Causal Embeddings for Recommendation

Presenter: YC Choi
2019.01.15

Contents

Abstract

- We are interested in finding the optimal treatment recommendation policy that maximizes the reward with respect to the control recommendation policy for each user
- known as the Individual Treatment Effect(ITE)

Definition and notation

Symbol	Definition
u_i	A user of the recommendation system
p_j	A product which the system can recommend
π_x	A recommendation policy (eq. 1)
π_c	The control recommendation policy. This represents the recommendation system used to create the training dataset.
π_t	The treatment recommendation policy. This represents the updated recommendation system.
π^{rand}	The fully random recommendation policy that shows any product with equal probability to all users.
r_{ij}	The true reward for recommending a product p_j to user u_i
y_{ij}	The observed reward for recommending product p_j to user u_i in the data. By comparison with r_{ij}, its value can be unknown.
R^{π_x}	The total reward for policy π_x (eq. 2)
$ITE_{ij}^{\pi_x}$	The difference between the reward for current and control policy (eq. 3)
p_i^*	The product with the highest reward for user u_i (eq. 6)
π^*	The best incremental recommendation policy (eq. 4)
S_c	A large set of training samples collected under the control recommendation policy
S_t	A smaller set of samples taken under a full-randomized recommendation policy

Definition and notation

- $p_j \sim \pi_x(.|u_j)$: a probability for the user u_i to be exposed to the recommendation of product p_j
- $r_{ij} \sim r(.|u_i, p_j)$: the true reward for recommending product p_j to user u_i
- $y_{ij} = r_{ij}\pi_x(p_j|u_j)$: the observed reward for the pair i,j of user-product according to the logging policy π_x

Definition and notation

- $R^{\pi_x} = \sum_{i,j} r_{ij}\pi_x(p_j, u_i)p(u_i) = \sum_{i,j} y_{i,j}p(u_i) = \sum_{i,j} R_{ij}$: the reward associated with a policy π_x
- $ITE_{ij}^{\pi_x} = R_{ij}^{\pi_x} - R_{ij}^{\pi_c}$: Individual treatment effect
- $\pi^* = \arg\max_{\pi_x} ITE^{\pi_x}$ where $ITE^{\pi_x} = \sum_{i,j} ITE_{ij}^{\pi_x}$

Lemma.1) 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.

Inverse Propensity Scoring

- In order to find the optimal policy pi^*, 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 $y_{ij} \sim r_{ij}\pi_x(p_j|u_j)$
- (IPS) Predict unobserved reward $\hat{r}_{ij} \approx \dfrac{y_{ij}}{\pi_c(p_j, u_i)}$
- Products with low probability under the logging policy π_c will tend to have higher predicted reward.
- One potential solution is then to use the biased data from the current π_c and learn to predict the outcomes under a randomized policy.

Inverse Propensity Scoring

- But Using uniform exposure recommendations(denoted as π^{rand}) is impossible in practice due to the resulting low recommendation quality.
- The proposed method will use randomized policy and control recommendation policy together.

The proposed method

- We assume the existence of two training samples
$S_c = \{(u_i, p_j^c, y_{ij}^c)\}_{i=1}^{M_c}$: very large sample of exposed users with outcomes collected with the control recommendation policy
$S_t = \{(u_i, p_j^t, y_{ij}^t)\}_{i=1}^{M_t}$: much smaller sample of exposed users with outcomes collected with the fully randomized recommendation policy

The proposed method

- The authors assume that both the expected factual control and treatment rewards can be approximated as linear predictors over the fixed user representation u_i
- $y_{ij}^c \approx < \theta_j^c, u_i >$
- $y_{ij}^t \approx < \theta_j^t, u_i >$
  where θ_j^c, θ_j^t are the control/treatment vectorial representations of product j.

The proposed method

- $l_{ij}^t = L(< \theta_j^t, u_i >, y_{ij}^t) + \Omega(\theta_j^t)$
  where L is an arbitrary loss function and $\Omega(.)$ is a regularization term over the weights of the model. Switching to matrix notation,

$$L_t = \sum_{(i,j,y_{ij}) \in S_t} l_{ij}^t = L(U\Theta_t, Y_t) + \Omega(\Theta_t)$$

