Preventing Contrast Effect Exploitation in Recommendations

Chris Nota
cnota@cs.umass.edu
University of Massachusetts
Amherst, MA, USA

Georgios Theocharous
theochar@adobe.com
Adobe Research
San Jose, CA, USA

Michelle Saad
msaad@adobe.com
Adobe
Austin, TX, USA

Philip S. Thomas
pthomas@cs.umass.edu
University of Massachusetts
Amherst, MA, USA

ABSTRACT

Contrast effects are caused by the tendency for mental evaluations of objects to be influenced by one or more contrasting objects. As this cognitive bias can unduly influence users' behavior, we present a method for preventing recommender systems from exploiting contrast effects. We then apply our method to a simulated online storefront and show that it is able to prevent contrast effect exploitation while maximizing the conversion rate of users.

KEYWORDS

recommender systems, cognitive biases, machine learning

ACM Reference Format:
Chris Nota, Georgios Theocharous, Michelle Saad, and Philip S. Thomas. 2021. Preventing Contrast Effect Exploitation in Recommendations. In *SIGIReCom '21: The 2021 SIGIR Workshop On eCommerce, July 15, 2021.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Recommender systems are often trained on simple metrics such as *click-through rate* (the percentage of users who click on an ad) or *conversion rate* (the percentage of users who buy the product). These metrics do not necessarily capture user satisfaction. Like a shady used car salesman, such systems will learn to use any mechanism at their disposal to "make the sale," without any regard for long-term or ethical considerations. While superficially desirable, this can, in the long run, drive away users and harm the reputation of the seller. Therefore, in order to ensure that systems are connecting users with products that will lead to high satisfaction, it is desirable to prevent the systems from using such tactics [8].

One way that recommender systems can exploit users is by taking advantage of humans' *cognitive biases*, which are a type of systematic deviation from rational decision making [3]. *Contrast effects* are one category of bias particularly relevant to the problem of online recommendations. Contrast effects come from the tendency for mental evaluations of objects to be influenced by one or more contrasting objects [7]. For example, the *compromise effect* is the tendency of consumers to avoid the most expensive and most inexpensive options, instead choosing items at a median price point, regardless of the absolute scale of the pricing [9]. If a shady used car salesman wanted to encourage the customer to purchase a specific vehicle, they could show the customer a less expensive and a more expensive vehicle in order to exploit this effect. Because recommender systems often show users not just a single recommendation, but a *set* of recommendations, contrast effects can be similarly exploited in online recommendations.

We present a method for detecting contrast effects and allowing the system implementer to control the extent to which the algorithm is allowed to exploit them—at one extreme, completely preventing the algorithm from exploiting contrast effects, and at the other, allowing it to exploit them as much as possible. We demonstrate the effectiveness of this algorithm on a simulated environment based on a dataset of Amazon reviews [6].

2 BACKGROUND

We consider a recommender system for an online storefront. The store contains some set of items, $\mathcal{I}$. At each timestep, t, the system selects some subset of items, $\mathcal{I}_t$, based on some context vector, X_t, which contains information about the customer and their current browsing session. The system then selects a single item, I_t, from this subset. Finally, the system selects a *contrast group*, G_t, containing the target item and two similar items. We use $\mathcal{G}$ to represent the set of all such groups, and $\mathcal{G}_i$ to represent the set of all groups containing item i. The selected item and the contrast group are then displayed to the user. If the user purchases the item, the system receives a reward, R_t, of 1, otherwise it receives a reward of 0. The end-to-end system is shown in Figure 1.

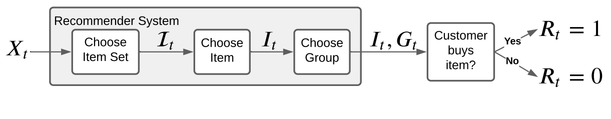

Figure 1: The three-phase recommender system.

