due to data limitations.

The pointer at which a user exits a session can be approximated by,

- approximation \(E_1\), the beginning of segment (assuming the user exits after the start of the segment),
- approximation \(E_2\), the middle of the segment (assuming the user exits in the middle of the segment).

2.3. Cleaning and structuring

We first aggregate the logs to a session table, in which a row is a session. And from the session table we estimate segment length for each video. Then for each session we measure the session progress with the four approximated measurements introduced above. For the duplicate sessions (same user watches same video several times), we only keep the session with highest session progress. Besides, since smaller number of segments in video will introduce more error in the estimation of session progress, we only keep the video with 12 segments, the maximum we have. The last step is aggregate the session table with measurements of session progress to user by video matrix.

Considered that work in this section is both memory and storage intensive, we use Elastic MapReduce and EC2 on Amazon Web Service.

3. DATA ANALYSIS USING LOW RANK MATRIX APPROXIMATION

To make the prediction of session progress, we assume our \(Z_{sp}\) is approximately low-rank, i.e., there exists a low-rank
matrix that can approximate \( Z_{\text{sp}} \) with a low level fitting error. A relevant popular problem is the completion of User by Movie rating matrix in the Netflix challenge, in which methods based on low-rank model outperform other single model methods [1].

Let \( X_{M \times N} \) be a partially observed matrix we want to complete, \( \Omega \) be the set of indices of observed entries, and \( \Omega^c \) be the set of indices of unobserved entries. We want to find a matrix \( Z_{\text{sp}} \) that best approximates \( X \). To be more specific, in a lot of cases, \( X \) is large in size (typically tall, i.e., \( M \gg N \)), and only a few entries of \( X \) are observed.

Borrowed from [8], the projection operator \( P_{\Omega}(\cdot) \) is defined by:

\[
P_{\Omega}(Y)(i,j) = \begin{cases} Y_{i,j} & \text{if } (i,j) \in \Omega \\ 0 & \text{if } (i,j) \notin \Omega, \end{cases}
\]

In this setting we choose Root Mean Square Error (RMSE) as the measurement of performance, so our goal is to minimize \( \| P_{\Omega}(X - Z) \|_F^2 \), where \( \| \cdot \|_F \) denotes Frobenius Norm.

Consider the following optimization problem:

\[
\text{minimize} \quad \text{rank}(Z) \\
\text{subject to} \quad P_{\Omega}(X) = P_{\Omega}(Z)
\]

Generally (3) is rigid to noise and NP-hard [9]. Its noisy version is given by:

\[
\text{minimize} \quad \text{rank}(Z) \\
\text{subject to} \quad \| P_{\Omega}(X - Z) \|_F^2 \leq \delta
\]

But the rank objective function in (4) makes it combinatorially hard generally [10].

Considering that nuclear norm is often used as a convex relaxation of rank [3, 4, 11], Candès and Plan [12] proposed a convex relaxation to (4):

\[
\text{minimize} \quad \| Z \|_* \\
\text{subject to} \quad \| P_{\Omega}(X - Z) \|_F^2 \leq \delta
\]

Problem (5) can be solved efficiently for using modern convex optimization methods [13] when the problem size is small. However, these second order methods can become prohibitively expensive if the dimensions of the matrix get large [8]. Mazumder et al. [6] reformulated (5) in Lagrange form

\[
\text{minimize} \quad \frac{1}{2} \| P_{\Omega}(X - Z) \|_F^2 + \lambda \| Z \|_*.
\]

\( \lambda \) in (6) is the regularization parameter controlling the training error. Mazumder et al. [6] proposed an iterative algorithm SOFT-IMPUTE to compute a series of solutions to (6) with respect to a grid of values for \( \lambda \). SOFT-IMPUTE is efficient in dealing with matrix with large size, which is critically important for us, because the potential huge size of \( Z_{\text{sp}} \). We use SOFT-IMPUTE for our prediction in the rest of the paper.

4. EXPERIMENTS AND RESULTS

4.1. Experimental Setup

Our original structured data contains a large number of users and videos while observed session progresses are very limited, leading to a 99.85% sparse user by video matrix. And more importantly, the majority of the rows have only one observed entries. It’s generally true that, the less observed entries a row has, the more difficult it is to predict its unobserved entries, so does the columns. We remove those users with less than 4 observed entries and videos with less than 60 observed entries from the original user by video matrix, leading to a 3796 \times 117 matrix with 19,589 entries observed, or 95.59% sparse.\(^1\)

The data is split into two subsets, the training data and the test data, in the way that each row has approximately the same number of observed entries in test data. Prediction of test data is made only with knowledge of training data. And parameters for SOFT-IMPUTE, \( \lambda \) can be chosen by cross-validation within the training data.

