Scalar is Not Enough: Vectorization-based Unbiased Learning to Rank

Mouxiang Chen
Zhejiang University & Alibaba-Zhejiang University Joint Institute of Frontier Technologies
chenmx@zju.edu.cn

Chenghao Liu*
Salesforce Research Asia
Singapore
chenghao.liu@salesforce.com

Zemin Liu
Singapore Management University
Singapore
zmliu@smu.edu.sg

Jianling Sun*
Zhejiang University & Alibaba-Zhejiang University Joint Institute of Frontier Technologies
sunjl@zju.edu.cn

ABSTRACT

Unbiased learning to rank (ULTR) aims to train an unbiased ranking model from biased user click logs. Most of the current ULTR methods are based on the examination hypothesis (EH), which assumes that the click probability can be factorized into two scalar functions, one related to ranking features and the other related to bias factors. Unfortunately, the interactions among features, bias factors and clicks are complicated in practice, and usually cannot be factorized in this independent way. Fitting click data with EH could lead to model misspecification and bring the approximation error.

In this paper, we propose a vector-based EH and formulate the click probability as a dot product of two vector functions. This solution is complete due to its universality in fitting arbitrary click functions. Based on it, we propose a novel model named Vectorization to adaptively learn the relevance embeddings and sort documents by projecting embeddings onto a base vector. Extensive experiments show that our method significantly outperforms the state-of-the-art ULTR methods on complex real clicks as well as simple simulated clicks. [1]

CCS CONCEPTS

• **Information systems → Learning to rank**.

KEYWORDS

learning to rank, unbiased learning to rank, examination hypothesis

ACM Reference Format:
Mouxiang Chen, Chenghao Liu, Zemin Liu, and Jianling Sun. 2022. Scalar is Not Enough: Vectorization-based Unbiased Learning to Rank. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Madrid, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3534678.3539468

*Corresponding authors.
[1]Codes are provided at https://github.com/Keytoyze/Vectorization

1 INTRODUCTION

Learning to rank (LTR) with click data has been widely employed in modern information retrieval systems since this logged feedback reflects the utility of each document for each user [24], which is relatively easy to obtain on a large scale. However, they inherently contain a lot of bias from user behavior [25]. For example, users are more likely to observe documents at a higher position, known as position bias, which causes clicks to be biased with the position. Using unbiased learning to rank (ULTR) to remove these biases has attracted increasing research interest [1, 26]. The key idea is the **examination hypothesis** (EH): each document has a certain probability of being observed and is then clicked based on the relevance, where the observation depends on some bias factors (e.g., position), and the relevance depends on the features that encoding query and document. The EH can be written as:

$$P(\text{click}) = P(\text{observation} \mid \text{bias factors}) \cdot P(\text{relevance} \mid \text{features}).$$

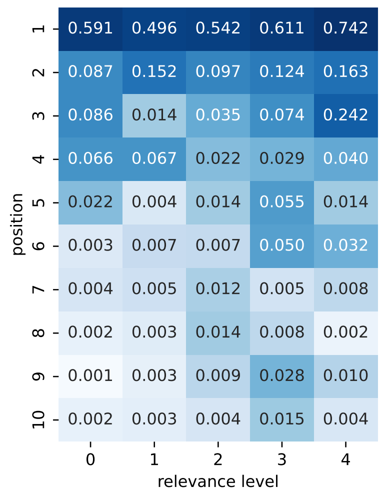


Figure 1: Real click rate matrix calculated on TianGong-ST. Darker colors indicate larger click rates.

Given this, ULTR methods try to model the observation probability using bias factors, and reweigh the click signals based on the reciprocal of observation probabilities, to recover an unbiased relevance probability for the ranking objective.

The EH relies on an implicit assumption - the click probability can be factorized into two *scalar* functions, one takes the bias factors as input only and the other takes features only. Unfortunately, this assumption isn't sufficient in practice, since the interaction between relevance and observation is rather complicated [2, 39, 44].

To illustrate it more intuitively, we performed a simple statistical analysis on the TianGong-ST dataset[2] [13]. This dataset contains both user click data (sampled from a real-world search engine) and relevance levels (annotated by humans). We grouped documents by their positions and relevance levels, and compute the click rates for each group as a click rate matrix. Figure 1 demonstrates the results. We assume the observation depends only on position [2, 4, 26, 41], and the relevance depends on the relevance level. Particularly, if the EH applies to it, then the matrix can be factorized into a 10×1 vector (denotes observation probability for each position) and a 1×5 vector (denotes relevance probability for each level) , which infers that the matrix rank must be 1. Obviously, this matrix does not satisfy this condition, since the singular values of this matrix are $(1.40, 0.15, 0.07, 0.06, 0.03)$. It shows that EH cannot describe real-world click rates accurately.

We identify the root cause of this problem to be the fact that the EH is *incomplete*: the click probability can be arbitrary functions related to bias factors and features due to the complicated interaction between observation and relevance, but the function family produced by the combination of such two scalar functions cannot cover all possible functions. Using this form to fit the click data could lead to model misspecification and bring approximation error no matter how much data we collect. Recent efforts extended EH and explicitly described the generative process of clicks in their specific scenarios [2, 35, 39], which, however, require prior knowledge and still cannot obtain the best performance in common click scenarios due to their insufficiency to cover all possible click functions.

To address this issue, we extend the EH into a vector-based formulation: the click rates can be written as the dot-product of two *vector* functions, one related to bias factors (named as observation embedding), and the other related to features (named as relevance embedding). Moreover, the universality of this factorization can be justified [27]: for any given click rate function, we can always find an appropriate dimension for these vectors such that the approximation error can be arbitrarily small. This suggests that our vector-based EH is complete and can catch complicated click patterns *adaptively* by optimization. Compared to traditional relevance scalars, relevance embeddings have a more powerful capacity to encode how relevant a document is.

However, unlike relevance scalars, embedding vectors cannot be sorted, which challenges the usage of this vector-based EH. To sort the documents with their relevance embeddings in the inference stage, we propose to use a common *base vector* and project each relevance embedding onto it to obtain relevance scalars, and sort the documents with these scalars. The challenge is how to find such a proper common base vector. We argue that we can find the most probable bias factors that ever appear in the training dataset together with all of these ranking features, and use the corresponding observation embedding as the base vector. This is because the dot-product of such a base vector and each relevance embedding is more closed to the real click rate when the features are assigned with the same bias factors. This product result can serve as a substitute for relevance. Given that some bias factors and ranking features may not overlap in the dataset, we further propose to use the most probable observation embedding that ever appears together with features as the base vector. Finally, we derive a closed-form of the base vector, which enables us to calculate it in a very efficient way.

[2]http://www.thuir.cn/tiangong-st/

Moreover, to evaluate the performance of our model in practice, we propose a method to apply the click pattern in the real world in semi-synthetic experiments. Extensive experiments conducted on two widely-used datasets showed that our Vectorization method significantly outperforms the state-of-the-art ULTR methods in both simple and complex click settings.

To the best of our knowledge, we are the first to study the limitation of scalar-based EH used by the current ULTR framework. The main contributions of this work are three-fold:

(1) We propose a vector-based examination hypothesis (vector-based EH) that can capture complicated interactions between clicks, features, and bias factors through a dot-product between relevance embeddings and observation embeddings. This hypothesis is complete for real-world click data.
(2) We propose a method that can sort documents with relevance embeddings, in which each relevance embedding is projected onto a base vector. The base vector can be calculated efficiently.
(3) We provide a method to apply the real click pattern into semi-synthetic experiments.

2 RELATED WORK

Debiasing Click Data. Most of the current approaches to debiasing click data for ranking are based on the examination hypothesis. They can be divided into two groups. The first is to model user's behavior to infer relevance from biased click signals, known as *click models* [7, 8, 15, 17, 18]. However, most click models focus on predicting clicks, and the relevance inference is an afterthought [4]. The second group tries to directly learn unbiased ranking models from biased clicks, known as *unbiased learning to rank* (ULTR). Based on it, Joachims et al. [26] proposed the inverse propensity scoring (IPS) method to reweigh the click signals based on the reciprocal of observation probabilities (called propensity scores) and provide an unbiased estimate of the ranking objective. The propensity scores are estimated by randomized experiments [26, 40], which hurts users' experience, unfortunately. To address it, Agarwal et al. [3] and Fang et al. [16] proposed to do intervention harvest by exploiting click logs with multiple ranking models. Nevertheless, they have a relatively narrow scope of application due to the strict assumption to construct interventional sets [14]. Recently, some researchers proposed to jointly estimate relevance and bias [4, 20, 22, 41]. Similar to them, our proposed method could jointly train the ranking model and observation model without intervention.

