OFF-Set: One-pass Factorization of Feature Sets for Online Recommendation in Persistent Cold Start Settings

Michal Aharon, Natalie Aizenberg, Edward Bortnikov, Ronny Lempel
Yahoo! Labs, Haifa, Israel
{michala,anatalia,ebortnic,rlempel}@yahoo-inc.com

Roi Adadi, Tomer Benyamini, Liron Levin, Ran Roth, Ohad Serfaty
Yahoo! Smart Ad, Tel Aviv, Israel
{roiaddi,tomerb,liron,ranr,ohad}@yahoo-inc.com

ABSTRACT

One of the most challenging recommendation tasks is recommending to a new, previously unseen user. This is known as the *user cold start* problem. Assuming certain features or attributes of users are known, one approach for handling new users is to initially model them based on their features.

Motivated by an ad targeting application, this paper describes an extreme online recommendation setting where the cold start problem is perpetual. Every user is encountered by the system just once, receives a recommendation, and either consumes or ignores it, registering a binary reward.

We introduce One-pass Factorization of Feature Sets, *OFF-Set*, a novel recommendation algorithm based on Latent Factor analysis, which models users by mapping their features to a latent space. OFF-Set is able to model non-linear interactions between pairs of features, and updates its model per each recommendation-reward observation in a pure online fashion. We evaluate OFF-Set against several state of the art baselines, and demonstrate its superiority on real ad-targeting data.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithms, Experimentation

Keywords

Dynamic Ad Optimization, Matrix Factorization, Collaborative Filtering, Persistent cold start problem

1. INTRODUCTION

In recent years, Collaborative Filtering (CF) based recommender systems have gained both commercial success and increased focus from the research community. Generally speaking, CF discovers and exploits recurring consumption patterns of items by users, at scale, in order to recommend items to users. Studied consumption types include ratings of items by users, which explicitly articulate users' likings and dislikings of items; binary indication of users' consumption of items, without explicit evidence regarding the users' opinions on their consumed items; richer implicit interaction settings where users, while still not explicitly providing ratings on items, can perform multiple operations on items (e.g. click or comment on news stories), etc.

In order to provide high-quality recommendations of items to a user, recommender systems must observe some past activity of the user. When new users first arrive to a system, no previous activity is available and they are said to be "cold"; recommending to such users is challenging, and has been coined the "user cold-start problem" [12, 6, 8].

Motivated by an online ad targeting application (Section 3), this paper addresses an extreme online recommendation setting where the cold start problem is perpetual - practically every user encountered by the system is seen just once, i.e. every recommendation request is for a previously unseen user. Nevertheless, certain attributes of the users are known, and those are leveraged for recommendation. For every recommendation made, a binary indication (i.e. reward) is observed. The goal is to maximize the reward.

We introduce One-pass Factorization of Feature Sets - *OFF-Set* – a novel recommendation algorithm based on *latent factor analysis* [7], which models both users and items by mapping their features to a latent space. OFF-Set offers a non-linear solution by designing a latent space that models individual as well as pairwise combinations of features. OFF-Set is formulated for purely online recommendation, performing lightweight updates of its model per each recommendation-reward observation. We evaluate OFF-Set against several state of the art baselines, and demonstrate its advantage on real ad-targeting data.

2. RELATED WORK

The ever-growing demand for automated recommendations at scale has prompted the development of a variety of recommendation techniques. Collaborative Filtering (CF) algorithms are often the alternative of choice, among which, the *matrix factorization* (MF) [7] technique is successful and popular [3, 2]. In its most basic form, MF associates each user and each item with a latent factor (vector). The match score between a user and an item is represented as an inner product between the corresponding vectors.

RecSys '13, October 12 - 16 2013, Hong Kong, Hong Kong
ACM 978-1-4503-2409-0/13/10 ...$15.00.

Although collaborative filtering algorithms have been gaining great success in recent years, they are often challenged by the well-known *user cold-start* problem [6] – namely, making recommendations to previously unobserved users. In oder to overcome this problem at scale, reference to user features is often done. One such approach is *factorization machines* (FM), first presented by Rendle [9, 10]. FM is known for its capability to model sparse feature spaces. It is a combination of *support vector machines* (SVM) [4] and matrix factorization, and it closely fits our problem specification. FM was shown to be superior to other approaches, including "regular" matrix factorization [11].

