UNIVERSITY OF CALIFORNIA

Los Angeles

Applications of Generative Modeling for Recommender Systems

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Statistics

by

Ryan H Cheong

2022

ABSTRACT OF THE THESIS

Applications of Generative Modeling for Recommender Systems

by

Ryan H Cheong
Master of Science in Statistics
University of California, Los Angeles, 2022
Professor Yingnian Wu, Chair

Recommender systems have the difficult task of not only filtering out an overwhelming amount of information, but also learning user preferences across a large set of items. Beyond that, the order of the items recommended can be just as important as what is suggested. The practice of recommender systems has largely been grounded in latent variable models founded on linear algebra methods. However, their assumed linearity and normality limit the models' full potential. Despite their earliest inception in computer vision, we share two popular frameworks of generative modeling in the context of recommender systems and argue how they are well-suited to tackle the cold-start problem. We do so by first introducing a single implementation of each model, highlighting its key differences to its vanilla version, and then elaborating on another implementation designed to solve the cold-start problem. We find that, especially in the cold-start tasks, these approaches record impressive results.

To my family . . .

for their love and support

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

Whether one is aware or not, recommender systems surround each and every one of us across more mediums than one can think of. While most commonly associated with video streaming platforms and e-commerce sites led by industry pioneers such as Netflix and Amazon for movie and product recommendations respectively, less obvious applications of recommender systems can be found in nearly every day-to-day instrument that people interact with. Not unlike serving up movie recommendations, music streaming service Spotify compiles personalized playlists that combine tracks that a user has liked recently with others that he/she might not have discovered yet. Similarly, just as how news publishers customize their suggested "next reads" to a subscriber, social media giant TikTok determines a user's interests towards curating a specialized "For You" feed tailored specifically to his/her tastes. As such, it is not difficult to imagine the importance of strong, robust recommender systems for engaging the right users with the right items. Beyond that, its importance in downstream processes such as inventory management [6] and upsell opportunities [7] can also can also drive significant growth across organizations.

For the most part, recommender systems have traditionally practiced approaches grounded in collaborated- or content-based principles. For each situation, the task remains the same, that is: given a set of m users and n items and some information in the form of either item details and/or ratings, what are the top-K item to recommend to each user? Traditionally, latent variable models, specifically matrix factorization techniques [8, 9, 10], have long been considered reliable methods that have continued to produce strong results for decades – all while retaining the underlying interpretability in many cases by drawing on their foundations and interpretations from linear algebra.

Still, through the process of filtering the vast amount of information for what is most relevant to defining each user's individual tastes, traditional methods are limited by the linearity and normality assumptions underlying these latent variable models. Hence, deep neural networks empower practitioners to explore the non-linear relationships embedded in the user-item interactions. In particular, generative modeling approaches which first emerged in the domain of computer vision and whose earliest inception pertains to generating realistic images represent an intriguing area of study for use-cases in the recommendation arena.

In the sections to follow, we explore implementations of generative modeling, specifically variational autoencoders (VAEs) [11] and generative adversarial networks (GANs) [12], in the context of recommender systems after brief overviews of the existing recommender systems in practice and the vanilla architectures for VAEs and GANs. We then proceed to expanding this discussion to implementations of each of the two methods that are able to overcome the so-called "cold-start" problem. At last, we evaluate how the models perform against domain-specific metrics that borrow from the space of information theory. Finally, we offer commentary on the future of generative modeling in recommender systems.

CHAPTER 2

Overview

This chapter is intended to provide the contextual groundwork for defining the recommender system problem and describing two popular generative modeling architectures. The concepts presented in this chapter, including its definitions and notations, will carry through the remainder of the paper.

2.1 Recommender Systems

To begin, let us first open the discussion on the topic of recommender systems. As mentioned before, the implementation of recommender systems and its wide adoption across almost all domains has made it ubiquitous in more ways than not. In its most raw form, recommender systems are used by many for their ability to extract the relevant information from an ever-expanding and often-overwhelming cache of data in order to present users with the right items that most closely align with their interests. From a relaxed statistical perspective, recommender systems optimize for a proxy measure for successful or ideal alignment between users and items through such metrics like click-through-rate (CTR) or mean squared error (MSE) of the ratings, to name a few.

In most cases, the information at hand takes the form of either explicit feedback, such as ratings, or more commonly and easier to collect, implicit feedback, such as clicks. Moreover, said information is often given in the form of a user-item matrix $X \in \mathbb{R}^{m \times n}$ describing m users and n items and where an element x_{ij} represents user i's interaction with item j. For convenience and interoperability, we fix this and the following notation for the remainder of this paper.

Oftentimes, when referring to the overwhelming amount of data, we are implying in part that the vast number of items in the complete item set, $I = \{i_1, \ldots, i_n\}$, completely outnumbers the number of items that even the most "active" users have interacted. Hence, the recommendation system problem can often be formulated as how to use the limited observations in X to detect each of the users' from the user set, $U = \{u_1, \ldots, u_m\}$, preferences and predicting not only whether item j is relevant, but also where in the sequence of recommended items it should be placed.

Unsurprisingly, personalization is a core component of this process and so is filtering through the information to be able to uncover the underlying insights between users and items. This action alone lends itself to two of the most widely applied techniques in the practice, holding the names: collaborative and content-based filtering [13, 14].

Loosely said, collaborative filtering methods leverage other users' historical data to identify items that others with comparable taste also liked whereas content-based filtering methods utilize the items' attributes, like genre or synopsis information, to select other similar items. In other words, collaborative-based methods suggest that similarity should be derived from what other users with comparable taste have liked while content-based methods follow the definition that similarity should be based on items that share qualities with the other items the user has enjoyed during previous sessions. Common appearances of these approaches in practice may be found in the form of, respectively, "others also liked" or "you might also like" sections.

Despite strong performance from both content- and collaborative-based methods, the insufficient resolutions to such obstacles as sparsity and cold start lead many towards hybrid-based methods that combine both sources of information [15]. Hybrid methods incorporate both aspects of the aforementioned approaches either as a combination of the two systems ran separately, called loosely coupled, or as a single, unified model, called tightly coupled.

Nevertheless, in practice, even hybrid models still face headwinds when either a new user or item is introduce for which there is no prior information (cold start) or when the observed user-item interactions are limited (sparsity) [16]. This is simply due to the nature of the problems and while the above techniques attempt to navigate around these issues, they still remain difficult to model on and provide sound recommendations for. Thus, in such a situation, generative modeling presents themselves as an intriguing candidate for its ability to seemingly "create something out of nothing."

