Introduction to Causal Inference via Papers

Contents

1 HW2: causal effect of NICU on premature baby delivery [paper]

We hope to find evidence about whether it is truly useful to invest in strengthening perinatal regionalization system, i.e., build more high-level NICU (neonatal-ICU with advanced life support equipments) to improve survival rate for premature babies. Binary variable D is defined as: $D = 1$ indicates a baby being delivered in high-level NICU, and $D = 0$ indicates a baby being delivered in low-level NICU. Binary variable Y is the outcome: $Y = 1$ indicates a survived baby, and $Y = 0$ indicates a passed-away baby. The goal is to estimate

$$E\left(Y(1) - Y(0) \middle| \text{confounders}\right)$$

note that the authors are not using $Y(D)$, and we will explain what $Y(1)$ and $Y(0)$ are later. If the conditional expectation is positive, then we would suggest

to the policy maker to invest more in building high-level NICUs. Because babies are precious to families expecting them, making sure to improve their surviving rate is crucial.

Continue with variable identification. Besides treatment D and outcome Y, we need to address the idea of having high-level NICU close to neighborhoods. The authors propose a binary random variable Z, where $Z = 1$ means there is a high-level NICU within 10 minutes of drving distance, and $Z = 0$ means the traveling time is longer than 10 minutes. Now

$$Y = Y(Z)$$

is the potential outcome we focus on. Next, there are some covariates X, which includes infant's gestational weeks, the month of pregnancy that prenatal care started, and mother's education. Finally, there is an unmeasured confounder U, which characterizes whether a mother is willing to go to high-level NICU: $U = `c'$ means a mother is compliant to the distance factor $Z$, $U = `a'$ means a mother will always go to high-level NICU no matter what, $U = `n'$ means a mother will not go to high-level NICU no matter what, and $U = `d'$ means a mother chooses the opposite to Z.

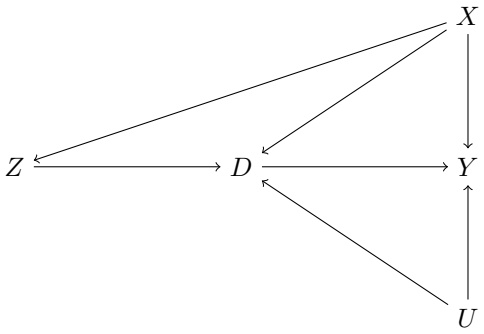
With the notations above, the goal is to estimate

$$E\left(Y(1) - Y(0) \,\middle|\, X = x, U = c\right)$$

Now, let's go through main assumptions.

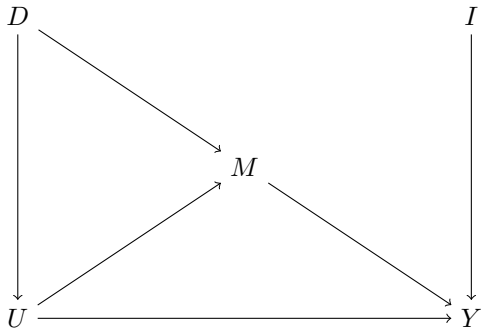
- (SUVTA) This allows us to write $D_i(z) = D_i(z_i), Y_i(z) = Y_i(z_i)$ meaning i-th individual's potential outcome is not affected by others' status. Since each mother is independent, this is plausible.
- Z affects D: i.e., $E(D(1) - D(0)) \neq 0$
- (Exchangeability) $(Y(0), Y(1), D(0), D(1)) \perp Z \mid X$. This is because X in some sense fully defines Z, i.e., when a mother decides where to live, X is the only deciding factor because we assume mothers are not in general expecting premature babies. And this roughly means Z will not suddenly become 1 if $Y(0)$ goes low, etc.
- $D(1) \geq D(0)$: This is plausible because if $D(1) = 0$, i.e., don't go to high-level NICU even it's close, then very likely $D(0) = 0$ given a longer distance.

