Adversarial Factorization Autoencoder for Look-alike Modeling

Khoa D. Doan
Virginia Tech
Arlington, Virginia
khoadoan@vt.edu

Pranjul Yadav
Criteo AI Labs
Palo Alto, California
p.yadav@criteo.com

Chandan K. Reddy
Virginia Tech
Arlington, Virginia
reddy@cs.vt.edu

ABSTRACT

Digital advertising is performed in multiple ways, for e.g., contextual, display-based and search-based advertising. Across these avenues, the primary goal of the advertiser is to maximize the return on investment. To realize this, the advertiser often aims to target the advertisements towards a targeted set of audience as this set has a high likelihood to respond positively towards the advertisements. One such form of tailored and personalized, targeted advertising is known as look-alike modeling, where the advertiser provides a set of seed users and expects the machine learning model to identify a new set of users such that the newly identified set is similar to the seed-set with respect to the online purchasing activity. Existing look-alike modeling techniques (i.e., similarity-based and regression-based) suffer from serious limitations due to the implicit constraints induced during modeling. In addition, the high-dimensional and sparse nature of the advertising data increases the complexity. To overcome these limitations, in this paper, we propose a novel Adversarial Factorization Autoencoder that can efficiently learn a binary mapping from sparse, high-dimensional data to a binary address space through the use of an adversarial training procedure. We demonstrate the effectiveness of our proposed approach on a dataset obtained from a real-world setting and also systematically compare the performance of our proposed approach with existing look-alike modeling baselines.

CCS CONCEPTS

• **Information systems** → **Computational advertising**; *Content analysis and feature selection*; Information retrieval; • **Computing methodologies** → **Neural networks**; Unsupervised learning; • **Theory of computation** → **Adversarial learning**.

KEYWORDS

Look-alike modeling, hashing, autoencoder, adversarial training, deep learning, factorization.

ACM Reference Format:
Khoa D. Doan, Pranjul Yadav, and Chandan K. Reddy. 2019. Adversarial Factorization Autoencoder for Look-alike Modeling. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM'19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357384.3357807

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357807

1 INTRODUCTION

To maximize an advertising effort, advertisers execute their campaigns in multiple ways, i.e. digital advertising, search-based advertising and contextual advertising. The major aim of the advertisers across these avenues is to maximize the return on their investment. To realize this, the advertiser often aims to *target the advertisements* towards a *targeted set of audiences* as this set has a high likelihood to respond positively towards the advertisements. One such form of tailored and personalized targeted advertising is known as Look-alike Modeling.

Specifically, look-alike modeling, also known as "audience expansion" in the literature, is the task of finding new and relevant users given an existing, often much smaller set (known as seed-set) of users under the assumption that the newly identified look-alike users are similar to the seed-set's users with regards to the activity over the internet (e.g., click, sale or browsing). Computational look-alike modeling is a fundamental and challenging problem in the advertising domain for which various techniques have been proposed [12–14, 17, 22]. The nature of the advertising data, coupled with constraints induced when modeling look-alike approaches have been a major challenge in the development of robust and sophisticated techniques. In particular, digital advertising data is a collection of information across sources (e.g., user interest, product information, engagement information), thereby making advertising data highly complex and multi-dimensional in nature. In computational advertising, users are often represented by hundreds to thousands, if not more, of categorical and continuous raw features. This often leads to the issue of high-dimensionality (millions of features) when interactions between the raw features or categorization of the raw features are taken into account. Another challenge that arises in modeling advertising data revolves around sparsity. This arises as advertisers collect users' data only when the users are displayed with some kind of product-related advertisements, thus the collected data results in a significant amount of examples where the values are absent (or missing features). For example, a user is not shown advertisements for all the products, but is typically only shown advertisements of products for which the user might have a propensity for a favourable outcome.

Additionally, it is critical for computational look-alike techniques to be efficient, especially in programmatic advertising platforms where there can be hundreds (or thousands) of campaigns that are executed daily. Each campaign requires a different set of look-alike users from the campaign's different, initial seed-set, which potentially involves the creation of many look-alike models on high-dimensional advertising data. For such reasons, despite the improved performance, *regression-based look-alike methods* [17], which separately model the membership probability of a user to the seed-set of a campaign using classification-based methods, are often too computationally expensive to be employed. This is because for

each campaign, a classification model is created on a carefully constructed labeled dataset whose positive examples are users from the seed-set. When there are many campaigns, this process is prohibitive, especially when the data is highly sparse and high-dimensional. In addition, when sufficient amount of training data is not available on a newly created campaign, regression-based methods do not perform well; this issue is also known as the "cold-start" problem.

Similarity-based look-alike methods [12, 14, 15], which involve modeling the pair-wise similarity functions between a pair of users, are preferred because they can be scaled out to millions of users and they do not require creating different models for each campaign. Similarity-based methods include the current state-of-the-art research methods employed Locality Sensitive Hashing (LSH) [12, 14, 15]. However, LSH-based techniques are known to ignore some of the vital properties of the data (for e.g., they pay little attention to the distribution of the data during the modeling process) and have been shown to be less accurate and slower than data-dependent hashing schemes such as Semantic Hashing [19]. Replacing LSH-based techniques with semantic hashing based techniques is not straightforward, as Semantic Hashing does not work well on sparse, high-dimensional data and typically produces much inferior results in the presence of complex feature interactions. In addition, to motivate the hidden sigmoid-neuron activations more often near 0 and 1, the learning process relies on employing training tricks, such as adding a deterministic Gaussian noise, with additional hyper-parameters that increase the complexity of using semantic hashing in a real-world environment. To address the aforementioned challenges, in this paper, we propose a novel Adversarial Factorization Binary Autoencoder that can efficiently learn a mapping from sparse, high-dimensional data to a binary address space through the use of an adversarial training procedure. Our model, which is a "hashing" based approach [23], learns compact, storage-efficient binary codes. The cost of finding look-alike users in the original high-dimensional space is reduced to Hamming-distance calculations (using bit-wise XOR operation which takes only one CPU instruction) in the binary address space (in which a data point's representation requires only 4 to 8 bytes of storage). The main contributions of our work are as follows:

- We propose a novel autoencoder based look-alike model that employs adversarial training procedure to systematically "encourage" the hidden-code layer to be binary. Our method introduces an entirely new and efficient alternative to train binary neurons, which can be employed in other problems that require conditional computations [1].
- We extend the semantic hashing method to be able to efficiently handle extremely-sparse data by modeling an additional feature embedding layer and embedding interaction layer that can automatically learn higher-order feature interactions in an *end-to-end*, completely *unsupervised* manner.
- We demonstrate the effectiveness of our proposed look-alike method on a large real-world advertising data and show both quantitative and qualitative results of the performance.
- The strong offline performance of our proposed model across different partners demonstrate the practical applicability of our approach and thus will motivate advertisers to opt for a targeted advertisement (i.e., look-alike) as compared to a generic re-targeting based advertising.

