Modeling Impression Discounting in Large-scale Recommender Systems

Pei Lee, Laks V.S. Lakshmanan
University of British Columbia
{peil, laks}@cs.ubc.ca

Mitul Tiwari, Sam Shah
LinkedIn Corporation
{mtiwari, samshah}@linkedin.com

ABSTRACT

Recommender systems have become very important for many online activities, such as watching movies, shopping for products, and connecting with friends on social networks. User behavioral analysis and user feedback (both explicit and implicit) modeling are crucial for the improvement of any online recommender system. Widely adopted recommender systems at LinkedIn such as "People You May Know" and "Endorsements" are evolving by analyzing user behaviors on impressed recommendation items.

In this paper, we address modeling impression discounting of recommended items, that is, how to model user's no-action feedback on impressed recommended items. The main contributions of this paper include (1) large-scale analysis of impression data from LinkedIn and KDD Cup; (2) novel anti-noise regression techniques, and its application to learn four different impression discounting functions including linear decay, inverse decay, exponential decay, and quadratic decay; (3) applying these impression discounting functions to LinkedIn's "People You May Know" and "Endorsements" recommender systems.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Filtering; J.4 [**Computer Applications**]: Social and Behavior Sciences

Keywords

Impression discounting; Recommender system

1. INTRODUCTION

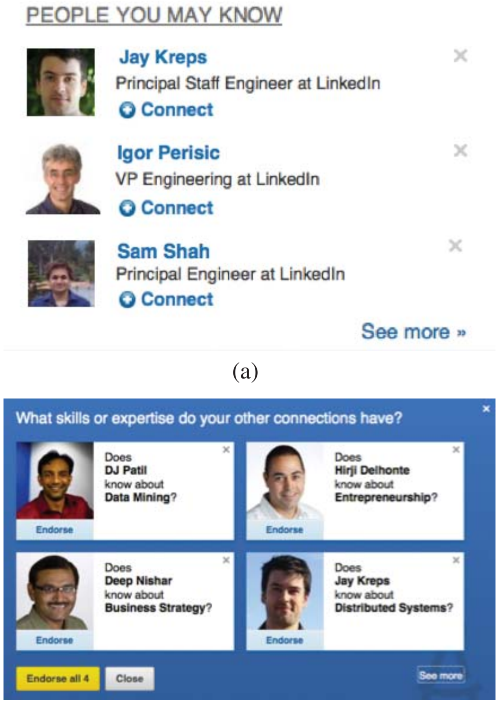
Analyzing and incorporating user feedback on recommended items is crucial for evolving any recommender systems, which have become important for many online activities and especially for social networks. At LinkedIn, the largest professional network with more than 300 million members, recommender systems play significant role in engaging users, discovering new content, connections, opportunities, and satisfying users needs. LinkedIn exposes connection or link recommender systems through features such as "People You May Know" (PYMK) (Figure 1(a)), which recommends other members to connect with, and skills endorsements, which recommends to endorse connections with their skills & expertise to add to their profiles (Figure 1(b)).

Modeling user behavior and incorporating feedback on live recommender systems such as PYMK and Endorsements, which are exposed to millions of users every day, are critical for enhancing these recommender systems. User feedback on recommended items viewed by users or impressions, can be explicit or implicit. User action such as inviting or dismissing a recommended member to connect with is explicit feedback in PYMK. On the other hand, viewing a recommended member or clicking to view the profile of a recommended member, but not inviting to connect, are examples of implicit feedback in PYMK. There has been extensive research on modeling explicit feedback in terms of ratings and acceptance of recommended items [9, 21, 26]. However, modeling certain implicit feedback such as no action on impressed recommended items for conversion, has been under explored. There has been some work on estimating Click-through rate (CTR) using past impressions [1], but CTR estimation is different from conversion prediction as also argued recently in [5, 17].

Usually a recommender system generates a list of items, ordered by a score, to show to users. We say a recommended item is impressed when a user views the item. Acceptance of an impressed recommended item results in conversion. No action from users on impressed recommended items can be because of multitude of reasons, for example, a user might be busy with other things, or the recommended item is not relevant or the purpose of the site visit is very different. In the impression discounting problem we aim to maximize conversion of recommended items generated by a recommender system by applying a discounting factor, derived from past impressions, on top of scores generated by the recommender system. There are two basic challenges in the impression discounting problem: (1) how can we build an effective model between conversion and various user behaviors from the large amount of past impression data? (2) how can the model be applied to improve the performance of existing recommender systems? There are various factors that can be used in modeling the impact of impressions, such as, (1) the number of times an item is impressed or recommended to a user, (2) when the item was impressed, and (3) frequency of user visits on the site or user seeing any of the recommended items.

In this work, we present models for discounting impressions of recommended items based on various factors such as the number of impressions and the last time the item was seen. The intuition behind these models is simple; for example, in the case of PYMK, a recommended member that results in a number of impressions, but does not lead to an invitation, should be removed or lowered in the recommended list of members. We performed detailed analy-

KDD '14, August 24 - 27 2014, New York, NY, USA
Copyright 2014 ACM 978-1-4503-2956-9/14/08$15.00.
http://dx.doi.org/10.1145/2623330.2623356


PEOPLE YOU MAY KNOW

Jay Kreps
Principal Staff Engineer at LinkedIn
Connect

Igor Perisic
VP Engineering at LinkedIn
Connect

Sam Shah
Principal Engineer at LinkedIn
Connect

See more »

(a)

What skills or expertise do your other connections have?

Does DJ Patil know about Data Mining?
Endorse

Does Hirji Delhonte know about Entrepreneurship?
Endorse

Does Deep Nishar know about Business Strategy?
Endorse

Does Jay Kreps know about Distributed Systems?
Endorse

Endorse all 4
Close
See more

(b)

Figure 1: (a) People You May Know at LinkedIn recommends other members to connect with. (b) Suggestions for skills endorsements, a recommender system at LinkedIn to endorse connections for their skills and expertise, which can be added to their profile.

sis of past impressions over large amount of tracking data to find correlation between various factors and conversion rate, for example, invitation rate for PYMK. We learned four different regression models that represent linear decay, inverse decay, exponential decay, and quadratic decay. We developed novel anti-noise regression techniques to detect outliers and lower the effect of outliers in learning regression functions. We show empirically that our impression discounting models perform significantly better both in offline analysis using past data as well as in online A/B testing. In offline analysis for PYMK, these impression discounting models show up to 31% improvement in invitation rate, and in online A/B test experiments, the model improves invitation rate by up to 13%.

The main contributions of this paper are as follows:

- Perform large scale correlation studies between impression signals and conversion rate of impressed items;
- Design effective impression discounting models based on linear/multiplicative aggregation, and propose novel anti-noise regression model to deal with the data sparsity problem;
- Evaluate these regression models on real-world recommendation systems such as "People You May Know" and "Endorsements" to demonstrate their effectiveness both in offline analysis and in online systems by A/B testing.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 formalizes concept of impressions in the context of recommender systems and details data sets used in this work. Section 4 presents exploratory data analysis to figure out correlations between various user behavior and conversion rate. This section also describes our impression discounting framework, and various discounting functions learned through regression. Section 4.5.3 describes our anti-noise regression techniques. Section 5 presents experimental results including offline analysis and online A/B testing results. Finally we conclude in Section 6.