2.1.1 Metrics

Among recommender systems in literature and in practice, precision, recall, and normalized discounted cumulative gain are among the most frequently used and are detailed below. Precision (P@K) and recall (R@K) evaluate the number of correct items in the recommendation set, $\hat{y}$, as a measure for relevancy of the recommendation. Precision is calculated as a fraction of the number of items recommended, K, while recall is computed as a fraction of the user i's entire relevant item set, I_{u_i}. Broadly speaking, precision measures the relevancy of a set of recommendations while recall evaluates the model's ability to retrieve users' positively rated items. In practice, precision and recall are calculated for each user's top K items and then averaged across all m users.

Precision@K and Recall@K for a single user u_i is expressed as:

$$\text{Precision@K}(u_i, \hat{y}) = \frac{\sum_{k=1}^{K} \mathbb{1}\left[\hat{y}(k) \in I_{u_i}\right]}{K},$$
$$\text{Recall@K}(u_i, \hat{y}) = \frac{\sum_{k=1}^{K} \mathbb{1}\left[\hat{y}(k) \in I_{u_i}\right]}{\min(m, |I_{u_i}|)},$$

where we define $\hat{y}(k)$ to be the k-th item from the recommendation set.

Normalized discounted cumulative gain (NDCG@K) considers the position of the recommendations. Whereas recall treats each ranked item with equal weight, discounted cumulative gain accounts for higher ranked items with a larger weight by means of a monotonically increasing discount.

To calculate NDCG@K, DCG@K is first computed to be for a single user u_i:

$$\text{DCG@K}(u_i, \hat{y}) = \sum_{k=1}^{K} \frac{2^{\mathbb{1}\left[\hat{y}(k) \in I_{u_i}\right]}}{\log(k+1)}$$

and then normalized to arrive at NDCG@K after normalized by the highest DCG@K when recommending the most relevant items first.

By nature of the problem construction, to measure the success of a recommendation system is partially synonymous to assessing the accuracy of (1) whether the model is able to distinguish said items of interest (classification) and (2) how well the model is able to order the relevant items (rank) [17].

Other performance metrics, such as mean reciprocal rank, were considered, but it is decided that precision, recall and normalized discounted cumulative gain are sufficient measures for the classification and rank problems in our design.

2.2 Generative Modeling

We delay the introduction of the key models until the succeeding chapter for the information covered here pertaining to distinction between these two classes of statistical models serves as a necessary prerequisite first.

In short, generative models learn either the joint probability $P(x, y)$ or marginal probability $P(x)$ towards uncovering the likelihood of the instance x, thus enabling the generation new data instances, $\tilde{x}$; whereas discriminative models learn the conditional probability $P(y|x)$ so as to inform the likelihood of a label y belonging to a certain instance x, thereby providing a means to discriminate different kinds of data with its predicted labels $\hat{y}$. In general, generative models are considered to be more difficult in that it must learn the underlying structure of the data x and where the instances fall in the subspace. This contrasts with discriminative models that have the comparatively "simpler" task of learning only the boundaries separating x according to y within the subspace [18].

We will namely focus on the class of generative modeling, specifically of deep generative models [19] which lie at the intersection of deep neural networks and generative models. This set of models again follow the general goal of learning the underlying data distribution of the training set so as to be able to generate new instances with slight variations or perturbations; however, the deep neural network backbone leverages the power of the universal approximation theorem to learn essentially any kind of complex, high-dimensional data distribution with little or no supervision. In practice, it is common for the hidden layers to nonlinearly transform a sample, z, from a known distribution, such as the standard normal, to get as close to a true, multimodal data distribution as possible. Therefore, this approach naturally evokes the use of the Kullback-Leibler (KL-) and Jensen-Shannon (JS-) divergence in its formulation as a measure of the differences between the posterior and prior distributions.

These models have proven to achieve high performance and successful results, especially given a large number of training examples, and have become some of the hottest areas of study in artificial intelligence research as of late. In particular, we will explore two of the most popular methods in this family of models: variational autoencoders (VAEs) and generative adversarial networks (GANs).

2.2.1 Variational Autoencoders (VAEs)

We begin by explaining the core architecture for a variational autoencoder (VAE) [11, 20]. In order to do so, it is appropriate to preface our discussion with a brief explanation of the vanilla autoencoder. In short, autoencoders are cherished in the unsupervised learning community for their ability to learn embeddings of high-dimension, complex input data from a transformed sample in a lower-dimension, simpler latent space. Broadly speaking, it does so in two manners:

1. Exploiting the underlying structure of the input data, and

2. Recreating the original object in a more efficient manner.

Following this design, it can be useful to define its two main parts: the encoder and decoder. The former, the encoder, receives the original object, x, as an input, and is trained to learn a mapping onto a meaningful, yet smaller representation, $z \in \mathbb{R}^k$, in a lower k-dimension coordinate system. To simplify notation, we define the encoding function to be $q_\phi(z|x)$. This compression from a high to low dimension space is commonly referred to as a bottleneck effect. The latent vector representation, z, is then fed into the latter, the decoder, where the hidden representation is used to reconstruct the original input from $p_\theta(x|z)$.

Naturally, during this process, there will be some measure of information lost as a consequence of the bottleneck effect. In totality, the autoencoder is evaluated on the difference between the original data and the reconstructed output, as measured by the classical reconstruction loss functions. A regularization term is commonly added to discourage the model from simply memorizing the input and overfitting.

Our focus is on the application of autoencoders in the context of generative modeling for which variational inference is used to redefine the loss function, giving way for an alternative framework called variational autoencoder (VAE). Specifically, VAEs replace the fixed latent representation with an entire probability distribution $z \sim p_\theta(z)$ instead that can be used for interpolation and sampling. In other words, doing so imposes that the encoded representation onto the new latent coordinate space be organized and structured. In effect, the distribution can be used to perform generative tasks for new instances.

In this approach, the conformity to the standard normal distribution can be thought of as a regularization term measured by the addition of the KL-divergence term in the loss function, thereby forcing the learned latent distribution to conform to a standard normal distribution.

And so, if we suppose the true unknown distribution of $x \sim p_{data}(x)$, then in the encoder, we can define $[z|x] \sim q_\phi(z|x)$. Most commonly, a standard normal distribution is used for $z \sim \mathcal{N}(0, I_k)$. In turn, we would have that the encoder model learns the mean and covariance parameters from $\mathcal{N}(\mu_\phi(z), \sigma_\phi(z)I_k)$ and then sampled from to create the sampled latent vector that is used to recreate the higher dimension object. Continuing with the above notation, we have that in the decoder model, $[x|z] \sim p_\theta(x|z)$.