This paper is organized is as follows. We discuss the related works in Section 2. In Section 3, we present information about the data generation process, feature engineering along with some basic data statistics. In Section 4, we describe the background, problem formulation and our proposed approach. In Section 5, we describe our experimental procedure, baseline methods and results. Lastly, in Section 6, we present our conclusion and future work.

2 RELATED WORKS

In this section, we will describe three research directions that are closely related to the proposed work, namely, look-alike modeling, autoencoders and high-dimensional sparsity.

2.1 Look-alike Modeling

Look-alike modeling is the task of reaching new and relevant users in an advertising campaign by extending an existing, smaller set of users (also known as the seed-set). Typically, this extension process can be accomplished by finding "similar" users to those within the seed-set [13–15, 17, 22]. These approaches can be classified broadly into two categories:

(1) *Regression-based look-alike models*: The simplest approach is to predict the class-membership of a user belonging to the seed-set. Given a user with a set of features x, these models compute the propensity score $p(x \in S|x)$ – the probability of the user belonging to the seed-set S – which can be modeled using the sigmoid function $\frac{1}{1+\exp(-\theta^T x)}$ [17]. **Density estimation** approaches can be used to directly estimate $p(x \in S|x)$, but this is generally very difficult both because of the inherent challenges of density estimation and a higher chance of over-fitting when the size of the seed-sets is small. A more popular regression-based look-alike modeling approach is **classification** where negative samples (of the negative class in training the propensity model) can be randomly sampled from all the available users (minus the seed-set). Random negative samples have the following effect: features that correlate with the advertising conversions (such as click-through rate or product purchases) play less important role in similar-user selection. This motivates the sampling of previously exposed (having seen the campaign's advertisements) but non-converted users. However, doing this will lead to the cold-start problem; for example, on new advertising campaigns where there have not been sufficient number of previously exposed users. In site of this problem, there are a couple of primary advantages of using regression based models. First, using such approaches, the decision to select the extended users is compressed within a model. Second, the performance of regression-based techniques is often significantly better when there is enough, well-constructed supervised training data.

(2) *Similarity-based look-alike models*: Instead of learning a class-membership model, using a similarity measures such as Cosine or Jaccard [11], similarity-based systems find look-alike users by directly comparing all possible pairs between seed users and available users using a "pre-defined" pairwise similarity function [12, 14, 15]. A seed-set can be extended by selecting top

users from a ranked list of users which is ordered based on their maximum similarity scores to any users in the seed-set. These systems can be scaled out to millions of users using **hashing methods** [23] such as Locality Sensitive Hashing (LSH) [11], but the performance of such systems depend greatly on selecting the right features and similarity functions. Often, they are chosen independent of the data (for e.g., using random projection) and hence, important intrinsic properties of the data are completely ignored.

A key advantage of similarity-based approaches is the fact that its learning process is independent of the number of seed-sets. An implementation of regression-based models requires learning one model for each individual seed-set, which significantly increases the training time when there are thousands of such models to be trained on a regular basis (for e.g., daily), which is a common scenario for many advertisers. Furthermore, extending the seed-sets require a full linear-scan of the users since the model needs to calculate the propensity score for each user in order to rank them. Conversely, similarity-based approaches focus on learning only one desired similarity function, irrespective of the number of seed-sets. In addition, because of the advantage of hashing, these approaches can also limit the retrieval to a small list of candidate users [11]. However, because of the performance superiority of regression-based models, most production look-alike modeling systems either employ regression-based models or a hybrid version of regression-based and hashing models (to reduce its learning/training time while preserving the overall performance). In this paper, we focus on a similarity-based approach (specifically hashing) that automatically learns an efficient similarity function in an end-to-end manner directly from the raw data and demonstrates significant performance improvements compared to both existing regression-based and similarity-based approaches.

2.2 Autoencoders and Adversarial Training

2.2.1 Autoencoders. Most of the widely used similarity-based look-alike models employ variants of LSH because such models do not require pre-training and are computationally efficient to deploy in a real production environment [12, 14, 15]. Despite their efficiency, most LSH approaches are data-independent because their hash functions employ random projection, which makes them inappropriate for complex scenarios, especially for high-dimensional sparse data.

Deep neural networks have the ability to discover good representations from the raw input data [10]. The authors of [19] proposed an alternative to the LSH-based approach for approximate similarity search with a pre-defined similarity function by learning a binary vector representation of the input data using autoencoders [10]. The binary vector captures the semantic dependency of the input data in a lower-dimensional space. The advantage of semantic hashing is that, when more hidden layers are used, the network can capture complex interactions among the input features. In addition, finding similar items, after the model is trained, is experimentally shown to be faster while achieving better accuracy compared to LSH-based approach for document retrieval tasks in the text domain [19].

2.2.2 Generative Adversarial Network (GAN). GAN models have recently gained popularity due to their generative power [7]. One important feature of GAN is the adversarial training procedure which is able to implicitly "match" an output of a deep network to a pre-defined distribution. In fact, the adversarial training mechanism has been used in several works as a regularization of the latent representation of autoencoders [9, 16]. This improves the quality of the projected, low-dimensional manifolds that the autoencoders are able to learn, especially when the high-dimensional input data has a complex structure [9].

Regularizing the autoencoders with an adversarial procedure allows our model to learn an effective representation of the complex, high-dimensional sparse input data while ensuring the latent representation to implicitly match with a pre-defined binary-like distribution. The latter contribution encourages the bottleneck layer of the autoencoder to generate efficient hash codes (similar to that of LSH) but stays closer to the manifold of the original input data.

2.3 High-dimensional Sparsity

Advertising data is high-dimensional in nature with many missing feature values and hence, most of the standard machine learning models are prone to the problem of over-fitting. Traditionally, manual feature engineering is used to improve machine learning models' performance but such solutions fail to scale well for higher order feature interactions where the number of features grows exponentially (consequently making the features even sparser). Feature embedding techniques, including both factorization and neural-network models, solve this problem by learning feature interaction directly from raw data [8, 18]. For example, Factorization Machines (FM) [18] model interactions between each pair of features as a dot product between two low-dimensional embedding vectors of the features.