2. RELATED WORK

The related work of this paper falls mainly under one of the following categories.

Implicit Feedback in CF. User implicit feedback plays an essential role in the recommendation models [2]. Collaborative filtering (CF) is a typical recommendation model, with the advantage of not needing the user/item profiles [9, 10, 21, 26]. Hu et al. [8] studied implicit feedback on CF and proposed a factor model by treating implicit feedback as the indication of positive and negative preference associated with varying confidence levels. Koren [9] discovered that incorporating implicit feedback into a neighborhood latent factor model (SVD++) could offer significant improvement. On the application side, a local implicit feedback model mining users' preferences from the music listening history is proposed in [24]. Moling et al. [15] exploited implicit feedback derived from a user's listening behavior, and modeled radio channel recommendation as a sequential decision making problem. Yang et al. [25] proposed Collaborative Competitive Filtering (CCF) to learn user preferences by modeling the choice process with a local competition effect. Gupta et al. [7] modeled user response to an ad campaign as a function of both interest match and past exposure, where the interest match is estimated using historical search/browse activities of the user. All these methods [8, 9, 25, 7] exploit implicit feedback by integrating historical signals with existing recommendation models. In contrast, our impression discounting model serves as an external plugin of the existing recommendation model, which has advantages in modularization.

CTR Estimation. The estimate of click-through rate (CTR) for online resources such as news articles, Ads or search results is a hot topic, and we categorize the related work into two types: classification-based estimate and regression-based estimate. For the first type, Agichtein et al. [2] showed that incorporating user behavior data can significantly improve the ordering of top results in real web search engines, by treating user behaviors as features and using classification to re-rank. Richardson et al. [18] employed logistic regression to predict the CTR for newly created Ads, using features of ads, terms, and advertisers. For the second type, Agarwal et al. [1] proposed a spatio-temporal model to estimate the first impression CTR of Yahoo! Front Page news articles.

Impression discounting shares some commonalities with the estimate of CTR: they both try to predict a user's future action. However, they differ in many aspects. First, impression discounting aims to improve the conversion rate of recommendations, which is a different goal from CTR estimate, as a click can happen multiple times and may not result in a conversion action. Second, the existing work on CTR estimation is mainly focused on new items; for example, the first exposure of news articles [1] or Ads [18]. The link between impression history and conversion rate is not covered in existing studies.

Density-based Approach. Density concepts, first introduced by density-based clustering [6] and recently adapted to social network and media analysis [13, 12], is a traditional way to detect noises. In the context of linear regression, Ristanoski et al. [19] proposed segmenting the input time series into groups and simultaneously optimizing both the average loss of each group and the variance of the loss between groups, to produce a linear model that has low overall error. Chen et al. [4] introduced a nonlinear logistic regression model for classification, and the main idea is to map the data to a feature space based on kernel density estimation. Kernel regression is another relevant approach, which estimates the conditional expectation of a random variable by non-parametric techniques [11]. However, all these approaches [19, 4, 11] cannot combat the noise in linear regression.

3. IMPRESSION DATA IN LARGE-SCALE RECOMMENDER SYSTEMS

In large-scale recommender systems, user actions, such as impressions, are readily captured [23]. Before jumping to the modeling of impression discounting, we start to extract and formalize the impression data sets in a uniform format, crossing various application scenarios, including PYMK, Skill Endorsements, and Keyword Search Ads. These impression data sets are typically reflecting the interaction between users and popular online recommender systems.

3.1 Formalizing Impressions

Our study on the daily interaction logs of different recommender systems reveals that there is a common structure behind impression records from different sources. Without the loss of generality, we formalize each impression record as a tuple $\mathcal{T}$ with six attributes, as explained below.

Definition 1. (Impression) An impression in the recommender system is modeled as a tuple $\mathcal{T}$ with six attributes:

$$\mathcal{T} = (user, item, conversion, [behavior1, behavior2, \cdots], t, R)$$

where:

– $user$ is a user ID;

– $item$ is a recommendation item ID;

– $conversion$ is a boolean type to describe whether or not $user$ takes an action on $item$ in this impression;

– $behavior$ is an observed feature of interaction;

– t is the time stamp of impression;

– R is the recommendation score of $item$ to $user$, which is provided by the recommendation engine.

As a convention, we use $\mathcal{T}.x$ to refer to attribute x in $\mathcal{T}$.

Conversion. We use the term $conversion$ to indicate that $user$ explicitly accepts the recommendation. A $conversion$ action may differ from a user click. For example, a conversion action in PYMK means that a user does not simply click on the profile page of another user, but instead takes action to invite that user. In job recommendations, $conversion$ means that a $user$ actually applies for the job $item$.

Behaviors. User interactions with the recommender system typically follow a "see-think-do" procedure. The interaction between users and recommender systems has many facets. We aggregate the following user behaviors:

- *LastSeen*: the day difference between the last impression and the current impression, associated with the same $(user, item)$;
- *ImpCount*: the number of historical impressions before the current impression, associated with the same $(user, item)$;
- *Position*: the offset of $item$ in the recommendation list of $user$;
- *UserFreq*: the interaction frequency of $user$ in a specific recommender system.

We use X to represent the set of behaviors and use m to describe the set size, i.e., $m = |X|$.

Impression Sequence. Given a large impression data set with each record formatted as tuple $\mathcal{T}$, we can perform a "group-by" operation on $(user,\ item)$ to obtain a sequence with the same $(user,\ item)$. This sequence can be ordered by time and denoted as $seq(user, item) = (\mathcal{T}_1, \cdots, \mathcal{T}_n)$. In a given observation time window, the sequence length is indicated by *ImpCount*. Especially, if $seq(\mathcal{T}.user, \mathcal{T}.item)$ has the property $ImpCount = 1$, we call it $\mathcal{T}$ the first impression.

Conversion Rate. In most recommender systems, once the conversion is true, $item$ will not be recommended to $user$ again. So $conversion = True$ only possibly occurs once on the tail of a sequence. Thus, the overall conversion rate can be defined as the ratio between the number of impressions with $conversion = true$ and the number of all impression sequences. Let $\mathcal{S}$ denote the set of impressions. The global conversion rate is computed by

$$ConveRate = \frac{|\{\mathcal{T} \in \mathcal{S} | conversion = true)\}|}{|\mathcal{S}|} \tag{1}$$

Conversion rate can also be measured on the basis of each behavior set X. We can "group-by" all impression sequences on X, and then define the local conversion rate by the formula

$$ConveRate(X) = \frac{|\{\mathcal{T} \in \mathcal{S} | conversion = true \,\&\, behavior = X)\}|}{|\{\mathcal{T} \in \mathcal{S} | behavior = X)\}|} \tag{2}$$

We normalize conversion rates in this work as only relative changes are important.

3.2 Data Set Descriptions

To deploy the impression discounting model, we collect two real-world data sets from online recommender systems in LinkedIn and one external public data set from Tencent. In the following section, we describe the salient characteristics of these data sets.