Hence, at the root, the VAE would like to maximize the probability of each data in x, or $p_\theta(x)$ whose log-likelihood is given as:

$$\log p_\theta(x) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x, z) - \log q_\phi(z|x)\right] + D_{KL}(q_\phi(z|x)|p_\theta(z|x)),$$

where the first term is the ELBO and the second term is the KL-divergence. However, since the posterior $p_\theta(z|x)$ is intractable, we must use ELBO to maximize the log-likelihood by proxy, using the fact that the first term serves as the variational lower bound to $\log p_\theta(x_i)$. Therefore, to maximize the likelihood of the data is to also maximize the following objective:

$$\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - D_{KL}(q_\phi(z|x)|p_\theta(z)).$$

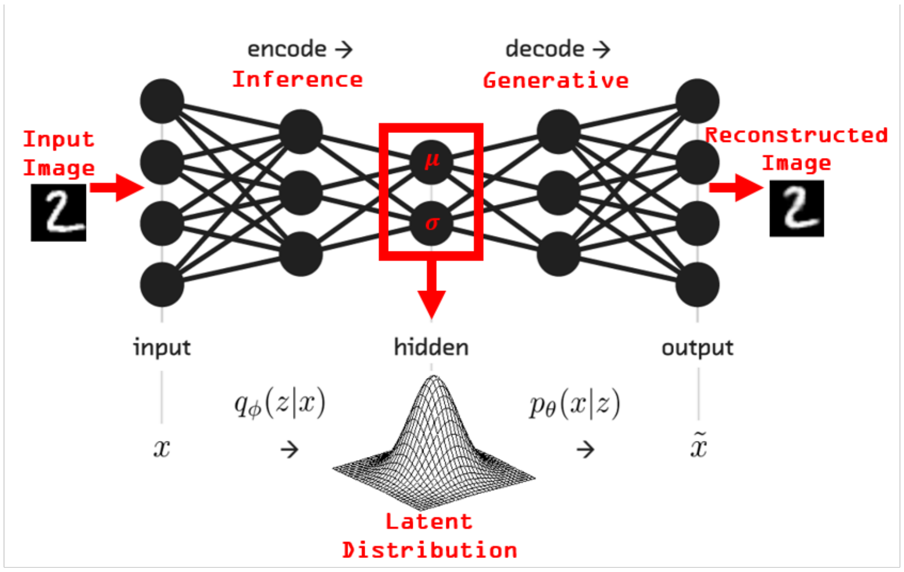

Figure 2.1: Architecture for vanilla VAE in context for image generation [1]

2.2.2 Generative Adversarial Networks (GANs)

We now proceed with a general overview of our second generative model, the generative adversarial network (GAN) [2, 12]. Much like autoencoders, a GAN is also an unsupervised learning technique using actually a combination of two models, but what makes it unique is its use of adversarial training. Similar to our discussion of VAEs, we continue by detailing both parts to the architecture as follows:

1. The generator, G, which learns a mapping to represent the underlying data x given a noise vector z, and
2. The discriminator, D, which aims to distinguish between the samples drawn from the distribution of the training data $x \sim p_{data}(x)$ against samples drawn from the reconstructed distribution $\tilde{x} \sim p_\theta(x)$.

Oftentimes, we sample z from the standard normal so that $z \sim \mathcal{N}(0, I_k)$ which is easy to sample from. The generator maps the sampled z onto a new coordinate system that shares the same domain as x, or with notation, $G(z) \sim p_\theta(x)$. In theory, the instance $G(z) = g_\theta(z)$ should be a replicate of the original data x and so by extension, $p_\theta(x)$ should resemble $p_{data}(x)$.

Next, both the remapped data $g_\theta(z)$, again, meant to be a proxy to x and assigned a label $y = 0$, and the instance x from the original distribution, assigned a label $y = 1$, are fed into the discriminator. Through this process, we are able to largely reframe the unsupervised learning task to that of a supervised learning problem since at its core, the discriminator is simply a binary classifier that predicts the authenticity of each instance according to the probability of an instance belonging to the original distribution.

Hence, the combination of the two models together are designed so that the discriminator aims to maximize the chances of predicting the correct label whereas the generator seeks to fool the discriminator by generating an instance $g_\theta(z)$ that is near indistinguishable from original data and appears to have originated from the $p_{data}(x)$. Under this architecture, the two models compete with each other in a minimax game in what is commonly referred to as a zero-sum game. A common analogy in this setup is a cat-and-mouse or cop-and-thief scenario under an actor-critic model.

Starting from the binary cross entropy loss function, the classical GAN value function is:

$$V(G, D) = \mathbb{E}_{p_{data}}\left[log[D(x)]\right] + \mathbb{E}_{p_z}\left[log[1 - D(G(z))]\right].$$

In practice, to mitigate the vanishing and exploding gradient issue, we can adjust the loss function to maximize $\mathbb{E}\left[log[D(G(z))]\right]$.

During training, it is typical that the generator is first fixed when updating the discriminator; then after the generator's parameters, θ, have been updated, the discriminator's parameters, ϕ, can be updated next. With the JS-divergence, under a fixed D, we can prove that $V(G, D)$ is minimized when the generated distribution $p_\theta(x)$ is equal to the target distribution $p_{data}(x)$. In this state, the generator is trained to a point where it is able to produce examples indistinguishable from its original counterpart, which is synonymous to where the discriminator only being able to classify the data correctly about half the time.

