电子科技大学
University of Electronic Science and Technology of China

Matrix Completion and Hawkes Process in Recsys

((Bai)|(Bo))lin Feng

Data Mining Lab, UESTC

December 23, 2019

"Bai" preferred.

Outline

Matrix Completion and Its Applications

- Matrix Completion[?][?]
- Scalable Demand-Aware Recommendation[?]

Hawkes Process and Its Applications

- Hawkes Process
- Neural Hawkes Process[?]
- CTRec[?]

Q&A

Keywords:

recommendation system · matrix completion · Hawkes process

Motivation

"Dear Amazon, I bought a toilet seat because I needed one. Necessity, not desire. I do not collect them. I am not a toilet seat addict."[0]

[0]https://twitter.com/GirlFromBlupo/status/982156453396996096

1. Matrix Completion

(1). Motivation

Matrix completion is the task of filling in the missing entries of a partially observed matrix.

$$\begin{bmatrix} \checkmark & ? & ? & ? & \checkmark & ? \\ ? & ? & \checkmark & \checkmark & ? & ? \\ \checkmark & ? & ? & \checkmark & ? & ? \\ ? & ? & \checkmark & ? & ? & \checkmark \\ \checkmark & ? & ? & ? & ? & ? \\ ? & \checkmark & ? & ? & \checkmark & ? \\ ? & ? & \checkmark & \checkmark & ? & ? \end{bmatrix}$$

- In general, it is NP-hard.
- But under additional assumptions, it can be solved efficiently.

Under low-rank restriction (non-convex, still NP-hard):

$$\begin{aligned} \min_{X} \quad & \text{rank}(X) \\ \text{s.t.} \quad & X_{ij} = M_{ij} \quad (i,j) \in \Omega \end{aligned}$$

Convex relaxation:[1]

$$\begin{aligned} \min_{X} \quad & ||X||_* \\ \text{s.t.} \quad & X_{ij} = M_{ij} \quad (i,j) \in \Omega \end{aligned}$$

Now this can be solved efficiently.

[1] Candès, Emmanuel J., and Benjamin Recht. *Exact matrix completion via convex optimization*. Foundations of Computational mathematics 9.6 (2009): 717.

2. Scalable Demand-Aware Recommendation[2]

(1). Motivation

- Item utility = *form utility* + *time utility*.
- traditional CF models only considers *form utility* $\longrightarrow$ tries to incorporate both utilities.

[2] Jinfeng Yi, Cho-Jui Hsieh, Kush R. Varshney, etc. *Scalable demand-aware recommendation*. In NIPS, 2017.

2. Scalable Demand-Aware Recommendation

(2). PU learning for matrix completion[3]

- Aims to recover the underlying matrix $M \in \mathbb{R}^{m \times n}$.
- Y is a **clean** 0-1 matrix observed from M by thresholding process: $Y_{ij} = \mathbb{1}\{M_{ij} > \tau\}$ and assume that only a subset of positive entries of Y are observed, denoted as A. But unfortunately, it is impossible to recover M, recover Y instead.
- Label-dependent loss:

$$\begin{aligned}\hat{X} &= \underset{X:||X||_* \leq t}{\arg\min} \sum_{i,j} l_\alpha(X_{ij}, A_{ij}) \\ &= \underset{X:||X||_* \leq t}{\arg\min}\ \alpha \sum_{i,j:A_{ij}=1} (X_{ij}-1)^2 + (1-\alpha) \sum_{i,j:A_{ij}=0} X_{ij}^2\end{aligned}$$

  which can be solved very efficiently by optimization methods.
- Recover Y by $\bar{X}_{ij} = \mathbb{1}\{\hat{X}_{ij} > \tau\}$.

---

[3] Cho-Jui Hsieh, Nagarajan Natarajan, and Inderjit S. Dhillon. *PU learning for matrix completion*. In ICML, 2015.