Finally, the authors present a DAG as follows,



2 HW 3: Recommendation System [paper]

This paper incorporates some do-calculus techniques to alleviate the population bias issue in classic recommendation systems. We start off with a brief discussion on what is population bias and why it happens in classic algorithms. Population bias refers to an item from a popular category receiving high score even when the user may not like it that much. One of the main reasons is that classic algorithms use click-frequency as a factor in computing user-item interaction. And items from popular category will receive more clicks no matter whether the user like them or not. In term of causal inference, the user's click distribution over different categories acts like a confounder as explained below.

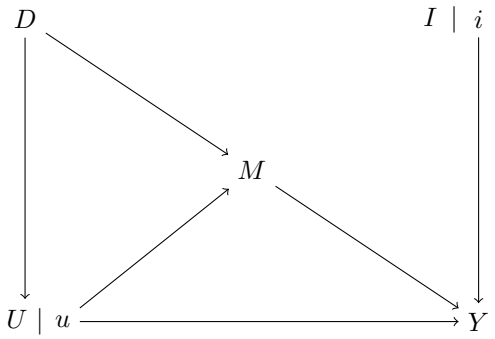


We first provide definitions of the notations. In general, we use H dimensional vector encoding. $U = (u_1, ..., u_K)$ with $u_i \in \mathbb{R}^H$ denotes the K user related features. $x_u = (a_{u,1}, ..., a_{u,K}) \in \mathbb{R}^K$ is the coefficient vector of a particular user u. So, to encode a user u, we use $\sum_1^K a_{u,i}u_i \in \mathbb{R}^H$. Similarly, we let $V = (v_1, ..., v_N)$ with $v_i \in \mathbb{R}^H$ encode the N item categories. $z_{\text{item}} = (b_{\text{item},1}, ..., b_{\text{item},N}) \in \mathbb{R}^N$ is the coefficient vector for an item. Like before, we can use $\sum_1^N b_{\text{item},i}v_i \in \mathbb{R}^H$ to encode for a particular item (symbol I in the above DAG). Next, D is for user's click history frequencies. Suppose there are N item groups, $d_u = (p_u(1), ..., p_u(N)) \in \mathbb{R}^N$ is this user's click frequency. For example, if there are 3 item categories, $d_u = (0.5, 0.4, 0.1)$ means the user doesn't really click into the third item group. Then, M denotes the interaction between user and his/her click history. Formally, we write $M_u = f(d_u, u) \in \mathbb{R}^H$. A reasonable choice for f can be defined as,

$$M_u = \sum_{i=1}^{N}\sum_{j=1}^{K} d_u(i)v_i \odot x_{u,j}u_j$$

where $\odot$ stands for component-wise product.

Classic recommender system algorithms tend to implicitly have the arrow from D to U, meaning the user's click history will affect how the algorithm would characterize the user. The goal of a recommender system is to estimate $E(Y \mid U = u, I = i)$. But when we fix U, I, node D still keeps a path from U to Y: $U \to D \to M \to Y$. And we observe node D indeed satisfies the backdoor criteria (we present a SWIG),



Since a user's click history constantly changes, we wouldn't fix D. Before we present the backdoor adjustment, we first go through a quick computation from the original paper to see how exactly D affects Y (another way to see why D is a confounder),

$$
\begin{aligned}
E(Y \mid U = u, I = i) &= \frac{\int_{\mathcal{Y}} y \cdot (\sum_{d \in \mathcal{D}} \sum_{m \in \mathcal{M}} P(d)P(u|d)P(m|d,u)P(i)P(y|u,i,m))dy}{P(u)P(i)} \\
&= \int_{\mathcal{Y}} y \cdot \left(\sum_{d \in \mathcal{D}} \sum_{m \in \mathcal{M}} P(d|u)P(m|d,u)P(y|u,i,m) \right) dy \\
&= \int_{\mathcal{Y}} y \cdot \left(\sum_{d \in \mathcal{D}} P(d|u)P(y|u,i,M(d,u)) \right) dy \\
&= \int_{\mathcal{Y}} y \cdot P(y|u,i,M(d_u,u))dy
\end{aligned}
$$

The first equality follows from Bayes's rule with the assumption that I is independent from other random variables. The third equality follows from our definition that M is deterministic when given U and I. The last equality follows from the fact that given a user, his/her history click frequency is fixed. The last equality shows that the history click distribution affects Y via the mediator M. The good news is that D can be measured (online platform can collect users' purchase history information). So, in order to cut the arrow from D to U, we

can use the *backdoor adjustment*.

