Technical Report for BigData Cup 2021 (Track 1)

Tzu-Heng Lin
Peking University
lzhbrian@gmail.com

Chen Gao
Tsinghua University
chgao96@gmail.com

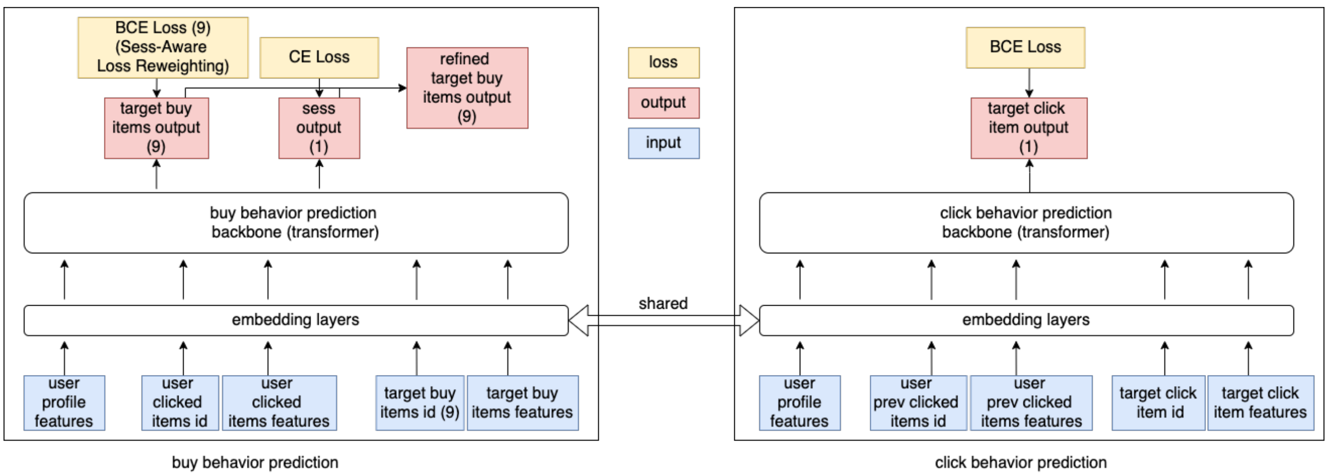

Figure 1: Overall structure of our method.

Abstract

In this paper, we detailedly describe our solution for the BigData Cup 2021 (Track 1: Item Combination Prediction). We first conduct an exploratory data analysis on the dataset and them utilize the findings to design our framework. Specifically, we use a ***two-headed transformer-based network*** *to predict user feedbacks and unlocked sessions, along with the novel* ***session-aware reweighted loss***, ***multi-tasking with click behavior prediction***, *and* ***randomness-in-session augmentation***. *In the final private leaderboard on Kaggle, our method ranked 3rd with an accuracy of 0.39224.*[1]

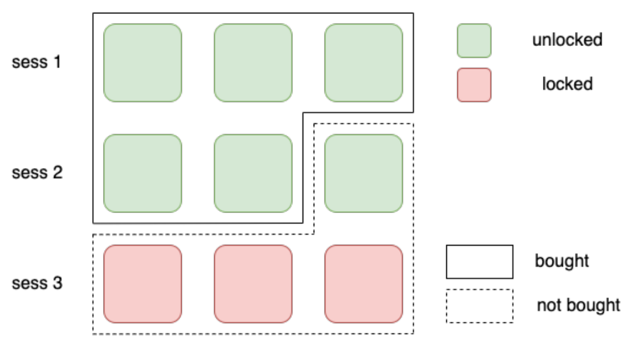

Figure 2: Problem Definition [1]

1. Introduction and EDA

1.1. Problem Definition

The task of the BigData Cup 2021 (Track 1) is to predict each user's feedback to 9 exposed items, given this user's click history, portrait features, and the items' features. The special things about this task is that the 9 items are grouped into 3 sessions (*c.f.* Figure 2). The user can only unlock the following session after he/she buys all 3 items in the previous session.

1.2. Exploratory Data Analysis (EDA)

Before diving into the feedback prediction framework, we conduct exploratory data analysis firsthand to master the whole picture of the dataset.