2. Scalable Demand-Aware Recommendation

(3). Loss Functions

Tensor nuclear norm minimization(TNNM) problem:

$$\min_{\mathcal{X}\in\mathbb{R}^{m\times n\times l},\mathbf{d}\in\mathbb{R}_+^r} \eta \sum_{ijk:p_{ijk}=1} \max[1-\overbrace{(\underbrace{x_{ijk}}_{\text{form utility}} - \underbrace{\max(0, d_{c_j} - t_{ic_jk})}_{\text{time utility}})}^{\text{overall utility}}, 0]^2$$

$$+ (1-\eta) \sum_{ijk:p_{ijk}=0} x_{ijk}^2 + \lambda ||\mathcal{X}||_*^2$$

- $\mathcal{P} \in \{0,1\}^{m\times n\times l}$ denotes the purchase history.
- $\mathcal{X} \in \mathbb{R}^{m\times n\times l}$ denotes the underlying form utility.
- $d \in \mathbb{R}_+^r$ with d_i denoting the underlying duration of category i.
- $\mathcal{T} \in \mathbb{R}^{m\times n\times l}$ where t_{ic_jk} denotes the number of time slots between user i's most recent purchase within item category c_j until time k.
- A binary utility indicator $\mathcal{Y} \in \{0,1\}^{m\times n\times l}$ with

$$y_{ijk} = \mathbb{1}[x_{ijk} - \max(0, d_{c_j} - t_{ic_jk}) > \tau].$$

3. Hawkes Process[4]

- The Hawkes process is a counting process that models a sequences of 'arrivals' of some type over time and each arrival excites the process.
- Conditional intensity function(CIF) of a counting process $N(t)$ with associated histories $\mathcal{H}(\cdot)$:

$$\lambda^*(t) = \lim_{h \to 0} \frac{\mathbb{E}\{N(t+h) - N(t) | \mathcal{H}(t)\}}{h}$$

- CIF for Hawkes process:

$$\lambda^*(t) = \boxed{\lambda} + \boxed{\sum_{i:t_i<t} \mu(t - t_i)}$$

background intensity

excitation function

Generally, $\mu(\cdot)$ takes the form of $\mu(t) = \alpha e^{-\beta t}$.

[4]Patrick J. Laub, Thomas Taimre, Philip K. Pollett. *Hawkes Process*. arXiv:1507.02822.

3. Hawkes Process

- The log-likelihood of the Hawkes model given observations I is:[5]

$$
\begin{aligned}
l &= \left(\sum_{i=1}^{k} \log P\left(t_i | \mathcal{H}_i\right) \right) + \log P(t_{k+1} > T | \mathcal{H}_I) \\
&= \sum_{i=1}^{k} \log \lambda^*(t_i) - \int_0^{t_k} \lambda^*(u) \mathrm{d}u
\end{aligned}
$$

[5]Proof can be found in the supplementary material of [**?**]

4. Neural Hawkes Process[7]

(1). Motivation

$$\lambda^*(t) = \lambda + \sum_{i:t_i<t} \mu(t - t_i)$$

- The Hawkes process always assumes that the excitation is ① positive, ② linearly additive over past events, ③ exponentially decaying with time.
- Neural Hawkes process expands the expressivity by generalizing it to ① both positive and negative, ②non-linear, ③ waving-curve influences.[6]

[6] Author's answer at zhihu in Chinese.

[7] Hongyuan Mei and Jason M Eisner. *The neural hawkes process: A neurally self-modulating multivariate point process*. In NeurIPS, 2017.

4. Neural Hawkes Process

(2). A simple generalization

Original intensity function (with only self-exciting):

$$\lambda_k(t) = \mu_k + \sum_{h:t_h<t} \alpha_{k_h,k} e^{\left(-\delta_{k_h,k}(t-t_h)\right)}, \quad \mu_k \geq 0, \alpha_{j,k} \geq 0$$

A generalization with self-inhibition:

$$\tilde{\lambda}_k(t) = \mu_k + \sum_{h:t_h<t} \alpha_{k_h,k} e^{\left(-\delta_{k_h,k}(t-t_h)\right)}, \quad \mu_k \in \mathbb{R}, \alpha_{j,k} \in \mathbb{R}$$

$$\lambda_k(t) = f_k\left(\tilde{\lambda}_k(t)\right) = s_k \log\left(1 + \exp\left(\frac{\tilde{\lambda}_k(t)}{s_k}\right)\right)$$

where the "*softplus*" function $f_k(x)$ ensures positive results of the intensity function.

