# Recommendation Using Reward Modelling and Sophisticated Practical Compromises

Amine Benhalloum
Criteo
Paris, France

Guillaume Genthial
Criteo
Paris, France

David Rohde
Criteo
Paris, France

Flavian Vasile
Criteo
Paris, France

## ABSTRACT

It might seem that algorithms that learn to optimize reward are perfectly suited to the task of recommendation. We outline how reward models might be applied to real world recommender systems and difficulties with applying them in practice. A pragmatic compromise is described that has proven value in very large scale recommendation tasks. We discuss trade-offs between direct reward optimization and more scaleable and robust heuristic approaches. We outline what a reward modelling and optimizing approach might look like and discuss if this is practically feasible outlining possible benefits and challenges. Limitations of the proxy approach and challenges in the reward modelling approach are discussed.

**ACM Reference Format:**
Amine Benhalloum, Guillaume Genthial, David Rohde, and Flavian Vasile. 2018. Recommendation Using Reward Modelling and Sophisticated Practical Compromises. In . ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

## 1 INTRODUCTION

A recommender system for computational advertising has a clear well defined goal i.e. the recommender system is successful if it increases the overall sales or profitability of the client. In short the profit motive in computational advertising provides one of the clearest performance metrics of any recommender system.

Given this clear objective we might imagine that by applying machine learning techniques that optimize reward would be highly applicable. There are two broad classes of machine learning methods that appear well suited to this task: model based approaches e.g. [8] or alternatively inverse propensity score based approaches [3].

This paper addresses the question: *Why do large scale production recommender systems ignore this elegant theory and instead optimize proxies of performance?.*



, ,

https://doi.org/10.1145/1122445.1122456

## 2 A FAST RECOMMENDATION ENGINE

A user with context $X$ must have $K$ recommendations delivered to them. We assume the following architecture:

A user representation is created using a deep network such that $\omega = f_\Xi(X)$. Here $\omega$ is a low dimensional dense representation of a user with context $X$. We then obtain the $K$ recommendations by computing:

$$a_1, ..., a_K = \text{argsort}[f_\Xi(\boldsymbol{X})^T \boldsymbol{\beta}]_{1:K}$$

here $\beta$ represents the item embeddings. There are two key advantages to this architecture:

- By performing a fast approximate maximum inner product search [1, 6] it is possible to solve the otherwise slow argsort operation. This allows the method to be applied at high speed and high scale.
- It is possible to use deep learning methods in order to find good solutions to $\Xi, \boldsymbol{\beta}$.

We make a parenthetical remark here. This is just one type of constrained decision rule, that has appealing properties in a production setting. We could also consider decision rules that are less constrained e.g. that optimize $a_1, ..., a_K$ jointly and can exploit (if they exist) virtuous combinations of recommendations.

## 3 OPTIMIZING FOR REWARD

In this paper we will assume that optimizing for short term reward (or clicks) is our objective. In order to optimize for clicks a central object is to quantify for a given context $X$ and a banner (or slate) size $K$ what is the probability of obtaining a reward/click $c$ - that is we require: $P(c|a_{1:K}, \boldsymbol{X})$. The model $P(c|a_{1:K}, \boldsymbol{X})$ can be trained using logs of the recommender system either using a modeling framework e.g. [8] or using inverse propensity score estimators [3]. Both methods can be challenging especially in cases where the recommendations $a_{1:K}$ have never (or rarely) been delivered historically on context $X$. Assuming we have obtained a $P(c|a_{1:K}, \boldsymbol{X})$ that we trust we can then perform the following optimization problem in $\Xi, \boldsymbol{\beta}$ space to find the appropriate decision rule

$$U(\Xi, \boldsymbol{\beta}) = \sum_n P(c|a_{1:K} = \text{argsort}[f_\Xi(\boldsymbol{X}_n)^T \boldsymbol{\beta}]_{1:K}) \quad (1)$$

While elegant the above formulation has two key challenges which means, that to the best of our knowledge, it has never been applied in practice in large scale recommendation:

- Modelling $P(c|a_{1:K}, \boldsymbol{X})$ accurately is extremely difficult especially for actions that differ from what was done previously. If the old recommender system always recommended a phone to a specific user it is not easy to model the impact of recommending shoes. - in these cases inverse propensity score methods fail and model based methods are highly conditional on assumptions.
- Optimizing an argsort efficiently a neglected problem. Methods for learning of the decision rule/policy in the literature are often restricted to the case where $K = 1$ in which case the argsort can be replaced with an argmax which can be approximated using a differentiable softmax allowing $\Xi, \boldsymbol{\beta}$ to be optimized using deep learning tools. The softmax itself can be a computational bottleneck which can be (approximately) dealt with by using a sampled softmax adapted to policy learning setting [5]. A partial solution to these issues is present in [4] which replaces the argsort with a method that optimizes an un-orderd set (within the top K). A more direct solution to the problem could be obtained with a differentiable argsort e.g. [2].