[1]Our code is available at https://github.com/lzhbrian/bigdatacup2021

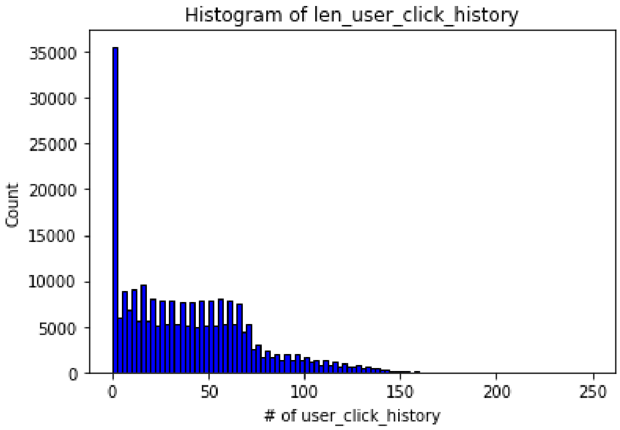

Figure 3: Click Behavior

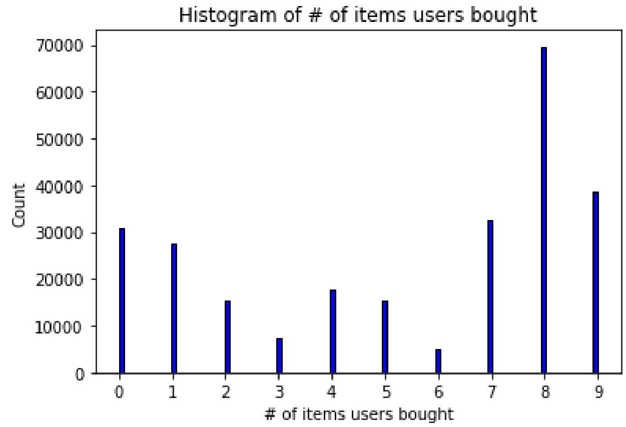

Figure 4: Buy Behavior

Table 1: Click and Buy Statistics in Different Sessions.

Session	Item IDs	# Clicks	# Buys
1	1∼39 (39)	4,606,977	80,231
2	40∼147 (108)	3,608,173	63,297
3	148∼381 (234)	2,220,648	287,482

1.2.1 Data Statistics

In total, there are 260,087 entries for training, and 206,254 entries for testing in track 1.

Click We first plot the histogram of how many times do each user clicked in Figure 3. We can see that most of the users only click few times.

Buy We also plot the histogram of how many items do each user bought in Figure 4. We can see that 30912 users buy nothing, and 50267, 38191, 140717 users buy 1∼3, 4∼6, 7∼9 items, respectively. It is also interesting that very few people buy 3 or 6 items. We hypothesize that this is because the main reason why a user buy 3 or 6 items, is to unlock and buy the 3 items in the next session. For the usage in the later paragraphs, we classify user states into 4 sessions:

- sess0: user has bought 0 item.
- sess1: user has bought 1∼3 items.
- sess2: user has bought 4∼6 items.
- sess3: user has bought 7∼9 items.

From Table 1, we can see that items in sess3 possess the majority of clicks and buys.

1.2.2 User Portrait and Item Features

Since the meaning of user portrait features and item features are not given in the dataset description, we arrange their stats in Table 2 and Table 3. We can see that all user portrait are discrete features, while two of the item features are continuous features.

2. Method

The overall structure of our method is shown in Figure 1. In what follows, we will introduce each of our designs respectively.

2.1. Two-headed (Buy and Session) Prediction

We start with a two-headed prediction structure (Figure 1 left). The network takes the following inputs: user profile features, user clicked items' id and features, 9 exposed target items' id and features. These inputs are further processed by their corresponding embedding layers, and a backbone network (*e.g.* MLP). Finally, the network predicts whether the user will buy the 9 exposed target items or not, and to which session (*c.f.* Section 1.2.1) will the user unlock. The predicted session will be used to refine the predicted results of the 9 exposed target items.

We design this framework for the following reasons: Firstly, the 9 item feedbacks are correlated. For example, users might buy all of the first 6 items, only to buy the 9th item. It is not suitable to predict the 9 feedbacks independently. So we design a network that is able to predict 9 feedbacks simultaneously. Secondly, if we only predict the 9 feedbacks, there might be mistakes where the predicted results are impossible to happen in the real world. For example, the network might predict one only buys the 1st and the 9th items. This is impossible since the user has to buy all of the first 6 items, in order to buy the 9th item. So we design another head to predict the unlocked sessions of users. The results are used to fix unreasonable outputs of the buy feedback predictions.