4. Neural Hawkes Process

(3). The neural generalization

$$\lambda_k(t) = f_k\left(\boldsymbol{w}_k^\top \boldsymbol{h}(t)\right), \qquad \boldsymbol{h}(t) = \text{cLSTM}(x)$$

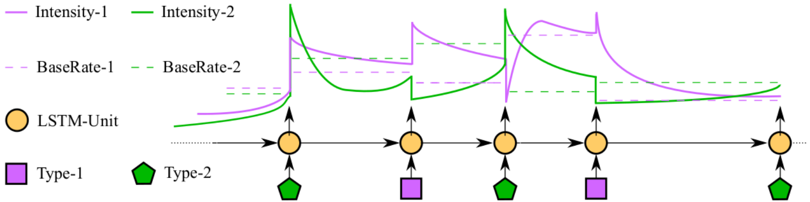

Figure: Drawing an event stream from a neural Hawkes process. Type-1 excites itself but inhibits Type-2. Type-2 excites itself, and excites or inhibits Type-1 according to #(Type-2) is odd or even.

---

cLSTM stands for **continuous-time LSTM**.

Image source: [**?**]

4. More About Hawkes Process and Point Process

(4). References

Papers:

- A Dirichlet Mixture Model of Hawkes Processes for Event Sequence Clustering, NeurIPS'17[**?**]
- Learning Granger Causality for Hawkes Processes, ICML'16[**?**]
- Learning Hawkes Processes from Short Doubly-Censored Event Sequences, ICML'17[**?**]
- Learning Conditional Generative Models for Temporal Point Processes, AAAI'18[**?**]
- Wasserstein Learning of Deep Generative Point Process Models, NeurIPS'17[**?**]
- Time is of the Essence: A Joint Hierarchical RNN and Point Process Model for Time and Item Predictions, WSDM'19[**?**]
- . . .

Group:

- Hongyuan Zha @ Georgia Institute of Technology

5. CTRec[8]

(1). Framework: demand-aware Hawkes process

- Conditional intensity function:

$$\lambda_i(t;\theta) = f(\underbrace{\boldsymbol{w}_i^{\text{item}\top} \cdot \boldsymbol{h}(t)}_{\text{short-term}} + \underbrace{w_i^{\text{attri}\top} \cdot \boldsymbol{\vartheta}(t)}_{\text{long-term}} + \underbrace{\boldsymbol{w}_i^{\text{user}\top} \cdot \boldsymbol{u}}_{\text{basic demands}})$$

, where $f(x) = \frac{s}{1+\exp(\frac{-x}{s})}$.

- The parameters here $\boldsymbol{w}_i^{\text{item}}$, $\boldsymbol{w}_i^{\text{attri}}$, $\boldsymbol{w}_i^{\text{user}}$, are learnt by neural Hawkes process.
- Then the probability of item i will be purchased at time t:

$$p_i(t;\theta) = \lambda_i(t;\theta) \exp\left(-\int_{t_o}^{t} \lambda_i(s;\theta)ds\right)$$

- Expectation next purchase time $\hat{t}_{\text{next}}$ of item i is:

$$\hat{t}_{\text{next}} = \int_{t_o}^{+\infty} t \cdot p_i(t;\theta)\mathrm{d}t$$

[8] Ting Bai, Lixin Zou, Wayne Xin Zhao, etc. *CTRec: A Long-Short Demands Evolution Model for Continuous-Time Recommendation*. In SIGIR, 2019.