However, to the best of our knowledge, this idea has not been applied in the unsupervised domain, specifically in autoencoders which learn low-dimensional representation of high-dimensional sparse input with complex feature interaction. In the presence of sparse data, the autoencoder's reconstruction output loses its sparsity and easily collapses to the average of the input features [6]. Simply designing autoencoders to be wider and deeper does not help because with too much capacity, autoencoders can learn to "copy" the input without finding a good latent, low-dimensional representation of the data. We propose to tackle the high-dimensionality and sparsity challenges by fusing factorized models with autoencoders, regularized by an adversarial learning procedure in order to find an efficient latent representation of the data.

3 CRITEO DATA

3.1 Data Generation Process

Criteo Look-alike Modeling (CLM) is a recent product from Criteo that was launched at the start of 2018 on selected markets and partners. In CLM, partners (i.e., advertisers selling products) provide a selected set of users (i.e., seed-set) and then the goal of CLM is to identify a set of look-alike users (i.e., users who are not in the seed-set, but are similar to the seed-set of users with respect to their online behaviour) from the pool of available users. After the identification of look-alike users, Criteo uses its re-targeting technology to display partner-related advertisements to the look-alike users across publisher websites.

CLM combines different types of data to train machine learning models for identifying look-alike users. CLM collects data about

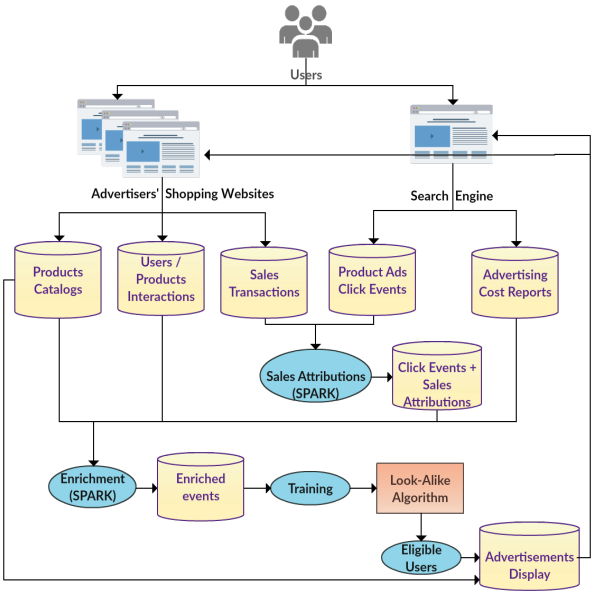


Figure 1: Data collection and model training Pipeline for the Criteo Look-alike Modeling product.

users interacting with product related advertisements primarily from advertisers in an anonymous setting. Also, it collects additional detailed data related to products' baskets, products' sale transactions and clicks of the relevant product advertisements. Figure 1 represents the entire end-to-end pipeline of the large-scale data collection process, model training and advertisement display methodology for CLM.

3.2 Feature Engineering

Each user is represented by an aggregated set of online activities that are performed across publishers and products. In this paper, we only consider three major action types for the user (i.e., click on the product advertisement, addition of product to the basket and sale of the product). Furthermore, to overcome the high-dimensionality associated with billions of products that are available in Criteo catalog, we use the hierarchical taxonomy from Google [1] to map multiple products to a single relevant category. Hence, Billions of products in Criteo catalog are mapped to approximately thousands of product categories. In summary, every user is represented using categorical representation across the kind of interaction (i.e., click, basket or sale) and product category. For the sake of replicability of our experiments and results, we have already released the data that is used in this work.[2]

3.3 Data Statistics

The data setting for CLM is as follows: each partner provides a seed-set consisting of approximately a few thousand users. CLM, then, uses its robust ML technology to identify look-alike users (roughly 10 times) who are similar to that of the seed-set users from a pool of billions of users. However, considering the fact that a partner is often interested in a specific geography (i.e., Europe, North America or Asia-Pacific), the pool of users often is limited to hundreds of millions of potential users (instead of billions of users).

Among the hundreds of millions of potential users, only a few millions will have a significant online activity. Now we will present some data statistics, provided in Table 1. The data contains information about six partners, i.e., the size of the seed-set provided by the partner and the size of the non-seed users available in the data after multiple rounds of filtering. The partners are selected from broad consumer segments, thus bringing significant diversity to the partners' datasets and the numbers of users in the seed-sets.

[1] https://www.google.com/basepages/producttype/taxonomy.en-US.txt
[2] https://s3-us-west-1.amazonaws.com/criteo-lookalike-dataset/data.zip

Table 1: Statistics of the partner's datasets.

Partner	Consumer Segment	Seeds	Non-seeds
Apparel-1	*Apparel & Accessories*	18,898	3,038,404
Electronics-1	*Electronics*	398	3,030,559
Home/Garden	*Home & Garden*	34,523	3,020,830
Electronics-2	*Electronics*	114	3,047,652
Animal/Pet	*Animals & Pet-supplies*	3,544	3,047,785
Apparel-2	*Apparel & Accessories*	75,646	3,009,994

4 THE PROPOSED LOOK-ALIKE MODEL

In this section, we will first define the problem statement. We will then describe our proposed framework.

4.1 Problem Formulation

Given a seed-set of N users $X_o = \{x^{(1)}, x^{(2)}, ..., x^{(N)}\}$, the goal of look-alike modeling is to extend X_o with M new users X_e such that the extended users X_e are similar to X_o with respect to their online internet shopping behaviour. Each user $x^{(i)}$ is represented by a sparse vector in $\mathbb{R}^n$. We will now discuss various components of our proposed framework for the look-alike modeling problem. The notations used in our paper are given in Table 2.

Table 2: Notations used in our paper.