On the other side, researchers developed models to extend the scope of bias factors. The bias factors contain position [4, 10, 20, 41], contextual information [16, 37], clicks in the same query list [14, 38], presentation style [32, 44], search intent [36] and result domain

[21]. In our work, we don't limit the exact meaning of bias factors, which makes our model more flexible and generic.

Clicks beyond examination hypothesis. There is much work finding that the click functions of features and bias factors are complicated, and cannot be written in the form of scalar-based EH. For instance, in trust bias [2, 24, 39], users are more likely to click incorrectly on higher-ranked items, and the relevance scalar function requires an affine transformation about the position, to fit the clicks. As we will mention in this paper, trust bias can be written explicitly as a 2-dimensional vector-based EH.

Beyond trust bias, there exist other click patterns that cannot be written in the form of EH. Williams et al. [43] and Zheng et al. [44] argued that some documents with low click necessity will lower the click probability, while the click necessity is related to relevance and bias factors (like presentation style). Liu et al. [31] found that users may examine results in several stages, and different bias factors and features take effect in the different stages. Even though the click function in these scenarios may not be written explicitly as a vector dot-product, the vector-based EH can approximate it thanks to its universality.

Vector-based factorization. Vector-based factorization is widely used in the field of recommendation systems, known as matrix factorization, where a user-item rating matrix is approximated by the product of two low-rank matrices (latent factor vectors) [19, 29, 30, 34, 42]. Besides, [27] proved the universality of product effect, which is a theoretical guarantee for our vector-based EH. However, this work didn't provide a solution to sort the vectors, which is crucial in the LTR task.

3 PRELIMINARIES

In this paper, we use bold letters to denote vectors (e.g., $\mathbf{r}$), and thin letters to denote scalars (e.g., r). Generally, the core of LTR is to learn a ranking model f which assigns a relevance score to a document with its ranking feature. For a query, documents can be sorted in descending order by their scores. In the full information setting that we already know the true relevance for each document, the observational data related to a query $q \in \mathcal{Q}$ can be notated as $\mathcal{D}_q^{\text{full-info}} = \{(\mathbf{x}_i, r_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathcal{X}$ denotes the ranking features encoding query, document and user, and $r_i \in \mathbb{R}$ denotes its true relevance score. The ranking target is to optimize f by minimizing the empirical risk:

$$\mathcal{R} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \sum_{(\mathbf{x}_i, r_i) \in \mathcal{D}_q^{\text{full-info}}} L(f(\mathbf{x}_i), r_i),$$

where L denotes a loss function based on any specific IR metric of interest [26]. The true relevance score r_i denotes how relevant a document with the query related to the ranking features $\mathbf{x}_i$, which is typically obtained by human annotation.

In practice, the relevance scores are often unknown and are costly to estimate through human labeling [11]. Instead, offline Unbiased Learning to Rank (ULTR) methods try to learn the ranking model from offline click logs, which are cheap and timely to obtain at scale. This is because click logs can be seen as implicit feedback which reflects users' preferences to some extent. Nevertheless, click logs are often biased. For example, higher-ranked documents are more likely to be observed and clicked (known as position bias).

In this click setting, the observational data related to q can be notated as $\mathcal{D}_q = \{(\mathbf{x}_i, \mathbf{t}_i, c_i)\}_{i=1}^n$, where $c_i \in \{0, 1\}$ denotes the click signals of $\mathbf{x}_i$, and $\mathbf{t}_i \in \mathcal{T}$ denotes bias factors that cause clicks to be biased, such as document position [26], context information [16], other clicks around the document [14, 38] or the presentation style [32]. In this work, we do not limit the exact meaning of $\mathbf{t}_i$, which allows us to generalize our conclusion to most of the previous ULTR methods. For convenience, let $\mathcal{D} = \{(\mathbf{x}, \mathbf{t}) \mid (\mathbf{x}, \mathbf{t}, c) \in \mathcal{D}_q, q \in \mathcal{Q}\}$ denote all pairs of ranking features and bias factors that ever appear in the dataset.

We assume that the click rate of a document only depends on its ranking features and its bias factors. Denote $c(\mathbf{x}, \mathbf{t}) = \Pr(c = 1 \mid \mathbf{x}, \mathbf{t})$ as the click rate function, with $(\mathbf{x}, \mathbf{t}) \in \mathcal{D}$. In order to derive relevance from click data, most of the current ULTR methods [4, 14, 16, 22, 26, 41] are based on **examination hypothesis** (EH) to model user's click behavior. It assumes that the user clicks on a document if this document is observed and relevant. If we further assume that the relevance r depends on the ranking features $\mathbf{x}$ and observation o depends on the bias factors, we have:

$$c(\mathbf{x}, \mathbf{t}) = r(\mathbf{x}) \cdot o(\mathbf{t}), \quad \forall (\mathbf{x}, \mathbf{t}) \in \mathcal{D}, \tag{1}$$

where $r(\mathbf{x})$ denotes the probability of relevant and $o(\mathbf{t})$ denotes the probability of being observed by user. Both $r : \mathcal{X} \to \mathbb{R}$ and $o : \mathcal{T} \to \mathbb{R}$ are scalar functions. By explicitly modeling the bias effect via observation probability, it is able to attain an unbiased estimate of the ranking objective.

4 VECTORIZATION-BASED ULTR

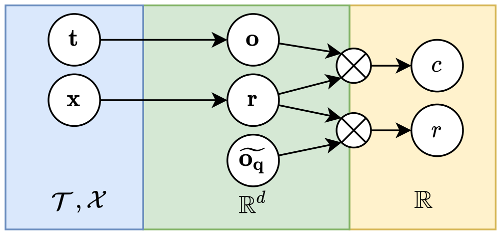

t
x
o
r
$\widetilde{\mathbf{o}_\mathbf{q}}$
c
r
$\mathcal{T}, \mathcal{X}$
$\mathbb{R}^d$
$\mathbb{R}$

Figure 2: Graph representation of our method.

In this section, we propose our Vectorization to deal with complicated click patterns. Figure 2 illustrates our framework. We first map bias factors and features into vectors and combine them onto clicks for training (§ 4.1). For final ranking, we use a common base vector to project relevance embeddings onto scalar (§ 4.2). Finally, we describe how to find such a base vector (§ 4.3). In addition, we further discuss the relation with trust bias in § 4.4.

4.1 Vector-based EH

Unfortunately, the interaction between clicks, bias factors and features is extremely complicated in real-world [9, 31, 32, 43]. An ideal factorization for the click in Eq.(1) often does not exist in practice, since the function family produced by this form cannot cover all possible click rate functions. It leads to model misspecification and brings approximation error no matter how much data we collect.

To capture the complicated interaction between relevance and observation, we first extend the scalar-based EH to a vector-based formalization. That is, we assume a *product effect*: the click function $c(\mathbf{x}, \mathbf{t})$ can be written as a dot product of two functions, one over the ranking features $\mathbf{x}$ and the other over the bias factors $\mathbf{t}$.

$$c(\mathbf{x}, \mathbf{t}) = \mathbf{r}(\mathbf{x})^\top \mathbf{o}(\mathbf{t}), \tag{2}$$

where $\mathbf{r} : \mathcal{X} \to \mathbb{R}^d$ is the relevance function and $\mathbf{o} : \mathcal{T} \to \mathbb{R}^d$ is the observation function. The outputs of them are referred to as **relevance embedding** and **observation embedding**. Note that, Eq.(1) is a special case of Eq.(2) when $d = 1$.

The universality of product effect can be formally justified: if we increase the dimensionality d of $\mathbf{r}$ and $\mathbf{o}$, any arbitrary bounded continuous function in $c(\mathcal{X} \times \mathcal{T})$ can be approximated. A formal proof of the universality of product effect is provided in the Appendix § A. It enables us to decompose the biased click into an unbiased part (relevance) and a biased part (observation), no matter how complicated the interaction between relevance and observation is.