Following the theorem from lecture notes (here we put $X_i = (U, I)$, $X_j = Y$, and $X_A = D$), we have

$$E(Y(u,i)) = E(E(Y \mid U = u, I = i, D)) = E(g(D)) = \sum_{d \in \mathcal{D}} g(d)P(d)$$

where $g(D) = E(Y \mid U = u, I = i, D)$ is some unknown function, which may be approximated/fitted via machine learning methods.

In summary, the only tool we see here is the backdoor adjustment. So the assumptions needed are two-folds: firstly, we assume our SWIG is correct (mainly there are no other arrows connecting I and other nodes); secondly, we would need the assumptions for performing backdoor adjustment: consistency of Y and d-separation (which follows from SWIG).

3 HW 4: Estimation Algorithm (continuation of the paper from section 2)

Using backdoor adjustment, the paper reduces the problem to estimating

$$\sum_{d \in \mathcal{D}} g(d)P(d)$$

where $g(d) = E(Y \mid U = u, I = i, D = d) = f(u, i, d)$, for some unknown function f. We hope to make use of shopping distribution d more specifically instead of just saying it's an input to f. Recall in the paragraph above where I go through notations, I mentioned a quantity $M(d_u, u)$, which is defined as a component-wise product. This quantity basically captures the interaction between the user u and his/her shopping history d_u. Thus, we write

$$g(d) = f(u, i, M(d, u))$$

An important note is that instead of using $M(d_u, u)$, we use $M(d, u)$. The reason is two-fold: $g(d)$ takes an arbitrary shopping history distribution d instead of a particular user's history d_u; secondly, this choice has the benefit of exposing this user to other people's shopping history to avoid the so-called filter-bubble: our recommendation scope will gradually become narrower and narrower because the user's click history and our recommendation system form a feedback loop. If we take a look at our computation above for $E(Y \mid U = u, I = i)$, we notice $M(d_u, u)$ is used. This shows classical recommendation system tend to have the filter-bubble issue, and the backdoor adjustment helps to alleviate this issue by taking into account other people's shopping history. To summarize, the task now is to estimate

$$\sum_{d \in \mathcal{D}} f(u, i, M(d, u))P(d)$$

3.1 Estimating d

There are infinitely many possible shopping distributions, so we need to only consider the available users' history. For a particular user u, we compute the distribution $d_u = (p_u(1), ..., p_u(N))$ (recall the distribution is for N item categories) as follows,

$$p_u(j) = \frac{\sum_{i \in \mathcal{H}_u} q_j^{(i)}}{|\mathcal{H}_u|}$$

where $\mathcal{H}_u$ is a collection of items this user clicked in the past, and $q_j^{(i)} = 0$ or 1 tells whether this item i belongs to category j. Next, the choice for $P(d)$ can be tricky. There is no simple way to compute it, and the paper uses the following method,

$$P(d_u) := \frac{|\mathcal{H}_u|}{\sum_{v \in \text{all users}} |\mathcal{H}_v|}$$

This says if a user bought many many items, his/her opinion is more important, so we weigh the distribution d_u heavier.

3.2 Estimating Eg(D)

Recall the goal is to estimate $Eg(D)$. The paper chooses to estimate $g(ED)$ instead. We observe that if g is a linear function, then $g(ED) = Eg(D)$. The paper also mentions that under some conditions, $g(ED)$ is not too far away from $Eg(D)$.