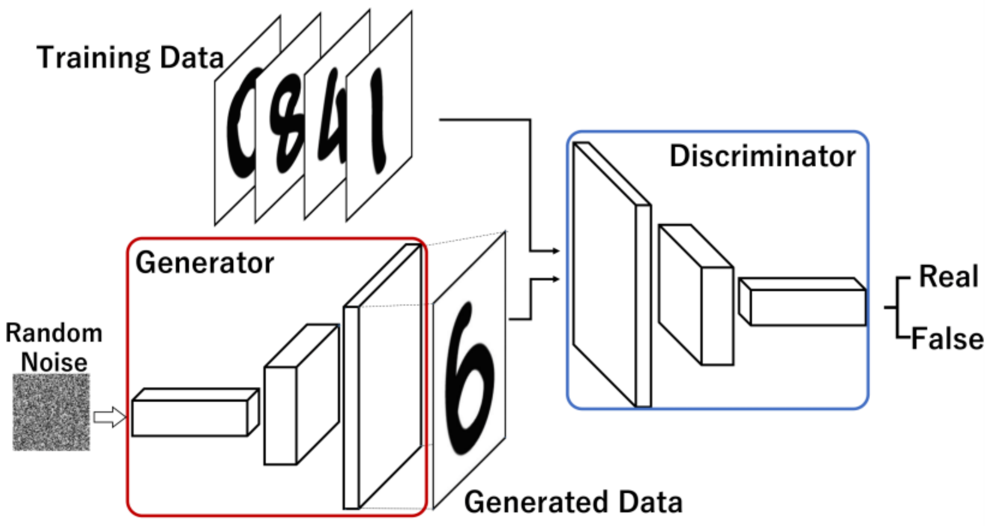

Figure 2.2: Architecture for vanilla GAN in context for image generation [2]

CHAPTER 3

Methods

In this chapter, we elaborate on four generative models designed for use in the recommender system domain. For each framework, VAE and GAN, we explain the mechanics of one implementation each followed immediately by another implementation that specifically takes into account the cold-start problem. Each section details the objective functions as well as how predictions are made from the trained models.

3.1 VAE-CF

Fundamentally speaking, VAEs can be viewed as a generalization of a latent variable model, not unlike those that populate the industry today and were described in the prior chapter, namely matrix factorization. Simply put, these methods all attempt to reduce the dimension of the user-item matrix into an expression from a lower dimensional space. In this regard, matrix factorization methods, such as SVD and SLIM, are not unlike the bottleneck effect that is seen in VAE architectures. In fact, the vanilla VAE also assumes a Gaussian latent distribution like in SVD.

Most importantly however, VAEs are not bound by the same limitations of linearity as their matrix factorization counterparts. Rather, the neural network framework unleashes the full potential for the model to learn the entire latent distribution. This effect from multi-layer perception models is frequently sought after and continues to provide significant advantages over its non-linear counterparts. Added to that, VAEs inherently lend themselves to be more robust due to the option of tapping into a wider range of likelihood functions outside of the commonly-used Gaussian likelihoods underneath most linear-based techniques.

It is with these key distinctions and advantages in mind that we begin our discussion of generative modeling approaches to recommender systems with a VAE model whose design seeks to incorporate collaborative filtering principles. Justly named, Variational Autoencoders for Collaborative Filtering (VAE-CF) [21], the model receives implicit data in the form of binary user-item interactions towards learning an underlying multinomial distribution that drives the generative model. Additionally, the multinomial likelihood tends to outperform other likelihood functions in ranking problems since its design makes it convenient to also think of the more relevant items to be ranked higher, and thus have a higher probability weight.

To start, let us consider the $m \times n$ matrix X to be the sparse user-item matrix containing implicit interactions between user i and item j. $x_i = [x_{i1}, \dots, x_{in}]$ can then be thought of as the row vector of all the items that user i has interacted with. The task of learning the statistical relationship between users and items is fundamentally characteristic in all collaborative filtering approaches; here, the VAE-CF model accomplishes this through the encoder ϕ and decoder θ. In other words, x_u is inputted into the inference network to yield a latent representation z_u, assumed to be drawn from a Gaussian prior, $\mathcal{N}(0, I_k)$. Then, after z_{u_i} is passed through the encoder, the output f_θ is treated with a softmax layer to arrive at a probability vector $\pi(z_{u_i})$ of user i interacting across the item set. As we assume x_u to have been drawn from a multinomial distribution, the reconstructed $\tilde{x}_{u_i}$ can also be thought of as a sample from $\text{Mult}(|I_{u_i}|, \pi(z_u)$, where $|I_{u_i}| = \sum_{j=1}^{n} x_{ij}$ is the total number of interactions from user i.

As before, the posterior is intractable and so variational inference with ELBO is required in order to arrive at following objective function for a single user i:

$$\mathcal{L}_{\theta,\phi}(x_{u_i}) = \mathbb{E}_{q_\phi(z_{u_i}|x_{u_i})}[\log p_\theta(x_{u_i}|z_{u_i})] - KL(q_\phi(z_{u_i}|x_{u_i})|p(z_{u_i})),$$

where the reparametrization trick has been performed on $z_{u_i} = \mu_\phi(x_{u_i}) + \epsilon \odot \sigma_\phi(x_{u_i})$ and $\epsilon \sim \mathcal{N}(0, I_k)$.

From the information theoretic perspective, the objective function, specifically the KL-divergence measure, can be treated as a tradeoff between the observed data and the assumed prior. This can further be contextualized to how closely the approximate posterior follows either the observed data, $q_\phi(z_{u_i}|x_{u_i})$, or the assumed prior, $p(z_{u_i})$. It is following this rhetoric that the authors found it beneficial to add a regularization coefficient β to the second term that specifies how freely the model can use the inferred signals of the user preferences from past interactions.

The decoded latent representation vector, $f_\theta(z_{u_i})$, is all that is needed to sort the items into a ranked list for predictions.

3.2 CVAE

Now, we extend our discussion to another implementation of VAEs, but this time with a heavier emphasis on solving the cold-start problem, called the collaborative variational autoencoder (CVAE) model [3]. Contrary to the VAE-CF model, this approach combines both collaborative- and content-based principles to better tackle the cold-start problem. By design, the CVAE model is a Bayesian generative model whose inference and generation network is used to learn a latent content variable that is passed along into a latent item variable that also receives input from a latent collaborative variable, thereby qualifying the approach as a tightly coupled hybrid model. Most importantly, this hybrid model combines the embedded information in both ratings and content information by means of a twofold approach:

1. Learning latent representations of the content data, and

2. Identifying implicit relationships between items and users from both the content and

rating data.

In particular, the former portion is accomplished through a Bayesian generative model following the classical VAE design.

Because our task involves learning the implicit relationships between items and between users, traditional deep learning models that require i.i.d. inputs must be enhanced by Bayesian modeling, thus enabling the learning of the content data as a probabilistic latent variable. And thus, by using Bayesian inference to capture the stochastic nature of the latent space, this architecture is able to obtain an accurate representation of the content information that is agnostic to modality. Components of the baseline model, namely the generation and inference networks, can be implemented by other architecture not limited to convolution and deconvolution networks for image data (like movie posters) or recurrent neural networks for sequential data (ie. textual data like movie synopses).

