

Part 3. Towards more Causal Recommendations

Flavian Vasile

Criteo Research

June 28, 2018

Collaborators:

Special thanks to my colleagues:

- Stephen Bonner
- David Rohde

Structure of the talk

- What is wrong with our Recommender Systems?
- Product Recommendation in Ad Optimization
- Learning to Optimize for Incremental Treatment Effect
- CausE: Causal Embeddings For Recommendation
- Next steps

What is wrong with our
Recommender Systems?

Modern Recommendation Systems Research

Q: How does the literature on Large Scale Recommendation Systems look right now?

A: Most of the latest publications are talking about ways of using Deep Learning for Recommendation:

- **Matrix Factorization extensions:** Word2Vec, Deep and Wide, Neural MF, Node2Vec
- **Content-based recommendation:** CNNs for Image, Text, Sounds to compute item similarities
- **Next event prediction / user activity modeling:** RNNs, TCNs

Modern Recommendation Systems Research

We see great improvements in offline metrics!

- **Regression/Classification metrics:** MSE, AUC, NLL
- **Ranking metrics:** MPR, Precision@k, NDCG

What is wrong with this picture?

In the same time, as practitioners, we see difficulties in improving the Real World Recommendation models!

Offline - online metrics alignment for Recommendation is a recognized problem:

- **Previous work:** Missing Not At Random (MNAR) framework for Matrix factorization, Bandits Literature
- **New avenues:** *REVEAL: Offline evaluation for recommender systems Workshop at RecSys 2018* (this October in Vancouver)
<https://sites.google.com/view/reveal2018/home>

Let's take a step back...

How are we improving large-scale Recommender Systems in Real World

- Learn a supervised model from past user activity
- Evaluate offline and decide or not to A/B test
- A/B test
- If positive and scalable, roll-out
- If not, try to understand what happened and try to create a better model of the world using the same past data
- Repeat

The relationship between Reinforcement Learning and Recommendation

- We are doing Reinforcement Learning by hand!
- Furthermore, we are trying to do RL using the Supervised Learning Framework

Supervised learning with the wrong objective

Most of the time, we frame the problem either as a:

- Missing link prediction
- Next user event prediction

Supervised learning with the wrong objective

We are operating under the assumption that the best Recommendation Policy is in some sense the **optimal auto-complete of natural user behavior**

Supervised learning with the wrong objective

From the point of view of business metrics, **learning to autocomplete behavior is a great initial recommendation policy**, especially when no prior user feedback is available.

Supervised learning with the wrong objective

However, after a first golden age, where all offline improvements turn into positive A/B tests, the *naive Recommendation* optimization objective and the business objective will start to diverge.

Aligning the Recommendation objective with the business

- Of course, we could start incorporating user feedback that we collected while running the initial recommendation policies
- We should be able to continue bringing improvements using feedback that is now aligned with our business metrics (ad ctr, post click sales, dwell time, number of videos watched, ...)

Aligning the Recommendation objective with the business

However, this does not come without caveats...

Application

In the following sections, we will take a look on how to improve product recommendations in the context of online performance advertising.

Product Recommendation for Ad Optimization

Objective: Ad CTR Optimization

Toy World: An online advertising ecosystem with **one e-commerce website** (where the user can buy) and **one publisher website** (where we can show the ads).

Simplifying assumption: constant probability of buying once on-site.