4.2. Results

4.2.1. Performance of low-rank matrix completion method

In this paper, we want to compare the prediction results of different datasets, which have different noise levels or measurement scales. So it’s not fair to simply look at RMSE when comparing the performances of prediction across these matrices. Here we use the ratio of improvement from mean methods, Global Mean or Best Mean, as measurement of performance across different datasets.

Table 3 and Table 4 compare performance of low-rank matrix completion (LM) method\(^2\) with that of Best Mean and Global Mean, which does show that low-rank matrix completion method achieves an 83.7% average improvement from Best Mean, and an 13.288% average improvement from Global Mean. However, compared to Netflix data, Column Mean (‘Best Mean’ for Netflix data) achieves an RMSE of 1.0528, Global Mean achieves an RMSE of 1.1296 [1], and a rank-95 solution achieves an RMSE of 0.9497 [6] with a 9.79% improvement from Best Mean and 15.93% improvement from Global Mean, which is considerably higher than on our data. Note that Netflix data has an even higher sparsity level.

The unremarkable performance of LM on the session progress prediction leads us to wonder that, whether our data has high-level noises or the low-rank assumption simply doesn’t hold. While due to the limitations of our data,

\(^1\)Netflix dataset is 98.8% sparse, more than 3% higher than ours. However, later in Section 4.3 the empirical results indicates that, when the matrix size varies, sparsity level is no longer a good indicator of performance of SOFT-IMPUTE.

\(^2\)We use SOFT-IMPUTE to get the low-rank matrix completion results, same for the rest of the paper.
we are not able to conduct further verification experiments on our current data. Instead, we design a series of experiments on Netflix data, which indicates further improvement of LM on the session progress prediction when more entries observed or matrix size grows, as is shown in Section 4.3.

### Table 1. Performances of Best Mean and LM

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Best Mean</th>
<th>LM</th>
<th>Improve</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_1S_1$</td>
<td>0.2111</td>
<td>0.2110</td>
<td>0.003%</td>
</tr>
<tr>
<td>$E_1S_2$</td>
<td>0.2271</td>
<td>0.2230</td>
<td>1.816%</td>
</tr>
<tr>
<td>$E_2S_1$</td>
<td>0.2085</td>
<td>0.2088</td>
<td>-0.139%</td>
</tr>
<tr>
<td>$E_2S_2$</td>
<td>0.2320</td>
<td>0.2281</td>
<td>1.666%</td>
</tr>
<tr>
<td>average</td>
<td>0.2197</td>
<td>0.2177</td>
<td>0.837%</td>
</tr>
</tbody>
</table>

### Table 2. Performances of Global Mean and LM

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Global Mean</th>
<th>LM</th>
<th>Improve</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_1S_1$</td>
<td>0.2379</td>
<td>0.2110</td>
<td>11.291%</td>
</tr>
<tr>
<td>$E_1S_2$</td>
<td>0.2529</td>
<td>0.2230</td>
<td>11.826%</td>
</tr>
<tr>
<td>$E_2S_1$</td>
<td>0.2350</td>
<td>0.2088</td>
<td>11.190%</td>
</tr>
<tr>
<td>$E_2S_2$</td>
<td>0.2811</td>
<td>0.2281</td>
<td>18.842%</td>
</tr>
<tr>
<td>average</td>
<td>0.2517</td>
<td>0.2177</td>
<td>13.288%</td>
</tr>
</tbody>
</table>

### 4.3. Comparison with Netflix data

#### 4.3.1. Generating ‘natural’ sub-matrix with controlled size and sparsity

Consider the potential effects of matrix size, sparsity and other features like distribution of observed entries, on performance of matrix completion methods, we want to generate sub-matrix with given size and sparsity. An intuitive and simple method to do that is, pick a dense submatrix and then randomly remove observed entries until reaching given sparsity. While the process of removing entries from rows or columns might loose the distribution of observed entries, which has been showed to affect the completion with low rank methods [14, 15], and make the data ‘unnatural’, i.e., having different features with the original data.

An alternative way to do that is by removing entire rows and columns. However, with this method it’s not intuitive to simultaneously guarantee the size and sparsity of the sub-matrix, yet not disturbing distribution of observed entries. Here we propose an efficient method to do it, formulated as follows.

Given a partially observed matrix $A_{M \times N}$, we want to generate one or more sub-matrices $A'$ by ‘picking’ rows and columns without manually removing observed entries. And we want that $A' \in \mathbb{R}^{(M' \pm 1) \times (N' \pm 1)}$ and sparsity of $A'$ is approximately $s$, where $M' \times N'$ and $s$ are given. Moreover, to make $A'$ have as similar features as $A$ as possible, we want the distribution of row sparsities and column sparsities of $A'$ similar to $A$.

Without loss of generality (WLOG), we assume that $M' \geq N'$. First we randomly choose $N'$ columns from $A$ (note that different $A'$ can be generated by choosing different columns at this step), and let the selected sub-matrix be the initial $A'$ where we start with. Then we generate an associative array $i2n$, in which each cell is the key-value pair corresponding to each user: $i \rightarrow n$, where $i$ is the index of the user and $n$ is the number of observed entries in current $A'$, with its pairs sorted by an ascending order of $n$. The main algorithm is given in Algorithm 1.