Notation	Description
$x, \tilde{x}, \hat{x}$	input vector, bi-interaction embedding vector, and reconstructed input vector, respectively
x_i	i^{th} input feature
$x^{(i)}$	i^{th} input sample
e_i	embedding vector of feature i
D_x	data distribution
$I(.)$	bi-interaction function whose output is $\tilde{x}$
f, g	encoding and decoding functions
b	low-dimensional, binary vector – output of $f(.)$
$q(b)$	the posterior distribution of b
L_{BA}, L_{FA}	reconstruction cost of the binary autoencoder and factorization binary autoencoder, respectively
D	the discriminator network
L_D, L_G	original loss function for the generator and discriminator, respectively
Θ_G	parameters of the generator, including parameters of both functions $i(.)$ and $f(.)$
Θ_D	parameters of the discriminator $D(.)$

4.2 Binary Autoencoder

Given a dataset $X = \{x|x \sim D_x\}$, a binary autoencoder (BA) defines an encoding function $f : x \rightarrow b$ that maps each data point $x \in \mathbb{R}^n$ into a point $b \in \mathbb{R}^B$ in the coding space and a decoding function $g : b \rightarrow \hat{x}$ that recovers x, as $\hat{x}$, from b. In semantic hashing [19], the RBM-based autoencoder uses a sigmoid encoding layer – or bottleneck layer – and *motivates* each component of b, activations of the bottleneck layer, to take values closer to either 0 or 1. As a result, thresholding the activations b will return a binary code vector $c \in \{0, 1\}^B$ which can be considered as the discrete hash code of the input x.

For high-dimensional, advertising data, it is important that we can exploit the higher-order interaction for better model performance, as discussed in Section 2. In spite of some drawbacks (which will be discussed later in Section 4.3), deep multi-layer encoders can model feature interaction and find a useful discrete, low-dimensional representation of the input. Thus, the binary autoencoder has the following structure:

- *An encoder function* $f : x \rightarrow b$ that maps x into a vector b. In our work, we model f as a multi-layer perceptron (MLP) whose last layer has the sigmoid activation function.
- *A decoder function* $g : b \rightarrow \hat{x}$ that reconstructs the input x as $\hat{x}$. Similar to the encoder, we model the decoder with an MLP.
- Similar to Semantic Hashing [19], we employ fixed, random normal noise to force b's values to be binary (closer to 0 or 1).
- BA learns its parameters by minimizing the following squared-error reconstruction cost:

$$L_{BA} = ||\hat{x} - x||_2^2 = ||g(f(x)) - x||_2^2 \tag{1}$$

4.3 Factorization Binary Autoencoder

Autoencoders are popularly used in learning non-linear data embedding, or data representation. Unfortunately, when the input is high-dimensional and highly sparse, it becomes difficult for the autoencoders to learn a useful representation [6]. Wider and deeper encoders/decoders result in too much model capacity, thus the autoencoders can easily learn to **copy** the input, without extracting any useful information about the data distribution in the low-dimensional latent space [6]. In other words, it is difficult for the binary autoencoders to utilize non-linear, higher-order interactions of the input features to learn a good hash function of the input. Therefore, we propose to model the the binary autoencoder with an explicit feature interaction layer. Inspired by the supervised NFM [8], given an embedding vector $e_i \in \mathbb{R}^e$, where e is the dimension of the embedded space, for each feature i, we model the interaction of features using a bi-interaction function $I(x)$ as follow:

$$I(x) = \tilde{x} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i e_i \odot x_j e_j \tag{2}$$

where $\odot$ is the element-wise multiplication operation of two vectors and the last term can be efficiently calculated using only non-zero feature positions [8]:

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i e_i \odot x_j e_j = \frac{1}{2} \left[(\sum_{i=1}^{n} x_i e_i)^2 - \sum_{i=1}^{n} (x_i e_i)^2 \right] \tag{3}$$

Equation (2) captures higher-order feature interactions in the low-dimensional, discrete representation space because the first term achieves memorization, correlation of the input features, while the second term achieves generalization, transitivity of the features' correlation [3]. In our paper, we model second-order interaction of the input x in the bi-interaction embedding layer, but it is possible to model the higher-order interaction by extending the embedding vectors e_i into higher-dimensional tensors. It is important to note that computation of the high-order, non-linear interaction is linear in the number of non-zero raw input features, which makes it very efficient for high-dimensional, sparse input domains such as computational advertising.

In our proposed factorization binary autoencoder (FA) model, the encoder function becomes $f : \tilde{x} \rightarrow b$. On the other hand, instead of employing the similar decoder function $g : b \rightarrow \hat{x}$ as in Section 4.2, our decoder, instead, learns to reconstruct the bi-interaction embedding input $\tilde{x}$ by minimizing similar squared-error, reconstruction cost as follows:

$$L_{FA} = ||\hat{x} - \tilde{x}||_2^2 = ||g(f(I(x))) - I(x)||_2^2 \tag{4}$$

Reconstructing $\tilde{x}$ instead of x overcomes the difficulties of autoencoders to reconstruct high-dimensional, sparse input. Different from NFM, FA learns to simultaneously find a good interaction representation and a good low-dimensional, discrete representation of the input x because the gradient of the loss function propagates to the embedding space both immediately at the decoder's output and through the autoencoder's network. More importantly, the learning process is performed in a completely unsupervised manner.

4.4 Adversarial Factorization Binary Autoencoder

From an alternative perspective, an autoencoder defines an encoding distribution $q(b|x)$ and a decoding distribution $p(x|b)$. Consequently, the encoding step imposes a posterior distribution on the coding space as follows:

$$q(b) = \int_x q(b|x) D_x(x) dx \tag{5}$$

Our proposed FA model has several network parameters, from both the bi-interaction layer and autoencoder module, that need to be learned. It is previously shown that when the encoder and decoder are too powerful, the autoencoder can get stuck in bad local-minima and perform "copying" the input without learning any useful representation from the data [2, 9]. In order words, the distribution $p(x|b)$ easily collapses to $D_x(x)$, and the encoder maps data points in D_x to discrete codes **scattered** in the latent space, thus the locality information in the original input space are not well preserved in the latent, discrete space [9]. To overcome such limitations, we propose to regularize the discrete, hash layer with a discriminator through an adversarial learning procedure [9]. We also take advantage of the **distribution matching** property of the adversarial network to force the low-dimensional, discrete space $q(b)$ to **match** a binary-like distribution such as a "discrete Bernoulli" prior. If we enforce $q(b|x)$ to "agree" with a distribution that has all desired characteristics of a good hashing function, such as "bit