3.3 Put Everything Together

From section (2.1.2), the goal now becomes estimating

$$f(u, i, M(ED, u))$$

Now,

$$ED = \sum_{d \in \mathcal{D}} d \cdot P(d)$$

and for the same reason as mentioned in section (2.1.1), we only consider $\widetilde{\mathcal{D}}$: the users' history available to us. Denote

$$\overline{d} = (\overline{p}(1), ..., \overline{p}(N)) \approx \sum_{d_u \in \text{ available dataset}} d_u \cdot P(d_u)$$

which can be computed using the method in section (2.1.1). Next, for $M(\overline{d}, u)$, we can use the component-wise product definition:

$$M(\overline{d}, u) = \sum_{a=1}^{N} \sum_{b=1}^{K} \overline{p}(a) v_a \odot x_{u,b} u_b$$

for notations, please refer to the paragraph discussing them (second half of the first page in section 2.) Finally, function f can be a neural network or some linear function.

3.4 Parameters and Training

We need to find the best value for v_a and u_b, which are vector encoding of item category and user features. My understanding is that $x_{u,b}$ is specified by us, not from model training. We apply the classical gradient descent method for,

$$\arg\min_{\theta} \sum_{(u,i,y_{u,i}) \in \text{ training data}} l(f(u, i, M(\overline{d}, u)), y_{u,i})$$

where l can be a 0-1 loss function or log-loss function, and $y_{u,i} = 0$ or 1 is whether or not the user u bought item i.

3.5 Output $EY(u, i)$

The paper uses a combination of classic recommendation system and causal inference aided recommender. Suppose $A = E(Y \mid U = u, I = i)$ is the output of some classic recommender system, and $B = EY(u, i)$ is the causal inference aided result. We can combine them via,

$$(1 - \eta)A + \eta B$$

where η is some parameter. The paper uses an interesting approach to set η. Recall causal inference helps to avoid filter-bubble by exposing the user to other people's shopping history. If the user's preference is very consistent, i.e., he/she is not easily affected by others, then we should set η to be smaller, meaning a classic recommender should suffice. However, if the user's taste is changing, then it would be useful to take into account other shopping distributions, thus a higher η is desirable. Therefore, the paper uses KL-divergence to see how vary a user's taste is changing. Suppose the user's shopping distribution from last year is $d_u^{(1)}$, and $d_u^{(2)}$ is from this year. We compute the symmetric KL divergence,

$$\eta_u = \mathrm{KL}(d_u^{(1)} || d_u^{(2)}) + \mathrm{KL}(d_u^{(2)} || d_u^{(1)})$$

Finally, we set the η for user u to be

$$\eta = \frac{\eta_u - \eta_{min}}{\eta_{max} - \eta_{min}}$$

and we finish the estimation step.

3.6 Summary

The estimation part is not using causal-inference methods like outcome regression, matching, or IPW. Instead, causal inference is used in the setup of the estimator, and the training part is still following the classic machine learning methods like gradient descent. The reason of improvement made by causal inference and detailed computation steps are shown in the subsections above, and here I summarize them again,

- Compute $ED = \sum_{d \in \widetilde{\mathcal{D}}} d \cdot P(d)$
- Compute $M(ED, u)$
- Train $f(u, i, M(\overline{d}, u))$ using machine learning methods.
- output our estimate of $EY(u, i)$

3.7 Performance

I will summarize the authors' finding here. The authors tested the causal-inference-aided algorithm using two datasets: ML-1M (about movie ratings)

and Amazon-Book (about Amazon products). The output is compared with two representative recommenders: FM and NFM models. The task is to check if there are improvements in accuracy and alleviating bias amplification. For testing accuracy, metrics *Recall@K* and *NDCG@K* are used. These metrics basically test that among the top K items recommended, how likely the users may interact with them. And the higher the score, the better. The resulting table is attached below. Next, for checking if bias amplification is reduced, the authors used the calibration metric C_{KL}. This basically monitors how much the recommending-item list drifts due to the user's history. A larger C_{KL} implies a bigger drift, i.e., the recommender is easily affected by a user's history, eventually leads to a filter-bubble.

