Causal Embeddings For Recommendation

Stephen Bonner & Flavian Vasile

Criteo Research

September 28, 2018

criteo.

Introduction

Classical Recommendation approaches:

- *A distance learning problem between pairs of products or between pairs of users and products* - measured with MSE and AUC.
- *A next item prediction problem that models the user behavior and tries to predict next action* - ranked Precision@K and Normalized Discounted Cumulative Gain (NDCG).

- **However** - both fail to model the inherent interventionist nature of recommendation, which should not only attempt to model the organic user behavior, but to *actually attempt to optimally influence it according to a preset objective.*

Recommendation Policy

- We assume a *stochastic policy* π_x that associates to each user u_i and product p_j a probability for the user u_i to be exposed to the recommendation of product p_j:

$$p_j \sim \pi_x(.|u_i)$$

- For simplicity we assume showing no products is also a valid intervention in $\mathcal{P}$.

Policy Rewards

- Reward r_{ij} is distributed according to an unknown conditional distribution r depending on u_i and p_j:

$$r_{ij} \sim r(.|u_i, p_j)$$

- The reward R^{π_x} associated with a policy π_x is equal to the sum of the rewards collected across all incoming users by using the associated personalized product exposure probability:

$$R^{\pi_x} = \sum_{ij} r_{ij}\pi_x(p_j|u_i)p(u_i) = \sum_{i} R_{ij}$$

criteo

Individual Treatment Effect

- The *Individual Treatment Effect (ITE)* value of a policy for a given user i and a product j for a policy π_x is defined as the difference between its reward and the control policy reward:

$$ITE_{ij}^{\pi_x} = R_{ij}^{\pi_x} - R_{ij}^{\pi_c}$$

- We are interested in finding the policy π^* with the *highest sum of ITEs*:

$$\pi^* = arg \max_{\pi_x} \{ITE^{\pi_x}\}$$

where: $ITE^{\pi_x} = \sum_{ij} ITE_{ij}^{\pi_x}$

Optimal ITE Policy

- For any control policy π_c, the best incremental policy π^* is the policy that shows deterministically to each user the product with the highest associated reward.

$$\pi^* = \pi_{det} = \begin{cases} 1, & \text{if } p_j = p_i^* \\ 0, & \text{otherwise} \end{cases}$$

IPS Solution For π^*

- In order to find the optimal policy π^* we need to find for each user u_i the product with the highest personalized reward r_i^*.
- In practice we do not observe directly r_{ij}, but $y_{ij} \sim r_{ij}\pi_x(p_j|u_i)$.
- Current approach: *Inverse Propensity Scoring (IPS)*-based methods to predict the unobserved reward r_{ij}:

$$\hat{r}_{ij} \approx \frac{y_{ij}}{\pi_c(p_j|u_i)}$$

criteo

Addressing The Variance Issues Of IPS

- Main shortcoming: IPS-based estimators do not handle well big shifts in exposure probability between treatment and control policies (products with low probability under the logging policy π_c will tend to have higher predicted rewards).
- Minimum variance $\pi^c = \pi^{rand}$. However, low performance!
- Trade-off solution: Learn from π^c a predictor for performance under π^{rand}

Our Approach: Causal Embeddings (CausE)

- We are interested in building a good predictor for recommendation outcomes under random exposure for all the user-product pairs, which we denote as $\hat{y}_{ij}^{rand}$.
- We assume that we have access to a large sample S_c from the logging policy π_c and a small sample S_t from the randomized treatment policy π_t^{rand}.
- To this end, we propose a multi-task objective that jointly factorizes the matrix of observations $y_{ij}^c \in S_c$ and the matrix of observations $y_{ij}^t \in S_t$.

Predicting Rewards Via Matrix Factorization

- We assume that both the expected factual control and treatment rewards can be approximated as linear predictors over the fixed user representations u_i:

$$y_{ij}^c \approx < u_i, \theta_j^c >, \text{ or } Y^c \approx U\Theta_c$$

$$y_{ij}^t \approx < u_i, \theta_j^t >, \text{ or } Y^t \approx U\Theta_t$$

- As a result, we can approximate the ITE of a user-product pair i, j as the difference between the two:

$$\widehat{ITE}_{ij} = < u_i, \theta_j^t > - < u_i, \theta_j^c > = < \theta_j^\Delta, u_i >$$

Joint Objective

$$L_t = L(U\Theta_t, Y_t) + \Omega(\Theta_t)$$
$$L_c = L(U\Theta_c, Y_c) + \Omega(\Theta_c)$$

Θ_t, Θ_c	parameter matrix of product representations for t, c
U	parameter matrix of user representations
L	arbitrary element wise loss function
$\Omega(\cdot)$	element wise regularization term

criteo

Joint Objective

$$L_t = L(U\Theta_t, Y_t) + \Omega(\Theta_t)$$
$$L_c = L(U\Theta_c, Y_c) + \Omega(\Theta_c)$$

Θ_t, Θ_c	parameter matrix of product representations for t, c
U	parameter matrix of user representations
L	arbitrary element wise loss function
$\Omega(\cdot)$	element wise regularization term

$$L_{CausE}^{prod} = \underbrace{L(U\Theta_t, Y_t) + \Omega(\Theta_t)}_{treatment\ \ task\ \ loss} + \underbrace{L(U\Theta_c, Y_c) + \Omega(\Theta_c)}_{control\ \ task\ \ loss} + \\ + \underbrace{\Omega(\Theta_t - \Theta_c)}_{regularizer\ \ between\ \ tasks}$$

Experimental Setup: Datasets

- We use the *MovieLens100K* and *MovieLens10M* explicit rating datasets (1-5). We process it as follows:
- We binarize the ratings y_{ij} by setting 5-star ratings to 1 (click) and everything else to zero (view only).
- We then create two datasets: regular (REG) and skewed (SKEW), each one with 70/10/20 train/validation/test event splits.