Outside of collaborative filtering there may be found other solutions for feature-based prediction modeling to offer a different approach to the cold-start and sparsity issues. The *gradient-boosted decision trees* (GBDT) algorithm [5] is one such approach. In contrast with linear methods, GBDT models the scoring process as an aggregation over an ensemble of decision trees. GBDT has been shown to be successful in non-latent feature spaces, including those with interdependent features.

Both FM and GBDT are not designed to work in an online setting, as they require several passes over the data in order to converge. In this sense they differ from our approach, as OFF-Set is completely online – i.e., every input element is processed exactly once and the ensuing update of the model is lightweight.

3. DYNAMIC ADVERTISING CAMPAIGNS

Dynamic advertising campaigns are a relatively new form of digital display advertising, where the advertiser offers multiple products or multiple ad creative alternatives (referred to as 'ad variants') to be presented to users. Unlike traditional display advertising settings, where the ad exchange assigns a specific ad to every user impression, in dynamic campaigns the exchange designates a campaign to the impression. It is up to a follow-up process - a selection algorithm - to determine which variant, i.e. which specific ad instance, to actually serve to the user. That decision is based upon data (context) provided to the selection algorithm, e.g. properties about the user, the page content surrounding the ad slot, the time of day, etc. In this work we assume only basic demographic information about the users (age, gender, geographic location) is known.

A typical optimization task in dynamic campaigns is to maximize the campaign's click-through rate (CTR), i.e. the CTR over all user impressions which were served by some ad variant of the campaign. We formulate dynamic ad campaign optimization as a recommendation task which attempts to assign, to each user impression, the variant that a user is most likely to click on.

Display ads, as opposed to content (e.g. articles, search results) and other digital advertising types (e.g. sponsored search) are characterized by very low CTR. In addition, many campaigns *target* (i.e. compete over) the same set of users, resulting in most users being exposed to a campaign no more than once. These two factors lead to an extremely sparse user signal - most users do not register a single click in most campaigns - and impose a great challenge to any recommendation algorithm.

4. NOTATIONS

Our problem is about recommending items, in our case ad variants $\mathbf{A} = \{a_1, a_2, \ldots, a_L\}$, to users $\mathbf{U} = \{u_1, u_2, \ldots, u_M\}$. Each user u_i is associated with a set of feature values $[u_i^1, u_i^2, \ldots, u_i^K]$, where $u_i^k \in \mathbf{F}_k$, and $\mathbf{F}_k$ is the set of available values for feature k (e.g., one feature can be 'gender', where the set of available values is 'male', 'female' and 'unknown'). We now define $\mathbf{C}$ as the set of (user, ad variant) pairs that resulted in a click Similarly, we denote by $\mathbf{NC}$ the set of (user, ad variant) pairs that did not result in a click. Finally, let $\mathbf{N} = \mathbf{C} \bigcup \mathbf{NC}$ be the union of the two sets.

Figure 1: Example of Dynamic Ad Campaign (for Yahoo! Shopping)

5. THE OFF-SET ALGORITHM

Dynamic ad selection is yet another ranking problem. Given a new user entering the system, we aim at selecting the best possible ad variant that will maximize the probability of a click, thus maximizing the CTR of the overall campaign. Like in many other ranking problems, we chose to use a latent factors approach, where each ad variant and each user are represented by a latent vector, $\mathbf{v}_{a_j} \in R^D$ and $\mathbf{v}_{u_i} \in R^D$ respectively. The score of the match for a pair (u_i, a_j) is defined by the inner product of the two corresponding vectors, $S(u_i, a_j) = \langle \mathbf{v}_{u_i}, \mathbf{v}_{a_j} \rangle = \mathbf{v}_{u_i}^T \cdot \mathbf{v}_{a_j}$, where higher inner products represent better match, or a higher probability of a click. Hence, given a user, the OFF-Set algorithm suggests the ad variant that results in the highest matching score, in order to maximize the CTR of the campaign. As previously mentioned, the user vector is not directly trained by the algorithm, but rather the feature vectors that construct it, as will be detailed in section 6.