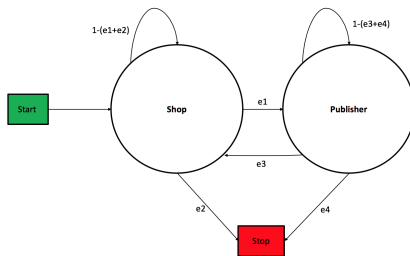


Figure 1: FSM of organic user behavior

Objective: Ad CTR Optimization

The goal of Recommendation: Show the product ad that maximizes CTR

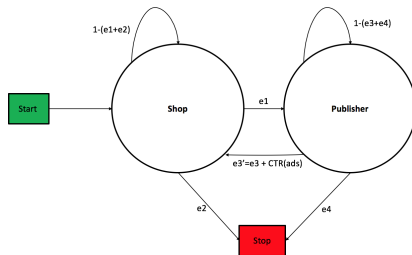
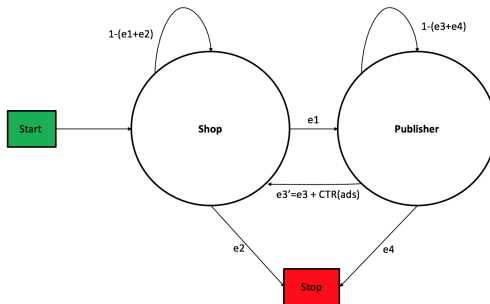


Figure 2: FSM of ad-influenced user behavior

Two alternative sources for the Implicit Feedback Matrix



- **Standard recommendation literature:** Look at the user shopping behavior matrix (left hand side matrix)
- **Computational advertising/Bandits literature:** Look at the user CTR matrix (right hand side matrix)

Factorization of Implicit Feedback User-Product Matrix

- **Implicit Feedback Matrix:** $\text{NbUsers} \times \text{NbProducts}$, Binary response, Very Sparse
- Many factorization objectives: MLE, Ranking ...
- **Very similar online performance!**

The Supervised Objective

Choose the best personalized ad ad_i^* for a user u_i , by framing the problem of CTR prediction as a *Maximum Likelihood Estimation (MLE)* problem which leads to the following formula:

$$ad_i^* = \operatorname{argmax}_j \{P^{\pi_0}(ad_j^{click} | u_i, ad_j^{view})\} \quad (1)$$

The Supervised Objective

$$ad_i^* = \operatorname{argmax}_j \{P^{\pi_0}(ad_j^{click} | u_i, ad_j^{view})\}$$

where:

- π_0 is the current recommendation policy: $ad_j^{view} \sim \pi_0(\cdot | u_i)$
- ad_j^{view} is the event of displaying an ad for product j
- ad_j^{click} is the event of clicking an ad for product j
- u_i is the profile of the user i

Shortcomings of the supervised objective

Three issues:

- **First**, in the worst case, simply fitting supervised models to your historical data (collected with an older logging policy) will suffer from selection bias and could give you with high confidence a bad recommendation policy - *Simpsons Paradox*
- **Second**, for models with limited capacity, you might spend too much fitting power on the wrong area of logging policy
- **Third**, even if the models have high capacity (DNNs) and the resulting estimators are somehow unbiased, the variance will be huge outside of the support of the logging policy!

Shortcomings - degree of severity

Three issues:

- **First** - Low (under some assumptions on how the causal graph looks like)
- **Second** - Medium (there are classes of models that are supposed to be able to handle changes in domain - see Gaussian Processes)
- **Third** - Very high! (counterfactual inference and exploration)

Shortcomings of the supervised approach

Let's look at the 3 problems in more detail

Simpsons paradox.

Simpsons paradox. An Example

	Overall CTR	Male	Female
Sport	0.5	0.4	0.8
Movie	0.6	0.3	0.7

Figure 3: Simpson's paradox example - Image source: Lihong Li WSDM Tutorial

<https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tutorial.pdf>

Simpsons paradox. An Example

Fraction of **males** and **females** who saw
"Sport" and "Movie"

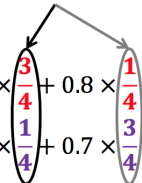

$$0.5 = 0.4 \times \frac{3}{4} + 0.8 \times \frac{1}{4}$$
$$0.6 = 0.3 \times \frac{1}{4} + 0.7 \times \frac{3}{4}$$

Figure 4: Mixing weights effect - Image source: Lihong Li WSDM Tutorial
<https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tutorial.pdf>