LinkedIn PYMK Impressions. PYMK is one of the most popular recommender systems in LinkedIn. The PYMK $item$ is actually another 2nd or higher degree LinkedIn user. We select an observation time with a total of 1.08 billion impressions in tracking data. Out of them, a significant portion of impressions are multiple, which means at least half the impressions were previously presented to the users, but get no invitation action. The relatively long impression sequences in PYMK make it an ideal data set for modeling impression discounting.

LinkedIn Skill Endorsement Impressions. Skill Endorsement is another popular recommender system to recognize the skills and expertise of 1st-degree connections. We collect impressions tracking data with 0.19 billion impression tuples, and a small portion of them are multiple. The $item$ in a tuple is the combination of a user and his skill. Because endorsements are only allowed for 1st-degree connections, impression sequences are usually very short and the conversion rate is relatively high, which implies that the skill recommendations are more likely endorsed by connections in the first impression.

Tencent SearchAds Impressions[1]**.** SearchAds is a data set made publicly available for KDD Cup 2012 by the Tencent search engine. It contains various ingredients of the interaction between a user and the search engine; for example, user, query, Ad ID, depth, position, etc. In total, this data set has 0.15 billion impression sequences, and the CTR of the Ad at the 1st, 2nd and 3rd position of the search session is 4.8%, 2.7%, and 1.4% respectively.

[1] http://www.kddcup2012.org/c/kddcup2012-track2

4. IMPRESSION DISCOUNTING MODEL

There are two basic challenges in the impression discounting problem: (1) how can we learn an effective model between the $conversion$ and various user behaviors from the big data? (2) how is the model applied to new impressions and improving the performance of existing recommender systems? In this section, we introduce the impression discounting model, which is learned from the actual impression data sets and integrated as a plugin in the existing large-scale recommender systems.

4.1 Conventions

For convenience of presentation, we introduce the following concepts and notations:

- **Behavior List**: a list of behaviors that describe the user's interaction with the recommender system, denoted as X. The list size is denoted by m and each element X_i can be observed from the impression records of recommender systems, e.g., X=(*LastSeen*, *ImpCount*).
- **Conversion Rate**: the actual conversion rate in data sets is defined by Eq. (2) and denoted by $y = ConveRate(X)$ for the ease of discussion. The predicted conversion rate by the model is denoted by $\hat{y}$.
- **Observation**: an observation (X, y) is composed by a behavior set X and its conversion rate y.
- **Support of Observation**: the support of an observation (X, y) is the number of impression sequences with the same X that produces the same y in Eq. (2), denoted by $support(X, y)$.
- **Observation Space**: the set of all possible observations (X, y) in the data set, denoted by D.

4.2 Impression Discounting Framework

As an example in LinkedIn PYMK, assume a common scenario that a user called Alice gets the recommendation for a connection called Bob in September 20 but takes no invitation action. This assumption has two folds of meaning: (1) the recommendation engine thinks that Bob has a relatively high recommendation score R to Alice; (2) it is very likely that Alice is not satisfied with the recommendation item Bob. If the recommender system ignores this implicit negative feedback from Alice, most likely Bob will be recommended to Alice again in another day, which only reduces the overall satisfaction of Alice. We believe that this kind of scenarios can be easily found in nearly every recommender system, e.g., Ad recommendation, job recommendation and even web search engines. Motivated by this example, we design the impression discounting framework, which serves as a module in the existing ecosystem of recommender systems, by taking full consideration of the implicit negative feedback from users. We formalize the impression discounting problem as follows.

Problem 1. (**Impression Discounting**). Given a large-scale impression data set with each record defined by Definition 1, for each unique $(user, item)$, supposing the historical impression sequence is $seq(user, item) = (\mathcal{T}_1, \cdots, \mathcal{T}_n)$ $(n \geq 1)$ and $\mathcal{T}$ is the current impression (i.e., $\mathcal{T}.t > \mathcal{T}_n.t$), the problem of impression discounting on $\mathcal{T}$ is to determine a discounting factor d $(0 < d \leq 1)$, which updates

- $\mathcal{T}^*.R = \mathcal{T}.R \cdot d$;
- $seq(user, item) = (\mathcal{T}_1, \cdots, \mathcal{T}_n, \mathcal{T})$;
- $seq^*(user, item) = (\mathcal{T}_1, \cdots, \mathcal{T}_n, \mathcal{T}^*)$;

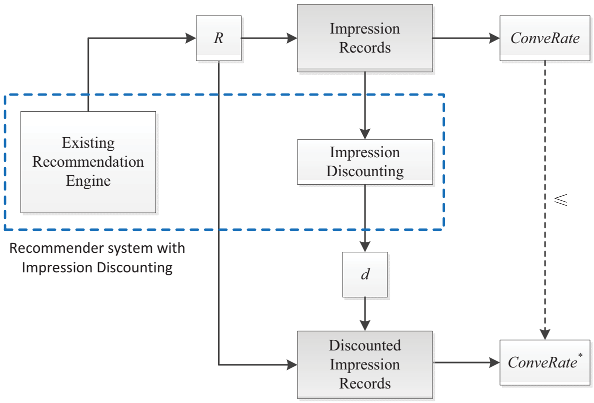

R
Impression Records
ConveRate
Existing Recommendation Engine
Impression Discounting
≤
Recommender system with Impression Discounting
d
Discounted Impression Records
ConveRate*

Figure 2: The impression discounting framework. Our proposed impression discounting model only relies on the historical impression records, and can be treated as a plugin for existing recommendation engines. We use the dotted rectangle to circle the newly built recommender system with impression discounting, which produces discounted impressions with a higher overall conversion rate.

and maximizes

$$Improvement = \frac{ConveRate^* - ConveRate}{ConveRate} \qquad (3)$$

where $ConveRate^*$ and $ConveRate$ are computed by Eq. (1) based on $seq^*(user, item)$ and $seq(user, item)$ respectively.

To explain why impression discounting helps improve the overall conversion rate, we use the following example. Suppose that in a user interaction session, the recommendation engine fetches top 100 recommendation items ranked by $\mathcal{T}.R$ and 50 of them will be observed by the user. If without impression discounting, existing recommendation engine will present the top 50 items ranked by $\mathcal{T}.R$ to the user. However, since a large portion of item impressions are multiple, their historical negative feedback indicates that their conversion rate will be low. Now we consider a newly built recommender system with impression discounting. Since impressions are discounted properly by d where $d \leq 1$, all 100 items will be re-ranked. Ideally, items with less negative feedback will bubble up into top 50 list and be exposed to user's observation, which results in a higher overall conversion rate.

We show the impression discounting framework in Figure 2. Basically, we assume that R is already produced by the best-known recommendation method, and the computation of R is not the focus of this paper. Our proposed impression discounting model only relies on the historical impression records in the past observation time window. In other words, impression discounting can work independently with the recommendation engine. Once added as a plugin into the recommender system, impression discounting model will produce a discounting factor d, which penalizes items that not likely gain a conversion action and makes the recommendation list re-ranked. A properly designed impression discounting model will make the overall conversion rate increasing and improve user's satisfaction. The challenge of the impression discounting problem is how to design an effective model that determines the optimal d to re-rank the recommendation list with intuitive explanation.