Method	FM								NFM							
	ML-1M				Amazon-Book				ML-1M				Amazon-Book			
	R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20
FM/NFM [16, 29]	0.0676	0.1162	0.0566	0.0715	0.0213	0.0370	0.0134	0.0187	0.0659	0.1135	0.0551	0.0697	0.0222	0.0389	0.0144	0.0199
Unawareness [15]	0.0679	0.1179	0.0575	0.0730	0.0216	0.0377	0.0138	0.0191	0.0648	0.1143	0.0556	0.0708	0.0206	0.0381	0.0133	0.0190
FairCo [21]	0.0676	0.1165	0.0570	0.0720	0.0212	0.0370	0.0135	0.0188	0.0651	0.1152	0.0554	0.0708	0.0219	0.0390	0.0142	0.0199
Calibration [32]	0.0647	0.1149	0.0539	0.0695	0.0202	0.0359	0.0129	0.0181	0.0636	0.1131	0.0526	0.0682	0.0194	0.0335	0.0131	0.0178
Diversity [47]	0.0670	0.1159	0.0555	0.0706	0.0207	0.0369	0.0131	0.0185	0.0641	0.1133	0.0540	0.0693	0.0215	0.0386	0.0140	0.0197
IPS [30]	0.0663	0.1188	0.0556	0.0718	0.0213	0.0369	0.0135	0.0187	0.0648	0.1135	0.0544	0.0692	0.0213	0.0370	0.0137	0.0189
DecRS	**0.0704**	**0.1231**	**0.0578**	**0.0737**	**0.0231**	**0.0405**	**0.0148**	**0.0205**	**0.0694**	**0.1218**	**0.0580**	**0.0742**	**0.0236**	**0.0413**	**0.0153**	**0.0211**
%improv.	4.14%	5.94%	2.12%	3.08%	8.45%	9.46%	10.45%	9.63%	5.31%	7.31%	5.26%	6.46%	6.31%	6.17%	6.25%	6.03%

Figure 1: Accuracy comparison among different models

	ML-1M					
FM	**R@20**			**N@20**		
Threshold	**FM**	**DecRS**	**%improv.**	**FM**	**DecRS**	**%improv.**
0	0.1162	0.1231	5.94%	0.0715	0.0737	3.08%
0.5	0.1215	0.1296	6.67%	0.0704	0.0730	3.69%
1	0.1303	0.1412	8.37%	0.0707	0.0741	4.81%
2	0.1432	0.1646	14.94%	0.0706	0.0786	11.33%
3	0.1477	0.1637	10.83%	0.0620	0.0711	14.68%
4	0.1454	0.1768	21.60%	0.0595	0.0737	23.87%

Figure 2: Accuracy for different types of users

Few words about accuracy. As we can see from the first table, DecRS (proposed in this paper) has higher accuracy comparing to other methods (those many methods are all based on FM/NFM models). A snapshot of the second table shows accuracy for different types of users. Recall we define η_u (in section 2.1.5) to indicate how fluctuating a user's taste is. The bigger the η_u, the more unstable the user's taste. Recall in this causal-inference-aided method, we look at all shopping distributions in $M(\overline{d}, u)$, instead of only looking at a single person's history $M(d_u, u)$. So, if a user's taste is unstable, i.e. d_u is constantly changing, it would be better to expose the model to other distributions. Intuitively, if we recommend based on current d_u, and the user changes his/her taste in the future, then our recommendation would be inaccurate. So, it's better to broaden our view by looking at how this user interacts with all kinds of distributions. With this in mind, we would expect a bigger improvement for users with fluctuating preferences. And this is confirmed in table 2 (the "threshold" is η_u.) Finally, the diagram below shows the improvement on bias amplification,