A parenthetical remark: Finding the optimal decision rule can be achieved by solving for $\Xi, \boldsymbol{\beta}$ using optimisation methods e.g. deep learning. It also may be unnecessary to do this optimisation if the parametric form of the model has a form sufficiently similar to the decision rule. For example if

$$P(c = 1|a_1, a_2, X) = \frac{\exp(g_\Omega(\boldsymbol{X})^T \boldsymbol{\Psi}_{a_1}) + \exp(g_\Omega(\boldsymbol{X})^T \boldsymbol{\Psi}_{a_2})}{\phi + \exp(g_\Omega(\boldsymbol{X})^T \boldsymbol{\Psi}_{a_1}) + \exp(g_\Omega(\boldsymbol{X})^T \boldsymbol{\Psi}_{a_2})},$$

then $g_\Omega(X)^T \boldsymbol{\Psi}_a$ is a scalar "score" of action $a$ and the argsort is optimal and can be achieved by setting the decision rule to $f_\Xi(\boldsymbol{X}) = g_\Omega(X)$ and $\boldsymbol{\beta} = \boldsymbol{\Psi}$. In other situations $\Xi, \boldsymbol{\beta}$ must be found using numerical optimization.

## 4 A PRACTICAL ALTERNATIVE THAT DELIVERS FAST RECOMMENDATIONS AT SCALE

In our practical formulation we propose the following method. For a given context $\boldsymbol{X}$ we specify a desirable list of actions $a_1^*, ...a_K^*$. We produce this list not by looking at the logs of the recommender system but at associations between items in the recommender system logs. That is we assume that if a user with context views item $a$ it suggests that we are likely to get a reward if we recommend item $a$ to that user.

In practice we do not explicitly define the optimal list $a_{1:K}$ for a given $\boldsymbol{X}$ rather we define a heuristic where we 'sample positives' for items that we consider good recommendations for user $\boldsymbol{X}$ and sample negatives for items that we consider bad for user $\boldsymbol{X}$.

If for each context $\boldsymbol{X}_n$we define a positive actions $a^+_{n,1..R}$ and a negative action $a^-_{n,1..S}$ then the loss becomes:

$$\mathcal{L}(\Xi, \boldsymbol{\beta}) = \sum_n \{ \sum_r \log \sigma \{ f_\Xi(\boldsymbol{X}_n)^T \boldsymbol{\beta}_{a^+_{n,r}} - f_\Xi(\boldsymbol{X}_n)^T \boldsymbol{\beta}_{a^-_{n,s}} \}$$

The characteristics of this procedure are:

| a | X | position |
|---|---|---|
| phone 1 | telephony | 1 |
| phone 2 | telephony | 2 |
| phone 3 | telephony | 3 |
| phone 4 | telephony | 4 |
| phone 5 | telephony | 5 |
| book 1 | telephony | 6 |
| book 2 | telephony | 7 |
| book 3 | telephony | 8 |
| book 4 | telephony | 9 |
| book 5 | telephony | 10 |
| beer | telephony | 11 |
| book 1 | literature | 1 |
| book 2 | literature | 2 |

**Table 1: Selected rows of what the proxy ranking method produces - for each context X there is an optimal ranking defined. In practice these rankings are defined by heuristics that sample positives and negatives for each context. Optimisation is used to minimize distance with an implementable ranker and the "optimal rank" for each context.**

- A heuristic is used instead of explicit and direct use of the reward logs, this negates the difficulty in obtaining reliable rewards for actions rarely done by the production system at the cost that the reward model is replaced with a proxy. In simple terms the recommender system has previously never recommended rice to a user with phones in their history - so the bandit feedback is silent on this action - in contrast rice and phones infrequently co-occur so subject to co-occurrence being a good proxy of reward it can be estimated more easily.
- No softmax or other operations that scale with the catalogue size are performed, this allows fast training even when the catalogue sizes are vast. The trade-off with this approach is that what is effectively optimised is the 'edit distance' of an optimal action $a_1^*, ...a_K^*$ and the list delivered $a_1, ...a_K$ - this is simplistic e.g. for some users there might be precisely $K$ good items so delivering precisely those items is important for other users there may be many good recommendations that have similar value if they were delivered in the top $K$. The simple edit distance cannot represent this complexity.