We focus on the problem of selecting I_t and G_t given a subset of items $\mathcal{I}_t$, and assume that $\mathcal{I}_t$ is chosen by some other subsystem. Let π_t be the *policy* at time t for selecting items and groups, such

SIGIReCom '21, July 15, 2021, Online
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

that $\pi_t(\mathcal{I}_t, I_t, G_t) := \Pr(I_t, G_t|\mathcal{I}_t)$. Let $r : \mathcal{I}, \mathcal{G} \to [0,1]$ be the *conversion rate*, that is, the probability of the user purchasing the item when shown item i and group g:

$$r(i,g) = \mathbb{E}[R_t | I_t = i, G_t = g]. \tag{1}$$

The naive goal of the system is to choose a sequence of policies that maximizes the expected sum of rewards, $\sum_{t=0}^{T} \mathbb{E}[r(I_t, G_t)|\pi_t]$, where T is the lifetime of the system. We decompose the conversion rate as the sum of two components, the "unbiased conversion rate," which represents the probability that the user would select item i assuming that it was displayed independently of any other items, and the "contrast effect," which captures the impact of the group g on the customer's perception of i:

$$r(i,g) = \underbrace{r_i(i)}_{\text{unbiased conversion rate}} + \underbrace{b(i,g)}_{\text{contrast effect}}. \tag{2}$$

We propose to solve the optimization problem

$$\max_{\pi} \mathbb{E}[b(I,G)|\pi] \tag{3}$$

$$\text{s.t. } \pi \in \arg\max_{\pi'} \mathbb{E}[r_i(I)|\pi']. \tag{4}$$

That is, we constrain the solution to first maximize the "unbiased" conversion rate, $r_i(I)$, and maximize the contrast effect, $b(I,G)$, subject to this constraint. The difficulty comes from the fact that we do not have unbiased samples of $r_i(I)$, only samples of $r(I,G)$.

3 METHODOLOGY

The core idea of our approach is to estimate the unbiased conversion rate for each item in the catalog by learning the bias function, $b(I,G)$, and extrapolating the unbiased conversation rate by compensating for this bias. We will then combine this with a novel variant of UCB in order to efficiently learn the unbiased conversion rate. One of the basic assumptions of our approach is that because contrast effects are universal human biases, they do not need to be estimated for each individual item. Instead, it is possible to share information between items and even between catalogs.

3.1 Distance Scores

For each contrast group, we compute a *distance* score, $d : \mathcal{G} \to \mathbb{R}$, which measures the similarity of the items in a given contrast group using features available in the catalog. A single score is computed for the group of three items. We then assume that the biased conversion rate is a function of the following form:

$$\hat{r}(i,g) = \hat{r}(i) + \hat{b}(d(g)), \tag{5}$$

where i is an item and g is a contrast group. That is, it is some function of the distance added to the unbiased conversion rate. We then learn $\hat{r}$ and $\hat{b}$ using least squares regression. Once we have learned these functions, we can simply choose an item in $\arg\max_{i \in \mathcal{I}_t} \hat{r}(i)$ and a group in $\arg\max_{g \in \mathcal{G}} \hat{b}(d(g))$. Below, we will discuss how these functions can be learned and how exploration and learning can be balanced with exploitation in the online setting.

3.2 Bias Functions

For a simplified example of how this approach works, consider the case shown in Figure 2. While this example is unlikely to accurately reflect human biases, it shows the high-level idea of the approach.

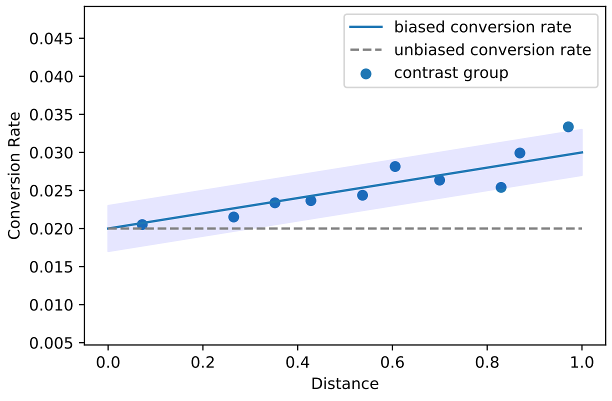

Figure 2: A demonstrative (but unrealistic) example of how the unbiased conversion rate could be estimated, if a distance score of 0 corresponded to "unbiased" behavior. The dots represent the observed conversation rates for each contrast group containing the hypothetical item.