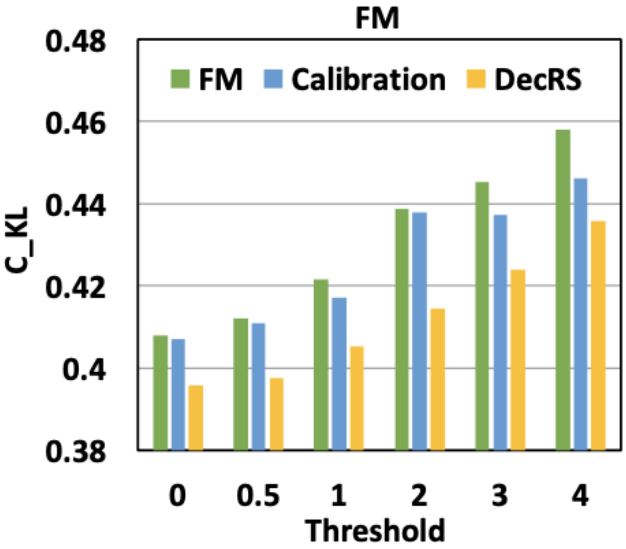


Figure 3: C_{KL} scores

For users with high η_u, their interests are constantly drifting, so we expect high C_{KL} scores for all methods. DecRS always has a lower C_{KL} score, which confirms the improvement on reducing bias amplification. As a conclusion, DecRS proposed by this paper indeed provides improvements on both accuracy (especially for high fluctuating users) and reducing bias amplification.

4 HW 5: Sensitivity Analysis (continue with the same problem of recommender system)

In the original paper, there is no discussion on sensitivity analysis, so I have to come up with something (hopefully they somewhat make sense.) I think of couple reasons on why the authors didn't perform sensitivity analysis. Recall the treatments here are U for user and I for item, and there is also a backdoor node D, which is the user's browsing/shopping history distribution. The main difficulty is that the treatment is not binary (the potential outcome Y is continuous, but it's fine with Manski's bound and also Rosenbaum's gamma sensitivity.) Secondly, a possible unmeasured confounder can be a user's personality, which can greatly affect his/her taste when shopping. But in the paper, the author simply encoded users as vectors: $U \cdot x_u \in \mathbb{R}^{H\times 1}$, where $x_u \in \mathbb{R}^{K\times 1}$ is for user u, and $U \in \mathbb{R}^{H\times K}$ is the vector encoding for K characteristics of users (each characteristic is vector encoded in $\mathbb{R}^{H\times 1}$, for more details on notation, please refer to the first page of section 2.) Then, the authors used machine learning methods with neural network to train for parameters of matrix U. What this does is that instead of explicitly identify treatment A (using the notation from lecture notes), where we may leave some confounders unmeasured, we implicitly encode everything into the matrix U (there is also a similar matrix I for item characteristics.) So, in some sense, we "measure" everything via neural network.

Now, I will try to propose some strategies to perform sensitivity analysis. Firstly, I make a reduction to the original problem: instead of putting (u, i) as treatment, I will just use u (suppose we simply **want to see if a user would like a particular type of items**.) I'll follow the SWIG below (I changed the name of nodes to reflect the notations from lecture,) and this new SWIG is the same as before except now I add a top node for unmeasured confounders and delete the node I,

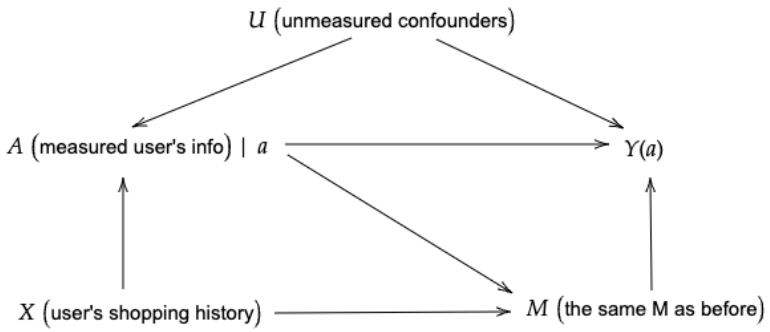


Figure 4: SWIG with unmeasured U

Let's put aside the original paper because it's hard to add sensitivity analysis to that framework. We first identify what treatment A is. Instead of using a matrix like in the original paper, I now propose to identify it explicitly as follows. Consider,

{gender, age level, culture/ethnicity, current living city type, education, job type}

Those information seems to be commonly collected with online platforms, so I will use those (one can remove or add some entries.) Suppose $A = \{\boldsymbol{a_1}, ..., \boldsymbol{a_k}\}$, where $\boldsymbol{a_i}$ is a vector recording a possible combination of those information. Notice each information entry only has finite options: 0/1 for gender, 0/1/2 for teen/young adult/old, etc. Hence, we can say there are only finitely many treatment levels: making A a finite set. Then, there are some important unmeasured confounders: namely user's personality, and we denote it as node U.