Simpsons paradox. An Example

- CTR without knowing gender:

$$P(\text{click} | \text{action} = \text{sport}) = 0.5$$

$$P(\text{click} | \text{action} = \text{movie}) = 0.6$$

- CTR knowing gender:

$$P(\text{click} | \text{action} = \text{sport}, \text{user} = m) = 0.4$$

$$P(\text{click} | \text{action} = \text{movie}, \text{user} = m) = 0.3$$

$$P(\text{click} | \text{action} = \text{sport}, \text{user} = f) = 0.8$$

$$P(\text{click} | \text{action} = \text{movie}, \text{user} = f) = 0.7$$

Simpsons paradox. An Example

Comparing the decisions before and after observing gender

Online traffic: 4 f, 4 m

Before: Decision: show movie recommendation to everyone

Offline Expected Payoff = $(4 * 0.6 + 4 * 0.6)/8 = 0.6$

Online Observed Payoff = $(4 * 0.3 + 4 * 0.7)/8 = 0.5$

The offline estimator is too optimistic, the ABTest will go bad!

After: Decision: show best arm (news sports/movie) for each gender

Offline Expected Payoff = $(4 * 0.4 + 4 * 0.8)/8 = 0.6$

Online Observed Payoff = $(4 * 0.4 + 4 * 0.8)/8 = 0.6$

The offline estimator is unbiased, the ABTest will go as expected!

Simpsons paradox in Recommendation Systems

Q: Can we check?

A: Yes, using Pearl's *Backdoor Criterion* we can check that under the following causal graph representing a generic Recommender System there is no Simpsons paradox:

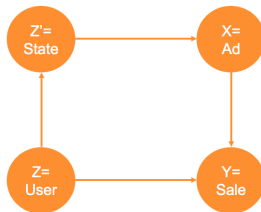


Figure 5: Causal Graph for Ad Recommendations

Simpsons paradox in Recommendation Systems

- The principle is simple: The paths connecting X and Y are of two kinds, causal and spurious.
- Causative associations are carried by the causal paths, namely, those tracing arrows directed from X to Y .
- The other paths carry spurious associations and need to be blocked by conditioning on an appropriate set of covariates. All paths containing an arrow into X are spurious paths, and need to be intercepted by the chosen set of covariates.

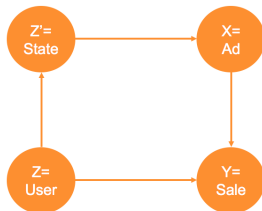


Figure 6: Causal Graph for Ad Recommendations

Simpsons paradox in Recommendation Systems

When dealing with a singleton covariate Z , as in the Simpsons paradox, we need to merely ensure that:

- Z is not a descendant of X - True
- Z blocks every path that ends with an arrow into X - True

Lesson: Always condition on all features used for acting!

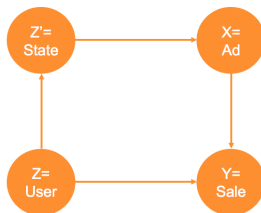


Figure 7: Causal Graph for Ad Recommendations

Simpsons paradox in Recommendation Systems

So what is different in Reco vs. the cases where Simpson's paradox is present?

Real World cases that are facing Simpson's paradox:

- Observational study data: The criteria used for past actions/decisions are not fully logged/observable.
- Outcome: first formal policy

Fitting the wrong problem

Fitting the wrong problem

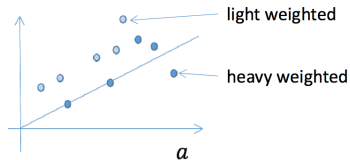


Figure 8: Fitting a linear model of reward on past data

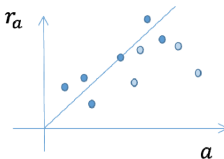


Figure 9: The true best linear predictor

Domain Shift affects misspecified models

Fitting $P(a|x)$ vs fitting $P(a, x)$

- If we can fit an unbiased model for $P(a|x)$ we should not care about the change of $P(x)$
- Of course we won't be good at predicting everywhere but only in the places we historically observed $x \in X$.
- What we need is a model that can predict well and have a model of their own uncertainty everywhere.