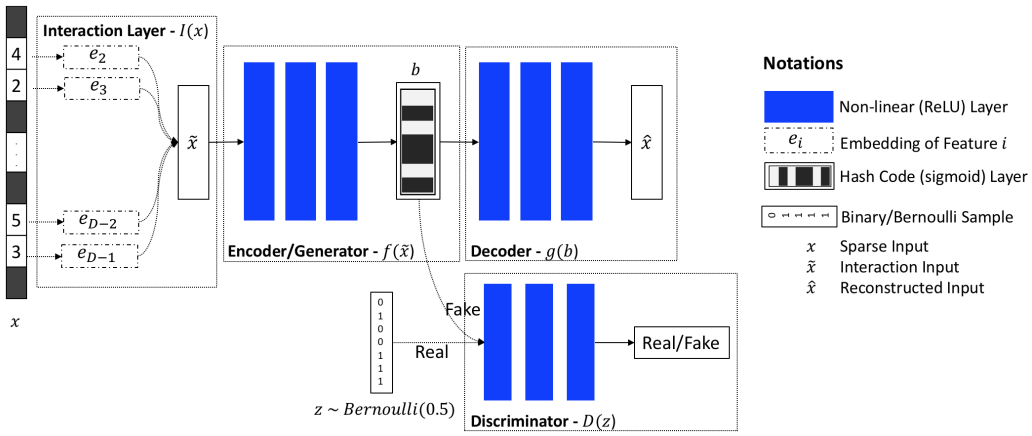


Figure 2: Network architecture of the proposed Adversarial Factorization Binary Autoencoder model.

balance" and "bit uncorrelation" [23], the adversarial training procedure will force the encoder function $f(x)$ to generate binary codes with the same desired characteristics. Specifically, given an input vector $z \sim Bernoulli(p)$, defined as a vector where each of its components z_i is independently drawn from the same underlying Univariate Bernoulli distribution with parameter p, we regularize the posterior distribution $q(b)$ to be similar to $Bernoulli_z(p)$. The adversarial binary autoencoder (ABA) is trained with dual objectives as follows:

(1) Minimize the reconstruction error from the coding space in the decoding step.
(2) Constrain the coding posterior distribution, $q(b)$ to agree with the Bernoulli prior, $Bernoulli_z(p)$, where p is set to 0.5. An intuitive explanation of the regularization is that every dimension of b will learn to divide as optimal as possible the original space into two halves (thus its activation has about 50% chance of being closer to 1 or closer to 0 – a "bit balance" [23]), where the points in each half are more similar to those in the same half compared to those in the other half – "bit uncorrelation" [23].

In the adversarial setting, the autoencoder's encoding module takes the role of a generator which is trained to fool the discriminator. The discriminator tries to accurately divide its input into two classes, real and fake for the Bernoulli samples z and the autoencoder's encoding outputs b, respectively. The solution of this adversarial procedure (or the minimax game) is obtained by solving the following objective function:

$$\begin{aligned}\min_{\Theta_E}\max_{\Theta_D} L(E,D) = \min_{\Theta_E}\max_{\Theta_D}\ & E_{z\sim Bernoulli_z(p)}[\log D(z)] \\ & + E_{x\sim D_x}[\log(1-D(f(I(x))))]\end{aligned} \tag{6}$$

where Θ_G and Θ_D are network parameters of the generator and discriminator, respectively. For an optimal discriminator with parameters Θ_D^*, it can be proven that the training criterion of the generator is as follows:

$$C(G) = -log(4) + 2 * \text{JSD}(Bernoulli_z(p) \,||\, q(b)) \tag{7}$$

where $\text{JSD}(p \,||\, q)$ is the Jensen-Shannon Divergence between distributions p and q. The global minimum of $C(G)$ is achieved when the non-negative, symmetric divergence

$$\text{JSD}(Bernoulli_z(p) \,||\, q(b)) = 0 \tag{8}$$

and its solution is

$$q(b) = Bernoulli(p) \tag{9}$$

In other words, the encoder network perfectly replicates the Bernoulli-data generating process. Since the Bernoulli-data generating process samples each component z_i of z independently (hence, each bit is uncorrelated) from a Univariate Bernoulli distribution with probability 0.5 (hence, each bit has 0.5 probability of being 1), matching $q(b)$ against the $Bernoulli_z(p)$ is exactly equivalent to generating bit-uncorrelated and bit-balance codes. Consequently, our proposed regularized autoencoder learns an effective and optimal hash function because it satisfies the following key characteristics:

- Achieve *low quantization loss* by generating samples that are similar to the sampled, binary vectors z. Note that, using adversarial training, our model minimizes $JSD(Bernoulli_z(p) \,||\, q(b))$ instead of minimizing the expected quantization error, $E_{D_x}||b-c||^2$ [5].
- Achieve *bit uncorrelation* and *bit balance* by forcing the encoder to replicate the bit-uncorrelated and bit-independence data generation process.
- *Preserves the data manifold* in the low-dimensional discrete space, i.e., the encoder learns to map the original data points to an effective manifold-preserving low-dimensional space in order to have low-reconstruction cost.

We train the network using alternating stochastic gradient descent (SGD) procedure in three stages:

- Train the autoencoder to reconstruct accurate samples from the encoding space b by minimizing L_{FA}
- Train the generator's encoder to confuse the discriminator by minimizing the familiar heuristic generator cost:

$$L_G = E_{x\sim D_x}[-\log(D(f(I(x))))] \tag{10}$$

- Train the discriminator to distinguish the true (bernoulli) samples from the fake (autoencoder) ones by minimizing:

$$L_D = E_{z\sim Bernoulli_z(p)}[-\log(D(z))] + E_{x\sim D_x}[-\log(1-D(f(I(x))))] \tag{11}$$

The detail of the training algorithm is described in Algorithm 1.

ALGORITHM 1: AFA Model Training

Input: Training data X,
Feature embedding size e,
Binary code size B,
Bernoulli sampling parameter p,
Batch size N_b,
Number of training iterations L, l.
Output: W all parameters of the network
for *number of training iterations* L **do**
  for *a few iterations* l **do**
    - Sample a minibatch of N_b examples $\{x^{(1)}, ..., x^{(N_b)}\}$.
    - Sample m vectors $\{z^{(1)}, ..., z^{(N_b)}\}$ where $z^{(i)} \sim Bernoulli_z(p)$.
    - Update the autoencoder parameters by minimizing L_{FA} in Equation (4).
    - Update Θ_E by minimizing L_G using Equation (10).
  end
  - Sample a minibatch of N_b examples $\{x^{(1)}, ..., x^{(N_b)}\}$.
  - Sample m vectors $\{z^{(1)}, ..., z^{(N_b)}\}$ where $z^{(i)} \sim Bernoulli_z(p)$.
  - Update Θ_D by minimizing L_D in Equation (11).
end

5 EXPERIMENTAL ANALYSIS

5.1 Evaluation Procedure

Our hypothesis is that the carefully constructed segments (i.e., seed-sets), obtained from advertisers, essentially define the advertising targets; for example, instead of executing a campaign with an explicit target such as "reaching users who are movie-goers", the advertiser instead provides a seed-set of users and the look-alike modeling system finds "similar" new users to reach during the campaign. However, it should be noted that advertisers rarely provide the rationale behind the creation of the seed-set. In such a scenario, the only way to assess the off-line performance of look-alike modeling algorithm is to analyze the recall of the seed-set from the test data. A high value of recall indicates that the method is able to discover the rationale behind the creation of a seed-set by the advertiser.