4.1 Manski's bound

Like before, we are interested in estimating $EY(\boldsymbol{a_i})$ for each $\boldsymbol{a_i}$. Manski's bound is a general approach without referring to unmeasured confounders. Following the idea from lecture notes (we work out $Y(\boldsymbol{a_1})$, and the same approach works for other $Y(\boldsymbol{a_i})$),

$$\begin{aligned}
E(Y(\boldsymbol{a_1}) \,|\, X) &= E_{A|X}(E(Y(\boldsymbol{a_1}) \,|\, X, A)) \\
&= \sum_{a_i} E(Y(\boldsymbol{a_1}) \,|\, X, A = \boldsymbol{a_i}) \cdot P(A = \boldsymbol{a_i} \,|\, X) \\
&= E(Y(\boldsymbol{a_1}) \,|\, X, A = \boldsymbol{a_1}) \cdot P(A = \boldsymbol{a_1} \,|\, X) \\
&\qquad + \sum_{a_i \neq a_1} E(Y(\boldsymbol{a_i}) \,|\, X, A = \boldsymbol{a_i}) \cdot P(A = \boldsymbol{a_i} \,|\, X) \\
&\leq E(Y \,|\, X, A = \boldsymbol{a_1}) \cdot P(A = \boldsymbol{a_1} \,|\, X) + \max \cdot (1 - P(A = \boldsymbol{a_1} \,|\, X))
\end{aligned}$$

where max is the biggest value possibly taken by score Y. Suppose we use logit function for propensity score, and a function of choice for $E(Y \,|\, X = x, A = \boldsymbol{a_i}) = f(x, \boldsymbol{a_i})$. Then we re-write the above upper bound as,

$$\begin{aligned}
E(Y(\boldsymbol{a_1}) \,|\, X = x) &\leq f(x, \boldsymbol{a_1}) \cdot \text{logit}(x, \boldsymbol{a_1} \,|\, \theta) \\
&\qquad + \max \cdot (1 - \text{logit}(x, \boldsymbol{a_1} \,|\, \theta)) \\
&= \text{Upper}(x, \boldsymbol{a_1} \,|\, f(x, \boldsymbol{a_1}), \theta) \qquad \text{(for notional simplicity)}
\end{aligned}$$

Finally,

$$E(Y(\boldsymbol{a_1})) = E(E(Y(\boldsymbol{a_1}) \,|\, X)) \leq \sum_x \text{Upper}(x, \boldsymbol{a_1} \,|\, f, \theta) \cdot P(x) \qquad (\star)$$

This upper bound (RHS) remarkably has the same form as what we had in section 3 (the X here is the user's shopping history distribution, which is denoted as D in the previous section.) We can borrow the idea from the original paper

and run a neural network to find the best f and θ. For the loss function, we can again use 0/1 loss, e.g., 1 for RHS $\geq \hat{E}(Y(\boldsymbol{a_1}))$ in $(\star)$, where $\hat{E}$ can come from the direct standardization method. For training purpose, we can keep sampling for the training data set, so each time we get a probably slightly different $\hat{E}(Y(\boldsymbol{a_1}))$. We use the same f and θ for other $Y(\boldsymbol{a_i})$'s throughout the training.

After the training is done, we can produce the RHS of $(\star)$. So, we can answer our question proposed in blue on the first page of this section: given a user's collected information a, what's the highest possible score that user may rate for a type of items. If this upper bound is low, then we wouldn't recommend this type of items to the user.