Domain Shift affects misspecified models

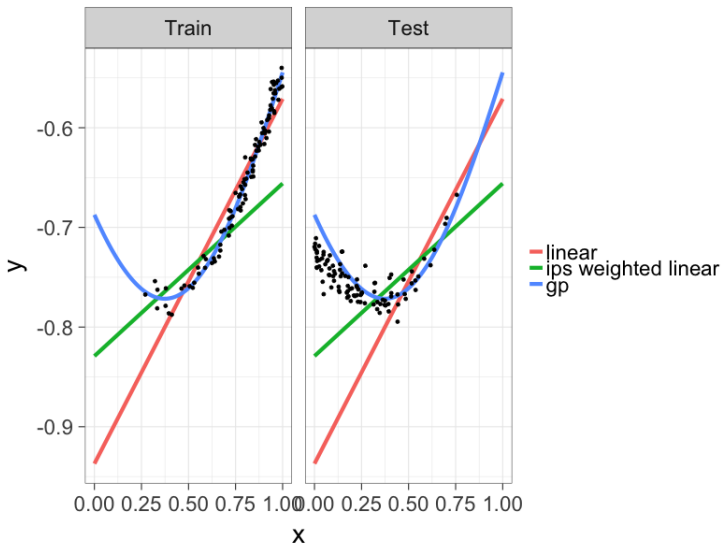


Figure 10: Gaussian Processes vs. Linear Models

Fitting the wrong problem

Example: Gaussian Processes (GPs)

- GPs are non-parametric models that can model any continuous function and can be arbitrarily complex.
- DNNs + Dropout = Fast GPs!

Fitting the wrong problem

Conclusion: This is definitely not a fully solved problem but non-linear models suffer from it a lot less than linear models shown in toy examples.

Variance

Variance

The main real problem remains variance outside of the current support of $P(x)$.

The only answers: Exploration and Counterfactual Reasoning.

We will introduce the Counterfactual/Causal framework in the next section.

Causal Inference Framework

Causal Vocabulary

- **Classical setup:** One single action - Treatment: The real action, Control: The placebo action (do vs. not do)
- **Stochastic setup:** Multiple actions - Treatment and Control are both distributions (aka stochastic policies) over all actions (classical setup is a special case)
- In the case of Criteo: **Control policy = Existing Reco System**

Causal Vocabulary

- **Average Treatment Effect (ATE):**

$$p(\text{outcome}|\text{treatment}, \text{pop}_B) - p(\text{outcome}|\text{control}, \text{pop}_A)$$

where pop_B and pop_A are populations of randomly split users

- **Individual Treatment Effect (ITE):**

$$p(\text{outcome}|\text{treatment}, \text{user}_x) - p(\text{outcome}|\text{control}, \text{user}_x)$$

where both probabilities are computed for the same user_x

- **Notes:**
 - ATE is what we measure with an A/B test
 - In Recommendation, we are interested to optimize the ITE (personalization!)

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 :

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

- 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 :

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

- 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:

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

Individual Treatment Effect

- The *Individual Treatment Effect (ITE)* value of a policy π_x for a given user i and a product j 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

- It is easy to show that, starting from any control policy π_c , the best incremental policy π^* is the policy that shows deterministically to each user u_i the product p_i^* with the highest personalized reward r_i^* :

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

- Note: This assumes non-fatigability, e.g. non-diminishing returns of recommending to the user the same product over and over (no user state/ repeated action effects at play)

State of art in Estimating ITE from Observational Data

3 Main Areas of Research

- **Causal literature:** Counterfactual estimators (IPS, Doubly Robust, Self-Normalized IPS)
- **Offpolicy in RL:** Robust/Safe, worst-case estimators (Sample Variance Penalization, Natural Gradient, Trust Region Policy Optimization (TRPO), Proximal Policy Optimization (PPO), Adversarial Attacks)
- **Causality as Domain Adaptation:** Optimal embeddings for Counterfactual Reasoning