#### Algorithm 1 Generating ‘natural’ sub-matrix with controlled size and sparsity

**Require:** $A, i2n, M', N', s$.

1. **# WLOG,** assume that sparsity of initial $A' > s$.
2. **WHILE** (current sparsity of $A' > s$) {
   1.Advance the pointer of $i2n$;
   2. **IF** ($n$ of pointed cell in $i2n < s \times N'$) {
      1. unset pointed row from $i2n$;
      2. remove the corresponded row from $A'$;
      3. set the pointer of $i2n$ to the next cell;
   } **ELSE** {
      1. reset pointer of $i2n$;
   }
3. $i2n_r \leftarrow$ reverse($i2n$);
4. reset the pointers of $i2n, i2n_r$;
5. **WHILE** (size of $i2n \geq M'$) {
   1. **IF** (sparsity of current $A' \geq s$) {
      1. Advance the pointer of $i2n$;
      2. **IF** ($n$ of pointed cell in $i2n \leq s \times N'$) {
         1. unset pointed row from $i2n, i2n_r$;
         2. remove corresponded row from $A'$;
         3. set the pointer of $i2n$ to the next cell;
      } **ELSE** {
         1. reset pointer of $i2n$;
      }
   } **ELSE** {
      1. Advance the pointer of $i2n_r$;
      2. **IF** ($n$ of pointed cell in $i2n_r > s \times N'$) {
         1. unset pointed row from $i2n_r, i2n$;
         2. remove corresponded row from $A'$;
         3. set the pointer of $i2n_r$ to the next cell;
      } **ELSE** {
         1. reset pointer of $i2n_r$;
      }
   }
}
4.3.2. Results Comparison and Interpretation

First we generate a sub-matrix of Netflix data, denoted as $Z_{Nf}$, which has the same size and sparsity as $Z_{sp}$. The performances of LM on these two ‘similar’ matrices are given in Table 5, which shows that LM works even worse on $Z_{Nf}$ than on $Z_{sp}$. This indicates that, with this matrix size, 3796 × 117, such a sparsity level, 95.59%, is not enough to recover the low-rank structure for Netflix data.

Further we generated another two sets of sub-matrices, one with various sparsity levels while same matrix size as $Z_{Nf}$, and one with various matrix size while same sparsity level and matrix shape (the ratio of number of rows to number of columns) as $Z_{Nf}$. The results of LM method on the two sets of sub-matrices are given in Fig. 3, and Fig. 4 respectively. Fig. 3 shows that, when the sub-matrix size is fixed, the improvement of LM generally increases when sparsity decreases, i.e., performance of LM might be improved by obtaining more observations for the same set of users and videos. Fig. 4 shows more consistent increase of LM’s improvement when the matrix size grows but sparsity remains the same. This means that, even when the sparsity level cannot be decreased, LM’s performance might still be improved when the matrix size grows.

Table 3. Comparison of $Z_{Nf}$ and $Z_{sp}$

<table>
<thead>
<tr>
<th>Item</th>
<th>$Z_{Nf}$</th>
<th>$Z_{sp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>3796 × 117</td>
<td>3796 × 117</td>
</tr>
<tr>
<td>Sparsity</td>
<td>95.59%</td>
<td>95.59%</td>
</tr>
<tr>
<td>RMSE: Global Mean</td>
<td>0.2179</td>
<td>0.2517</td>
</tr>
<tr>
<td>RMSE: Best Mean</td>
<td>0.2018</td>
<td>0.2197</td>
</tr>
<tr>
<td>RMSE: SOFT-IMPUTE</td>
<td>0.2063</td>
<td>0.2177</td>
</tr>
<tr>
<td>Improve from Best Mean</td>
<td>-2.2%</td>
<td>0.837%</td>
</tr>
<tr>
<td>Improve from Global Mean</td>
<td>6.099%</td>
<td>13.288%</td>
</tr>
</tbody>
</table>

5. CONCLUSION AND FUTURE DIRECTIONS

We have shown that LM method does achieve certain improvements from Best Mean on $Z_{sp}$, even though the improvements are not so significant as we expected. While our experiments on Netflix data indicate that the matrix size and sparsity level might be the factors that limit the performance of LM method. Better improvements over Best Mean are expected by either observing more entries with given users and videos, or including more users and videos while maintaining same sparsity level.

Future work may go in several directions. First is to look for theoretical support for the empirical conclusions made in this paper. Besides, in some cases there is feature information available for the user (geographic, local hour, etc.) and video (genre, language, etc.), and some features might have implicit effects on the session progress. We might seek to incorporating the feature information into LM method.

6. REFERENCES