5. CTRec

(2). Short-term demands

- Convolutional representation of item i_{t_j}:

$$\boldsymbol{v}_{t_j} = \text{avg}\left\{\boldsymbol{\iota}_{t_j}^{k_1}, \ldots, \boldsymbol{\iota}_{t_j}^{k_m}\right\}$$

- Then feed all the item vectors $\boldsymbol{v} = \{\boldsymbol{v}_{t_1}, \boldsymbol{v}_{t_2}, \cdots, \boldsymbol{v}_{t_n}\}$ into a time-aware LSTM to capture the short-term demands:

$$h(t) = \text{LSTM}(v)$$

$$\lambda_i(t;\theta) = f(\underbrace{\boldsymbol{w}_i^{\text{item}\top} \cdot h(t)}_{\text{short-term}} + w_i^{\text{attri}\top} \cdot \boldsymbol{\vartheta}(t) + \boldsymbol{w}_i^{\text{user}\top} \cdot \boldsymbol{u})$$

5. CTRec

(3). Long-term demands

- Let $\mathcal{D} \in \mathbb{R}^{|U| \times |A| \times |A|}$ be the estimated purchase time distance matrix of items for all users.
- Self-attentive component:

$$\alpha_{t,t_j} = \boldsymbol{h}\left(t_j\right)^\top \boldsymbol{i}_t - \lambda \log\left(\max\left\{\gamma, d^u_{a_t,a_{t_j}} - \Delta^u_{a_t,a_{t_j}}\right\}\right)$$

- Long-term demands:

$$\boldsymbol{\vartheta}_t = \sum_{j=1}^{n} \frac{\exp\left(\alpha_{t,t_j}\right)}{\sum_{q=1}^{n} \exp\left(\alpha_{t,t_q}\right)} \boldsymbol{h}\left(t_j\right)$$

$$\lambda_i(t;\theta) = f(\boldsymbol{w}_i^{\text{item}\top} \cdot \boldsymbol{h}(t) + \underbrace{w_i^{\text{attri}\top} \cdot \boldsymbol{\vartheta}_t}_{\text{long-term}} + \boldsymbol{w}_i^{\text{user}\top} \cdot \boldsymbol{u})$$

5. CTRec

(4). Loss function

Maximize the log-likelihood of ovserving items in $I_{t_n}^u$, which can be defined as:[9]

$$
\begin{aligned}
\ell\left(I_{t_n}^u;\theta\right) &= \sum_{j=1}^{n} \log \Pr\left(i_{t_j} | I_{t_j}^u, \Delta t_j\right) \\
&= \underbrace{\sum_{j=1}^{n} \log \lambda_{i_{t_j}}\left(t_j;\theta\right)}_{\text{purchase}} - \underbrace{\sum_{i_{\text{neg}}\in I} \int_{t_1}^{t_n} \lambda_{i_{\text{neg}}}(t)dt}_{\text{non-purchase}} \\
&= \sum_{i_{\text{neg}}\in I} \sum_{j=1}^{n} \left(\frac{1}{|I|} \log \lambda_{i_{t_j}}\left(t_j;\theta\right) - \int_{t_{j-1}}^{t_j} \lambda_{i_{\text{neg}}}(t)dt \right)
\end{aligned}
$$

[9] Hongyuan Mei and Jason M Eisner. *The neural hawkes process: A neurally self-modulating multivariate point process*. In NeurIPS, 2017.

Q&A

Thanks!

fengbailinn@gmail.com

Data Mining Lab, UeSTC

continuous-time LSTM

The continuous-time LSTM is defined as follows:

$$\lambda_k(t) = f_k(\boldsymbol{w}_k^\top \boldsymbol{h}(t))$$

$$\boldsymbol{h}(t) = \boldsymbol{o}_i \odot (2\sigma(2\boldsymbol{c}(t)) - 1), \quad \text{for } t \in (t_{i-1}, t_i]$$

$$\boldsymbol{c}(t) \stackrel{\text{def}}{=} \bar{\boldsymbol{c}}_{i+1} + (\boldsymbol{c}_{i+1} - \bar{\boldsymbol{c}}_{i+1})\exp(-\delta_{i+1}(t - t_i)), \quad \text{for } t \in (t_i, t_{i+1}]$$

Click to go back

Bibliography I