CausE: Causal Embeddings For Recommendation

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 $y_{ij} \sim r_{ij}\pi_c(p_j|u_i)$.
- **Quick parenthesis:** Normal MF for Reco would just decompose the matrix of y_{ij} !
- **Current approach to estimate r_{ij} :** *Inverse Propensity Scoring (IPS)*-based methods to predict the unobserved reward r_{ij} :

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

- Note: This assumes we have incorporated randomization in the current policy π_c

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 attained when:** $\pi_c = \pi_{rand}$.
- π_{rand} = **Recommend all products uniformly.** This means zero recommendation performance!
- **Our question:** Could we learn from π_c a predictor for performance under π_{rand} and use it to compute the optimal product recommendations p_i^* ?

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 $\pi_{t=rand}$ (e.g. the logging policy π_c uses *e-greedy* randomization).
- 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 $y_{ij}^t \in S_t$.

Causality as Domain Adaptation - Related Work

- Johansson, Fredrik, Uri Shalit, and David Sontag. *Learning representations for counterfactual inference*. ICML 2016.
- Louizos, Christos, et al. *Causal effect inference with deep latent-variable models*. NIPS 2017.
- Rosenfeld, Nir, Yishay Mansour, and Elad Yom-Tov. *Predicting Counterfactuals from Large Historical Data and Small Randomized Trials*. WWW 2017.

Predicting Rewards Via Matrix Factorization

- Using the MF model, we assume that both the expected factual control and treatment rewards can be approximated as linear predictors over the **shared** user representations u_i :

$$y_{ij}^c \approx \langle p_j^c, u_i \rangle$$

$$y_{ij}^t \approx \langle p_j^t, u_i \rangle$$

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

$$\widehat{ITE}_{ij} = \langle p_j^t, u_i \rangle - \langle p_j^c, u_i \rangle = \langle w_j^\Delta, u_i \rangle$$

Proposed joint optimization solution

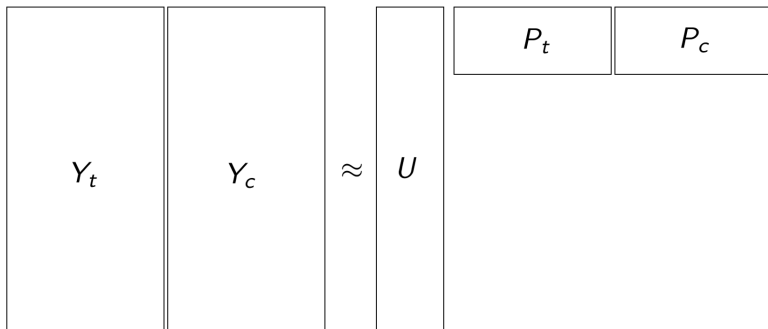


Figure 11: The joint MF problem.

Sub-objective #1: Treatment Loss Term L_t

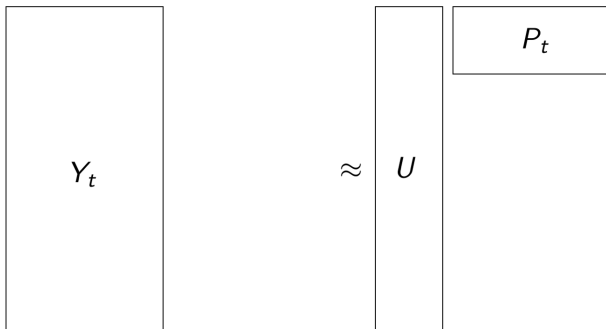


Figure 12: MF of the Y_t matrix

Sub-objective #1: Treatment Loss Term L_t

- We define the first part of our joint prediction objective as the supervised predictor for y_{ij}^t , trained on the limited sample S_t :

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

where:

- P_t is the parameter matrix of treatment product representations.
- U is the fixed matrix of the user representations.
- Y_t is the observed rewards matrix.
- L is an arbitrary loss function.
- $\Omega(\cdot)$ is a regularization term over the weights of the model.