Experimental Setup: SKEW Dataset

- **Goal: Generate a test dataset that simulates rewards uniform expose π_t^{rand}.**
- Method:
  - Step 1: Simulate uniform exposure on 30% of users by rejection sampling.
  - Step 2: Split the rest of 70% of users in 60% train 10% validation
  - Step 3: Add to train a fraction of the test data (e.g. S_t) to simulate a small sample from π_t^{rand}.
- NB: In our experiments, we varied the size of S_t between 1% and 15%.

Experimental Setup: Exploration Sample S_t

We define 5 possible setups of incorporating the exploration data:

- **No adaptation** *(no)* - trained only on S_c.
- **Blended adaptation** *(blend)* - trained on the blend of the S_c and S_t samples.
- **Test adaptation** *(test)* - trained only on the S_t samples.
- **Product adaptation** *(prod)* - separate treatment embedding for each product based on the S_t sample.
- **Average adaptation** *(avg)* - average treatment product by pooling all the S_t sample into a single vector.

Method	MovieLens10M (SKEW)		
	MSE lift	**NLL lift**	**AUC**
BPR-no	–	–	0.693($\pm$0.001)
BPR-blend	–	–	0.711($\pm$0.001)
SP2V-no	+3.94%($\pm$0.04)	+4.50%($\pm$0.04)	0.757($\pm$0.001)
SP2V-blend	+4.37%($\pm$0.04)	+5.01%($\pm$0.05)	0.768($\pm$0.001)
SP2V-test	+2.45%($\pm$0.02)	+3.56%($\pm$0.02)	0.741($\pm$0.001)
WSP2V-no	+5.66%($\pm$0.03)	+7.44%($\pm$0.03)	0.786($\pm$0.001)
WSP2V-blend	+6.14%($\pm$0.03)	+8.05%($\pm$0.03)	0.792($\pm$0.001)
BN-blend	–	–	0.794($\pm$0.001)
CausE-avg	+12.67%($\pm$0.09)	+15.15%($\pm$0.08)	0.804($\pm$0.001)
CausE-prod-T	+07.46%($\pm$0.08)	+10.44%($\pm$0.09)	0.779($\pm$0.001)
CausE-prod-C	+**15.48**%($\pm$**0.09**)	+**19.12**%($\pm$**0.08**)	**0.814**($\pm$**0.001**)

Table 1: Results for MovieLens10M on the Skewed (SKEW) test datasets. We can observe that our best approach *CausE-prod-C* outperforms the best competing approaches *WSP2V-blend* by a large margin (21% MSE and 20% NLL lifts on the MovieLens10M dataset) and *BN-blend* (5% AUC lift on MovieLens10M).

criteo

Results

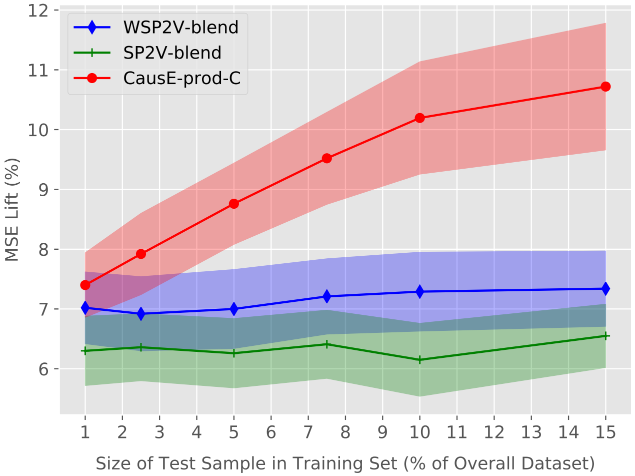

Figure 1: Change in MSE lift as more test set is injected into the blend training dataset.

Results

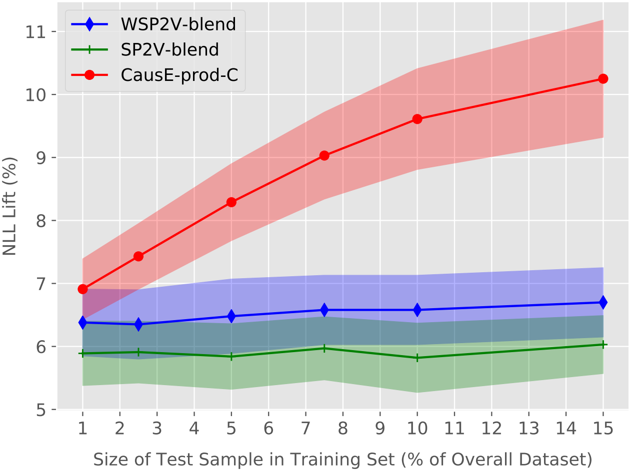

Figure 2: Change in NLL lift as more test set is injected into the blend training dataset.

Conclusions

- We have introduced a novel method for factorizing implicit user-item matrices that optimizes for *incremental recommendation outcomes.*
- We learn to predict user-item similarities under the uniform exposure distribution.
- *CausE* is an extension of matrix factorization algorithms that adds a regularizer term on the discrepancy between the product embeddings that fit the training distribution and their counter-part embeddings that fit the uniform exposure distribution.

https://github.com/criteo-research/CausE

Thank You!