4.2 Rank with relevance embedding

However, it's not possible to apply the vector-based EH immediately by reason that relevance embedding cannot be sorted according to their values. We need to find a method that can rank documents with their relevance embeddings. For a given query q with n ranking features $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$, our goal is to rank these features with their relevance embeddings $\mathbf{r}(\mathbf{x}_1), \cdots, \mathbf{r}(\mathbf{x}_n)$. Obviously, it is not suitable to simply average the elements in vectors and sort all vectors based on the average values[3]. Consider the form of Eq.(2), a natural solution is to find a common **base vector** $\widetilde{\mathbf{o}_q} \in \mathbb{R}^d$ for a query q, and project each relevance embedding onto $\widetilde{\mathbf{o}_q}$:

$$r(\mathbf{x}_i) = \mathbf{r}(\mathbf{x}_i)^\top \widetilde{\mathbf{o}_q}, \quad i \in [n]. \tag{3}$$

Then we can sort with the scalar $r(\mathbf{x}_i)$, like traditional LTR methods. In fact, for a set of relevance embeddings, there exists a base vector that can sort them in any given order if we allow d to grow, which shows the universality of this projection method. This is because when $r(\mathbf{x}_i)$ and $\mathbf{r}(\mathbf{x}_i)$ are fixed, Eq.(3) can be seen as linear equations in which $\widetilde{\mathbf{o}_q}$ is a variable, $\mathbf{r}(\mathbf{x}_i)$ is a coefficient and $r(\mathbf{x}_i)$ is a constant. Suppose that $\{\mathbf{r}(\mathbf{x}_i) : i \in [n]\}$ are linearly independent. If $d \geq n$, then the linear equations must have a solution of $\widetilde{\mathbf{o}_q}$. It suggests that if the dimension d is large enough, we can always find a base vector such that $r(\cdot)$ can equal arbitrary values.

Now, the next problem is how to find such a base vector. If there exists a few labeled data, it can be done by solving the above linear equations. In this paper, we focus on a general case that the labeled data is unknown.

4.3 Find the base vector

In this section, we suppose the function $\mathbf{r}(\cdot)$ and $\mathbf{o}(\cdot)$ are given and fixed, and we aim to find the base vector *unsupervisedly*. To start with, we assume that for any two documents, their click rate order equals to their relevance order if we fix their bias factors:

[3]This is because we do not impose a non-negative constraint on the embeddings. We can flip the signs of the relevance embedding and the observation embedding at the same time without changing their product, but the average of relevance embeddings is quite different after flipping.

$$c(\mathbf{x}_1, \mathbf{t}) \geq c(\mathbf{x}_2, \mathbf{t}) \iff \mathbf{x}_1 \succeq_r \mathbf{x}_2, \quad \forall (\mathbf{x}_1, \mathbf{t}), (\mathbf{x}_2, \mathbf{t}) \in \mathcal{D}, \tag{4}$$

where $\mathbf{x}_1 \succeq_r \mathbf{x}_2$ means that $\mathbf{x}_1$ is more relevant than $\mathbf{x}_2$. This is owing to that if two documents have the same bias factors, they are in the same environment, thus a more relevant document tends to receive more clicks.

We start with a toy example. For a query q and the corresponding ranking features $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, suppose that there exist common bias factors $\mathbf{t}$ such that they ever appear together with these features in $\mathcal{D}$. Here, we can just set:

$$\widetilde{\mathbf{o}_q} = \mathbf{o}(\mathbf{t}), \quad \text{s.t. } (\mathbf{x}_i, \mathbf{t}) \in \mathcal{D}, \forall i \in [n]. \tag{5}$$

This is because $\mathbf{r}(\mathbf{x}_i)^\top \widetilde{\mathbf{o}_q} = \mathbf{r}(\mathbf{x}_i)^\top \mathbf{o}(\mathbf{t})$ indicates the click rate of $\mathbf{x}_i$ and $\mathbf{t}$, which reflects the relevance because of Eq.(4). Note that if there exists more than one $\mathbf{t}$, it's hard to decide which one to use. Thus, we can use maximum likelihood estimation (MLE) to choose the bias factors t^* that maximize the probability of appearing together with $\mathbf{x}_1, \cdots, \mathbf{x}_n$. Suppose $\mathcal{D}$ is generated from a joint distribution $P(X, T)$, where X are the ranking features and T are the bias factors. Then we set:

$$\widetilde{\mathbf{o}_q} = \mathbf{o}(\mathbf{t}^*), \quad \text{where } \mathbf{t}^* = \arg\max_{\mathbf{t}} \prod_{i=1}^{n} P(T = \mathbf{t} \mid X = \mathbf{x}_i), \tag{6}$$

and $P(T \mid X)$ can be estimated from $\mathcal{D}$. Generally, the click rate estimation will be more accurate for the bias factors that have a larger probability of $P(T \mid X)$ [45]. Thus we select the *most possible* bias factors related to the ranking features as the base vector $\widetilde{\mathbf{o}_q}$, since the combination $\mathbf{r}(\cdot)^\top \widetilde{\mathbf{o}_q}$ can be seen as an accurate click rate.

However, it may be intractable in practice since common bias factors for all ranking features may *not exist* in the training dataset. For example, we assume $\mathbf{t}$ denotes the position of documents. Suppose that a ranking feature $\mathbf{x}_1$ is always assigned to the first position, and another ranking feature $\mathbf{x}_2$ is always assigned to the second position. No matter how we choose a position $\mathbf{t}$, there must exist a ranking feature $\mathbf{x} \in \{\mathbf{x}_1, \mathbf{x}_2\}$ such that $P(\mathbf{t} \mid \mathbf{x}) = 0$, which challenges finding base vector by Eq.(6).

We identify the root cause to be the fact that X and T may not *overlap*: $P(X, T)$ may be zero for some $(X, T) \in \mathcal{X} \times \mathcal{T}$. To address this problem, we first transform $\mathcal{D}$ into $\mathcal{D}_{\mathbf{o}} = \{(\mathbf{x}, \mathbf{o}(\mathbf{t})) : (\mathbf{x}, \mathbf{t}) \in \mathcal{D}\}$. This time we use the observation embedding that maximizes the probability of appearing together with $\mathbf{x}_1, \cdots, \mathbf{x}_n$, rather than the raw bias factors:

$$\widetilde{\mathbf{o}_q} = \arg\max_{\mathbf{o}} \prod_{i=1}^{n} P(O = \mathbf{o} \mid X = \mathbf{x}_i), \tag{7}$$

where $P(O \mid X)$ can be estimated from $\mathcal{D}_{\mathbf{o}}$. Eq.(7) is similar to but better than Eq.(6) in use, for that O is more dense than the raw bias factors, which alleviates the overlap problem.

Furthermore, we propose to model the $P(O \mid X)$ as a multivariate Gaussian distribution (all of its components are independent), since that $P(O \mid X) > 0$ can always be established, which avoids the overlap problem and make the estimation more stable. More importantly, Eq.(7) will have a closed-form solution, which allows us to calculate $\widetilde{\mathbf{o}_q}$ in a very efficient way[4]. Suppose that

[4]We admit that the Gaussian distribution may not be the best model, but we found that it was good enough in the experiment. One explanation is that the relevance embeddings are robust enough to tolerate the small perturbation of the base vector.