Linking the control and treatment effects

- We can use the translation factor in order to be able to use the model built from the treatment data S_t to predict outcomes from the control distribution S_c :

$$\langle p_j^c, u_i \rangle = \langle p_j^t - w_j^\Delta, u_i \rangle$$

Sub-objective #2: Control Loss Term L_c

- Now we want to leverage our ample control data S_c and we can use our treatment product representations through a translation:

$$L_c(P_t, W^\Delta) = \sum_{(i,j,y_{ij}) \in S_c} l_{ij}^c = L(U(P_t - W^\Delta), Y_c) + \Omega(P_t, W^\Delta)$$

which can be written equivalently as:

$$L_c(P_t, P_c) = \sum_{(i,j,y_{ij}) \in S_c} l_{ij}^c = L(UP_c, Y_c) + \Omega(P_c, P_t - P_c)$$

- Where we regularize the control P_c against the treatment embeddings P_t ($W^\Delta = P_t - P_c$).

Sub-objective #2: Control Loss Term L_c

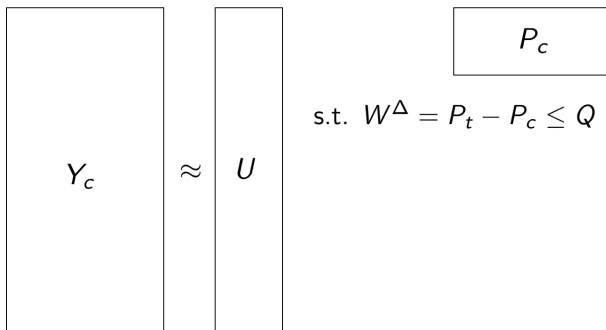


Figure 13: MF of the Y_c matrix

Proposed joint optimization solution - updated

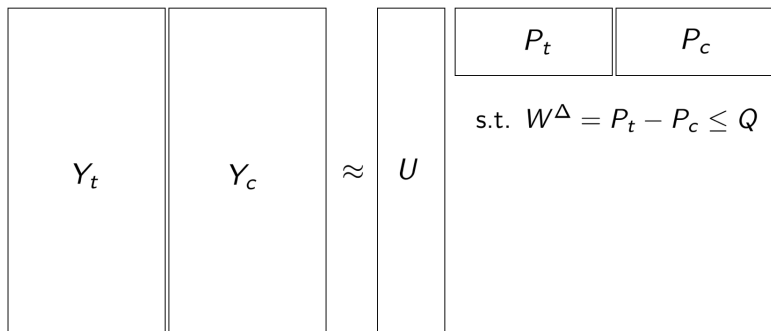


Figure 14: The joint MF problem.

Overall Joint Objective

- By putting the two tasks together (L_t and L_c) and regrouping the loss functions and the regularizer terms, we have that:

$$L_{CausE}^{prod}(P_t, P_c) = L(P_t, P_c) + \Omega_{disc}(P_t - P_c) + \Omega_{embed}(P_t, P_c)$$

Where:

- $L(.)$ is the reconstruction loss function for the concatenation matrix of P_t and P_c .
- $\Omega_{disc}(.)$ is a regularization function that weights the discrepancy between the treatment and control product representations.
- $\Omega_{embed}(.)$ is a regularization function that weights the representation vectors.

Relationship Between IPS And W^Δ

$$IPS_{ij} = \frac{\pi_t(p_j|u_i)}{\pi_c(p_j|u_i)} = \frac{\langle u_i, p_j^t \rangle}{\langle u_i, p_j^c \rangle} = 1 + \frac{\langle u_i, w_j^\Delta \rangle}{\langle u_i, p_j^c \rangle}$$

- We can see that IPS is a function of W^Δ .
- Regularizing $W^\Delta \approx$ regularizing IPS

Question: How about User Shift?