5.1 Applying Maximum Log Likelihood

We define the target function for maximization following the approach suggested by Aizenberg et al. [2]. Given the training data described above, that includes $|\mathbf{C}|$ click interactions, we define a multinomial distribution for a click over all our (user, ad variant) pairs as follows:

$$PC(u_i, a_j) = |\mathbf{C}| \cdot \frac{\exp^{S(u_i,a_j)}}{\sum_{(u_k,a_l)\in\mathbf{N}} \exp^{S(u_k,a_l)}}, \qquad (1)$$

where $PC(u_i, a_j)$ is the probability of a click for the (u_i, a_j) pair, and $S(u_i, a_j)$ is the score our model assigns this pair. Note that this is a general definition of probability that satisfies $\sum_{(u_i,a_j)\in\mathbf{N}} PC(u_i, a_j) = |\mathbf{C}|$. Also, it holds that for all i and j, $0 < PC(u_i, a_j) \leq 1$ under the constraint that $|S(u_i, a_j)| \leq \ell$, for $\ell = 0.5 \ln \frac{|\mathbf{N}|}{|\mathbf{C}|}$[1].

Our model's parameters (denoted hereafter by Θ) are now trained in order to maximize the mutual probability of the pairs that actually resulted in a click (assuming independence between all pairs). Using the log likelihood approach,

[1]As the score value is not important by itself, but rather the ranking it derives, scores can always be normalized in order to satisfy this constraint. Therefore, from now on, we will not refer to this constraint.

we get[2]

$$\Theta = \arg\max \log \prod_{(u_i,a_j)\in C} PC(u_i,a_j) = \quad (2)$$

$$\arg\max \sum_{(u_i,a_j)\in C} S(u_i,a_j) - |\mathbf{C}| \cdot \log \sum_{(u_i,a_j)\in \mathbf{N}} \exp^{S(u_i,a_j)}.$$

5.2 Training The Model

We extract Θ using a stochastic gradient ascent method. The derivative of the above target function depends on whether the (user, ad variant) pair resulted in a click or not. Whenever $(u_i, a_j) \in \mathbf{C}$, we get:

$$\Delta\Theta = (1 - PC(u_i,a_j)) \cdot \frac{\partial S(u_i,a_j)}{\partial\Theta}. \quad (3)$$

And when $(u_i, a_j) \in \mathbf{NC}$ we get

$$\Delta\Theta = -PC(u_i,a_j) \cdot \frac{\partial S(u_i,a_j)}{\partial\Theta}. \quad (4)$$

It is clear that a positive reward (a click) will always result in an increase in the value of $S(u_i, a_j)$, and vice versa. However, the increase and decrease step sizes are not equal.

If we begin the training process with random variables, assuming equal probability of a click to all pairs (which equals $\frac{|\mathbf{C}|}{|\mathbf{N}|}$), we get that the ratio (denoted by μ) between the step sizes of a negative and a positive rewards is $\mu = -\frac{|\mathbf{C}|}{|\mathbf{NC}|}$. This implies that for any positive reward we should apply a gradient ascent step that will increase $S(u_i, a_j)$ in some learning step α, while for any negative reward, we apply a gradient ascent step of size $\alpha \cdot \mu$, that will decrease $S(u_i, a_j)$.

Throughout the training process, assuming the model succeeds in representing the user preferences by representing the real probabilities, the step sizes should be decreased in both directions, and dependent on the specific pair. For simplicity reasons, we keep μ as the ratio between the update steps, and update its value throughout the training process, as detailed in Section 5.3.

5.3 Online Training

Up until now we have referred to the training data as a fixed and known set of examples. This, however, does not hold in the real world settings we operate in. The amount of data pairs and the rate in which they are produced are high. On the one hand, the data sparsity issue requires gathering many samples for training. On the other hand, as a single campaign may run for only a few hours, the training must be done with minimal latency. Considering these constraints together, an on-line training scheme is required. In such a solution, each sample updates the model's parameters only once (either positively, or negatively, according to the reward). Furthermore, the update depends only on this single sample, with no reliance on previous data samples. The only accumulative value we aggregate from the collection of data samples is the updated μ factor, the ratio between the step sizes of updates stemming from positive and negative rewards.