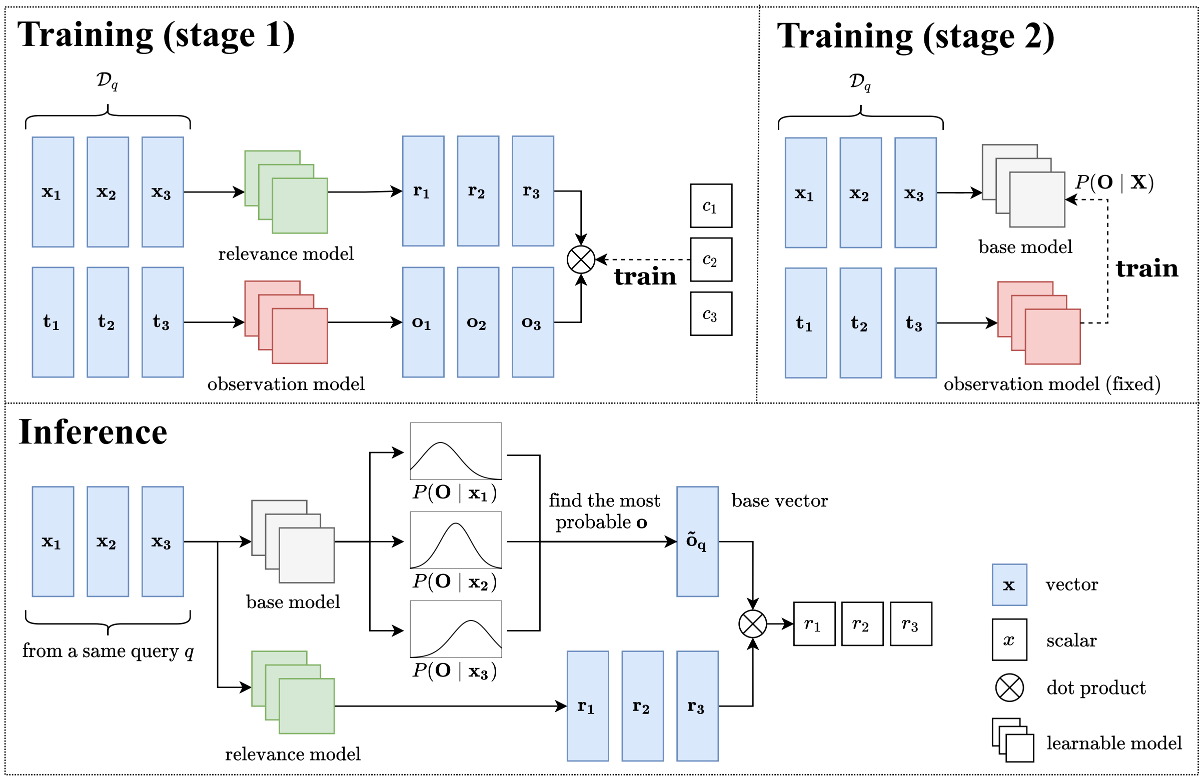

Training (stage 1)
$\mathcal{D}_q$
$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$
relevance model
$\mathbf{r}_1$ $\mathbf{r}_2$ $\mathbf{r}_3$
$\mathbf{t}_1$ $\mathbf{t}_2$ $\mathbf{t}_3$
observation model
$\mathbf{o}_1$ $\mathbf{o}_2$ $\mathbf{o}_3$
train
c_1 c_2 c_3
Training (stage 2)
$\mathcal{D}_q$
base model
$P(\mathbf{O} \mid \mathbf{X})$
train
observation model (fixed)
Inference
from a same query q
base model
$P(\mathbf{O} \mid \mathbf{x}_1)$
$P(\mathbf{O} \mid \mathbf{x}_2)$
$P(\mathbf{O} \mid \mathbf{x}_3)$
find the most probable $\mathbf{o}$
$\tilde{\mathbf{o}}_\mathbf{q}$
base vector
relevance model
r_1 r_2 r_3
vector
scalar
dot product
learnable model

Figure 3: Framework of the proposed Vectorization. In the first training stage, we jointly train a relevance model and an observation model with vector-based EH. In the second training stage, we train a base model to estimate the conditional observation embedding distribution. In the inference stage, we utilize the base model to infer the base vector based on input features and project each relevance embedding onto the base vector for ranking.

$P(O \mid X = \mathbf{x}_i) \sim \mathcal{N}\left(\boldsymbol{\mu}(\mathbf{x}_i), \boldsymbol{\sigma}(\mathbf{x}_i)^2\right)$, where $\boldsymbol{\mu}(\mathbf{x}_i)$ and $\boldsymbol{\sigma}(\mathbf{x}_i)$ are the mean and the standard deviation of O (given $X = \mathbf{x}_i$), we have:

$$\begin{aligned}\widetilde{\mathbf{o}_q} &= \arg\max_{\mathbf{o}} \prod_{i=1}^{n} P\left(O = \mathbf{o} \mid X = \mathbf{x}_i\right)\\ &= \arg\max_{\mathbf{o}} \prod_{i=1}^{n} \frac{1}{\boldsymbol{\sigma}(\mathbf{x}_i)\sqrt{2\pi}} \exp\left(-\frac{(\mathbf{o}-\boldsymbol{\mu}(\mathbf{x}_i))^2}{2\boldsymbol{\sigma}(\mathbf{x}_i)^2}\right)\\ &= \arg\max_{\mathbf{o}} \sum_{i=1}^{n} \left(-\frac{(\mathbf{o}-\boldsymbol{\mu}(\mathbf{x}_i))^2}{2\boldsymbol{\sigma}(\mathbf{x}_i)^2}\right),\end{aligned}$$

where every operators are element-wise. The second equality uses the PDF of Gaussian distribution. Let:

$$F(\mathbf{o}) = \sum_{i=1}^{n} \left(-\frac{(\mathbf{o}-\boldsymbol{\mu}(\mathbf{x}_i))^2}{2\boldsymbol{\sigma}(\mathbf{x}_i)^2}\right),$$

by solving the equation ${}^{\mathrm{d}F}/_{\mathrm{d}\mathbf{o}} = 0$ we reach the final form:

$$\widetilde{\mathbf{o}_q} = \frac{\sum_{i=1}^{n} \frac{1}{\boldsymbol{\sigma}(\mathbf{x}_i)^2}\boldsymbol{\mu}(\mathbf{x}_i)}{\sum_{i=1}^{n} \frac{1}{\boldsymbol{\sigma}(\mathbf{x}_i)^2}}. \tag{8}$$

It indicates that for a given query, the base vector can be calculated through a weighted average over the most probable observation embeddings of all ranking features related to q. The larger the variance, the smaller the weight. The variance can be explained as the model's uncertainty. Thus, a major advantage of our method is that it can comprehensively consider the base vector according to the uncertainty, which helps to increase the robustness.

4.4 Relation with trust bias

Most of current ULTR methods assume the scalar-based EH (Eq.(1)), while an exception is the trust bias model [2, 39], in which the click rate $c(\mathbf{x}, \mathbf{t})$ with regard to the ranking features $\mathbf{x}$ and the bias factors $\mathbf{t}$ (usually take the position) can be written as:

$$c(\mathbf{x}, \mathbf{t}) = \theta(\mathbf{t})\left(\epsilon^{+}(\mathbf{t})r(\mathbf{x}) + \epsilon^{-}(\mathbf{t})(1 - r(\mathbf{x}))\right),$$

where ϵ^{+}, ϵ^{-} and θ are some functions related to bias factors. This formula cannot be written in the form of scalar-based EH. Oppositely, it can be regarded as a special case of 2-dimensional vector-based EH (Eq.(2)), as long as we take:

$$\begin{aligned}\mathbf{r}(\mathbf{x}) &= [r(\mathbf{x}), 1]^{\top},\\ \mathbf{o}(\mathbf{t}) &= \theta(\mathbf{t})[\epsilon^{+}(\mathbf{t}) - \epsilon^{-}(\mathbf{t}), \epsilon^{-}(\mathbf{t})]^{\top},\end{aligned}$$

and always select $\widetilde{\mathbf{o}_q} = [1, 0]^{\top}$ as the base vector for final ranking. This proves the ability of our method to deal with trust bias. In the experiment, we find that in the trust bias click setting, when we take $d = 2$, the performance of our model is similar to Affine [39], which is designed to deal with trust bias specifically. This shows the method we used to find the base vector works.

5 MODEL IMPLEMENTATION

So far, we have shown that to effectively catch the complicated interaction among clicks, bias factors and features, we should learn a relevance embedding that can fully combine with the observation, and project the relevance embedding into the base vector to sort the documents. In this section, we introduce the implementation of the proposed Vectorization method, in the training stage and the inference stage. Figure 3 illustrates the overall model architecture.