Impression Discounting: Plugin vs. In-Model. There are typically two ways to incorporate impression discounting into an existing recommendation system: serve as an independent plugin or push in the recommendation model. The plugin approach does not change

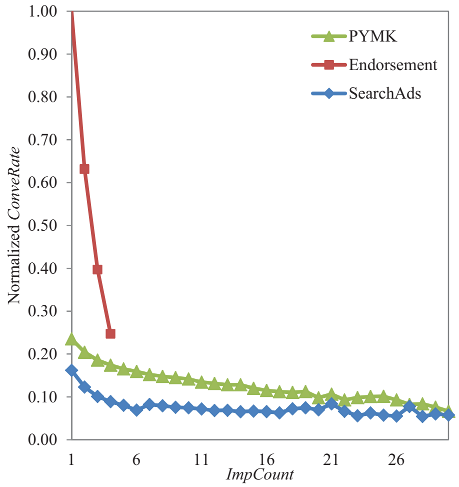


Figure 3: The change of Normalized $ConveRate$ with increasing *ImpCount* on three real-world impression data sets.

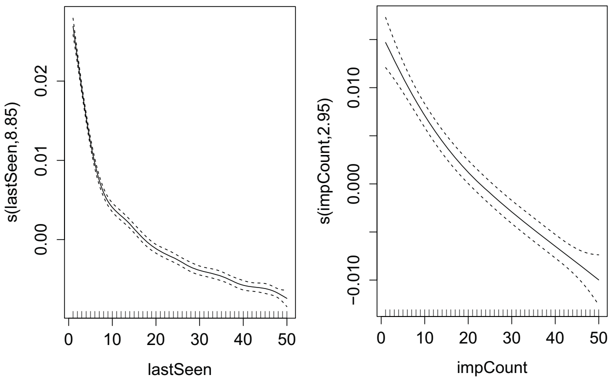
the existing recommendation model, and the impression discounting is performed by multiplying a discounting coefficient d with the recommendation score. In contrast, the in-model approach modifies the existing recommendation model by adding more signals. For example, supposing the recommendation model is based on classification, the in-model approach will add signals like *impCount*, *lastSeen* as additional features. The plugin-based approach provides model independence: we can be agnostic as to the underlying recommendation model, whether it is a sophisticated similarity ranker or simple cooccurance-based collaborative filtering, it is treated as a black box. Thus, impression discounting can be provided "as a service" for any client application. In this paper, we focus on the plugin-based approach, though these techniques can be adapted to the in-model approach without loss of generality.

4.3 Hypothesis Testing

We have a basic hypothesis for the impression discounting:

***Hypothesis** 1.* Impression $\mathcal{T}$ with $conversion = false$ is a negative feedback to the recommendation of $item$ to $user$.

If this hypothesis is false, the recommender system will always try to recommend items with a longer impression history to gain a higher conversion rate, and these long impression sequence will dominate the recommendation list, preventing new items being displayed. If this is the case, the discounting factor $d \leq 1$ in the problem definition will become invalid, as the conversion rate may increase if the impression sequence becomes longer. Thus, if this hypothesis is false, the impression discounting framework will be problematic. To verify this hypothesis on real-world impression data sets, we measure the change of $ConveRate$ with increasing impression sequence length, as shown in Figure 3. As we can see, on all three data sets, $ConveRate$ decreases very fast as increasing *ImpCount* in the beginning. Later, the decreasing trend becomes steady, and when *ImpCount* is high (which is rare in observation), there are some turbulences on the tail of the curves due to the data sparsity problem. The impression sequences of the Endorsement data set are very short in length and $ConveRate$ decreases even faster than the other two data sets. In a nutshell, Figure 3 verifies that Hypothesis 1 is true, and also reveals that the discounting factor in the Endorsement data set should be more severe and smaller than the other two data sets.



(a) *LastSeen* Correlation (b) *ImpCount* Correlation

Figure 4: Correlation confidence analysis between *ConveRate* and two behaviors on PYMK data.

4.4 Exploratory Analysis

Correlation Analysis. It is essential to explore the correlation confidence between conversion rate and user behaviors. An effective way to tackle the correlation analysis is using non-parametric smoothers in a generalized additive model (GAM) [14] to fit the conversion rate. We perform GAM fitting on the PYMK data set and show the correlation analysis between conversion rate and *ImpCount*, *LastSeen* in Figure 4. The narrow intervals in Figure 4(a) and (b) suggest that the curvatures of the correlation are both strong. Moreover, as the interval in Figure 4(a) is even narrower than the interval in Figure 4(b), we conclude that $ConveRate$ has a stronger correlation with *LastSeen* than the correlation with *ImpCount*. In other words, *LastSeen* plays a more important role than *ImpCount* in determining conversion rate. The monotone decreasing trends suggest that the correlations are both negative. The correlation analysis for other user behaviors and in Endorsement and SearchAds data sets delivers similar messages. We omit the details due to space constraints.

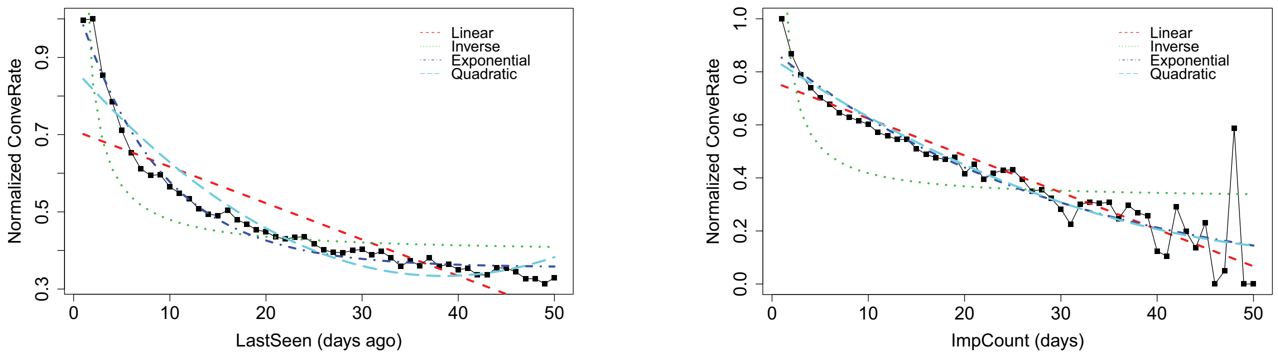
Discounting Functions. Based on the correlation analysis, we define discounting functions to model the negative relationship between the conversion rate and a specific user behavior. In particular, we introduce the following four kinds of discounting functions:

- Linear Discounting: $f_L(x) = \alpha_1 \cdot x + \alpha_2$;
- Inverse Discounting: $f_I(x) = \frac{\alpha_1}{x} + \alpha_2$;
- Exponential Discounting: $f_E(x) = e^{\alpha_1 \cdot x + \alpha_2}$;
- Quadratic Discounting: $f_Q(x) = \alpha_1(x - \alpha_2)^2 + \alpha_3$.

We use PYMK as the default data set, and show the conversion rate vs. *LastSeen* and *ImpCount* in Figure 5. Since the supports of observations for a high *LastSeen* or *ImpCount* is really low, the curve tails in Figure 5(a) and (b) become fluctuant, which brings new challenges for curve-fitting, a problem we will study in Section 4.6. So far, we perform regression and show the root mean square error (RMSE) of different discounting functions to the actual observations in Table 1. As we can see, exponential discounting achieves the overall minimal RMSE with the best fitting quality. More evaluation of discounting functions can be found in Section 5. In our impression discounting model, these discounting functions will serve as the atoms in the multiple variable regression process.

