STREAMING RANKING BASED RECOMMENDER SYSTEMS

Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, Quoc Viet Hung Nguyen
University of Queensland, Australia & Griffith University, Australia

July 17, 2018
Outline

• Introduction

• Challenges of Streaming Recommender Systems

• Effective Streaming Models

• Evaluation
Introduction

- Recommender systems
- Streaming data everywhere
  - eBay: more than 10 million transactions per day. [1]
  - Twitter: half a billion tweets are generated every day (around 6,000 tweets every second). [2]

Introduction

• Streaming data everywhere
  • eBay: more than 10 million transactions per day. [1]
  • Netflix: more than 3 million subscribers from mid-March 2013 to April 2013. [2]
  • Twitter: half a billion tweets are generated every day (around 6,000 tweets every second).

• Characteristics
  • Temporally ordered
  • Large-volume
  • High-velocity

Introduction

- **Traditional recommender systems - static**
  - Collaborative filtering & matrix factorization
  - Nearest neighbors and correlation coefficients are precomputed
  - New coming data in the streams cannot be integrated into the trained model efficiently
Introduction

• **Traditional recommender systems - static**
  
  • Collaborative filtering & matrix factorization
  
  • Nearest neighbors and correlation coefficients are precomputed
  
  • New coming data in the streams cannot be integrated into the trained model efficiently
Outline

• Introduction

• Challenges of Streaming Recommender Systems

• Effective Streaming Models

• Evaluation
Challenges

• One possible way
  • Learn parameters online with SGD
  • Update users’ preferences based on each new instance in stream

• Challenge
  • Short-term “memory”
    • The updates are only based on the most recent data
  • Tend to “forget” past observations
    • Cannot users’ long-term interests


Challenges

• Capture users’ drifted interests & model new users/items
  • Data in streams are temporarily ordered
  • Users’ preference tend to drift over time
    • A mother tends to be interested in different products for children as her child grows up
  • Avoid being overwhelmed by past observations
  • New users and new items arrive continuously
    • From 2007 to 2015, the number of registered users on Amazon increase from 76 millions to 304 millions

• Challenges
  • Large-volume and high-velocity
  • Identify and model new users and new items
**Challenges**

- **Overload** \(^{[1, 2, 3]}\)

  1 activity/sec → **Cost=2 sec/activity** → Throughput=0.5 activity/sec

  ![Input arrival timeline](image1)

  ![Output delivery timeline](image2)

  **Figure 1: A Simple Overload Scenario**

- Input rates > Available computing capacity
  - Process time for one activity: 2 secs
  - Input rate: 1 activity per second
  - Impossible to update model over every activity

---


[3] Nesime Tatbul, Ugur 
Challenges

• Reservoir based\(^1\)
  • Maintain and update only based on the instances in reservoir
  • Aim of reservoir: keep a memory of long-term records
  • Sampling rate: sample \(t^{th}\) instances with the probability \(\frac{1}{t}\)
  • Shortcoming: cannot capture users’ drifted interests and modeling new users or items

Outline

- Introduction
- Challenges of Streaming Recommender Systems
- Effective Streaming Models
- Evaluation
Effective Streaming Model

- **Offline model**
  - PMF model with BPR optimization
  - Ratings -> binary
  - Each user and item are assigned with a z-dimensional latent factor vector, denoted as $u_i$ and $v_j$ respectively.
  - Assumption: all scores are conditionally independent with each other given latent factors of users and items.
  - Score function of a rating:
    \[
    f(i, j; \Theta) = \sum_{z'=1}^{z'=z} u_{iz'} \times v_{jz'} \tag{1}
    \]
    \[
    P(D|\Theta) = \prod_{i=1}^{I} \prod_{j=1}^{J} Ber(x_{ij}|\sigma(f(i, j; \Theta))) \tag{2}
    \]
    where
    \[
    Ber(x|p) = \begin{cases} 
    p, & x = 1 \\
    1 - p, & x = 0 
    \end{cases}
    \]
Effective Streaming Model

**Offline model**

- Introduce the Gaussian priors over the latent parameters $\Theta$, the generative process of our model is as follows:
  - For each $i \in (1, \ldots, I)$, generate $u_i \sim \mathcal{N}(0, \rho_U^2 I)$;
  - For each $j \in (1, \ldots, J)$, generate $v_j \sim \mathcal{N}(0, \rho_V^2 I)$;
  - For each user $u_i$ and each item $v_j$, generate $x_{ij} \sim \text{Ber}(x_{ij} | < u_i, v_j >)$.