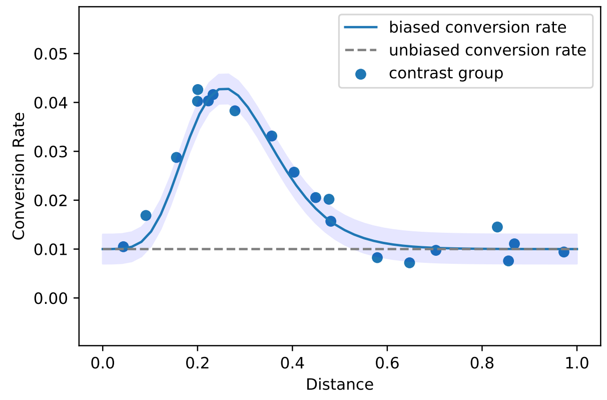

Figure 3: A more plausible bias curve for a user. The dots represent the observed conversation rates for each contrast group containing a hypothetical item. If the items in the group are too different or too similar to the target item, the effect is not present. At some ideal "hot spot," the effect is maximized.

Each point represents a particular contrast group for a given item. The x-axis represents the distance between the items in the group, and the y-axis represents measurements of the biased conversion rate from customer behavior. Suppose that as the distance between the items in the group approaches 0, the customer behavior will become more and more like that of a customer that was not shown a contrast group at all, that is, the unbiased conversion rate. At distance 0, the customer is simply shown three of the same item, and grouping effects should not be present at all (note that this is for illustrative purposes, and we do not assume this). By fitting a line to the measured points, we can extrapolate the unbiased conversion rate by simply looking at the y-intercept.

What if distance 0 does not correspond to unbiased behavior? Suppose unbiased behavior corresponds to some other arbitrary distance value, such as 0.1. Because we assume the shape of the curve (e.g., Figure 2 or Figure 3) is independent for all i given the distance score, the "ordering" of the items does not depend on which distance (if any) corresponds to an unbiased grouping. We will next discuss more details about how this function can be learned, including in the non-linear case.

3.3 Learning the Bias Function

In order to estimate the unbiased conversion rate for a given item, our general approach is to try to estimate the *biased* conversion rate for each item-group pair, (i, g), as a function of the distance, $d(g)$, and an independent parameter for each item, r_i. In order to learn more expressive functions of the distance, we use a basis expansion, $\phi(d(g))$, for example, the Fourier basis [5]. We estimate the biased conversion rate as

$$\hat{r}(i, g) = \mathbf{w}^T \phi(d(g)) + r_i, \tag{6}$$

where $\mathbf{w}$ is the learnable weight vector, and r_i is the learnable item-dependent scalar. Given some dataset of tuples, (i, g, r), where i is an item, g is a contrast group, and r is 1 if the customer purchased the item and 0 otherwise, $\hat{r}$ can be learned using any off-the-shelf linear regression package. Note that $\mathbf{w}$ is shared between all items and groups in the dataset. This means that the effects of contrast bias do not have to be independently estimated for all items.

In order to learn the function above, at time t, we construct a $|\mathcal{I}| + M$ feature vector $\mathbf{x}_t$ where M is the size of the basis expansion. $\mathbf{x}_{t,i}$ is 1 if item i was chosen at time t and 0 otherwise, and the remaining elements are the elements of $\phi(d(G_T))$. We then learn the parameters using ordinary least squares.

3.4 Debiasing UCB

In this section, we will extend the standard UCB1 [1] algorithm to minimize the unbiased conversion rate, rather than the standard biased conversion rate. The vanilla UCB1 algorithm, at each timestep, selects the item with the highest upper confidence bound according to

$$\text{UCB}(i) = \mu_i + C\sqrt{\frac{2\ln \sum_{j \in I_t} S_j}{S_i}}, \tag{7}$$

where μ_i is the observed conversion rate for item i, S_i is the number of times that item i has been recommended, I_t is the set of items available at time t, and C is a positive constant. The first term may be thought of as an estimate of the value, and the second term may be thought as an "exploration bonus." Notice that as time goes on, if item i is never recommended, then the exploration bonus will continue to grow. This guarantees that as t goes to infinity, every item will be recommended an infinite number of times and estimates of μ_i will converge to the true conversion rate by the law of large numbers.

In our setting, $r(i)$ cannot be estimated directly; rather, it must be extrapolated. Instead of directly computing the confidence interval using Hoeffding's inequality, as in UCB1, we can instead compute the uncertainty in the parameter b_i for a given confidence level. By increasing the confidence level over time, for example, by choosing a confidence level of $1 - 1/t$, we can achieve the same effect of guaranteeing that every item is eventually chosen. To compute the upper confidence bound for the item-dependent parameter, r_i, we compute a t-test-style confidence interval over the regression coefficients:

$$\text{CI}^+(i, \alpha) = r_i + t_{1-\alpha, n-2}\sqrt{\frac{\frac{1}{n-2}\sum_t (R_t - \hat{r}(I_t, G_t))^2}{(\mathbf{x}^\top \mathbf{x})^{-1}_{i,i}}}, \tag{8}$$

where i is the item number, α is a confidence level, I_t is the item chosen at time t, G_t is the group chosen at time t, R_t is the reward at time t, n is the number of time steps so far, and $\mathbf{x}$ is a matrix containing the feature vectors for each timestep. We then simply choose $\arg\max_{i \in \mathcal{I}} \text{CI}^+(i, \alpha)$.