Now, we will describe our evaluation procedure to analyze the performance of the look-alike modeling technique. The estimation of the off-line "Segment Recovery" task is similar to the work of [22]. Given a random subset of a segment A, the goal is to find similar users in order to recover A. In particular, we perform the following procedure:

(1) Approximately sample $|A|/2$ number of users from the segment A, i.e., randomly divide set A into two equal subsets denoted by A_1 and A_2. It should be noted that, in this case, the seed-set denoted by X_o will be equal to the set A_1.
(2) Given A_1, we assess how many users of A will a method recover at various extension thresholds m, where m is the multiple of the number of users in A_2. We define the following metrics for evaluation:

$$\text{recall}_{@m} = \frac{|X_e \cap A_2|}{|A_2|} \tag{12}$$

$$\text{precision}_{@m} = \frac{|X_e \cap A_2|}{|X_e|} \tag{13}$$

In addition, we calculate the limited area under the Precision-Recall curves (equivalent to limited mean average precision or mAP) of different methods [21]. For regression-based look-alike models, we rank the users using the probability score of a user being a seed user and select the top $|X_e|$, or $m * |A_2|$ ranked users to compute the precision and recall values. For similarity based look-alike methods, such as LSH or our proposed models, we rank the users by their minimum Hamming distances to seed-users and select the top $|X_e|$ users. When there are ties, we randomly select users in the tied ranks.

The reported recall metric indicates how well each method will recover the original segment X_o at various, specific values of m, while mAP indicates the expected precision under all possible m's values. In look-alike modeling, a method with a higher recall at the first few values of m ("early-rise"), for example, at $m = 1$, is typically more favorable. In order to efficiently compute the evaluation metrics on a single machine, we randomly select 20% of the non-seed users, instead of using the entire set of non-seed users. The reported metrics are averaged on several samples.

5.2 Comparison Methods

We compare the performance of our proposed method with various state-of-the-art methods developed in the literature for solving look-alike modeling problems.

- *Locality Sensitive Hashing using Random Projection* (**LSH**) [11]: the state-of-the-art similarity-based look-alike modeling method based on LSH with the cosine similarity function.
- *Iterative Quantization* (**ITQ**) [5]: another similarity-based look-alike modeling method based on the state-of the-art linear, hashing model.
- Binary Autoencoders (**BA**): a similarity-based look-alike modeling method using autoencoder with a fixed random noise for binarization similar to that of semantic hashing defined in Section 4.2.
- *1-class SVM* (**1-SVM**): a regression-based look-alike modeling method which learns a probability density function using only the seed-set X_o. We observe that "linear kernel" gives the best results in our experiments.
- *Degree-2 Polynomial* (**Poly2**): a regression-based look-alike modeling method employing the second-order polynomial classification model. The positive class includes seed users while the negative class is created from randomly selected negative users from the database. Different from LR, Poly2 includes second-order feature interaction but the effective input is significantly higher dimensional and sparser.
- *Linear Regression* (**LR**): the most popular regression-based look-alike modeling method by learning a linear classification model using seed users as positive class and randomly selected negative users from the database.
- *Factorization Machine* (**FM**) [18]: a very effective regression-based method that uses Factorization Machine as the classifier. FM can

Table 3: Recall values of the Segment Recovery Task at $m = 1$.

	LSH	ITQ	BA	1-SVM	Poly2	LR	FM	GBT	FA	AFA	*p-value*
Apparel-1	0.3487	0.4143	0.4656	0.5223	0.5798	0.6102	0.5769	0.6036	0.6606	**0.8528**	8e-03
Electronics-1	0.3485	0.3939	0.4343	0.2273	0.3535	0.5758	0.4595	0.6566	0.8116	**0.8434**	9e-11
Home/Garden	0.3783	0.4858	0.4570	0.5412	0.5783	0.5818	0.5711	0.6311	0.6389	**0.7581**	8e-08
Electronics-2	0.0217	0.0217	0.0217	0.0001	0.0001	0.0002	0.0068	0.0435	0.5975	**0.6522**	9e-11
Animal/Pet	0.4554	0.4870	0.5476	0.7568	0.8397	0.9137	0.8733	**0.9413**	0.8281	0.8997	4e-13
Apparel-2	0.4536	0.6665	0.6383	0.7568	0.7328	0.7548	0.7239	**0.7764**	0.6650	0.7659	3e-07

effectively learn second order feature interaction from the original high-dimensional sparse input.

- *Gradient Boosting Tree* (**GBT**) [4]: similar to LR, another popular regression-based method that employs a boosting-based classifier. GBT can learn high-order feature interactions.
- *Factorization Autoencoder* (**FA**): our proposed similarity-based look-alike modeling using factorized autoencoder with similar fix-random noise as that in BA as defined in Section 4.3.
- *Adversarial Factorization Binary Autoencoder* (**AFA**): our proposed FA that is regularized by a discriminator in an adversarial training procedure as defined in Section 4.4.

Implementation Details: We use the implementation of LSH provided in NearPy package [3]. For FM, we use the implementation from FastFM [4]. The implementations of LR, Poly2 and GBT are from Scikit-Learn [5]. All the autoencoders, including our proposed models, are implemented using Tensorflow [6] (Version 1.10). We use the same encoder's and decoder's network structures for all autoencoders. The encoder consists of two layers with 1000 units in each layer. For adversarial models, we use a similar two-layer architecture with 1000 units in each layer. We train the neural models using ADAM optimizer with a batch size of 100 examples, an initial learning rate of $1e-4$ and a learning decay of 0.96 at every 1000 training steps. We observe that using the batch normalization procedure stabilizes the training process and makes the training procedure converge faster and that dropout does not necessarily improve the generalization of the models. We also observe that employing the feature matching loss [20], which matches the statistics of the last hidden layer of the discriminator for real and fake samples, provides more stability in training our adversarial models.

In this experiment, we measure the performance of all the comparison methods for the '*segment recovery task*'. Tables 3 and 4 show the average recall values of the methods at $m = 1$ and mAP across different segments from six different partners (as described in Section 3.3), respectively. To determine if the performance of our proposed methods is significant, we calculate the paired t-tests between our methods and the baselines. The improvements of our models are statistically significant with p-values < 0.01. Due to space limitation, we only report the p-values between AFA and the next best baseline in Table 3.