Table 2: User portrait features.

user_portrait idx	1	2	3	4	5	6	7	8	9	10
# unique_values in trainset	3	1363	20	10	195	49	3	11	2	2164
# unique_values in testset_track1	3	1319	19	10	191	47	3	13	2	2054
# unique_values in testset_track2	3	1341	20	10	190	47	3	11	2	2060

Table 3: Item features.

item_feature idx	1	2	3	4	5
# unique_values in item_info	4	10	2	241	214
values	1,2,3,4	0,1,2,3,4,5,6,7,8,9	1,2	0∼1, float	0∼1, float

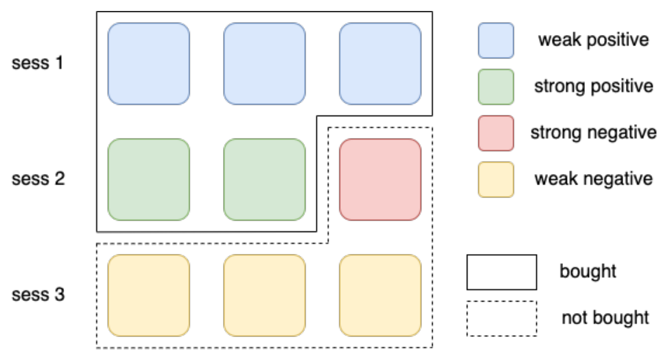

Figure 5: Session-Aware Loss Reweighting

2.2. Session-Aware Loss Reweighting

To better model users' buy behaviors, we classify the 9 exposed items into 4 categories according to each user's buy behavior (weak positive, strong positive, strong negative, weak negative) as shown in Figure 5. For sessions before the last session user has unlocked, items should be treated as weak positives, as user might buy these items only to unlock the later sessions. For the last session user has unlocked, items should be treated as strong positives and strong negatives. As user unlocked and stopped in this session, items bought or not bought should be classified as strong signals. For later locked sessions, items should be treated as weak negatives, as users haven't unlocked these sessions, we shouldn't assume too strong preference on these items.

In practice, we reweight the BCE loss of these 4 categories using weight 0.5, 1, 1, 0.5, respectively. In our experiments, reweighting loss provide huge improvements to the final score (from 0.365 to 0.388).

2.3. Multi-tasking with Click Behavior Prediction

Apart from the network described in 2.1, we use another network to predict user click behavior (Figure 1 right). The two networks share the same embedding layers. We argue that through this multi-tasking procedure, the embedding layers can be better learned. In our early ablation study, this can improve our score from 0.362 to 0.365.

2.4. Transformer Backbone

Instead of simple MLPs [6, 2], we switch the backbone part of buy behavior prediction and click behavior prediction into transformers [5], as their self-attention mechanism is proved to be effective on capturing inter-relations between different features. In our early ablation study, this can improve our score from 0.36 to 0.365.

2.5. Randomness-in-session Augmentation

To prevent over-fitting and make training more robust, we randomly shuffle items' order within the same sessions. This strategy is also used for test time augmentation, where orginal prediction and shuffled predictions are averaged to produce the final results. In our experiments, augmentation in training provides minor difference, while augmentation in testing provides an unstable improvement of around 0.002.

2.6. Other settings

Colab with a P100 GPU is used as our training platform. We use Adam [3] with default hyper-parameters in PyTorch [4]. Batch size is set to 32, and learning rate is set to 1e-2 for 10 epochs, and 1e-3 for another 10 epochs. Clicking data in the test set of both tracks is also used during our training. All continuous features are discretized into bins (including item features, prices).

3. Discussion & Limitation

3.1. Discussion

We have also tried several things that conceptually make sense but didn't improve the score. Firstly, we tried an attention-like deep interest network [6] to reweight user clicked items, however it didn't improve the final score. Given that we do not know how click behavior data is collected, we think that users might present different preference in the scenario where click data is collected. And thus making the model more complex in this aspect doesn't help. Secondly, we tried to add user embedding into the network yet encounter severe over-fitting in training. Adding mini-batch aware regularization [6] can reduce over-fitting, however it still cannot make improvements to the final score. Due to the fact that one user only have one training entry, this result is not very surprising. In addition, we also tried adding timestamp as a feature, however it also didn't help. We originally thought that weekends or holidays might affect user behaviors.

3.2. Limitation

Due to limited time during competition, the ablation studies of each component isn't very rigorous and sufficient. Some of the method statements and decisions are based on semi-finished experimental results and subjective judgements.

References

[1] Ieee bigdata cup 2021 compeition overview. `https://www.kaggle.com/c/bigdata2021-rl-recsys/overview/description`. Accessed: 2021-09-10.

[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 173–182, 2017.

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference for Learning Representations (ICLR)*, 2015.

[4] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems (NeurIPS)*, 32:8026–8037, 2019.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems (NeurIPS)*, pages 5998–6008, 2017.

[6] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1059–1068, 2018.