However, this only solves the problem of which item to choose. In order to solve the full problem, we must also consider how to choose the product grouping. Technically, the product grouping does not affect the unbiased conversion rate directly, so according to this metric, nearly any algorithm will do, with the caveat that it must maintain the asymptotic guarantee that each group will be chosen infinitely often. This is necessary in order to ensure that the linear regression converges correctly. For example, consider an algorithm which always chooses a grouping with a fixed distance score (e.g., $d = 0.5$). With just one x-value, there would be many solutions to the regression problem.

We propose to simply use standard UCB1 for choosing the product groupings. In the case that UCB1 is used for both the item selection and the group selection, the resulting algorithm is simply the UCT algorithm [4] and achieves the resulting convergence guarantees. However, if we replace the item selection algorithm with our modified linear regression approach, we get a new algorithm, Algorithm 1, that first attempts to minimize the debiased regret.

Algorithm 1: Debiased UCB

```
Initialize $\mathcal{D} = \{\}$
foreach episode do
    $i_t = \arg\max_i \text{CI}^+(\mathcal{D}, i, \alpha_t)$;
    $g_t = \text{choosegroup}(i_t)$;
    $d_t = \text{distance}(g_t)$;
    $r_t = \text{recommend}(g_t)$;
    $\mathcal{D} \leftarrow (i_t, d_t, r_t)$;
    $t = t + 1$;
end foreach
```

The schedule for α_t is a hyperparameter. As training progresses, we want a higher and higher confidence level in order to ensure sufficient explanation. In our experiments, we chose the schedule $\alpha_t = 1 - \frac{c_0}{1+\gamma t}$, where c_0 and γ are adjustable.

3.5 Adjusting the Exploitation Coefficient

Some implementers of this algorithm may wish to control the extent to which the recommender system exploits contrast effects. For example, we may wish to minimize the effects of contrast bias when the cost (in terms of the biased regret) is low, but not when it would cost a large number of sales. To achieve this, we introduce

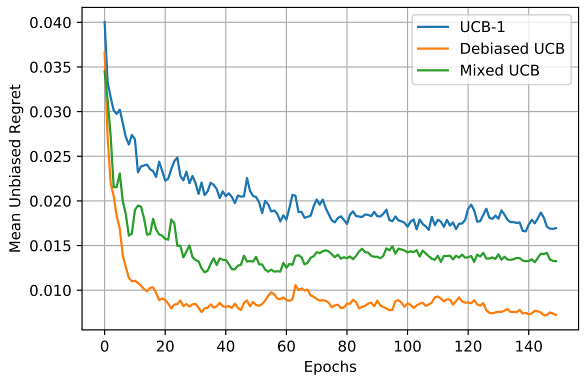

Figure 4: The unbiased regret for each algorithm.

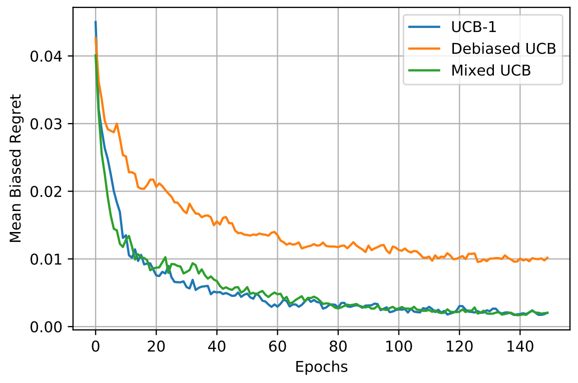

Figure 5: The biased regret for each algorithm.

an adjustable "exploitation" parameter, $\beta \in [0,1]$, which trades off between the biased and debiased algorithms, where $\beta = 0$ is equivalent to debiased UCB and $\beta = 1$ is equivalent to standard UCB. The algorithm works by taking a weighted average of the debiased and standard confidence bounds, with β determining the weight on each. That is, at each timestep we choose

$$\arg\max_i \left((\beta)\text{UCB}(i) + (1-\beta)\text{CI}^+(i,\alpha)\right), \tag{9}$$

where $\text{UCB}(i)$ is defined by Equation 7 and CI^+ by Equation 8.