These compromises turn out to be extremely powerful in practice - a more complete description with experiments on standard datasets and A/B tests is given in [7].

## 5 EXAMPLE AND DISCUSSION

Let's step through a toy example. Imagine that we have a catalogue of 11 items, Five of these items are phones which we denote phone 1, phone 2,..., phone 5 and five of these items are books that we denote book 1, book 2 ,..., book five the final item is beer.

We will observe a context $\boldsymbol{X}$ that has one of three states: interested in telephony, literature or drinks. We want to deliver recommendations in slates or banners of size 2. Finally we must adhere to engineering constraints we cannot do arbitrary recommendations exhaustively searching the catalogue.

| $P(c=1\|a_1,a_2,X)$ | $a_1$ | $a_2$ | X |
|---|---|---|---|
| 0.20 | phone 1 | phone 2 | telephony |
| 0.19 | phone 2 | phone 3 | telephony |
| 0.18 | phone 4 | phone 5 | telephony |
| 0.06 | book 1 | phone 2 | telephony |
| 0.01 | book 1 | book 2 | telephony |
| 0.01 | phone 1 | phone 2 | drinks |
| 0.10 | beer | phone 2 | drinks |

**Table 2: Selected rows of what a reward model might predict in our hypothetical example using slates of size 2.**

In order to pursue the proxy approach for each of our contexts we use whatever signals are available to us to produce an idealised itemized list. For example for a user interested in literature we may use co-occurrences of items (a strong signal in many contexts, but one that is only a proxy to reward) and we might find that the order of co-viewing X=literature and each item is book 1, book 2, book 3, book 4, book 5, phone 1, phone 2, phone 3, phone 4, phone 5. This ordering is not usually defined explicitly but rather it is defined by a heuristic definition of how to sample positive and negative examples. While the heuristic may be quite good, it is not easy to make fine judgements about different proposals for creating this ordering and it is necessary to use extensive A/B testing.

Once an ordering for each context is defined we optimize a decision rule that satisfies engineering constraints by minimizing a distance between the delivered list of recommendations and the preferred list of recommendations. We use optimization techniques to deliver the best list by this criterion. Fortunately optimization that minimizes the distance between two ordered lists can be performed using mini-batches of the lists resulting in fast learning algorithms for large catalogues this is in practice *a massive practical advantage*.

Now consider a full decision theoretic approach. We need to estimate $P(c=1|a_1,a_2,X)$ - this in general might be a very difficult problem although in a small problem if we had a random logging policy that is if we have sufficient historical data including historical data of *preposterous recommendations* such as recommending beer and book 1 to a user interested in telephony; then we can see how such an estimation might be possible. We should immediately note that this is a far more complex structure than the proxy approach has used (which had a single list per context). In this scenario we have for every context a two item list mapping to a reward. Table 1 and Table 2 illustrate the two formulations. Table 2 is also simplified by the fact that the slate size is 2 here in general it would be much larger.

The reward model as presented in Table 2 makes it clear that the cost incurred for doing a sub-optimal ranking is uneven. For example if the context is telephony then the optimal ranking might be phone 1, phone 2, phone 3, phone 4, phone 5, book 1, book 2, book 3, book 4, book 5, beer. The reward model might agree that a slate consisting of phone 1 and phone 2 is optimal - but it might also note that a slate consisting of phone 4 and phone 5 still has almost the same performance. This nuance is lost when we compute a ranking loss with a recommender system that produces a list for telephony starting with items phone 4 and phone 5.

Once the reward model is estimated we must find the decision rule that will deliver optimal recommendations. This represents a further optimization problem that we argue current state of the art methods do not quite handle. The key complexity is the argsort in Equation 1 which we need to be able to differentiate through to efficiently optimize we also need this to be scaleable for large catalogues. We present current state of the art methods for related problems and suggest a solution for finding a differentiable argsort leave scaling this to large catalogues to future work.

The policy optimization for exponential models (POEM) [9] solves this problem for cases where the slate/banner is of size 1 reducing the problem to an argmax which can be approximated using a softmax. In this formulation there is only one item in the slate so we have $P(c|a,X)$ and the decision rule is a categorical over the action space which can be viewed as a "probability" $\pi(a|X)$. The optimal solution places all the probability on the optimal action for a given context. They propose using the inverse propesnity score (Horwitz-Thompson) estimator for $P(c|a,X)$.