5.2.1 Early-rise Performance Results. Table 3 details how each method performs when the advertisers desire to find a narrower set of look-alike users, a favorable condition known as "early-rise". We observe that similarity-based look-alike models, including linear hashing models such as LSH and ITQ and deep models such as BA, have worse recall values than those of regression-based look-alike models, especially when there are more training data (for example, for partners Apparel-1, Home/Garden, Animal/Pet, Apparel-2). As discussed in Section 2, this is expected because regression-based models utilize the supervised signal from the labeled data when compressing the original data into the propensity scores. However, when there are less training samples (for example, for partners Electronics-1, Electronics-2), all similarity-based models outperforms regression-based models. This result shows an important advantage of similarity-based approaches.

While it is expected that regression-based models have significantly better performance than that of similarity-based models, our proposed deep similarity-based approaches significantly outperform even the best regression-based model, GBT, in most cases. Specifically, AFA outperforms all other models with a significant improvement in for the first four partners while it has comparable performance with GBT for the last two partners, both of which have more training data. The result is important because AFA not only reduces the complexity of building look-alike models without considering the supervised signal (as discussed in Section 2), a characteristic of similarity-based look-alike models, but also achieves superior performance compared to that of regression-based look-alike models.

5.3 Segment Recovery Results

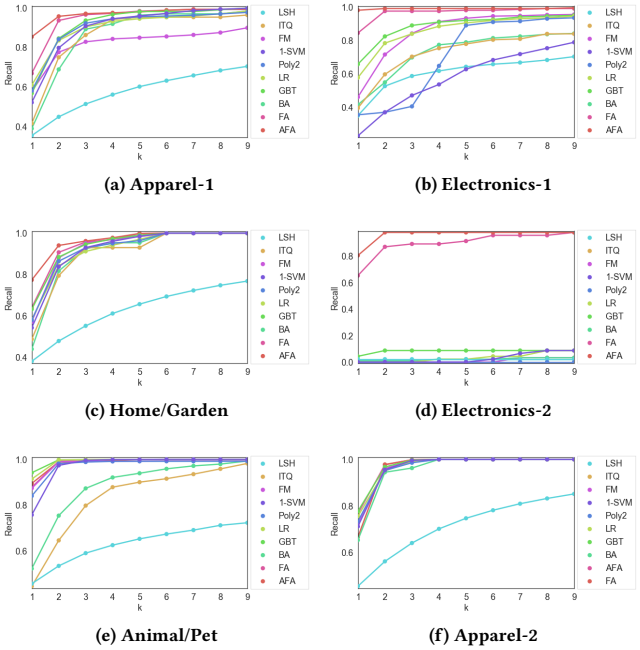
5.3.1 Quality of Look-alike Users. Complementary to Table 3, Figure 3 shows the recall curve for different values of m and Table 4 shows mAP values of the compared methods. Overall, the results exhibit consistent patterns as our observations from Table 3. While regression-based look-alike models outperform linear, similarity-based models and BA, our proposed methods show better look-alike quality in all values of m, as observed in Figure 3. Our proposed method, AFA, also achieves a significant improvement on the averaged precisions in most cases, as shown in Table 4. More importantly, when the sizes of the seed sets are smaller (for example, for the two Electronics partners), our proposed models significantly outperform all other look-alike models.

5.3.2 Ablation Study. Tables 3 and 4, and Figure 3 also show the contribution of each individual component of our proposed model with respect to all the performance metrics. FA extends BA with the bi-interaction layer and the bi-interaction embedding reconstruction as discussed in Section 4.3, and AFA further extends FA with the proposed adversarial training procedure as discussed in Section 4.4. The first observation is that the factorization component of FA contributes significantly to the improvement of the proposed

[3] https://github.com/pixelogik/NearPy
[4] http://ibayer.github.io/fastFM/
[5] https://scikit-learn.org
[6] https://www.tensorflow.org/

Table 4: mAP values for the segment recovery task of the partners.

	LSH	ITQ	BA	1-SVM	Poly2	LR	FM	GBT	FA	AFA
Apparel-1	0.2234	0.3411	0.5158	0.4187	0.4667	0.4846	0.4211	0.4655	0.7319	**0.7623**
Electronics-1	0.2178	0.2621	0.7591	0.1307	0.2219	0.4494	0.3627	0.5322	0.9339	**0.9705**
Home/Garden	0.278	0.4104	0.5253	0.458	0.4894	0.4844	0.4924	0.5385	0.6591	**0.6880**
Electronics-2	0.0005	0.0005	0.4979	0.0007	0.0006	0.0007	0.0008	0.0036	0.6427	**0.7015**
Animal/Pet	0.3994	0.3347	0.5171	0.6869	0.7781	0.8777	0.8215	**0.9151**	0.7200	0.8496
Apparel-2	0.455	0.3164	0.4247	0.6366	0.6565	0.6785	0.6502	**0.7054**	0.6678	0.6967



(a) Apparel-1 (b) Electronics-1

(c) Home/Garden (d) Electronics-2

(e) Animal/Pet (f) Apparel-2

Figure 3: Recall at values of m between 1 and 5 in the Segment Recovery Task of the partners.

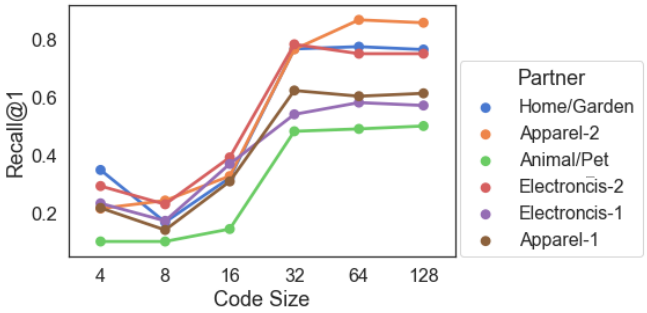


Figure 4: AFA's Recall values at $m = 1$ for the segment recovery task at different code sizes B.

models. It demonstrates that the presence of the factorization component is very important. Furthermore, employing the proposed adversarial training further improves the performance.