Not unlike the vanilla VAE, each of the latent variables is also drawn from a k-dimensional standard normal distribution so that:

$$\begin{gathered}\text{The latent user variable} = u_i \sim N(0, \lambda_u^{-1} I_k), \\ \text{the latent collaborative variable} = v_j^* \sim N(0, \lambda_v^{-1} I_k), \text{ and} \\ \text{the prior of the content variable} = z_j \sim N(0, I_k),\end{gathered}$$

where λ_u, λ_v are hyperparameters.

The hybrid framework of this model is highlighted by its inclusion of not only the information contained in the latent collaborative variable during rating prediction, but also the data embedded in the latent content variable generated from the Bayesian deep neural network.

Therefore, formally speaking, we have that the rating for item j and user i, or R_{ij}, is comprised of the inner product of the latent user variable and latent item variable; where the latter is further constituted from a latent collaborative variable and learned latent content variable. In particular, where we have specified the latent collaborative and latent content variables to follow prior normal distributions, the latent item variable can then be thought of as a product of the two Gaussian distributions and $R_{ij} \sim \mathcal{N}(u_i^\top v_{i_j}, C_{ij}^{-1})$ and C_{ij} is the precision parameter representing the confidence for R_{ij}.

Speaking in greater detail of the generative Bayesian model, we have that the original content data experiences the aforementioned bottleneck effect during the inference network so that the model can learn the parameters for the latent content variable which can be used in the generative network to reconstruct the content variable.

As with its vanilla version, the model's formulation is intractable and so we need to rely on approximate posterior inference where the original authors have elected to use the Stochastic Gradient Variational Bayes estimator. Further, by calling on the reparametrerization trick as before, the model parameters can be trained by backwards propagation and by applying ELBO, the resulting objective function can be expressed as, for a single item j and across of layers $l = \{1, ..., L\}$:

$$\mathcal{L}_{\theta,\phi}(x_{i_j}) = \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x_{i_j}|z_{i_j}^{(l)}) + \log p(v_{i_j}|z_{i_j}^{(l)}) - KL(q_\phi(z_{i_j}|x_{i_j})|p(z_{i_j})) + \text{ const.}$$

In turn, the gradients can be expressed as follows:

$$\nabla_{\mu_{i_j}}\mathcal{L} \approx -\mu_{i_j} + \frac{1}{L}\sum_{l=1}^{L} \Lambda_{v_{i_j}}(\mathbb{E}_{q_{\theta_V}}[v_{i_j}] - z_{i_j}^{(l)}) + \nabla_{z_{i_j}^{(l)}} \log p_\theta(x_{i_j}|z_{i_j}^{(l)}),$$

$$\nabla_{\sigma_{i_j}}\mathcal{L} \approx \frac{1}{\sigma_{i_j}} - \sigma_{i_j} + \frac{1}{L}\sum_{l=1}^{L} \Lambda_{v_{i_j}}(\mathbb{E}_{q_{\theta_V}}[v_{i_j}] - z_{i_j}^{(l)}) + \nabla_{z_{i_j}^{(l)}} \log p_\theta(x_{i_j}|z_{i_j}^{(l)}) \odot \epsilon^{(l)}.$$

The equations above offer some additional advantage to their maximum a posteriori (MAP) estimate counterparts that do not consider the precision parameter, C, to help introduce a measure of uncertainty.

Predictions can then be made according to $R_{ij} = u_i^\top(v_{i_j}^* + \mathbb{E}[z_{i_j}])$, or more broadly speaking, $\mathbb{E}[R_{ij}|X] = \mathbb{E}[u_i|X]^\top(\mathbb{E}[v_{i_j}^*|X] + \mathbb{E}[z_{i_j}|X]$. Here, in the presence of cold start items, the latent collaborative variable, $v^*_{i_j}$, will be 0, but $\mathbb{E}[z_{i_j}]$ continues to leverage the information from the learned latent content variable.

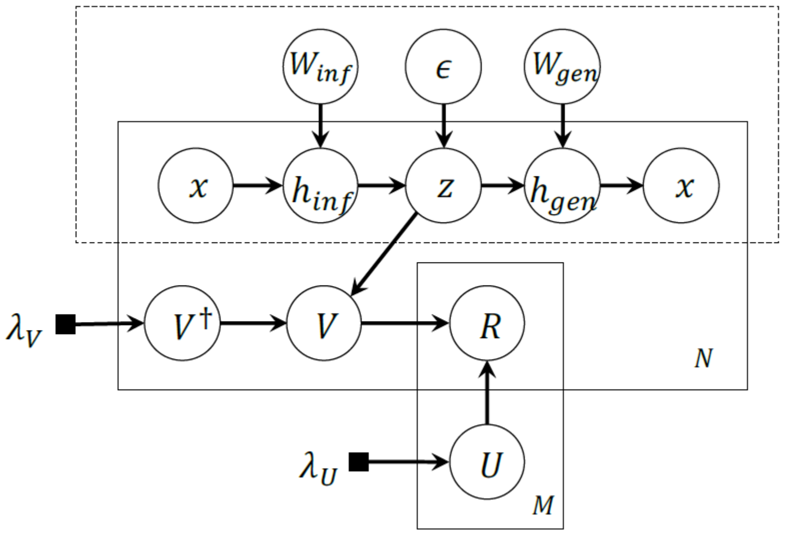

Figure 3.1: Architecture for CVAE where the top, contained portion represents the inference and generation networks [3]

3.3 CFGAN

Previously, we described the GAN architecture as how it was first introduced by Goodfellow et al. where the initial application was for image generation. Since then, the technique has been expanded upon for use-cases in natural language processing and music generation too, yielding results that are both promising and convincing. Likewise, GANs, although novel and young, are still no strangers to the domain of recommender systems [22, 23].

Characteristically, methods that follow the GAN framework are comprised of a generator, G, and discriminator, D. Likewise, the generator is taught to sample a plausible item-of-interest from the learned distribution of a user's preferences while the discriminator is trained to determine whether the item belonged to ground truth set or was synthesized. This is performed conditional on each user and across all n items. Introducing notation, we say that, conditional on user i, u_i, the generator samples a single item i_j from the item set with probability $\pi(f_\phi(i_j, u_i))$ which is treated with a soft-max and where $f(\cdot)$ is shorthand for the generator's MLP and ϕ is the model parameters belonging to the generator. Similarly, after treatment with a sigmoid function, the discriminator outputs a scalar representing the probability $\sigma(g_\theta(i_j, u_i))$ of item i_j coming from the ground truth set, where $g(\cdot)$ is shorthand for the discriminator's MLP and θ is the model parameters from the discriminator.

