

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Muhammad Kamal Memon

Deep hybrid collaborative filtering for E-commerce product recommendation system

Master's Thesis
Espoo, November 20, 2020

Supervisor: Professor Rohit Babbar
Advisor: Markku Mantere M.Sc. (Tech.)

Aalto University
School of ScienceMaster's Programme in Computer, Communication and
Information SciencesABSTRACT OF
MASTER'S THESIS

Author:	Muhammad Kamal Memon		
Title:	Deep hybrid collaborative filtering for E-commerce product recommendation system		
Date:	November 20, 2020	Pages:	42
Major:	Machine Learning, Data Science and Artificial Intelligence	Code:	SCI3044
Supervisor:	Professor Rohit Babbar		
Advisor:	Markku Mantere M.Sc. (Tech.)		
<p>Deep learning based approaches for collaborative filtering have proven highly successful in modern recommendation system research. In this work, we explore Variational Autoencoders to build a product recommendation system. We utilize modern frameworks of probabilistic deep learning to carry out an empirical analysis on real world datasets and compare it with traditional matrix factorization methods.</p> <p>Matrix factorization methods for collaborative filtering suffered from sparsity in the data and under performed on scalability measures. We use these methods as baselines and strive to outperform them. We augment a standard Variational Autoencoder with a regularization parameter that partially anneals the divergence term in the objective function and use multinomial likelihood to model the user-item interaction data.</p> <p>To analyse our methods, we use two recently collected datasets comprising of user and item interactions. We run various experiments on these datasets to find the best possible model. We report a quantitative comparison between the baselines and our approach on various metrics. We reach satisfactory results suggesting that the partially regularized Variational Autoencoder framework with a multinomial likelihood is well suited for the collaborative filtering task on real world data.</p>			
Keywords:	Collaborative Filtering, Variational Autoencoders, Deep learning, Recommendation systems, E-commerce		
Language:	English		

Acknowledgements

Firstly, I would like to thank my supervisor Rohit Babbar and my advisor Markku Mantere for their support and guidance throughout the process. In addition to that I would also like to thank the faculty and staff at Aalto University School of Science and to my colleagues at Fromso Oy, without their backing this thesis would never have been possible. I would also like to express special thanks to Tietokilta ry - the Computer Science Students Guild, AYY - Aalto Student Union and to Aalto Debating Society for enriching my journey to the master's degree with amazing experiences.

On a personal note, I would like to thank my parents, Shafiq and Humera, for their love and unflinching support throughout my academic career and life. Also I am incredibly grateful to all the wonderful friends and family here in Finland and back home in Pakistan and Brazil. Chiefly, I want to thank Daniyal Usmani for his perpetual help, to Azeem Akhter for being there through thick and thin and to everyone at Charskada for support and encouragement during the most exasperating times.

Lastly, but by no means least, I would like to mention Manchester United Football Club and Sir Alex Ferguson, to whom I will be eternally in debt for everything. And finally to Karachi and its Biryani, which sends one's taste bud to dance, the hope and longing of which keeps me going everyday.

Espoo, November 20, 2020

Muhammad Kamal Memon

Notice

This thesis was written as part of a research and development project at Frosmo Oy.

Abbreviations and Acronyms

B2C	Business to Customer
B2B	Business to Business
UX	User Experience
UI	User Interface
API	Application Programming Interface
MAE	Mean Absolute Error
CF	Collaborative Filtering
PCA	Principal Component Analysis
PMF	Probabilistic Matrix Factorization
SVD	Singular Value Decomposition
ANN	Artificial Neural Network
DAE	Denoising Autoencoder
ELBO	Evidence Lower Bound
VAE	Variational Autoencoder
SGVB	Stochastic Gradient Variational Bayes
SGD	Stochastic Gradient Descent
NCF	Neural Collaborative Filtering
MLP	Multilayer Perceptron
NMF	Non-negative Matrix Factorization
Multi-VAE	Variational Autoencoders with multinomial likelihood
NDCG	Normalized Discounted Cumulative Gain

To indicate matrices we use uppercase bold characters (e.g. \mathbf{X}) and to indicate vectors we use lowercase bold characters (e.g. \mathbf{x}).

Contents

Abbreviations and Acronyms	5
1 Introduction	8
1.1 Problem statement	9
1.2 Structure of the Thesis	10
2 Background	11
2.1 E-commerce	11
2.2 Frosmo Tools	11
2.3 Recommender Systems	12
2.4 Collaborative Filtering	13
2.4.1 Methods of Collaborative Filtering	14
2.4.2 Sparsity and Scalability	14
2.5 Representation Learning	15
2.5.1 Deep Collaborative Filtering	16
2.5.2 Variational Inference	16
2.5.3 Variational Autoencoder	18
3 Related work	20
3.1 Probabilistic Matrix Factorization	20
3.2 Matrix Factorizing Variational Autoencoder	20
3.3 Neural Collaborative Filtering	21
4 Method	23
4.1 Model	23
4.2 Amortized Variational Inference	24
4.3 Evidence Lower Bound (ELBO)	25
4.4 Training Variational Autoencoder	26
5 Empirical Analysis	28
5.1 Datasets	28