4.5 Aggregated Discounting Model

Conversion rate is determined by multiple facets of user behaviors. For example, it is confirmed that *ImpCount* and *LastSeen* will



(a) Conversion rate vs. *LastSeen*

(b) Conversion rate vs. *ImpCount*

Figure 5: The change of normalized conversion rate with increasing *LastSeen* and *ImpCount*. We simulate all observations by four discounting functions, and exponential discounting achieves the minimal RMSE.

RMSE	*ConveRate∼LastSeen*	*ConveRate∼ImpCount*
Linear	0.0041	0.0032
Inverse	0.0039	0.0053
Exponential	**0.0012**	**0.0028**
Quadratic	0.0024	0.0029

Table 1: RMSEs of various discounting functions.

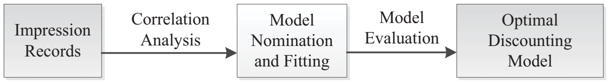


Figure 6: Impression discounting model workflow.

influence the invitation rate in PYMK. To model conversion rate accurately, we propose an aggregated discounting model in this paper, which integrates multiple discounting functions into a hybrid model. There are typically two ways to incorporate multiple variables into a model: linear or multiplicative combinations. We explore both of them in this section.

Workflow. The major workflow of the impression discounting model is presented in Figure 6. Without any prior knowledge, the correlation analysis is a great way to discover the correlation strength and polarity between the conversion rate and a specific user behavior. Correlation analysis also helps determine which discounting function fits the observations best by comparing RMSE, and narrows down the search space for the model candidates. Each candidate model will be learned by the regression from the training data set. Finally, we perform a model evaluation for each candidate on the testing data set, and the model with the best performance will be selected as the optimal discounting model for the corresponding recommender system.

Regression vs. Classification. Classification is a typical machine learning technique for decision making. Since our impression discounting framework follows the plugin approach, given that the recommendation score R is obtained from the best-available classification model, it will bring severe redundancy problem if we apply the classification model again on features in impression discounting. Besides, compared with the classification, regression can provide a better and more intuitive explanation of impression discounting by discounting functions.

4.5.1 *Linear Aggregation Model*

The first aggregated discounting model we explore is the linear model. Linear aggregation model combines discounting functions of different user behaviors by a linear function, in the form:

$$\hat{y} = \sum_{i=1}^{m} w_i f(X_i) \tag{4}$$

where $m = |X|$ and $f(X_i)$ is any one of discounting functions. Given a specific behavior X_i, the correlation analysis will help decide which discounting function is the best choice for $f(X_i)$. w_i is the weight for the discounting function and will be learned. It is well known that too many parameters in the linear model will result in over-fitting. Actually, once we select the specific discounting functions, the parameters in Eq. (9) can be reduced greatly. Here is an example with three different behaviors:

$$\begin{aligned}\hat{y} &= w_1 \cdot f_L(X_1) + w_2 \cdot f_I(X_2) + w_3 \cdot f_E(X_3) \\ &= w_1 \cdot (\alpha_1 X_1 + \alpha_2) + w_2 \cdot (\frac{\alpha_3}{X_2} + \alpha_4) + w_3 \cdot e^{\alpha_5 X_3 + \alpha_6} \\ &= w_1\alpha_1 \cdot X_1 + w_2\alpha_3 \cdot \frac{1}{X_2} + w_3 e^{\alpha_6} \cdot (e^{\alpha_5})^{X_3} + w_1\alpha_2 + w_2\alpha_4 \\ &= u_1 \cdot X_1 + u_2 \cdot \frac{1}{X_2} + u_3 \cdot \alpha^{X_3} + u_4\end{aligned} \tag{5}$$

where $\alpha = e^{\alpha_5}$. In general, combing like terms in Eq. (9) can reduce the parameters from the order of $3m$ to $(m + 1)$. These parameters can be learned by standard linear regression packages.

4.5.2 *Multiplicative Aggregation Model*

Multiplicative aggregation is another way to combine discounting functions, in the form:

$$\hat{y} = w \prod_{i=1}^{m} f(X_i) \tag{6}$$

Supposing each discounting function has two parameters on average, Eq. (6) has $(2m + 1)$ parameters to learn. We try to reduce the number of parameters, by performing a linearization, in the following way:

$$\ln \hat{y} = \ln w + \sum_{i=1}^{m} \ln f(X_i) \tag{7}$$

If $f(X_i)$ is the exponential discounting, Eq. (7) degrades to a simple linear regression with $(m+1)$ parameters without accuracy loss. Otherwise, we can discard the constants or lower order terms in discounting functions and obtain an approximated version of Eq. (7) with $(m + 1)$ parameters.

4.5.3 Impression Discounting Factor

Both linear aggregation and multiplicative aggregation can be linearized as a uniform form shown below

$$\tilde{y} = \sum_{i=0}^{m} u_i \tilde{X}_i \tag{8}$$

where $\tilde{y} = \hat{y}$ in linear aggregation and $\tilde{y} = \ln \hat{y}$ in multiplicative aggregation. We set $\tilde{X_0} = 1$ and when $i \geq 1$, we have $\tilde{X_i} \in \{X_i, \frac{1}{X_i}, \alpha^{X_i}, X_i^2\}$ for linear aggregation and $\tilde{X_i} \in \{\ln X_i, X_i\}$ for multiplicative aggregation. It is trivial to show $support(\tilde{X_i}) = support(X_i)$. The selection of $\tilde{X_i}$ depends on the kind of discounting functions for each behavior in the model. For example, in multiplicative aggregation, if $f(X_i)$ is the inverse discounting, we have $\tilde{X_i} = \ln X_i$. Eq. (8) can be written in matrix notation as

$$\tilde{y} = \tilde{X}^T u \tag{9}$$

Once we obtain the aggregated discounting model, the impression discounting factor d for a impression tuple $\mathcal{T}$ can be computed by the normalized value of $\tilde{y}$, which falls in $(0, 1]$. That is,

$$d = \frac{\tilde{y}}{\max \tilde{y}} \tag{10}$$

4.6 Anti-Noise Regression Model

Eq. (9) transforms the aggregated discounting model as a linear regression problem, whose objective function is to minimize the mean squared error, with a form

$$RMSE^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y})^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{X_i}^T u)^2 \tag{11}$$

In the practice of model-fitting, we observed a number of cases that the sparsity of observation supports may make the conversion rate go too high or too low. Examples are found on the curve tails in Figure 5(a) and 5(b). If y_i deviates from the overall trend of its neighboring observations because of sparsity, the objective function Eq. (11) will suffer a lot because it tries to minimize the difference between the local overall trend and sparse observations. Ideally, it would be perfect if outliers of observations can be removed by a well-designed mechanism and the observations with less supports can be penalized in the objective function.

Notice that the naive approach that truncates the curve tail by a threshold on the minimal support of an observation is problematic, because: (1) this will also remove the observations that don't deviate a lot from their local overall trends; (2) the threshold is difficult to decide, because on the scale of billions of records, the "sparsity" is a relative concept, where a "sparse" observation may still have millions of supports.