Yet, this single-item index approach struggles to utilize the full modeling capabilities of the adversarial training design since its discontinuous training inputs not only disallow the use of back-propagation, but also interfere with recommendation accuracy when contradictory labels are inevitably introduced by the discrete, finite sampling scheme.

Thus, the Collaborative Filtering-based GAN (CFGAN) model [4] improves on these shortcomings by pivoting the generator to generate a real-valued item vector with which the discriminator is to consequently discriminate the vectors obtained from the observations from those sampled by the generator. Then, as before, after several epochs and mini-batches and conditional on the user, the generator is said to arrive at an underlying distribution following the user's interests. And since the model is equipped with vector-wise training and real-values as with its vanilla counterpart, learning is able to be deployed by back-propagation with stochastic gradient descent (SGD) for ϕ and θ.

At this point though, the CFGAN is still not fully compatible with the recommender system problem since, unlike the classical GAN framework in image generation, the model here deals with a highly sparse vector, that although real-valued, still only contains 1's where an interaction is observed or is empty else. Therefore, in an effort to deter the generator from simply replicating a user's item vector and trivially setting all values to be 1, the notion of negative feedback is introduced. In execution, from the set of empty-valued items, we designate a proportion of them to be negative items by temporarily setting their values to be 0 during training. In the final implementation of the CFGAN, the authors explain a design for negative feedback that combines both a zero-reconstruction regularization and partial-masking. This combination has the effect of forcing the generator to consider both (1) generating high values close to 1 for the observed items and (2) assigning low values close to 0 to the unobserved items.

Now, we are able to arrive at the CFGAN's objective function, taking c_{u_i} to be the condition vector for user i:

$$\begin{aligned} V(G,D) = \sum_{i=1}^{m}(\log(1 - D((\tilde{x}_{u_i} \odot (e_{u_i} + k_{u_i})|c_{u_i}))) - \log D(x_{u_i}|c_{u_i}) \\ - \log(1 - D((\tilde{x}_{u_i} \odot e_{u_i})|c_{u_i})) + \alpha \sum_j (x_{u_ij} - \hat{x}_{u_ij})^2), \end{aligned}$$

where e_{u_i} is the n-dimensional indicator vector for which items user i has interacted with, k_{u_i} is the n-dimensional vector for partial masking, $\tilde{x}_{i_j}$ is the n-dimensional vector that is generated from G given c_{u_i} and a random noise vector $z \sim \mathbb{R}^n$, and α is the tuning parameter for the regularization portion. Further, we have that the first and last terms correspond to the generator and the terms in between are associated with the discriminator.

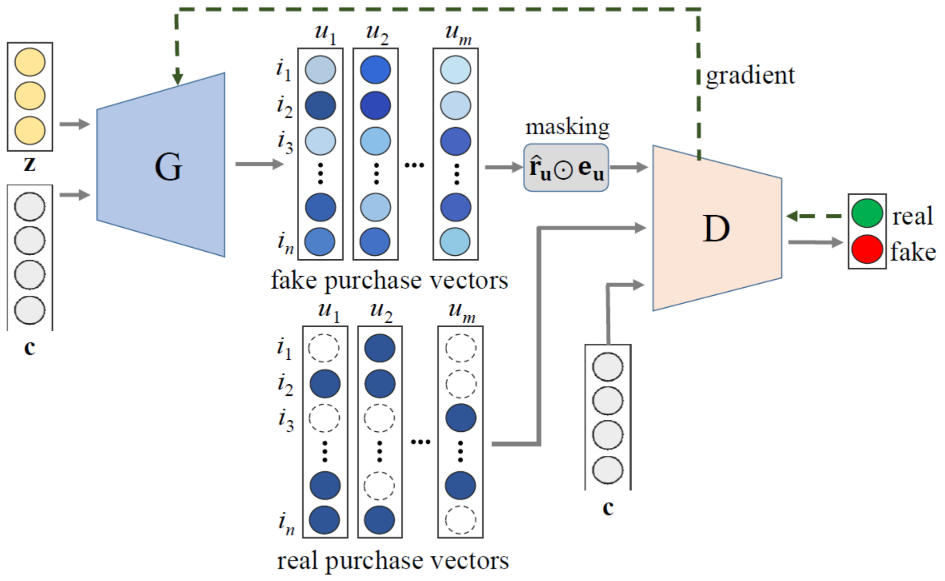

Figure 3.2: Architecture for CFGAN using zero regularization and partial masking [4]

3.4 LARA

Finally, we shift our discussion to another generative model architecture – specifically one that continues to use the approach of adversarial learning, but here, as a proposed solution to the cold-start problem. As is the case with the new arrival of items (or users even), the lack of historical data makes it difficult to exploit any user-item interaction. Moreover, the inability to compute similarity scores between users and items, much less to package as a recommendation, adds an additional obstacle.

Instead, Sun et al. [5] propose using attribution derived from multiple generators deployed through adversarial learning to generate users from each of the item's attributes. The similarity of user and item features, presented as attributes in some intermediate stage, can then be compared to infer the relationship between the new item and users from the user set U. From their paper, the original authors describe an example from the product recommendation space with the new item being a Nike shoe. The item's attributes may be its brand (Nike), category (sneaker), and selling price (discount). From there, the model is able to produce user profiles from each of the item attributes for the type of user profiles who would align closely with each of the attributes such as, favorite brand is Nike, likes to exercise, and prefers to shop for items on clearance. From these three user profiles, a real user can be matched to this generated, "ideal" candidate, and in turn, the new item. The use of multiple generators for different user profiles based on the different item attributes lends itself to its chosen name, adversariaL neurAl netwoRk with multi-generAtors, or LARA.

In order to ensure that the generated user profiles do not stray too far and eventually correspond to a real user belonging in U, a discriminator is trained to discriminate between the generated profiles and observed users. Beyond this, the discriminator also influences the modeled item attributes to stay relevant through the joint modeling of user profiles and item attributes, thereby qualifying this approach to be considered a tight-coupled hybrid technique.

Similar to how CFGAN was modeled conditional on a user, LARA is also designed to be conditional on an item which we will designated with the condition item vector c_i for item j. Continuing with this example, we say that item j has a representation of k-dimension denoted by $z_j = [z_{j1}, \ldots, z_{jk}]$. For each of these k attributes, individual generators $G_1, \ldots, G_k$ are trained to produce k user profiles. Finally, the proposed user vector u_c is created by deploying a neural network to concatenate each of the user profiles.