A proposal to go beyond this single item slate formulation was outlined in [4] using the concept of a Top K recommender. Here they use the same formulation of the decision rule over a single action i.e. $\pi(a|X)$ but modify the optimization procedure so that items in the top K do not find the optimal value at 1. They also propose scaling to large catalogues using a sampled softmax and the reinforce algorithm again combined with a Horwitz-Thompson style estimator for $P(c|a,X)$.

The approach of [4] while interesting doesn't quite handle our use case of handling ordered lists of size 2 (or more). It rather works with sets and in fact continues to use a decision rule designed for a single item slate setting $\pi(a|X)$.

We suggest that a simple alternative to handling this is to instead use $\pi(a_1,a_2|X,\boldsymbol{\beta},\Xi) = \frac{\exp(\boldsymbol{\lambda}_{a_1})}{Z}\frac{\exp(\boldsymbol{\lambda}_{a_2})}{Z-\exp(\boldsymbol{\lambda}_{a_1})}$ where $Z=\sum_p \exp \boldsymbol{\lambda}_p$ and $\boldsymbol{\lambda}_a = f_\Xi(\boldsymbol{X}_n)^T\boldsymbol{\beta}_a$ is a personalized score of recommendation $a$ on context $\boldsymbol{X}$.

To give intuition imagine that for context $X'$ the best recommendation will be item 1 and item 2 and item 3 and item 4 are not to be included but the order does not matter. In this case if $\boldsymbol{\lambda}_1$ is much greater than $\boldsymbol{\lambda}_2$ and $\boldsymbol{\lambda}_2$ is much greater than $\boldsymbol{\lambda}_3$ and $\boldsymbol{\lambda}_4$. Then we arrive at $\pi(a_1=1,a_2=2|X)\longrightarrow 1$. The relative positions of item 3 and item 4 scores' is not important. This differs markedly from the approach in [4] which uses a uni-action $\pi(a|X)$ and would try to share probability between $\pi(a=1|X)$ and $\pi(a=2|X)$. Although unfortunately the calculation $Z=\sum_p \exp \boldsymbol{\lambda}_p$ scales poorly with large catalogues.

## 6 CONCLUSION

We have discussed an idealised machine learning approach to optimizing reward of a recommender system and outlined the challenges that would occur when deploying such a system in practice. We also outlined a practical approach that has been proven in a production environment. The differences amount to using heuristics in place of an estimate of the true reward and the replacement of an argsort with a ranking loss. This discussion raises an obvious question: *Can the practical limitations of the reward approach be overcome and further improve performance?*

The key practical strengths of the heuristic approach are: nothing scales as per catalogue size and problems caused by uneven estimation of rewards is bypassed by the use of the proxy.

The key practical issues with the heuristic approach is that there is no clear way to establish how positive and negative recommendations should be sampled, this must be done experimentally with the ultimate arbiter being A/B testing. This approach can also suffer from Goodhart's law: *"Any observed statistical regularity will tend to collapse once pressure is placed upon it for control purposes."* Furthermore the ranking loss is insensitive to the variable cost of mis-ranking.

By contrasting the two approaches we hope to engage researchers and engineers on grappling with this difference between theory and practice.

## REFERENCES


[1] Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. 2014. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender systems*. 257–264.

[2] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. 2020. Fast differentiable sorting and ranking. In *International Conference on Machine Learning*. PMLR, 950–959.

[3] Denis Charles, Max Chickering, and Patrice Simard. 2013. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research* 14 (2013).

[4] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 456–464.

[5] Thorsten Joachims, Adith Swaminathan, and Maarten de Rijke. 2018. Deep learning with logged bandit feedback. In *International Conference on Learning Representations*.

[6] Omid Keivani, Kaushik Sinha, and Parikshit Ram. 2018. Improved maximum inner product search with better theoretical guarantee using randomized partition trees. *Machine Learning* 107, 6 (2018), 1069–1094.

[7] Olivier Koch, Amine Benhalloum, Guillaume Genthial, Denis Kuzin, and Dmitry Parfenchik. 2021. Scalable representation learning and retrieval for display advertising. *arXiv preprint arXiv:2101.00870* (2021).

[8] Otmane Sakhi, Stephen Bonner, David Rohde, and Flavian Vasile. 2020. BLOB: A Probabilistic Model for Recommendation that Combines Organic and Bandit Signals. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 783–793.

[9] Adith Swaminathan and Thorsten Joachims. 2015. Batch learning from logged bandit feedback through counterfactual risk minimization. *The Journal of Machine Learning Research* 16, 1 (2015), 1731–1755.