5.1 Training stage

Stage 1. We learn two models at first: relevance model $\mathbf{r}$ and observation model $\mathbf{o}$. For a query q with the data $\mathcal{D}_q = \{(\mathbf{x}_i, \mathbf{t}_i, c_i)\}_{i=1}^n$, we first combine the relevance embedding and the observation embedding with dot-product, by Eq.(2). Similar to [4], we use a list-wise loss based on softmax-based cross entropy:

$$L_q^{click} = -\sum_{i=1}^{n} c_i \log \frac{\exp\left(\mathbf{r}(\mathbf{x}_i;\theta_r)^\top \mathbf{o}(\mathbf{t}_i;\theta_o)\right)}{\sum_{j=1}^{n}\exp\left(\mathbf{r}(\mathbf{x}_j;\theta_r)^\top \mathbf{o}(\mathbf{t}_j;\theta_r)\right)}, \tag{9}$$

where θ_r and θ_o are the weights of the relevance model and the observation model respectively. The softmax-based cross entropy naturally converts the combinational output $\mathbf{r}(\cdot)^\top \mathbf{o}(\cdot)$ into click probability distributions.

Stage 2. After the convergence of the observation model, we need to fix it and learn a base model $\mathbf{v}$ that estimates the distribution $P(\boldsymbol{O} \mid \boldsymbol{X})$ to find the base vector for the inference stage. As mentioned above, we fix a Gaussian likelihood to model the distribution, and the output of $\mathbf{v}$ is composed of both predictive mean as well as predictive variance:

$$\left[\boldsymbol{\mu}(\mathbf{x}_i), \boldsymbol{\sigma}^2(\mathbf{x}_i)\right] = \mathbf{v}(\mathbf{x}_i;\theta_v),$$

where $\boldsymbol{\mu}(\mathbf{x}_i) \in \mathbb{R}^d$ and $\boldsymbol{\sigma}^2(\mathbf{x}_j) \in \mathbb{R}^d$ are the outputs of the base model $\mathbf{v}$, and θ_v is the weight. We want to make the Gaussian distribution parameterized by $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ close to the real distribution $P(\boldsymbol{O} \mid \boldsymbol{X})$. This can be done by minimizing the following regression loss, according to Kendall et al. [28]:

$$L_q^{base} = \frac{1}{2}\sum_{i=1}^{n}\left(\frac{||\boldsymbol{\mu}(\mathbf{x}_i) - \mathbf{o}(\mathbf{t}_i)||_2^2}{\boldsymbol{\sigma}^2(\mathbf{x}_i)} + \log \boldsymbol{\sigma}^2(\mathbf{x}_i)\right) + \lambda||\theta_v||_2^2, \tag{10}$$

where λ is the hyper-parameter controlling the L2 regularization. In practice, we train the model to predict the log variance, $\boldsymbol{s}(\mathbf{x}_i) := \log \boldsymbol{\sigma}^2(\mathbf{x}_i)$, because it is more numerically stable than regressing the variance, as the loss avoids a potential division by zero [28].

5.2 Inference stage

In the inference stage, we discard the observation model $\mathbf{o}$ and use $\mathbf{r}$ and $\mathbf{v}$ to estimate relevance scalars for ranking. For a query q with the ranking features $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, we first calculate the base vector $\widetilde{\mathbf{o}_q}$ by Eq.(8) with the base model, and then project each relevance embedding onto the base vector by Eq.(3).

We further present the algorithm for model training and inference in Appendix § B.

6 EXPERIMENTS

In this section, we describe our experimental setup and show the empirical results. We assume that the bias factors only contain the positions, so we modeled the observation model as a position-based model (PBM) in our experiments. One could easily extend it to other models that consider more bias factors. Correspondingly, all of our semi-synthetic setups used positions as the only bias factors.

6.1 Experimental Setup

Dataset. We didn't conduct our experiments on the TianGong-ST dataset, since the ranking features are highly limited and the performance of ranking models can be volatile, as reported in [6]. Instead, we follow the standard semi-synthetic setup in ULTR [4, 5, 39] and conduct experiments on two widely used public benchmark datasets: Yahoo! LETOR[5] [11] and Istella-S[6] [33]. These two datasets have more ranking features than TianGong-ST. More importantly, this enabled us to simulate clicks under different bias settings, including complicated clicks close to the real scene and simple clicks simulated by click models. We provide further details for these datasets in Appendix § C.

We followed the data split of training, validation and testing given by the datasets. To generate initial ranking lists for click simulation, we followed the standard process [4, 14, 26] and use 1% of the training data with relevance labels to train a Ranking SVM model [23]. Based on these initial ranking lists generated by it, we sampled clicks under different click settings.

Click Simulation. We considered the following two click settings. In both cases, only the top $n = 10$ documents were considered to be displayed. $y_{\max}$ denotes the maximum of the relevance level. For Yahoo! and Istella-S, $y_{\max} = 5$.

Real Click. Define $\boldsymbol{A} = [a_{ij}] \in \mathbb{R}^{y_{\max}\times n}$ as the real click matrix, in which each element a_{ij} represents the click rate of a document that locates at the position i and has a relevance level of j. The value of $\boldsymbol{A}$ is calculated from the TianGong-ST dataset and given in Figure 1. We sampled clicks on the two datasets according to this real click matrix $\boldsymbol{A}$:

$$\Pr(C = 1 \mid \boldsymbol{X} = \mathbf{x}, P = p) = a_{py}, \tag{11}$$

where $y \in [1, y_{\max}]$ is the relevance level. It allowed us to apply the real click rates to our experiments.

Trust Bias. We applied Agarwal et al. [2]'s trust bias model to simulate clicks as our second click setting. The click probability can be written as:

$$\Pr(C = 1 \mid \boldsymbol{X} = \mathbf{x}, P = p) = \theta_p\left(\epsilon_p^+\gamma_y + \epsilon_p^-(1-\gamma_y)\right), \tag{12}$$

where the relevance probability γ_y is based on the relevance level y. We followed previous work [4, 12, 14] and set:

$$\gamma_y = \frac{2^{y-1}-1}{2^{y_{\max}-1}-1}, \tag{13}$$

and we used the position bias parameter θ_p estimated by Joachims et al. [24]. For the trust bias parameters ϵ_p^+ and ϵ_p^-, we adopted the

[5]https://webscope.sandbox.yahoo.com/
[6]http://quickrank.isti.cnr.it/istella-dataset/

Table 1: Comparison of different algorithms on two datasets in the real click setting. Numbers are shown with their standard deviation in 8 repeated runs (i.e., ± x).

Algorithms	Yahoo! nDCG@1	Yahoo! nDCG@3	Yahoo! nDCG@5	Yahoo! nDCG@10	Istella-S nDCG@1	Istella-S nDCG@3	Istella-S nDCG@5	Istella-S nDCG@10
Labeled Data	$0.680_{\pm.002}$	$0.689_{\pm.001}$	$0.711_{\pm.001}$	$0.758_{\pm.001}$	$0.655_{\pm.001}$	$0.632_{\pm.001}$	$0.660_{\pm.001}$	$0.723_{\pm.000}$
Click Data	$0.615_{\pm.010}$	$0.635_{\pm.005}$	$0.662_{\pm.004}$	$0.718_{\pm.003}$	$0.596_{\pm.002}$	$0.576_{\pm.001}$	$0.607_{\pm.001}$	$0.676_{\pm.001}$
DLA	$0.625_{\pm.013}$	$0.643_{\pm.010}$	$0.671_{\pm.010}$	$0.725_{\pm.008}$	$0.599_{\pm.016}$	$0.580_{\pm.015}$	$0.610_{\pm.012}$	$0.679_{\pm.010}$
PairDebias	$0.612_{\pm.005}$	$0.633_{\pm.003}$	$0.661_{\pm.003}$	$0.717_{\pm.002}$	$0.595_{\pm.002}$	$0.576_{\pm.001}$	$0.607_{\pm.001}$	$0.676_{\pm.000}$
RegressionEM	$0.618_{\pm.010}$	$0.628_{\pm.007}$	$0.654_{\pm.006}$	$0.709_{\pm.005}$	$0.574_{\pm.010}$	$0.544_{\pm.009}$	$0.565_{\pm.012}$	$0.622_{\pm.015}$
Affine	$0.654_{\pm.003}$	$0.659_{\pm.003}$	$0.682_{\pm.003}$	$0.733_{\pm.003}$	$0.637_{\pm.005}$	$0.605_{\pm.004}$	$0.628_{\pm.005}$	$0.686_{\pm.006}$
Vectorization	$\mathbf{0.668_{\pm.002}}$	$\mathbf{0.674_{\pm.002}}$	$\mathbf{0.697_{\pm.002}}$	$\mathbf{0.747_{\pm.001}}$	$\mathbf{0.643_{\pm.004}}$	$\mathbf{0.613_{\pm.003}}$	$\mathbf{0.635_{\pm.002}}$	$\mathbf{0.694_{\pm.003}}$
Increment (vs. DLA)	6.88%	4.82%	3.87%	3.03%	7.35%	5.69%	4.10%	2.21%
Increment (vs. Affine)	2.14%	2.28%	2.20%	1.91%	0.94%	1.32%	1.11%	1.17%

following formula used by Vardasbi et al. [39]:

$$\epsilon_p^+ = 1 - \frac{p+1}{100}, \quad \epsilon_p^- = \frac{0.65}{p}. \tag{14}$$