- BPR optimization framework.

\[
P(U, V | \mathcal{D}, \rho_U^2 I, \rho_V^2 I) \propto P(X | U, V)P(U | 0, \rho_U^2 I)P(V | 0, \rho_V^2 I) = \prod_{i=1}^{I} \prod_{j=1}^{J} \text{Ber}(x_{ij} | \sigma(f(i, j; \Theta))) \prod_{i=1}^{I} \mathcal{N}(u_i | 0, \rho_U^2 I) \prod_{j=1}^{J} \mathcal{N}(v_j | 0, \rho_V^2 I)
\]

\[
\mathcal{L} = - \sum_{x_{ij} \in \mathcal{D}^+} \log(\sigma(f(i, j; \Theta))) - \sum_{x_{ij} \in \mathcal{D}^-} \log(1 - \sigma(f(i, j; \Theta))) + \frac{\lambda_U}{2} \sum_{i=1}^{I} \| u_i \|^2 + \frac{\lambda_V}{2} \sum_{j=1}^{J} \| v_j \|^2
\]

\[
\mathcal{O} = - \sum_{x_{ij} \in \mathcal{D}^+} \left( \log(1 - \sigma(f(x_{ijn}; \Theta))) \right) + \sum_{n=1}^{N} \log(1 - \sigma(f(x_{ijn}; \Theta))) + \frac{\lambda_U}{2} \sum_{i=1}^{I} \| u_i \|^2 + \frac{\lambda_V}{2} \sum_{j=1}^{J} \| v_j \|^2
\]
Effective Streaming Model

- **Offline model**
  - How to sample the negative instances?
    - A uniform sampling (simple but efficient)
    - GAN based sampling (expensive to maintain a generator)
    - A statistic-based sampling (data far away from being complete)
  - **BPR optimization framework.**
    
    **Input:** Training set $\mathcal{D}$, the dimension of embeddings $z$;
    
    **Output:** The latent vectors $\Theta$ for users and items;

    ```
    iter = 0;
    while iter < m do
        Sample a positive example $x_{ij}$ from $\mathcal{D}^+$;
        Sample $N$ negative examples $x_{in}$ from $\mathcal{D}^-$ by fixing the user $u_i$;
        Update the gradients of the latent variables associated with users and items in the batch based on Equation 10 and 11;
        Update the latent variables by batch SGD;
        iter = iter + 1;
    end
    ```

    **Algorithm 1:** Offline Model Training
Effective Streaming Model

• Online model - SPMF
  • A general framework

Input: The current set of parameters \( \Theta^t = \{U, V\} \),
the current reservoir \( R = \{s_1, s_2, \ldots, s_{|R|}\} \),
a window of new data streams
\( W = \{e_{t+1}, e_{t+2}, \ldots, e_{t+|W|}\} \);

Output: The updated latent vectors \( \Theta^t + |W| \) for users and items;

1. while next window of data have not arrived do
2.     Sample a data instance \( x_{ij} \) from \( R \cup W \);
3.     Sample \( N \) negative examples \( x_{in} \) from \( D_t^- \) by fixing the user \( u_i \);
4.     Update the gradients of the latent variables associated with users
      and items in the batch based on Equation 10 and 11;
5.     Update the latent variables by batch SGD;
6. end

Algorithm 2: Online Model Training

• Negative sampling: treat all the data outside \( W \cup R \) as negative
• How to sample positive data instances under the overload assumption?
Effective Streaming Model

- **Online model**
  - Efficiently and correctly sample positive instances

**Problem 1. (Model Updating in On-line Setting)** Given the set of parameters $\Theta^t$ until time $t$, the maintained reservoir $\mathcal{R} = \{s_1, s_2, \ldots, s_{|\mathcal{R}|}\}$, and a window of new data streams $\mathcal{W} = \{e_{t+1}, e_{t+2}, \ldots, e_{t+|\mathcal{W}|}\}$, the problem is updating the parameters $\Theta_t$ based on $\mathcal{W}$ and $\mathcal{R}$ before next window of new data streams arrive, with the goal to maximize the Equation 12.

$$P(\mathcal{W}, \mathcal{R}|\Theta_{\mathcal{W} \cup \mathcal{R}}^{t+|\mathcal{W}|}) = \prod_{w=1}^{|\mathcal{W}|} \text{Ber}(e_{t+w} | \sigma(f(e_{t+w}; \Theta_{\mathcal{W} \cup \mathcal{R}}^{t+|\mathcal{W}|}))) \times \prod_{r=1}^{|\mathcal{R}|} \text{Ber}(s_r | \sigma(f(s_r; \Theta_{\mathcal{W} \cup \mathcal{R}}^{t+|\mathcal{W}|})))$$

where $\Theta_{\mathcal{W} \cup \mathcal{R}}^{t+|\mathcal{W}|}$ is the model parameters obtained at time $t + |\mathcal{W}|$ after updating based on $\mathcal{W} \cup \mathcal{R}$, and $e_{t+w}$ and $s_r$ are data instances in new stream window $\mathcal{W}$ and the maintained reservoir $\mathcal{R}$ respectively.
Naïve idea: sample the data instances that are expected to change the model most

• Compute current model’s predictability of each data instance (defined by f)

• Maintain the reservoir

• Rank data according to f

• Compute the sampling probability of each data instance according to their ranks