On the other side of the adversarial training, the discriminator is trained with three pairs of training instances, each conditional on item j from above: i_j and the proposed user vector u_c as well as i_j paired with a true user u^+ and a negative user u^-. This type of negative feedback is comparable to the zero-regularization and partial masking techniques employed by CFGAN. Together, these three training pairs are used to help the discriminator output a scalar probability after passed through a sigmoid function with the aim of assigning a value close to 1 to the (i_j, u^+) pair.

Altogether, the addition of the third training pair, (i_j, u^-), discourages the generator from forming proposed users ill-suited for item j through the same mechanism as the negative feedback example from CFGAN. But its presence must also be accounted for in the objective function which is taken over all items $j = 1, \ldots, n$:

$$\mathbb{E}_{u^+ \sim p_{true}(u^+|i_j)}\left[\log D(u^+|i_j)\right] + \mathbb{E}_{u_c \sim p_\theta(u_c|i_j)}\left[\log(1 - D(u_c|i_j))\right]$$
$$+\mathbb{E}_{u^- \sim p_{false}(u^-|i_j)}\left[\log(1 - D(u^-|i_j))\right],$$

where the θ is the parameters of the generator. Both θ, ϕ are updated by SGD using the alternating optimization as with the vanilla GAN.

For predictions with the trained model, given an inputted item, user profiles are generated and concatenated to form a proposed user vector. From this point, similarity scores are used to find the top-K users from U by comparison of the vectors u_c and $u_1, \ldots, u_m \in U$.

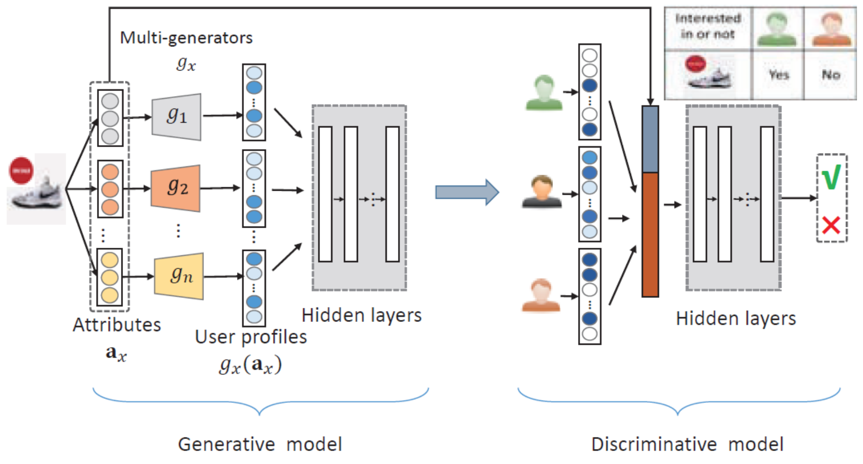

Figure 3.3: Architecture for LARA including the multi-generator design for the creation of multiple user profiles from the item attributes [5]

CHAPTER 4

Evaluation

This chapter describe how we measured the effectiveness of the aforementioned models with a real-world dataset. We evaluate the model performance and how well it solves the ranking and classification problems by measure of the metrics introduced in Section 2.1.

4.1 Data

We choose to measure the success of the models described in the previous chapter within the domain of movie recommendations and with the popular MovieLens-1M dataset that has been cited in countless literature pertaining to the topic of recommender systems. The dataset is comprised of 1,000,209 movie ratings on a scale of 1-5 from 6,039 users and 3,883 items. As warned in Section 2.1, it is often the case that these recommender system problems face sparsity as is the case here with 95.72% of the values in user-item matrix, X, empty.

Also included are some metadata on the users including gender, age, occupation, and zip code. Movie information is also provided in the form of 18 genres where some items may have more genres tagged than others.

4.2 Implementation

For the models that are designed to receive implicit, binary feedback, we present the ratings that are equal to or greater than 4 as 1's and all other ratings to be 0's. Recall that based on the use of negative feedback with the unobserved interactions, some of the empty elements may eventually be randomly imputed with 0's too during training.

In general, we retain the same technical architecture for each of the models as the authors originally intended. Hyperparameters, where present, were chosen across a number of different methods among the four models and the same optimal values determined from each tuning method by the authors are applied in this paper too.

To facilitate a more efficient training design, we only keep the users who have provided feedback to 20 or more movies. Additionally, although our models do not explicitly account for temporal effects and shifting preferences, as is the way users interact with these items in time, we hold out the most recent interactions for our test set in our 80/20 train-test split.

Models such as CVAE and LARA that make use of the item information, here in the form of specifically the genre information, receive the input as a one-hot encoded vector $\in \mathbb{R}^{18}$, where a movie belonging to three genres will have three 1's assigned to the corresponding indices and 0's elsewhere. Therefore, in the LARA implementation, 18 generators are trained, one for each genre attribute. In the CVAE implementation, several of the activation layers were altered from its original design that used the CiteULike datasets in order to accommodate the MovieLens dataset and its desired output.

Again, we measure the effectiveness of the models for their ability to solve the ranking and classification problems with precision (P@K), recall (R@K), and normalized discounted cumulative gain (NDCG@K) for $K = 20$.

4.3 Results

The results for each of the models are presented in Table 4.1.

Table 4.1: Summary of Performance for Proposed Models

Method	P@20	R@20	NDCG@20
CF-VAE	0.392	0.542	0.420
CVAE	0.362	0.453	0.674
CFGAN	0.412	0.484	0.487
LARA	0.286	0.334	0.839

CHAPTER 5

Conclusion

This chapter offers some insights and plausible explications for how the designs of the models explain their results from the previous chapter. Finally, we conclude this paper with some final thoughts on the intersection of recommender systems and generative modeling and possible future work.

5.1 Discussion

Speaking to the use of the multinomial likelihood in the CF-VAE model, we are able to recognize its utility in the classification problem as suggested by the P@20 and R@20 score as well as an appropriate proxy for the top-K ranking loss in the rank problem as indicated by the NDCG@20 score. In their original paper, the authors include comparisons against Gaussian and logistic likelihoods, both of which fall short in their ability to position the items in order of descending interest. This is almost to be expected since the multinomial likelihood is more akin to the recommender system problem by construct, as the probability weights can be thought of as a level of interest. All of the probability weights needing to sum up to 1 also forces the model to carefully consider where it assigns weights and to which items.