- Using same way

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

- $L_{CausE}^{prod} = L(U\Theta_t, Y_t) + \Omega(\Theta_t) + L_c = L(U\Theta_c, Y_c) + \Omega(\Theta_c) + \Omega(\Theta_t - \Theta_c)$
- $L_{CausE} = L(\Gamma_t\Theta_t, Y_t) + \Omega(\Gamma_t, \Theta_t) + L_c = L(\Gamma_c\Theta_c, Y_c) + \Omega(\Gamma_c, \Theta_c) + \Omega(\Theta_t - \Theta_c)$

Algorithm

Algorithm 1: CausE Algorithm: Causal Embeddings For Recommendations

Input : Mini-batches of $S_c = \{(u_i^c, p_j^c, \delta_{ij}^c)\}_{i=1}^{M_c}$ and $S_t = \{(u_i^t, p_j^t, \delta_{ij}^t)\}_{i=1}^{M_t}$, regularization parameters λ_t, λ_c for the two joint tasks L_t and L_c and λ_{dist} the regularization parameter for the discrepancy between the two representations for products and users, learning rate η

Output: $\Gamma_t, \Gamma_c, \Theta_t, \Theta_c$ - User and Product Control and Treatment Matrices

Random initialization of $\Gamma_t, \Gamma_c, \Theta_t, \Theta_c$;
while *not converged* **do**
  Read batch of training samples;
  for ***each*** *training sample s in the batch:* **do**
    if $s \in S_c$ **then**
      Lookup the product index j and user index i in Θ_c, Γ_c and
      Update control product vector:
      $\theta_j^c \leftarrow \theta_j^c - \eta \nabla L_{CausE}^{prod}$
      Update control user vector: $\gamma_i^c \leftarrow \gamma_i^c - \eta \nabla L_{CausE}^{user}$
    end
    if $s \in S_t$ **then**
      Lookup the product index j and user index i in Θ_t, Γ_t and
      Update treatment product vector:
      $\theta_j^c \leftarrow \theta_j^c - \eta \nabla L_{CausE}^{prod}$
      Update treatment user vector:
      $\gamma_i^c \leftarrow \gamma_i^c - \eta \nabla L_{CausE}^{user}$
    end
  end
end
return $\Gamma_t, \Gamma_c, \Theta_t, \Theta_c$

Experiments

1. MovieLens10M
   - 71567 unique users, 10677 unique products.
2. Netflix
   - 480189 unique users, 17770 unique products.

Experiments

Method	MovieLens10M (SKEW)			Netflix (SKEW)		
	MSE lift	**NLL lift**	**AUC**	**MSE lift**	**NLL lift**	**AUC**
BPR-no	–	–	0.693(±0.001)	–	–	0.665(±0.001)
BPR-blend	–	–	0.711(±0.001)	–	–	0.671(±0.001)
SP2V-no	+3.94%(±0.04)	+4.50%(±0.04)	0.757(±0.001)	+10.82%(±0.02)	+10.19%(±0.01)	0.752(±0.002)
SP2V-blend	+4.37%(±0.04)	+5.01%(±0.05)	0.768(±0.001)	+12.82%(±0.02)	+11.54%(±0.02)	0.764(±0.003)
SP2V-test	+2.45%(±0.02)	+3.56%(±0.02)	0.741(±0.001)	+05.67%(±0.02)	+06.23%(±0.02)	0.739(±0.004)
WSP2V-no	+5.66%(±0.03)	+7.44%(±0.03)	0.786(±0.001)	+13.52%(±0.01)	+13.11%(±0.01)	0.779(±0.001)
WSP2V-blend	+6.14%(±0.03)	+8.05%(±0.03)	0.792(±0.001)	+14.72%(±0.02)	+14.23%(±0.02)	0.782(±0.002)
BN-blend	–	–	0.794(±0.001)	–	–	0.785(±0.001)
CausE-avg	+12.67%(±0.09)	+15.15%(±0.08)	0.804(±0.001)	+15.62%(±0.02)	+15.21%(±0.02)	0.799(±0.002)
CausE-prod-T	+07.46%(±0.08)	+10.44%(±0.09)	0.779(±0.001)	+13.97%(±0.02)	+13.52%(±0.02)	0.789(±0.003)
CausE-prod-C	**+15.48%(±0.09)**	**+19.12%(±0.08)**	**0.814(±0.001)**	**+17.82%(±0.02)**	**+17.19%(±0.02)**	**0.821(±0.003)**

Table 2: Results for MovieLens10M and Netflix on the Skewed (SKEW) test datasets. All three versions of the *CausE* algorithm outperform both the standard and the IPS-weighted causal factorization methods, with *CausE-avg* and *CausE-prod-C* also out-performing BanditNet. 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).