4.6.1 Outlier Observation Detection

In related work, density-based clustering [6] defines clusters as areas of higher density than the remainder of the data set, and the key idea is that for each node in the cluster, the number of neighbors within a radius ε should exceed a threshold δ. Density-based clustering has the benefit of distinguishing clusters and outliers in a data space efficiently. Recently, density-based clustering emerged into the social network and media analysis [13, 12] and demonstrated its effectiveness. In impression discounting, we model the density of an observation as the support sum in a small neighborhood of this observation. The migration of density concepts to the linear regression is a contribution of this paper. To start, we define ε-neighborhood of a behavior list X in multiple-variable linear regression as follows.

***Definition** 2.* (ε-neighborhood of a behavior). The ε-neighborhood of a behavior set X, denoted by $N_\varepsilon(X)$, is defined by $N_\varepsilon(X) = \{X' \in D | dist(X, X') \leq \varepsilon\}$.

The shape of ε-neighborhood is determined by the choice of the distance function $dist(X, X')$. Without the loss of generality, we consider the Euclidean distance is the choice for multi-dimensional feature space, that is

$$dist(X, X') = dist(X', X) = \left(\sum_{i=1}^{n}(X_i - X_i')^2\right)^{\frac{1}{2}} \tag{12}$$

Density-based clustering does not define the concept of "connection" between points. In our anti-noise regression model, as each behavior list X is associated with a response value y, we use connection to describe the relationship between two pairs of observations, as defined below.

***Definition** 3.* (Connection in ε-neighborhood). Given two different observations (X, y) and (X', y'), they are connected if $dist(X, X') \leq \varepsilon$ and $|y - y'| \leq \gamma$, where γ is the maximal tolerance for local deviation.

We compute the density of an observation (X, y) as the sum of supports connected with (X, y) in the ε-neighborhood:

$$density(X, y) = \sum_{X' \in N_\varepsilon(X), |y - y'| \leq \gamma} support(X', y') \tag{13}$$

Density score will categorize observations into three types:

- **Core Observations**: (X, y) is a core observation, if the support of connected observations within the ε-neighborhood exceeds the threshold δ. That is, $density(X, y) \geq \delta$.
- **Border Observations**: (X, y) is a border observation, if (X, y) is not a core observation but connects to at least one core observation.
- **Outlier Observations**: (X, y) is an outlier observation, if (X, y) is neither a core nor a border observation.

Intuitively, outlier observations capture those observations that are unreliable by deviating from the local trends of nearby observations. Basically, all the outlier observations can be discarded before the linear regression. We can tune density parameters $\{\varepsilon, \delta, \gamma\}$ to control the percentage of outlier observations in the observation space. As a general guideline, $\{\varepsilon, \gamma\}$ is usually fixed for a specific impression data set, and we set 5% as the percentage of outlier observations out of all observations, which can be tuned by δ.

We explain the intuition of noise detection from the graph perspective. If we view each observation as a node and a connection between observations as an edge, an observation network will be built. Core observations will be those nodes with a high connectivity to other nodes. Those nodes represent high authority ranking in the network analysis such as PageRank [16]. Outlier observations are those nodes with low degrees and no edges to any high authority nodes, and they usually represent the marginal, isolated or noisy part of the network. Removing these outlier nodes will make the whole network structure more cohesive and accurate to model.

4.6.2 Density Weighted Regression

In RMSE shown in Eq. (11), the error between the actual conversion rate y and predicted conversion rate $\tilde{y}$ are weighted equally. However, we argue that for observations with high supports, their errors should be emphasized in the objective function, compared with the observations with a relatively low supports. To achieve the goal, we propose the *density weighted regression*, which adds a weight v_i for each error $(y_i - \tilde{X_i}^T u)$, with a modified objective function given below

$$RMSE_v^2 = \frac{1}{n}\sum_{i=1}^{n}(v_i(y_i - \tilde{X_i}^T u))^2 \tag{14}$$

The problem is how to decide v_i. One naive solution is using the ratio between the support of an observation and the total supports to weight the corresponding error, that is $v_i^* = \frac{support(X_i)}{\sum_{X \in D} support(X)}$. The assumption of this solution is that the distribution of $support(X_i)$ is "smooth" in X_i's local neighborhood. That is to say, compared with the support of X_i's neighbors, $support(X_i)$ should not be remarkably too high or too low. However, in real-world data sets, $support(X_i)$ is likely not smooth due to the sparsity problem. Instead, we propose an advanced solution, by adopting the density parameter ε:

$$v_i = \frac{Average(\sum_{X_j \in N_\varepsilon(X_i), |y_i - y_j| \leq \gamma} support(X_j))}{\sum_{X \in D} support(X)} \tag{15}$$

Notice that $X_i \in N_\varepsilon(X_i)$. The rationale behind Eq. (15) is that, the supports of all similar observations in the neighborhood of X_i contributes to the weight of (X_i, y_i). In the extreme case, if $\varepsilon = 0$, we have $N_\varepsilon(X_i) = \{X_i\}$ and v_i degrades to the naive solution. If $\varepsilon = \infty$ and $\gamma = 1$, $N_\varepsilon(X_i) = D$ and $v_i = 1$, making Eq. (14) degrades to the unweighted version (Eq. (11)). Typically, we set ε smaller than 5, making v_i as the ratio between the supports of ε-neighborhood and the total supports. v_i is smoother than v_i^* and highlights the observations with high supports effectively.

Next, we can write Eq. (14) in matrix notation as $n \cdot RMSE_v^2 = (Vy - VXu)^T(Vy - VXu)$, where $V = diag(v)$. If we take the derivative of $n \cdot RMSE_v^2$ with respect to u, we get $(VX)^T(Vy - VXu)$. By setting the derivative to zero, we solve u by the following reasoning:

$$\begin{aligned} \hat{u} &= ((VX)^T(VX))^{-1}(VX)^T(Vy) && (16)\\ &= (X^T(V^TV)X)^{-1}X^T(V^TV)y && (17)\\ &= (X^TV^2X)^{-1}X^TV^2y && (18) \end{aligned}$$

where $V^2 = diag(v \circ v)$ and $\circ$ is the Hadamard product [22] (a.k.a. entrywise product) for vectors. Eq. (18) can be actually viewed as a kind of locally weighted linear regression problem [20], where V^2 is the weighting matrix.

In the case that the matrix X^TV^2X is singular, we cannot solve $\hat{u}$ by Eq. (18) by the inverse operation. It is well known that Ridge regression [3] adds an additional matrix λI to the matrix X^TX in linear regression to make it non-singular. In our anti-noise regression model, we can apply a similar technique and estimate $\hat{u}$ by

$$\hat{u}_{ridge} = (X^TV^2X + \lambda I)^{-1}X^TV^2y \tag{19}$$

where λ is determined by minimizing RMSE.