Regarding the CVAE model, we observe an impressive NDCG@20 score by nature of its construction as a Bayesian generative model. In turn, this allows for a more robust performance since we are able to sample an entire vector from the learned latent distribution instead of relying on a single point estimate.

The strong P@20 and R@20 for the CFGAN model also speaks to the effectiveness of the vector-wise training in comparison to other GAN-based collaborative filtering models that still rely on single-item indexing [24] and other collaborative filtering approaches that utilize traditional pointwise optimization [25] thereby hindering the amount of learning by each model. In effect, the two models are able to consistently encourage one another to improve the capability for the generator to create more plausible samples and for the discriminator to provide feedback on how to produce a more realistic instance.

The LARA model achieved the highest NDCG@20 score of all the other models demonstrating the possible gains from exploring the user-item interactions and conceptualizing the user representations at the attribute-level. One possible explanation for this can be demonstrated through a brief illustration of a user who likes an item, but only for some attributes more so than others. At the interaction-level, all the item's attributes, here genre, are obfuscated into a single representation. This is harmful especially in a GAN model that is used to generating dense vectors that contrast with the sparse user vectors that we observe. But in the case of attribute-level user-item interaction, each attribute is passed through its own generator and generated item attributes are used to create user profiles whose similarities are allowed to be more idiosyncratic.

Overall, consistent with the vanilla version, the adversarial networks were the most difficult to train in terms of convergence as they are frequently cautioned as unstable techniques. While we did not encounter such obstacles as mode collapsing or vanishing gradients, the design of the problem naturally adds difficulty. Unlike the vanilla versions and in other instances of image or music generation, we do not have to deal with a random noise input the same way to generate new and varied instances as our aim is to produce realistic recommendations from our item set I or user set U.

We can also expand the input data to also receive explicit, non-binary feedback in future work. Some of these adjustments may be as simple as switching the underlying distribution, such as form multinomial to Gaussian for the CF-VAE model. Future work can also explore the consequences of changing the conditional input of the CFGAN model from user-conditional, as currently set, to item-conditional.

When dealing with the cold-start problem, we see how the Bayesian approach to generative modeling equips the model to still yield relevant outputs. This is suggestive of a strong prior and its influence on the likelihood through multiple training epochs. The presence of an assumed prior distribution also makes it robust when faced with sparsity.

5.2 Concluding Remarks

To conclude, we have described use-cases of generative modeling, specifically variations of the VAE and GAN frameworks, in the practice of recommender systems. Moreover, we have achieved impressive results for both the ranking and classification problem at hand. In particular, we have shown how the current linear-based modeling approaches such as matrix factorization are handicapped not only by their assumptions of linearity, but of normality too – especially when tasked with learning an entire latent distribution. Rather, by leaning into the universal approximation theorem, the true user-item interactions in the latent space can be untapped and modeled to achieve strong results for the recommender problem. Broadly speaking, VAEs and GANs both lend themselves well to the domain of recommender systems due to their interoperability with the popular latent variable models used in practice for the last few decades, but also because of their advanced modeling capabilities. In particular, we see that their architectures are positioned in a manner that makes it highly conducive and relatable to tightly coupled hybrid models that solve the elusive cold-start problem.

REFERENCES

[1] Visualizing mnist with a deep variational autoencoder, 2018.

[2] Shota Harada, Hideaki Hayashi, and Seiichi Uchida. Biosignal generation and latent variable analysis with recurrent generative adversarial networks, 2019.

[3] Xiaopeng Li and James She. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 305–314, New York, NY, USA, 2017. Association for Computing Machinery.

[4] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. Cfgan: A generic collaborative filtering framework based on generative adversarial networks. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 137–146, New York, NY, USA, 2018. Association for Computing Machinery.

[5] Changfeng Sun, Han Liu, Meng Liu, Zhaochun Ren, Tian Gan, and Liqiang Nie. Lara: Attribute-to-feature adversarial learning for new-item recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, page 582–590, New York, NY, USA, 2020. Association for Computing Machinery.

[6] C. Dadouchi and B. Agard. Recommender systems as an agility enabler in supply chain management. *Journal of Intelligent Manufacturing*, 32:1229–1248, 2021.

[7] Navin Dookeram, Zahira Hosein, and Patrick Hosein. A recommender system for the upselling of telecommunications products. In *2022 24th International Conference on Advanced Communication Technology (ICACT)*, pages 66–72, 2022.

[8] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[9] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. KDD '08, page 426–434, New York, NY, USA, 2008. Association for Computing Machinery.

[10] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. pages 497–506, 12 2011.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[13] Shuo Yang, Mohammed Korayem, Khalifeh AlJadda, Trey Grainger, and Sriraam Natarajan. Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive statistical relational learning approach. *Knowledge-Based Systems*, 136:37–45, 2017.

[14] Poonam B Thorat, Rajeshwari M Goudar, and Sunita Barve. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110(4):31–36, 2015.

[15] G Geetha, M Safa, C Fancy, and D Saranya. A hybrid approach using collaborative filtering and content based filtering for recommender system. In *Journal of Physics: Conference Series*, volume 1000, page 012101. IOP Publishing, 2018.

[16] Guibing Guo. Resolving data sparsity and cold start in recommender systems. In Judith Masthoff, Bamshad Mobasher, Michel C. Desmarais, and Roger Nkambou, editors, *User Modeling, Adaptation, and Personalization*, pages 361–364, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[17] Gunnar Schröder, Maik Thiele, and Wolfgang Lehner. Setting goals and choosing metrics for recommender system evaluations. In *UCERSTI2 workshop at the 5th ACM conference on recommender systems, Chicago, USA*, volume 23, page 53, 2011.

[18] J.A. Lasserre, C.M. Bishop, and T.P. Minka. Principled hybrids of generative and discriminative models. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 87–94, 2006.

[19] Lars Ruthotto and Eldad Haber. An introduction to deep generative modeling, 2021.

[20] Abu Kamruzzaman and Charles C. Tappert. Developing deep learning models to simulate human declarative episodic memory storage. *International Journal of Advanced Computer Science and Applications*, 10(3), 2019.

[21] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering, 2018.

[22] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 515–524, New York, NY, USA, 2017. Association for Computing Machinery.

[23] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets, 2017.

[24] Martin Saveski and Amin Mantrach. Item cold-start recommendations: Learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, page 89–96, New York, NY, USA, 2014. Association for Computing Machinery.

[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, page 452–461, Arlington, Virginia, USA, 2009. AUAI Press.