- **Example of user shift:** The current recommendation solution is targeting a subset of users, lets say active buyers on a website and the new recommendation targets mainly newly signed users (modulo randomization which should give non-zero probabilities for all user product pairs).

Generalization Of The Objective To User Shift

- Our objective function can be altered to allow for the user representations to change, we obtain the equation below:

$$L_{CausE}^{user}(U_t, U_c) = L(U_t, U_c) + \Omega_{disc}(U_t - U_c) + \Omega_{embed}(U_t, U_c)$$

- Putting the loss functions associated with the user and product dimension together ($L_{CausE}^{prod}, L_{CausE}^{user}$), we reach the final loss function for our method $L_{CausE}(P_t, P_c, U_t, U_c) =$

$$L(P_t, P_c, U_t, U_c) + \Omega_{disc}(P_t - P_c, U_t - U_c) + \Omega_{embed}(P_t, P_c, U_t, U_c)$$

Algorithm Overview

Input : Mini-batches of $S_c = \{(u_i, p_j^c, \delta_{ij}^c)\}_{i=1}^{M_c}$ and $S_t = \{(u_i, p_j^t, \delta_{ij}^t)\}_{i=1}^{M_t}$,
regularization parameters λ_{embed} and λ_{dist} , learning rate η

Output: P_t, P_c, U_t, U_c - Product and User Control and Treatment Matrices

Random initialization of P_t, P_c, U_t, U_c ;

while *not converged* **do**

 read batch of training samples;

for each *product* p_j **in** P_c, P_t **do**

 | Update product vector: $p_j \leftarrow p_j - \eta \nabla L_{CausE}^{prod}(p, \lambda_{embed}, \lambda_{dist})$

end

for each *user* u_i **in** U_c, U_t **do**

 | Update user vector: $u_i \leftarrow u_i - \eta \nabla L_{CausE}^{user}(u, \lambda_{embed}, \lambda_{dist})$

end

end

return P_t, P_c, U_t, U_c

Algorithm 1: CausE Algorithm: Causal Embeddings For Recommendations

Experimental Results

Experimental Setup

- The task is predicting the outcomes y_{ij}^{rand} under treatment policy π_{rand} , where all of the methods have available at training time a large sample of observed recommendations outcomes from π_c and a small sample from π_{rand} .
- Essentially it's a classical conversion-rate prediction problem so we measure *Mean-Squared Error (MSE)* and *Negative Log-Likelihood (NLL)*.
- We report lift over average conversation rate from the test dataset.

$$lift_x^{metric} = \frac{metric_x - metric_{AvgCR}}{metric_{AvgCR}}$$

Experimental Setup: Matrix Factorization Baselines

We compare our method with the following matrix factorization baselines:

- **Bayesian Personalized Ranking (BRR)** - Directly learn user and product representations via user/item matrix factorization, optimized using the BPR-OPT criteria and trained using LearnBPR.
- **Supervised-Prod2Vec (SP2V)** - Logistic MF method that approximates y_{ij} as a sigmoid over a linear transform of the inner-product between the user and product representations:
$$\hat{y}_{ij} = \sigma(\alpha \langle p_j, u_i \rangle + b_i + b_j + b)$$

Experimental Setup: Causal Inference Baselines

We compare our method with the following causal inference baselines:

- **Weighted-SupervisedP2V (WSP2V)** - SP2V algorithm on propensity-weighted data.
- **BanditNet (BN)-like** - MF with Self-Normalized IPS optimization objective

Experimental Setup: Datasets

- We use the *MovieLens10M* and *Netflix* 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.

Results- MovieLens10M

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

Table 1: Results for MovieLens10M 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.