4.7 Model Evaluation

Imagining that we have n choices for discounting functions, the number of different aggregation models can be as high as n^m for both linear aggregation and multiplicative aggregation. Although correlation analysis can rule out a large portion of aggregation models by selecting discounting functions with low RMSE, we still need standard evaluation metrics to compare different aggregation models and find the optimal discounting model for the recommender system. In this paper, we use the following evaluation metric to assess the performance of an aggregated discounting model:

- **Precision at Top** k. In the testing data set, given a $user$, we return the top k $item$ set L_k ranked by $\mathcal{T}.R \cdot d$ without looking at the $conversion$ attribute, where d is computed by the aggregated discounting model. Precision at Top k will be measured as the conversion rate of L_k, which is

$$P@k = \frac{seqSize(conversion = true)}{k} \tag{20}$$

5. EXPERIMENTAL EVALUATION

In this section, we test the proposed impression discounting model on three real online impression data sets: LinkedIn PYMK, LinkedIn Endorsement and Tencent SearchAds. The data set details are described in Section 3.2.

5.1 Correlation Analysis

Correlation analysis helps determine which discounting function fits the relationship between the conversion rate and a user behavior the best. We perform correlation visualization in both 2-D and 3-D.

5.1.1 2-D Correlation Analysis

The 2-D correlation analysis between the conversion rate and a specific behavior is an effective way to discover the strength and type of relationships between them. Figure 7 shows the conversion rate vs. *LastSeen*, *Position* and *UserFreq* on three real data sets. The conversion rate vs. *ImpCount* has already been shown in Figure 3. In Figure 7(a), we can see conversion rate in PYMK decreases faster with *LastSeen* than the conversion rate in the Endorsement data set, which reveals that *LastSeen* for Endorsement is less important than it for PYMK. *LastSeen* is not available in SearchAds data set. Figure 7(b) shows the relationships between the conversion rate and offset positions in the recommendation list. Clearly, if an $item$ is ranked lower in the recommendation list, its conversion rate will be lower. Endorsement data set has the steepest decreasing trend, compared with the other two data sets. In Figure 7(c), we show the conversion rate as the increasing of *UserFreq*. Surprisingly, we find that the conversion rate increases in both PYMK and Endorsement. The explanation may be that, a higher *UserFreq* indicates that $user$ is more active in the recommender system, and also more likely to take conversion actions. *UserFreq* does not play an important role in SearchAds, most likely because users do not have enough stickiness to a general web search engine.

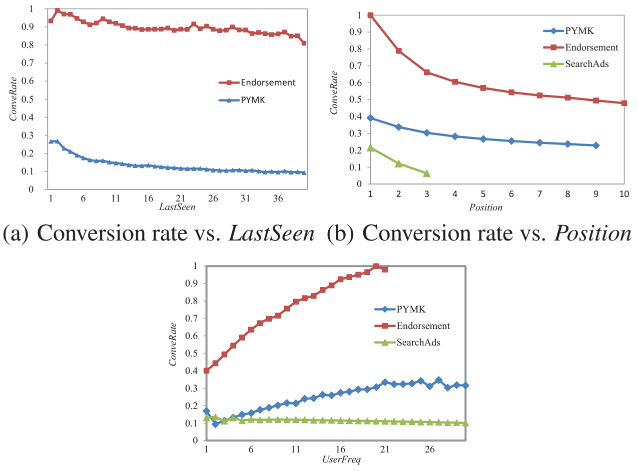
5.1.2 3-D Correlation Analysis

In Figure 8, we perform the 3-D visualization between conversion rate and two user behaviors on three data sets respectively. Each ball represents an observation, which is the conversion rate with respect to a specific value of user behaviors. As we can see, on the PYMK data set (Figure 8(a)), *ConveRate* decreases with both *ImpCount* and *LastSeen*. However, on the Endorsement data set (Figure 8(b)), *ConveRate* is not very sensitive with *LastSeen*, as supported by Figure 7(a). Figure 8(b) also shows there are many outlier observations even when $ImpCount < 10$, mainly because most impression sequences in Endorsement data set are very short and the sparsity problem becomes very critical. On the SearchAds data set (Figure 8(c)), because *LastSeen* is not available, we show $ConveRate$ vs. *ImpCount* and *Position*, which clearly decreases along both dimensions.

5.2 Model Evaluation

5.2.1 Precision at Top k

We split each impression data set into training and testing sets, and the impression discounting model is learned from the training



(a) Conversion rate vs. *LastSeen* (b) Conversion rate vs. *Position*

(c) Conversion rate vs. *UserFreq*

Figure 7: Conversion rate vs. *LastSeen*, *Position* and *UserFreq* on three real data sets.

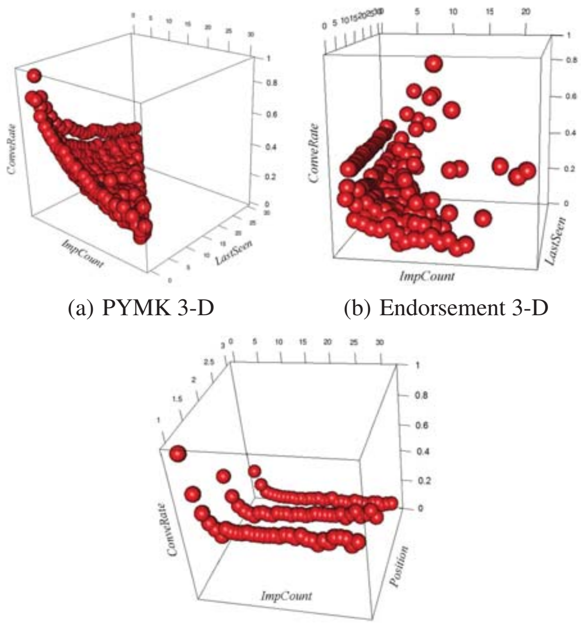
Behavior Set	Precision Improvement
PYMK (P@10)	
LastSeen, ImpCount	13.7%
LastSeen, ImpCount *Position, UserFreq*	31.3%
Endorsement (P@10)	
LastSeen, ImpCount	1.3%
LastSeen, ImpCount *Position, UserFreq*	3.4%
SearchAds (P@5, 10)	
ImpCount	0.53%(P@10)
ImpCount, Position	3.2%(P@10)
ImpCount, Position	6.87%(P@5)

Table 2: The improvement of precision at top k of the impression discounting model on different real-world data sets. The improvement rate is computed based on the precisions at top k without discounting.

set. We use precision at top k defined by Eq. (20) on the testing set to evaluate the performance of different aggregated discounting models. The precision improvement of the impression discounting model, compared with the baseline without a discounting model, is shown in Table 2. As we can see, if we integrate more user behaviors into the impression discounting model, the precision will be improved: the 4-behavior model on both PYMK and Endorsement gains at least two times higher precision growth compared with the 2-behavior model. Also, the precision growth on PYMK is obviously higher than the precision growth on Endorsement and SearchAds, because impression sequences in PYMK is distinctly longer than the impression sequences in the other two data sets. As proof, P@5 is obviously higher than P@10 on SearchAds, because a user has 5.8 items on average, and the precision on top 10 items of each user is not an effective evaluation metric.

5.2.2 A/B Testing

We implemented the impression discounting model in the LinkedIn PYMK recommender system, and the online A/B test results is shown in Table 3. This impression discounting model uses two behaviors, *LastSeen* and *ImpCount*. We fix the *ImpCount* dimension as the exponential discounting (as it has the minimal RMSE), and try the *LastSeen* by exponential discounting, inverse discounting, and linear discounting respectively. The results show that exponential discounting achieves a slightly better precision improvement, and all the Group B models gain significant improvements by incorporating impression discounting in recommendations.