Note that trust bias clicks follow 2-dimensional vector-based EH, while real clicks follow 5-dimensional vector-based EH since $\boldsymbol{A}$’s rank is 5.

Baselines. The baselines consist of the state-of-the-art ULTR methods, including RegressionEM [41], DLA [4], PairDebias [20], Affine [39], Labeled Data (uses human-annotated relevance labels to train the ranker directly, which provides an upper bound of ranker) and Click Data (uses the raw click data to train the ranker directly). Note that DLA and RegressionEM assume scalar-based EH. Affine is designed to deal with trust bias, which is a special case of 2-dimensional vector-based EH as we discuss in § 4.4. Training details can be found in Appendix § D.

6.2 Experimental Results

How does our method perform in the real click setting? Table 1 summarizes the results of the performance on the two datasets, where clicks are simulated according to the TianGong-ST real click matrix. Particularly, we have the following findings:

(1) Our method achieves better performance than all the state-of-the-art methods in terms of all measures. It demonstrates that the vector-based EH is more in line with real-world clicks.
(2) Affine works better than all the other baselines that follow the scalar-based examination hypothesis. Therefore, clicks in the real world are somewhat consistent with the trust bias since the trust bias model considers two-dimensional combinations.
(3) The standard deviation of our model is further less than that of Affine, which is less than that of DLA and RegressionEM. It shows that increasing the combination dimension can reduce the variance and obtain a more stable ranking model.

Can our method remove trust bias? Table 2 summarizes the results about the performance when clicks are generated by the trust bias model on Yahoo! dataset. We can observe that the performance

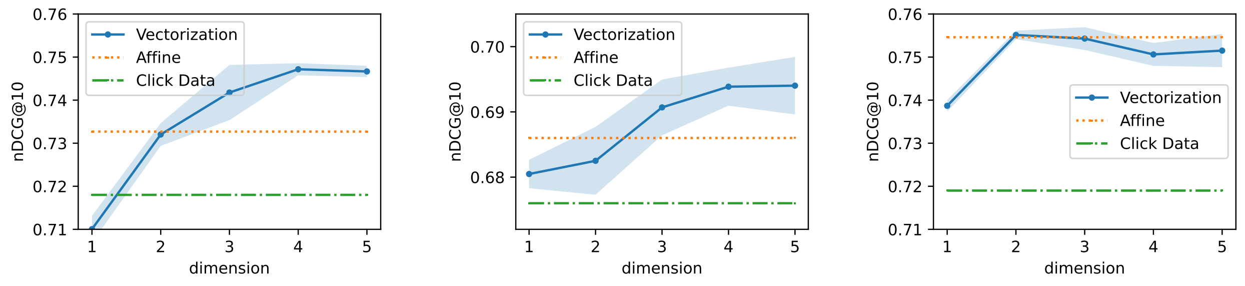
Table 2: Comparison of different algorithms on Yahoo! dataset in the trust bias click setting. Numbers are shown with their standard deviation in 8 repeated runs (i.e., ± x).

Algorithms	Yahoo! nDCG@1	Yahoo! nDCG@5	Yahoo! nDCG@10
Labeled Data	$0.680_{\pm.002}$	$0.711_{\pm.001}$	$0.758_{\pm.001}$
Click Data	$0.615_{\pm.008}$	$0.664_{\pm.003}$	$0.719_{\pm.002}$
DLA	$0.663_{\pm.004}$	$0.702_{\pm.002}$	$0.751_{\pm.001}$
PairDebias	$0.624_{\pm.005}$	$0.668_{\pm.001}$	$0.723_{\pm.001}$
RegressionEM	$0.646_{\pm.008}$	$0.686_{\pm.003}$	$0.737_{\pm.002}$
Affine	$0.678_{\pm.003}$	$\mathbf{0.707_{\pm.001}}$	$\mathbf{0.755_{\pm.001}}$
Vectorization	$\mathbf{0.679_{\pm.003}}$	$\mathbf{0.707_{\pm.001}}$	$\mathbf{0.755_{\pm.001}}$
Increment (vs. DLA)	2.41%	0.71%	0.53%
Increment (vs. Affine)	0.15%	0.00%	0.00%

of our model is fairly close to that of Affine, which shows the ability of our method for removing trust bias, and verifies its efficiency in finding the base vector. Both our method and Affine outperform the EH-based algorithms, which proves once again the limitation of the scalar-based EH. Besides, compared to Table 1, the performance of our method training on trust bias clicks is better than that of training on the real-world clicks. One explanation is that the trust bias model follows Eq.(4): the click rate relative order will not change with the position, which benefits the finding of base vectors. However, the real clicks are complicated and do not follow this assumption. There is still room for debiasing real clicks.

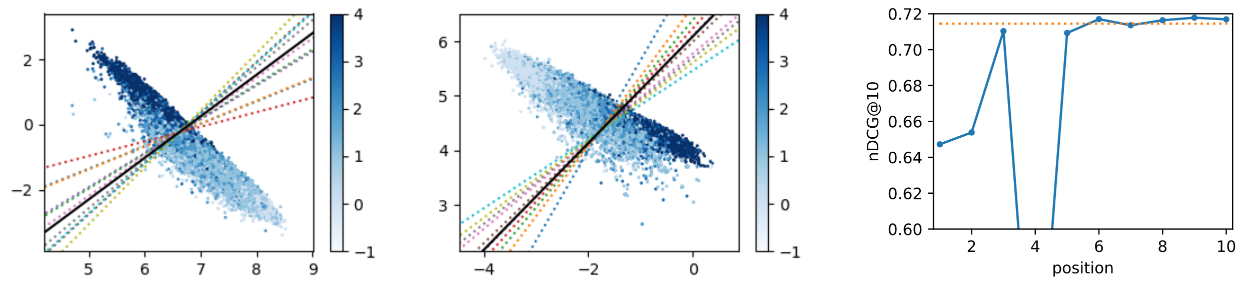
How many dimensions does the model need? We further tune the dimension in our method to verify its impact on performance.

- Figure 4(a) and 4(b) demonstrates the results in the real click setting. We can see that the model performance rises as the dimension increases, which shows that a higher dimension can better capture complex click patterns. When the dimension is



(a) real click (Yahoo!) (b) real click (Istella-S) (c) trust bias (Yahoo!)

Figure 4: Performance changes after adjusting the dimension in Vectorization, where the figure titles are in the form of "click setting (dataset)". The variance is displayed with the shadow areas.



(a) real click (Yahoo!) (b) trust bias (Yahoo!) (c) real click (Yahoo!)

Figure 5: (a)(b) Visualization of the embedding space. Points represent relevance embeddings, and their colors represent the relevance level. Directions of observation embeddings for each position are marked as colorful dotted lines, and the average directions of base vectors are marked as black solid lines. (c) Performance when projecting relevance embeddings in Figure 5(a) onto observation embeddings for each position (solid line), and when projecting the one onto base vectors (dotted line).

2, the performance of our method is similar to Affine, which is consistent with the fact that trust bias is a special case of 2-dimensional vector-based EH.