6. USER AND ITEM LATENT VECTORS

We now turn to describe the separate latent vector representations that map the ad variants and users into the same latent space. For ad variants which are repeating and stable along the campaign, we simply assign one D-dimensional latent factor per ad variant (item) a_j, $\mathbf{v}_{a_j} \in R^D$.

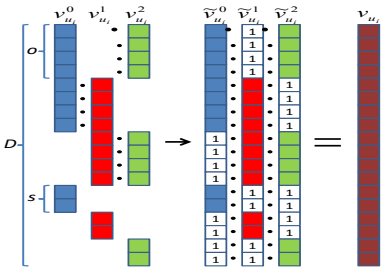

Figure 2: Example of a user latent factor construction from its feature value vectors ($K = 3$). In this example, $D = 18$, $d = 10$, $s = 2$ and $o = 4$. The middle illustration presents the auxiliary vectors $\widetilde{\mathbf{v}}^k_{u_i}$

We offer a novel construction of the latent representation on the user side. Recall that we assume that users are encountered once, and so are perpetually cold. Hence, they must be modeled by their features. Previous works in matrix factorization assigned a latent vector to each feature value, and modeled users (or items) as a linear combination of their features' vectors [2].While this approach can handle sparse user data, it cannot capture any dependencies between the features as the combination between the features is linear. Instead, we construct a more complex latent space over user features, that allows the representation of strong dependencies between each pair of features. Recall that items were mapped into R^D, and assume that each user has some values across K features. We select an *overlap dimension* o and a *standalone dimension* s such that $D = Ks + \binom{K}{2}o$. Each feature value is assigned a d-dimensional vector where $d = s + (K-1)o$, so that $\mathbf{v}^k_{u_i} \in R^d$ is the vector assigned to the kth feature value of user u_i. That vector has s entries that are attributed only to the specific feature, and $K-1$ blocks of o entries that are in overlap with each of the $K-1$ other features[3].

We now compose $\mathbf{v}_{u_i} \in R^D$ from $\binom{K}{2}$ chunks of size o, each holding the products of the values from a pair of feature vectors. The rest of $\mathbf{v}_{u_i}$ holds the Ks standalone values from each of the feature vectors. The process is illustrated in Figure 2. Now, let $\widetilde{\mathbf{v}}^j_{u_i} \in R^D$ be an extension of $\mathbf{v}^j_{u_i}$, so that the values of $\mathbf{v}^j_{u_i}$ appear in d of the entries of $\widetilde{\mathbf{v}}^j_{u_i}$, according to their indices and the rest of the entries are set to 1 (see middle part of Figure 2). Using this formulation, $\mathbf{v}_{u_i} = \prod_{1\le j\le K} \widetilde{\mathbf{v}}^j_{u_i}$ (for an element-wise product).

7. EVALUATION

We evaluate OFF-Set against two state-of-the-art methods mentioned in Section 2 as well as a simple 'popularity' algorithm. We use three baselines throughout our experiments: the Factorization Machines (FM) [9, 10] (which is given pairs of features, that allows it to represent dependencies between them), Gradient-Boosted Decision Trees

[2]A more comprehensive technical description was omitted, and can be found at[1]

[3]For simplicity, we assume each feature merits s unique entries and every pair of features merits an overlap of o entries. Non-uniform assignments of entries are possible and have no effect on the rest of the algorithm.

Campaign	clicks in warm-up phase	clicks in test phase	number of ad variants
A	100	2400	14
B	100	900	22

Table 1: Real-life campaign statistics. Significance was computed from Hoeffding's formula with confidence of 95%.