• Sample and update the model

Algorithm 3: Improved Online Model Training
Outline

• Introduction

• Challenges of Streaming Recommender Systems

• Effective Streaming Models

• Evaluation
Evaluation

• Important settings

• Datasets
  • Netflix: 100 millions ratings, 480 thousand movies, 17 thousand movies
  • Movielens: about 1 million ratings, 4 thousand movies, 6 thousand users
  • Taobao: 14 millions ratings, 462 thousand clothes, 1 million users

• Comparative methods

• State-of-art
  • sRec\textsuperscript{[1]}: latest, update based on all the incoming data and the posterior probabilities at the previous moment.
  • RMFX\textsuperscript{[2]}: pairwise, reservoir based, no positive sampling
  • RKMF\textsuperscript{[3]}: update model on selected data and assign larger probabilities to new users and new items
  • WRMF\textsuperscript{[4]}: not designed for streaming data and we assume that it can access the whole data for training. It is expected to set up an upper bound for the on-line approaches.

---

Evaluation

• Important settings
  
  • Comparative methods
    
    • Three variant versions of our SPMF model
      
      • SPMF-S1: sample the positive data instances randomly
      
      • SPMF-S2: without maintaining the reservoir, the positive data are sampled from $W$ only
      
      • SPMF-S3: without exploiting $W$. The positive data are sampled from $R$ only.

• Dataset division
  
  • Order all the instances based on temporary information
• **Dataset division**

  • Order all the instances based on temporary information
  • Divide all data into halves
    • Base training set: considered as the historical data pool to train the hyper parameters
    • Candidate set: used to mimic the on-line streaming inputs
  • Mimic the on-line stream
    • Divide the candidate set into five slices sequentially
    • Each slice is treated as a test set $\mathcal{D}^{test}$
    • All the data prior to the target test set in the candidate set are used for on-line learning
    • E.g., target test set is $\mathcal{D}_{i}^{test}$, all the sets $\mathcal{D}_{j}^{test}$, where $j < i$, are used for on-line learning and each one of them is treated as an input window $W$.
    • For users who have no activities before $\mathcal{D}_{i}^{test}$, we recommend the most popular items for all comparison methods
Evaluation

• **Evaluation metric**
  - The popular $Hits@n = \frac{\#hit@n}{|D_{test}|}$

• **Recommendation Effectiveness**
  - Simulation of overload situation
    - The time used to update RMFX (one comparison method) for $\frac{|W|}{2}$ iterations.
  - Size of reservoir
    - $\frac{|D|}{20}$
  - Statistics soundness
    - Ran all the methods 30 times and use the average results
Evaluation results

1. On both Movielens and Netflix, WRMF performs best among all the methods.

2. SPMF achieves very competitive results compared with WRMF under the limitation of memory and training time.

3. All methods perform worse on Taobao a much sparser data. Another possible reason is that Taobao has much larger numbers of test cases containing new items (about 45.86% on Taobao while less than 1% on other two datasets).

4. SPMF performs even better than WRMF in many cases on Taobao as WRMF is popularity based and is biased to the popular items.
Evaluation results

1. SPMF consistently outperforms the three variant versions, indicating the benefits brought by each factor.
2. SPMF-S2 performs second best on Movielens, demonstrating the advantage of wisely sampling positive outweigh the advantage of using reservoir.

Figure 5: Evaluation of Factors on Movielens dataset
Evaluation results

Figure 6: Performance for New Users and New Items

1. New users: test on the cases that contain users who have no history activities in the base training set $\mathcal{D}_{train}^\text{train}$
2. New items: no history records on $\mathcal{D}_{train}^\text{train}$
3. SPMF performs best in recommendation for new users and new items even compared with the batch based WRMF
4. RKMF performs best among the other streaming methods as it also prefers to update its model based on the data related to less popular users or items
Evaluation results

| n  | \[\frac{|W|}{16}\] | \[\frac{|W|}{8}\] | \[\frac{|W|}{4}\] | \[\frac{|W|}{2}\] | |W| | |W| \times 2 |
|----|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|
| 5  | 0.0123              | 0.0150              | 0.0163              | 0.0171              | 0.0173              | 0.0173              |
| 10 | 0.0237              | 0.0286              | 0.0306              | 0.0321              | 0.0324              | 0.0325              |
| 15 | 0.0338              | 0.0417              | 0.0437              | 0.0456              | 0.0460              | 0.0460              |
| 20 | 0.0435              | 0.0535              | 0.0573              | 0.0601              | 0.0609              | 0.0608              |
| 25 | 0.0519              | 0.0640              | 0.0683              | 0.0708              | 0.0717              | 0.0718              |
| 30 | 0.0601              | 0.0739              | 0.0793              | 0.0818              | 0.0828              | 0.0828              |

Table 1: \textit{Hits@n} of SPMF under Varying Loads

1. the results of $D_2^{test}$ on Movielens
2. As the number of iterations increase, the recommendation performance first increase rapidly, and then the values become steady.
Another line of work

• **Qinyong Wang**, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang and Zi Huang. "Neural Memory Streaming Recommender Networks with Adversarial Training". KDD'18
Thank you!