- Figure 4(c) demonstrates the results in the trust bias click setting. The best dimension is 2, which shows that a dimension that matches real clicks can get the best performance. Besides, increasing the dimension will not significantly degrade performance.

What does our Vectorization model learn? For convenience of visualization, we set the dimension to 2 and drawed all the relevance embeddings in Figure 5(a) and 5(b). The direction of observation embeddings and the average direction of base vectors are drawn as a straight line. We discuss more property of this embedding space in Appendix § E.

What if we choose other observation embeddings as the base vector? Since there are only ten positions, we can choose observation embeddings for each position as the base vector to see the ranking performance. Figure 5(c) shows the result based on Figure 5(a). We can see that the result projected onto the observation embedding changes drastically with the change of position, especially at the higher-rank position. As a comparison, the result of projecting them onto the base vector has always been well. It shows that our method of choosing the base vector is robust.

How much extra time will be introduced in the inference phase? Table 3 shows the inference time cost comparison of our Vectorization with the conventional scalar-based ranker used by baselines. The inference is 21.4% slower because of the more complicated model structure. Fortunately, since the base model and the ranker can run in parallel, we claim that this gap can be further narrowed through a more refined concurrency programming.

Table 3: Time spent sorting a batch (=256) of query lists. Each experiment was repeated for 3,000 times on a Tesla K80 GPU.

Vectorization	Scalar-based Ranker	Increment
0.136 s	0.112 s	+21.4%

7 CONCLUSION AND FUTURE WORK

Conclusions. In this work, we take the first step to studying the limitation of the scalar-based examination hypothesis (EH). To better catch the complicated click pattern in the real world, we propose vector-based EH, in which observation and relevance interact in a higher-dimensional embedding space. Its universality can be justified, which shows that this hypothesis is complete. To rank documents with relevance embeddings in the inference stage, we propose to find the most probable observation embedding that ever appears with the given features in the training dataset as the base vector, and project relevance embeddings onto it. We obtain a close form of the base vector by modeling the observation embedding distribution as Gaussian distribution. We propose a new method to simulate more realistic clicks for testing. Extensive experiments showed that our Vectorization significantly outperforms the state-of-the-art ULTR methods on complex real clicks as well as simple simulated clicks.

Future work. In future work, it would be interesting to investigate better solutions to sort relevance embeddings. Besides, how to directly optimize a certain ranking metric (e.g., nDCG) with the vector-based EH is also an open question.

ACKNOWLEDGMENTS

Thanks to Yusu Hong for proofreading the article, and the reviewers for their valuable comments and suggestions.

REFERENCES

[1] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. 2019. A general framework for counterfactual learning-to-rank. In *SIGIR 2019*. 5–14.

[2] Aman Agarwal, Xuanhui Wang, Cheng Li, Michael Bendersky, and Marc Najork. 2019. Addressing trust bias for unbiased learning-to-rank. In *TheWebConf 2019*. 4–14.

[3] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. 2019. Estimating position bias without intrusive interventions. In *WSDM 2019*. 474–482.

[4] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. 2018. Unbiased learning to rank with unbiased propensity estimation. In *SIGIR 2018*. 385–394.

[5] Qingyao Ai, Jiaxin Mao, Yiqun Liu, and W. Bruce Croft. 2018. Unbiased Learning to Rank: Theory and Practice. In *CIKM 2018* (Torino, Italy). ACM, New York, NY, USA, 2305–2306. https://doi.org/10.1145/3269206.3274274

[6] Qingyao Ai, Tao Yang, Huazheng Wang, and Jiaxin Mao. 2021. Unbiased Learning to Rank: Online or Offline? *TOIS* 39, 2 (2021), 1–29.

[7] Alexey Borisov, Ilya Markov, Maarten De Rijke, and Pavel Serdyukov. 2016. A neural click model for web search. In *WWW 2016*. 531–541.

[8] Alexey Borisov, Martijn Wardenaar, Ilya Markov, and Maarten de Rijke. 2018. A click sequence model for web search. In *SIGIR 2018*. 45–54.

[9] Georg Buscher, Susan T Dumais, and Edward Cutrell. 2010. The good, the bad, and the random: an eye-tracking study of ad quality in web search. In *SIGIR 2010*. 42–49.

[10] Hui Cai, Chengyu Wang, and Xiaofeng He. 2020. Debiasing Learning to Rank Models with Generative Adversarial Networks. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*. Springer, 45–60.

[11] Olivier Chapelle and Yi Chang. 2011. Yahoo! learning to rank challenge overview. In *Proceedings of the learning to rank challenge*. PMLR, 1–24.

[12] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. 2009. Expected reciprocal rank for graded relevance. In *CIKM 2009*. 621–630.

[13] Jia Chen, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2019. TianGong-ST: A New Dataset with Large-scale Refined Real-world Web Search Sessions. In *CIKM 2019*. ACM.

[14] Mouxiang Chen, Chenghao Liu, Jianling Sun, and Steven CH Hoi. 2021. Adapting Interactional Observation Embedding for Counterfactual Learning to Rank. In *SIGIR 2021*. 285–294.

[15] Georges E Dupret and Benjamin Piwowarski. 2008. A user browsing model to predict search engine click data from past observations.. In *SIGIR 2008*. 331–338.

[16] Zhichong Fang, Aman Agarwal, and Thorsten Joachims. 2019. Intervention harvesting for context-dependent examination-bias estimation. In *SIGIR 2019*. 825–834.

[17] Fan Guo, Chao Liu, Anitha Kannan, Tom Minka, Michael Taylor, Yi-Min Wang, and Christos Faloutsos. 2009. Click chain model in web search. In *WWW 2009*. 11–20.

[18] Fan Guo, Chao Liu, and Yi Min Wang. 2009. Efficient multiple-click models in web search. In *WSDM 2009*. 124–131.

[19] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR 2016*. 549–558.

[20] Ziniu Hu, Yang Wang, Qu Peng, and Hang Li. 2019. Unbiased LambdaMART: An unbiased pairwise learning-to-rank algorithm. In *TheWebConf 2019*. 2830–2836.

[21] Samuel Ieong, Nina Mishra, Eldar Sadikov, and Li Zhang. 2012. Domain bias in web search. In *WSDM 2012*. 413–422.

[22] Jiarui Jin, Yuchen Fang, Weinan Zhang, Kan Ren, Guorui Zhou, Jian Xu, Yong Yu, Jun Wang, Xiaoqiang Zhu, and Kun Gai. 2020. A deep recurrent survival model for unbiased ranking. In *SIGIR 2020*. 29–38.

[23] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *KDD 2006*. 217–226.

[24] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR 2005*. 154–161.

[25] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay. 2007. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *TOIS* 25, 2 (2007), 7–es.

[26] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *WSDM 2017*. 781–789.

[27] Jean Kaddour, Yuchen Zhu, Qi Liu, Matt J Kusner, and Ricardo Silva. 2021. Causal Effect Inference for Structured Treatments. *NeurIPS 2021* 34 (2021).

[28] Alex Kendall and Yarin Gal. 2017. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? *NIPS 2017* 30 (2017), 5574–5584.

[29] Yehuda Koren, Steffen Rendle, and Robert Bell. 2022. Advances in collaborative filtering. *Recommender systems handbook* (2022), 91–142.

[30] Chenghao Liu, Tao Jin, Steven CH Hoi, Peilin Zhao, and Jianling Sun. 2017. Collaborative topic regression for online recommender systems: an online and Bayesian approach. *Machine Learning* 106, 5 (2017), 651–670.

[31] Yiqun Liu, Chao Wang, Ke Zhou, Jianyun Nie, Min Zhang, and Shaoping Ma. 2014. From skimming to reading: A two-stage examination model for web search. In *CIKM 2014*. 849–858.

[32] Zeyang Liu, Yiqun Liu, Ke Zhou, Min Zhang, and Shaoping Ma. 2015. Influence of vertical result in web search examination. In *SIGIR 2015*. 193–202.

[33] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. 2016. Post-learning optimization of tree ensembles for efficient ranking. In *SIGIR 2016*. 949–952.

[34] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic matrix factorization. *NIPS 2007* 20 (2007).