5.3.3 Discussion. To summarize, our experimental results demonstrate the following conclusions:

- *BA has worse performance than regression-based look-alike models and our proposed models*: vanilla autoencoders could not efficiently handle sparse, high-dimensional data. Such data causes deep models, in general, and specifically autoencoders to settle in bad local minima, thus the low-dimensional discrete representation is sub-optimal. The empirical results are consistent with our discussion in Section 2. More importantly, our proposed models efficiently learn and embed feature interactions in the low-dimensional discrete representation.
- *FA has comparable performance with regression-based look-alike models while AFA performs even significantly better in most datasets.* It exploits higher-order interactions among the features and reduces the dimensionality of the input. The output of the autoencoder effectively allow the encoder to learn better low-dimensional discrete representation of the high-dimensional, sparse input.
- *AFA outperforms FA*: adversarial training effectively regularizes the autoencoder models.
- *Both FA and AFA outperforms regression-based look-alike models when the seed sets are small*: one advantage of regression-based models is their exploitation of the supervised signals of the seed sets but they require a more involved training process when there are many seed-sets and a full, computationally expensive linear-scan in order to find look-alike users. However, our models demonstrate that we can have a computationally efficient method which also produces high-quality look-alike users, even in the cases where regression-based models do not perform well.

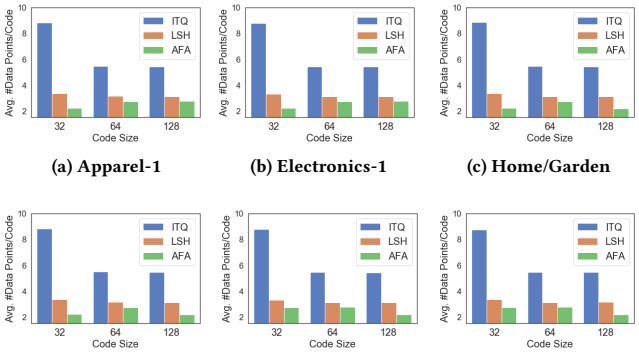
5.4 Parameter Sensitivity

An important hyper-parameter while using similarity-based look-alike models is the "size" of the discrete space, or the code size B. Figure 4 show AFA's recall values at $m = 1$ for various values of B for all the partners. We observe that AFA's performance does not noticeably change when B reaches a certain value. One possible reason is that when the discrete space is large enough, the encoder is able to map a very small number of data points to a unique discrete code (we will discuss how many data points, on average, are mapped to a unique code in the next Section 5.5), thus making it more flexible to **reconstruct** the original manifold in the discrete space. More specifically, the number of possible codes in a discrete space of size B is 2^B; for example, when $B = 4$, there are 16 possible codes and when $B = 32$ there are roughly 500 million possible codes, which makes the space large enough to accommodate the mappings of all the data points in our datasets.

5.5 Computational Efficiency

To find look-alike users, similarity-based look-alike models build a "candidate set" which likely has similar users and limit the search to only this subset. This is much more computationally efficient

than a full-linear scan as in the case of regression-based look-alike models. Although each model build its candidate set differently, one criteria to determine the computational efficiency of a similarity-based look-alike model is the distribution of the data points to the codes in the discrete space. A model which has better look-alike quality and lower number of data points, on average, assigned to each code is more favorable because it is also more suitable to distribute clustered data points in a distributed system for faster look-alike retrieval.



(a) Apparel-1 (b) Electronics-1 (c) Home/Garden

(d) Electronics-2 (e) Animal/Pet (f) Apparel-2

Figure 5: Expected number of data points assigned to each discrete code at different code sizes for similarity-based look-alike methods. Smaller values are better.

Figure 5 shows the expected number of data points assigned to a discrete code of our proposed AFA model and other baseline similarity-based look-alike models. We observe that AFA has a lower expected number of points per code compared to both LSH and ITQ in Figure 5. This shows that our proposed model produces better quality look-alike users with fewer computations when it is deployed in a production environment.

6 CONCLUSION

We proposed a novel and efficient similarity-based look-alike modeling approach that learns to embed sparse, high-dimensional users' data with complex feature interaction into a search-efficient, discrete representation space. To achieve this, we developed a new and effective neural network-based interaction layer which learns higher-order interactions between features and a mechanism to constrain the autoencoder's low-dimensional embedding to become binary by employing an adversarial training procedure. We have shown that the proposed model outperforms other existing state-of-the-art methods for the segment recovery task while being both computationally efficient and easier for parallelization in a distributed computing environment.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation grants IIS-1619028, IIS-1707498, and IIS-1838730, and the Criteo Faculty Research Award.

REFERENCES

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).
[2] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating Sentences from a Continuous Space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. 10–21.
[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.
[4] Jerome H Friedman. 2002. Stochastic gradient boosting. *Computational statistics & data analysis* 38, 4 (2002), 367–378.
[5] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 12 (2013), 2916–2929.
[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.
[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.
[8] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 355–364.
[9] Yoon Kim, Kelly Zhang, Alexander M Rush, Yann LeCun, et al. 2017. Adversarially regularized autoencoders for generating discrete structures. *arXiv preprint arXiv:1706.04223* (2017).
[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.
[11] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of massive datasets*. Cambridge university press.
[12] Haishan Liu, David Pardoe, Kun Liu, Manoj Thakur, Frank Cao, and Chongzhe Li. 2016. Audience Expansion for Online Social Network Advertising. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 165–174.
[13] Haishan Liu, David Pardoe, Kun Liu, Manoj Thakur, Frank Cao, and Chongzhe Li. 2016. Audience Expansion for Online Social Network Advertising. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 165–174.
[14] M Ma, Z Wen, Datong Chen, et al. 2016. A sub-linear massive-scale look-alike audience extension system. In *Proceedings of the 5th International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*.
[15] Qiang Ma, Eeshan Wagh, Jiayi Wen, Zhen Xia, Robert Ormandi, and Datong Chen. 2016. Score Look-Alike Audiences. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE, 647–654.
[16] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644* (2015).
[17] Yan Qu, Jing Wang, Yang Sun, and Hans Marius Holtan. 2014. Systems and methods for generating expanded user segments. US Patent 8,655,695.
[18] Steffen Rendle. 2010. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 995–1000.
[19] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50, 7 (2009), 969–978.
[20] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. 2234–2242.
[21] Gunnar Schröder, Maik Thiele, and Wolfgang Lehner. 2011. Setting goals and choosing metrics for recommender system evaluations. In *UCERSTI2 Workshop at the 5th ACM Conference on Recommender Systems, Chicago, USA*, Vol. 23. 53.
[22] Jianqiang Shen, Sahin Cem Geyik, and Ali Dasdan. 2015. Effective audience extension in online advertising. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2099–2108.
[23] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. 2017. A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence* (2017).