(a) PYMK 3-D (b) Endorsement 3-D

(c) SearchAds 3-D

Figure 8: The 3-D visualization between conversion rate and user behaviors on three real data sets.

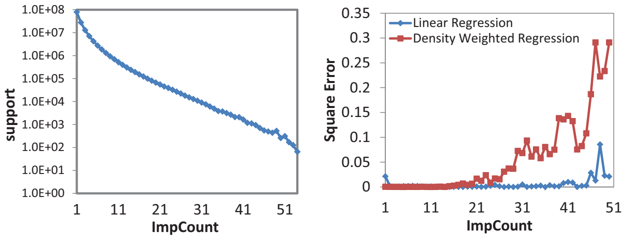
X_1: *LastSeen*; X_2: *ImpCount*	
Group A: Without Impression Discounting	
Group B:	Improvement
$\alpha_1 \cdot \alpha_2^{X_1} \cdot \alpha_3^{X_2}$	$11.97\% \pm 0.2\%$
$(\frac{\alpha_1}{X_1} + \alpha_2) \cdot \alpha_3^{X_2}$	$13.26\% \pm 0.2\%$
$(\alpha_1 \cdot X_1 + \alpha_2) \cdot \alpha_3^{X_2}$	$12.18\% \pm 0.2\%$

Table 3: The A/B test results of precision at top 10 of different impression discounting models on PYMK data set, with X =(*LastSeen*, *ImpCount*).

5.2.3 Enhancing Linear Regression

In this section, we evaluate the density weighted regression proposed in Section 4.5.3. Compared with traditional linear regression, density weighted regression removes the outlier observations and assigns weights to observations based on their supports. Figure 9(a) shows the distribution of observation supports is very skewed and follows power-low distribution. If we treat these observations as unweighted, the squared error of each observation using traditional linear regression is shown in the lower line in Figure 9(b). As we can see, although linear regression minimizes the sum of squared errors by reducing the area under the curve (AUC), squared errors are high when *ImpCount* is small. Since Figure 9(a) reveals that most observations have small *ImpCount*, traditional linear regression suffers from the problem that the majority of impression sequences have high prediction error.

In contrast, density weighted regression tries to minimize the prediction error of impression sequences. In this experiment, we set $\varepsilon = 2$, $\gamma = 0.01$ and $\delta = 2$. In the preprocessing, we detect



(a) (b)

Figure 9: (a) The support of observations with specific *ImpCount* in logarithmic scale. (b) The squared error of each observation by traditional linear regression and our anti-noise linear regression. Experiments are performed on PYMK data.

and remove 5 observations on the tail. We set the weights by Eq. (15), and the squared errors are shown by the upper line in Figure 9(b). Although the AUC of density weighted regression is larger than the traditional linear regression, the squared errors are minimized when *ImpCount* is small. To evaluate their performance in model-fitting, we instantiate the RMSE in Eq. (14) by

$$RMSE^2 = \frac{1}{\sum_{X \in D} support(X)} \sum_{i=1}^{n} (support(X_i, y_i)(y_i - \tilde{X_i}^T u))^2 \tag{21}$$

On PYMK data set, we compute the RMSE for traditional linear regression and density weighted regression as 0.1121 and 0.0188 respectively, which indicates density weighted regression performs much better than linear regression when the distribution of observation supports is skewed.

6. CONCLUSION

In this paper, we focuse on the impression discounting problem in large-scale recommender systems, in which a user's impressions on historical recommended items are discounted to re-rank the recommendation list. User satisfaction quantified by the conversion rate is expected to be improved due to the re-ranking. To precisely capture each facet of user interaction, we build impression discounting models by integrating discounting functions of user behaviors into a linear or multiplicative aggregation model. Moreover, to resolve the sparsity problem on observation supports, we propose anti-noise regression model to remove the outlier observations and perform a density weighted regression, which achieves a better prediction quality than the traditional linear regression. The proposed impression discounting framework is evaluated on three real-world data sets from LinkedIn and Tencent. The precision improvement on these impression data sets supports the effectiveness of our impression discounting model.

7. REFERENCES

[1] D. Agarwal, B.-C. Chen, and P. Elango. Spatio-temporal models for estimating click-through rate. In *WWW*, 2009.

[2] E. Agichtein, E. Brill, and S. T. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR*, pages 19–26, 2006.

[3] C. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[4] W. Chen, Y. Chen, Y. Mao, and B. Guo. Density-based logistic regression. In *KDD*, pages 140–148, 2013.

[5] B. Dalessandro, R. Hook, and C. Perlich. Evaluating and optimizing online advertising: Forget the click, but there are good proxies. In *Proceedings of Empirical Generalizations in Advertising Conference*, 2012.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[7] N. Gupta, A. Das, S. Pandey, and V. K. Narayanan. Factoring past exposure in display advertising targeting. In *KDD*, pages 1204–1212, 2012.

[8] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[9] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, 2008.

[10] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *TKDD*, 4(1), 2010.

[11] S. Kpotufe and V. K. Garg. Adaptivity to local smoothness and dimension in kernel regression. In *NIPS*, pages 3075–3083, 2013.

[12] P. Lee, L. V. S. Lakshmanan, and E. E. Milios. Keysee: supporting keyword search on evolving events in social streams. In *KDD*, pages 1478–1481, 2013.

[13] P. Lee, L. V. S. Lakshmanan, and E. E. Milios. Incremental cluster evolution tracking from highly dynamic network data. In *ICDE*, pages 3–14, 2014.

[14] X. Lin and D. Zhang. Inference in generalized additive mixed models by using smoothing splines. *Journal of the Royal Statistical Society, Series B*, 61(2):381–400, 1999.

[15] O. Moling, L. Baltrunas, and F. Ricci. Optimal radio channel recommendations with explicit and implicit feedback. In *RecSys*, pages 75–82, 2012.

[16] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, 1999.

[17] C. Perlich. Machine learning challenges in targeted advertising: Transfer learning in action. 2013.

[18] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW*, pages 521–530, 2007.

[19] G. Ristanoski, W. Liu, and J. Bailey. Time series forecasting using distribution enhanced linear regression. In *PAKDD (1)*, pages 484–495, 2013.

[20] D. Ruppert and M. P. Wand. Multivariate locally weighted least squares regression. *Annals of Statistics*, 22:1347–1370, 1994.

[21] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[22] G. P. Styan. Hadamard products and multivariate statistical analysis. *Linear Algebra and its Applications*, 6:217–240, 1973.

[23] R. Sumbaly, J. Kreps, and S. Shah. The big data ecosystem at linkedin. In *SIGMOD*, pages 1125–1134, 2013.

[24] D. Yang, T. Chen, W. Zhang, Q. Lu, and Y. Yu. Local implicit feedback mining for music recommendation. In *RecSys*, pages 91–98, 2012.

[25] S.-H. Yang, B. Long, A. J. Smola, H. Zha, and Z. Zheng. Collaborative competitive filtering: learning recommender using context of user choice. In *SIGIR*, pages 295–304, 2011.

[26] J. Zhang and P. Pu. A recursive prediction algorithm for collaborative filtering recommender systems. In *RecSys*, pages 57–64, 2007.