(GBDT) [5] and a global-popularity based algorithm (Popularity), which also follows the trend of the data, by applying a decay factor to the CTR values of all variants[4]

The evaluation metric we used was Mean Reciprocal Rank (mrr), applied only on positive observation ($\mathbf{C}$). Formally, for algorithm A,

$$mrr(A) = \frac{1}{|\mathbf{C}|} \sum_{(u_i,a_j)\in\mathbf{C}} \frac{1}{r_A^{u_i}(a_j)}, \quad (5)$$

where the function $r_A^{u_i}(a_j)$ is the rank of the ad variant a_j in algorithm A's computed ranked list for user u_i (1 for the first location). Under the assumption that the presented ad variants appear uniformly at random in the test data (and independently of the user), this metric quantifies well the ability of an algorithm to predict a click.

For evaluation on real data we chose two different dynamic campaigns, henceforth referred to as Campaign A and Campaign B. In order to simulate an on-line setting, the test (on both campaigns) was performed as follows. First, some initial number of observations is used for warm-up training. Subsequently, all remaining observations are iterated over, sorted by time of appearance. Observations that did not result in a click are used for further training. Any observation that resulted in a click is first used for evaluation, by requesting each algorithm to rank the available items for the given user. Afterward, the observation (along with its positive reward) is added as an additional training example. Table 1 provides some statistics on both campaigns, including the number of clicks contained in the warm-up and online test phases described above. Due to business sensitivity, we cannot disclose the exact number of non-clicks in the data. Rather, we roughly disclose that in both campaigns, the number of non-clicks was 2-3 orders of magnitude larger than the number of clicks.

For OFF-Set, being a single pass algorithm, the procedure above is straightforward. FM and GBDT, however, must be retrained from scratch once encountering a click in the online portion of the test[5].

The results are presented in Table 2. We can see a clear and significant advantage to OFF-Set as compared to the baselines (see Table 1 for the significant gap values for each campaign). In addition, its efficiency, being a single pass method, along with its low memory footprint, establish its superiority over the other methods for this problem.

8. CONCLUSIONS AND FUTURE WORK

In this work we introduced OFF-Set - an online single-pass algorithm that handles a perpetual user cold start problem.

[4]A more detailed evaluation section is given in[1]

[5]Since MRR is only measured on clicks, there is no need to retrain these models, for the purposes of the experiment, on each non-click.

Algorithm	MRR results campaign A	MRR results campaign B
Random	0.2114	0.2255
Popularity	0.2940	0.4360
FM w/paired features	0.3038	0.3824
GBDT	0.2613	0.3193
OFF-Set	0.5993	0.6234

Table 2: MRR results on real (offline) data. MRR differences higher than 0.055 and 0.091 for campaigns A and B respectively are significant (with 95% confidence).

OFF-Set is based on Matrix Factorization, and is motivated by a log-likelihood approach. It exploits repeating user features and captures pairwise feature dependencies. We evaluated OFF-Set on real-life data, and demonstrated its superiority over state-of-the-art methods.

9. REFERENCES

[1] M. Aharon, N. Aizenberg, E. Bortnikov, R. Lempel, R. Adadi, T. Benyamini, L. Levin, R. Roth, and O. Serfaty. Off-set - extended technical report. *arXiv.org:submit/0776700*, 2013.

[2] N. Aizenberg, Y. Koren, and O. Somekh. Build your own music recommender by modeling internet radio streams. *WWW '12 Proceedings of the 21st international conference on World Wide Web*, 2012.

[3] K. Barman and O. Dabeer. Analysis of a collaborative filter based on popularity amongst neighbors. *IEEE Transactions on Information Theory*, 58(12), 2012.

[4] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[5] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 29 (5), 2001.

[6] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. *IEEE 10th International Conference on Data Mining (ICDM)*, pages 176 – 185, 2010.

[7] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Published by the IEEE Computer Society*, 42(8), 2009.

[8] S. Park and W. Chu. Pairwise preference regression for cold-start recommendation. *Proceeding RecSys '09 Proceedings of the third ACM conference on Recommender systems*, pages 21–28, 2009.

[9] S. Rendle. Factorization machines. *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM), Washington, DC, USA*, 2010.

[10] S. Rendle. Social network and clickthrough prediction with factorization machines. In *KDD Cup*, 2012.

[11] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology*, Volume 3 Issue 3, May 2012.

[12] K. Zhou, S. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 315–324, 2011.