4 EXPERIMENTS

In order to test the effectiveness of the debiasing UCB algorithm, we implemented a recommendation system simulator based on the Amazon review dataset [2, 6]. Product groupings were created based on co-view information. Customer behavior was simulated based on product review data. We simulated a Gaussian contrast effect curve similar to Figure 3. A distance metric was created to measure the similarity of the items in a product group based on review data and pricing information. In each round, the simulator selects a subset of items randomly. This is intended to reflect the system selecting items based on customer context information. The agent must select an item from this subset and then select a product grouping based on this item.

The two primary metrics we consider are the "biased" and "unbiased" regret. The biased regret is the difference between the conversion rate for the best item/group pair available at a given time step and the conversion rate for the item/group pair that the algorithm actually chose, that is

$$\max_{i\in\mathcal{I}_t}\max_{g\in\mathcal{G}_i} r(i,g) - r(I_t, G_t). \tag{10}$$

The unbiased regret is the difference between the true unbiased conversion rate and the unbiased conversion rate for the item the algorithm chose:

$$\max_{i\in\mathcal{I}_t} r(i) - r(I_t). \tag{11}$$

We compare the standard UCB1 algorithm, Algorithm 1 ("Debiased UCB"), and the mixed algorithm with an exploitation coefficient of $\beta = 0.5$. We averaged results over 10 trials.

4.1 Results

The unbiased regret is plotted over time in Figure 4 and the biased regret is plotted in Figure 5. The standard algorithm, as expected, nearly minimizes the biased regret, but the unbiased regret plateaus. The biased and unbiased regret are correlated to some extent because some items are simply not likely to be purchased, regardless of the contrast group. Similarly, the debiased algorithm minimizes the debiased regret, while still substantially reducing the biased regret. The mixed algorithm achieves a biased regret score essentially on par with the standard algorithm, while still reducing the unbiased regret. This means that there is, at least in these experiments, no trade-off for using the mixed algorithm compared to the biased algorithm. That is, the mixed algorithm "sells" as many items as the biased algorithm while exploiting contrast effects less.

5 FUTURE WORK

While the simulator we used was based on a real-world dataset, the contrast effect was simulated. One of the most important areas of future work is to validated the existence of contrast effects in the target setting, and to determine the effectiveness of the proposed algorithm for realistic contrast effects.

6 CONCLUSIONS

We introduced the problem of contrast effect exploitation and described a variant of UCB1 which prevents the recommender system from exploiting contrast effects while maximizing the conversion rate of users. We showed that our method was effective in a simulated online storefront, and that the "mixed" version of our method was a strict improvement over standard UCB. We contribute to the literature on building systems that meet the needs of the users without taking advantage of human weaknesses.

ACKNOWLEDGMENTS

To Adobe for funding this research, Sridhar Mahadevan for providing helpful feedback and guidance, and Wes Cowley for assisting with preparation of the manuscript.

REFERENCES

[1] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47, 2-3 (2002), 235–256. https://doi.org/10.1023/A:1013689704352

[2] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*. 507–517. https://doi.org/10.1145/2872427.2883037

[3] Daniel Kahneman and Amos Tversky. 1972. Subjective probability: A judgment of representativeness. *Cognitive Psychology* 3, 3 (1972), 430–454. https://doi.org/10.1016/0010-0285(72)90016-3

[4] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*. Springer, 282–293. https://doi.org/10.1007/11871842_29

[5] George Konidaris, Sarah Osentoski, and Philip Thomas. 2011. Value function approximation in reinforcement learning using the Fourier basis. In *Twenty-fifth AAAI Conference on Artificial Intelligence*. https://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3569/3885

[6] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 43–52. https://doi.org/10.1145/2766462.2767755

[7] Scott Plous. 1993. *The psychology of judgment and decision making*. Mcgraw-Hill Book Company.

[8] Georgios Theocharous, Jennifer Healey, Sridhar Mahadevan, and Michele Saad. 2019. Personalizing with human cognitive biases. In *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*. 13–17.

[9] Jaewon Yoo, Hyunsik Park, and Wonjoon Kim. 2018. Compromise effect and consideration set size in consumer decision-making. *Applied Economics Letters* 25, 8 (2018), 513–517.