[35] Zohreh Ovaisi, Ragib Ahsan, Yifan Zhang, Kathryn Vasilaky, and Elena Zheleva. 2020. Correcting for selection bias in learning-to-rank systems. In *TheWebConf 2020*. 1863–1873.

[36] Yingcheng Sun, Richard Kolacinski, and Kenneth Loparo. 2020. Eliminating search intent bias in learning to rank. In *ICSC*. IEEE, 108–115.

[37] Mucun Tian, Chun Guo, Vito Ostuni, and Zhen Zhu. 2020. Counterfactual Learning to Rank using Heterogeneous Treatment Effect Estimation. *arXiv:2007.09798* (2020).

[38] Ali Vardasbi, Maarten de Rijke, and Ilya Markov. 2020. Cascade Model-based Propensity Estimation for Counterfactual Learning to Rank. *SIGIR 2020* (Jul 2020). https://doi.org/10.1145/3397271.3401299

[39] Ali Vardasbi, Harrie Oosterhuis, and Maarten de Rijke. 2020. When Inverse Propensity Scoring does not Work: Affine Corrections for Unbiased Learning to Rank. In *CIKM 2020*. 1475–1484.

[40] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to rank with selection bias in personal search. In *SIGIR 2016*. 115–124.

[41] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position bias estimation for unbiased learning to rank in personal search. In *WSDM 2018*. 610–618.

[42] Xin Wang, Steven CH Hoi, Chenghao Liu, and Martin Ester. 2017. Interactive social recommendation. In *CIKM 2017*. 357–366.

[43] Kyle Williams, Julia Kiseleva, Aidan C Crook, Imed Zitouni, Ahmed Hassan Awadallah, and Madian Khabsa. 2016. Detecting good abandonment in mobile search. In *WWW 2016*. 495–505.

[44] Yukun Zheng, Jiaxin Mao, Yiqun Liu, Cheng Luo, Min Zhang, and Shaoping Ma. 2019. Constructing click model for mobile search with viewport time. *TOIS* 37, 4 (2019), 1–34.

[45] Hao Zou, Peng Cui, Bo Li, Zheyan Shen, Jianxin Ma, Hongxia Yang, and Yue He. 2020. Counterfactual Prediction for Bundle Treatment. In *NeurIPS 2020*.

APPENDIX

A Universality of product effect

We prove that any bounded continuous function on $\mathcal{X} \times \mathcal{T}$ can be approximated with the dot-product of two vector functions on $\mathcal{X}$ and $\mathcal{T}$ respectively:

Theorem 1. *Let $\mathcal{H}_{\mathcal{X}\times\mathcal{T}}$ be a Reproducing Kernel Hilbert Space (RKHS) on the set $\mathcal{X} \times \mathcal{T}$ with universal kernel k. For any $\delta > 0$, and any $f \in \mathcal{H}_{\mathcal{X}\times\mathcal{T}}$, there is a $d \in \mathbb{N}$ such that there exist two d-dimensional vector fields $g : \mathcal{X} \to \mathbb{R}^d$ and $h : \mathcal{T} \to \mathbb{R}^d$, where $||f - g^\top h||_{L_2(P_{\mathcal{X}\times\mathcal{T}})} \leq \delta$.*

Proof. The proof can be found in Proposition 1 of [27]. □

B Algorithm

Training stage. We illustrate the model training of Vectorization in Alg. 1. In line 1, we initialize all the parameters. In lines 2-7, we jointly train the relevance model and the observation model through vector-based EH to make their dot product close to the clicks. In lines 8-12, we train the base model, to let the distribution estimation close to the observation embedding distribution.

Algorithm 1: Model Training for Vectorization

```
   Input: dataset $\{\mathcal{D}_q\}_{q\in Q}$, learning rate $\alpha$, $\lambda$
   Output: model parameters $\theta_r$, $\theta_o$, $\theta_v$
1  Initialize $\theta_r$, $\theta_o$ and $\theta_v$;
   /* Stage 1.                                        */
2  repeat
3      sample a query $q \in Q$, $\{(\mathbf{x}_i, \mathbf{t}_i, c_i)\}_{i=1}^n \leftarrow \mathcal{D}_q$;
4      compute $L_q^{click}$ with Eq.(9);
5      $\theta_r \leftarrow \theta_r - \alpha \cdot \partial L_q^{click} / \partial\theta_r$;
6      $\theta_o \leftarrow \theta_o - \alpha \cdot \partial L_q^{click} / \partial\theta_o$;
7  until $\theta_r$ and $\theta_o$ convergence;
   /* Stage 2.                                        */
8  repeat
9      sample a query $q \in Q$, $\{(\mathbf{x}_i, \mathbf{t}_i, c_i)\}_{i=1}^n \leftarrow \mathcal{D}_q$;
10     compute $L_q^{base}$ with Eq.(10);
11     $\theta_v \leftarrow \theta_v - \alpha \cdot \partial L_q^{base} / \partial\theta_v$;
12 until $\theta_v$ convergence;
13 return $\theta_r$, $\theta_o$, $\theta_v$
```

Inference stage. The overview of the algorithm in the inference stage is summarized in Alg. 2. In lines 1-2, we compute the observation embedding distributions for ranking features. In line 3, we compute the base vector. In lines 4-5, we project the relevance embeddings onto the base vector to obtain ranking scores.

C Further details of datasets

Table 4 shows the characteristics of the two datasets, Yahoo! and Istella-S, in addition to the TianGong-ST which we used to estimate the real click rate function.

Algorithm 2: Model Inference for Vectorization

```
  Input: a set of ranking features $\{\mathbf{x}_i\}_{i=1}^n$ related to a query,
         model parameters $\theta_r$ and $\theta_v$
  Output: ranking scores $r(\mathbf{x}_1), r(\mathbf{x}_2), \cdots, r(\mathbf{x}_n)$
1 for $i = 1$ to $n$ do
2     $\boldsymbol{\mu}(\mathbf{x}_i), \boldsymbol{\sigma}^2(\mathbf{x}_i) \leftarrow \mathbf{v}(\mathbf{x}_i; \theta_v)$;
3 compute $\widetilde{\mathbf{o}_q}$ with Eq.(8);
4 for $i = 1$ to $n$ do
5     $r(\mathbf{x}_i) \leftarrow \mathbf{r}(\mathbf{x}_i; \theta_r)^\top \widetilde{\mathbf{o}_q}$;
6 return $r(\mathbf{x}_1), r(\mathbf{x}_2), \cdots, r(\mathbf{x}_n)$
```

Table 4: Dataset statistics

	Yahoo!	Istella-S	TianGong-ST
queries	28,719	32,968	3,449
documents	700,153	3,406,167	333,813
features	700	220	33
relevance levels	5	5	5
avg. documents / query	24.38	103.32	96.79

D Training details

Similar to [4, 39], we trained an MLP as our ranking model, where the implementation was the same as ULTRA framework [5, 6]. To make fair comparisons, all the baselines and our model shared the same number of hidden units. The only difference in the ranking model between our model and baselines was the output dimensions. For our method, λ was set to be 0.001, the learning rate was tuned to be 0.05, and the dimension was selected from $\{1, 2, 3, 4, 5\}$. We used two layers with sizes $\{256, 64\}$ and *elu* activation as the implementation of the base model. We trained all these methods with a batch size of 256 and used AdaGrad to train the models. To ensure convergence, we train 15,000 epochs on the Yahoo! dataset, and 30,000 epochs on the Istella-S dataset. We used nDCG@k ($k = 1, 3, 5, 10$) as the performance metrics, and run each experiment for 8 times. We adopted the model with the best results based on nDCG@10 tested on the validation set and reported the average results testing on the test set.

E Discussion of relevance embedding

From Figure 5(a) and 5(b), we can see that the closer the true relevance is, the smaller the distance between them in the relevance embedding space. This is a good property since the distance relation can be kept approximately after linear transformation. That is to say, if we project them onto scalars, the close relevance embeddings should also have a close scalar distance, even if the base vector is inaccurate. It shows the robustness of the relevance embedding space.

Besides, one can observe that these relevance embeddings have a linear style. Why these points have this pattern is an open question. We tried to use PCA to find this direction and projected them onto it to see the performance. However, this method was not better than the method mentioned in this paper. This feature may be useful for finding better sorting methods in the future.