Results - Netflix

Method	Netflix (SKEW)		
	MSE lift	NLL lift	AUC
<i>BPR-no</i>	—	—	0.665(± 0.001)
<i>BPR-blend</i>	—	—	0.671(± 0.001)
<i>SP2V-no</i>	+10.82%(± 0.02)	+10.19%(± 0.01)	0.752(± 0.002)
<i>SP2V-blend</i>	+12.82%(± 0.02)	+11.54%(± 0.02)	0.764(± 0.003)
<i>SP2V-test</i>	+05.67%(± 0.02)	+06.23%(± 0.02)	0.739(± 0.004)
<i>WSP2V-no</i>	+13.52%(± 0.01)	+13.11%(± 0.01)	0.779(± 0.001)
<i>WSP2V-blend</i>	+14.72%(± 0.02)	+14.23%(± 0.02)	0.782(± 0.002)
<i>BN-blend</i>	—	—	0.785(± 0.001)
<i>CausE-avg</i>	+15.62%(± 0.02)	+15.21%(± 0.02)	0.799(± 0.002)
<i>CausE-prod-T</i>	+13.97%(± 0.02)	+13.52%(± 0.02)	0.789(± 0.003)
CausE-prod-C	+17.82%(± 0.02)	+17.19%(± 0.02)	0.821(± 0.003)

Table 2: Results for Netflix on the Skewed (SKEW) test datasets.

Results

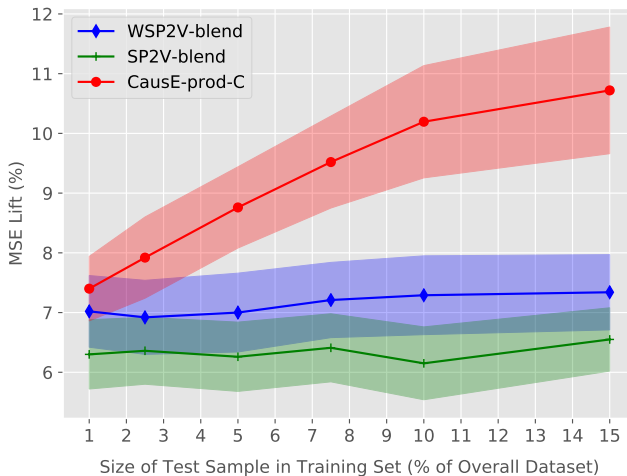


Figure 15: Change in MSE lift as more test set is injected into the blend training dataset.

Conclusions

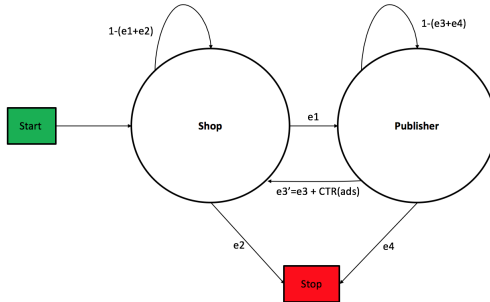
- We have introduced a novel method for factorizing implicit user-item matrices that optimizes for *incremental recommendation outcomes*.
- We learn to predict user-item similarities under the uniform exposure distribution.
- *CausE* is an extension of matrix factorization algorithms that adds a regularizer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

TL;DR

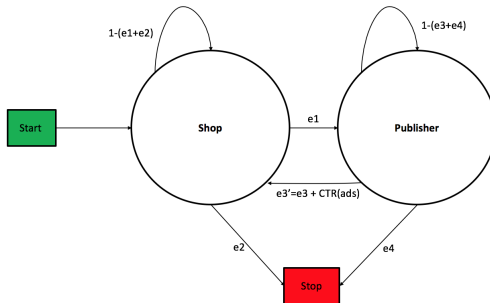
- **Simple algo for causal MF:** Word2Vec-like learning algo over two matrices with a regularised difference between product vectors.
- **What if we dont have e-greedy but a different randomization scheme?** You can use IPS trick to simulate data from π_{rand} .

Next Steps

In our current work we did not leverage at all the user shopping activity:



Second, by making it a MF task, we removed time from our models



In our upcoming work, we are incorporating both the organic user behavior and taking the time into account.

Challenges:

- Sparse rewards
- Delayed rewards
- Extremely large action space

We believe that with reasonable additional assumptions, we can build a realistic user sequence model, optimized for causal recommendations.

Thank You!