5.2	Evaluation Metrics	29
5.2.1	Normalized Discounted Cumulative Gain (NDCG) . . .	29
5.2.2	Recall	30
5.3	Experimental Setup	30
5.3.1	Baselines	30
5.3.1.1	Singular Value Decomposition	31
5.3.1.2	Non-negative Matrix Factorization	31
5.3.2	Data Preprocessing	32
5.3.3	Hyperparameter Optimization	32
5.3.4	Training	33
5.4	Results and analysis	34
6	Conclusion	36
6.1	Future work	37

Chapter 1

Introduction

In a world where businesses heavily rely on interwebs to sell products and where the shift towards a digital economy is accelerating every day, product recommendation systems are playing a pivotal role in helping the customers navigate the vast ocean of possibilities on online stores. The COVID-19 pandemic has also ushered in an era of unprecedented growth in E-commerce industry where more and more businesses are using their online presence to reach to their customers. The United Nations Conference on Trade and Development estimated in a survey [1] that there has been a 6 to 10 percentage point increase in online purchases across all product categories since the start of the pandemic.

Along with a heavy increase in traffic, the need to personalize the online retail experience is ever increasing. The inventories of online stores are huge and the traffic is enormous, for example, one of the biggest online store, Amazon has 488 million items listed for sale only in the United States, whereas the online store of Ikea, the world largest furniture retailer, was visited 2.8 billion times in 2019 [2]. It is then plainly a daunting task for any customer to explore inventories of such scale and consequently the amount of traffic on these sites necessitates the businesses to automate the product recommendation and personalization process on their sites.

An automated recommendation system works as an information agent and assist the customers in decision making by recommending the relevant products according to their needs. A common and efficient technique for recommendation systems to provide meaningful proposals is collaborative filtering. The collaborative filtering process in essence infer a customer's preferences from the historical behaviour of themselves and other similar customers. Traditionally, most collaborative filtering methods use matrix factorization (MF) which employs a joint latent space where the interaction of users with items are modelled as an inner products of that latent

space. However, matrix factorization methods have commonly suffered with sparsity and scalability issues such as when the user-item interactions are minimal or when the scale of operation grow enormously. However, recent advances in deep learning research [3, 4] have found neural architectures as a feasible solution for collaborative filtering problem. These neural network based methods have been found to alleviate the issues regarding sparsity of the data and the cold start of the recommendation system whilst remaining fairly scalable in nature.

In this thesis work, we endeavour to assess an artificial neural network called Variational Autoencoder (VAE) introduced by Kingma and Welling [5]. Based on probabilistic deep learning, VAEs have been applied to show significant performance improvement on collaborative filtering problem and we set out to specifically asses this approach on real world E-commerce data.

This work also involves Frosmo Oy, a Helsinki based software company, which provides personalized recommendation services to its clients. As part of the Frosmo platform's research and development, we have at our disposal their tools and real world E-commerce data to perform an empirical analysis on the feasibility of using deep learning networks on collaborative filtering task.

1.1 Problem statement

This thesis undertakes the application of recent advances in deep learning research on collaborative filtering problem. We base our approach on Variational Autoencoders (VAE) introduced by Kingma and Welling [5] and utilize the framework proposed by Liang et al., 2018 [6] for an E-commerce recommendation system. This approach aims to alleviate two critical issues: 1) The cold start problem when its not feasible to recommend new items or to recommend to new users based on learned latent factors and 2) to overcome the sparsity problem due to large user and item base.

The contribution of this thesis is to apply and evaluate the aforementioned approach by carrying out an empirical analysis on real world datasets and compare it to traditional matrix factorization methods. This leads us to the problem statement that this thesis will explore:

How well does a hybrid solution based on probabilistic deep learning performs on collaborative filtering problem for real world datasets?

1.2 Structure of the Thesis

Chapter 2 provides an overview on the domain of E-commerce and details on the working and tools of Frosmo Oy. It follows with the discussion on recommendation systems and how representation learning method of variational autoencoders can be employed for collaborative filtering task.

Chapter 3 describes the methods which are similar to our approach and gives formal definition and descriptions of methods closely related to our problem.

In Chapter 4 we outline the methodology used to conduct the experiments. It also includes the description of the model in detail and the various techniques employed to enhance the performance of the model.

Chapter 5 deals with the setup of experiments. It comprises of the details of the data used for the experiments and the baseline methods used for comparison. The evaluation metrics are also described in this chapter in detail along with the optimization process and the training of the model. Lastly, it includes a comparison and analysis on the obtained results where a quantitative comparison is presented between the proposed method and various baselines.

In Chapter 6 we reflect back on the whole process and summarize the approach. We also share our thoughts on the potential future work for this thesis.

Chapter 2

Background

2.1 E-commerce

E-commerce, in general terms, is defined as the act of selling and buying of goods or services electronically by using telecommunication networks such as the internet. The most common form of E-commerce is a business to customer (B2C) marketplace where online retail stores sell their products directly to the customers. There is also a significant presence of business to business (B2B) marketplaces where exchange of products, services or information is done between businesses.

In this thesis, our focus is on business to customer (B2C) online retail marketplaces where companies sell products to individual customers directly. Over recent years, this domain of E-commerce has become the most powerful mode of carrying online transactions and shopping for goods. Thus, naturally this B2C form is adopted by large multi-nationals which operate on a global scale. The Journal of Electronic Commerce Research [7] notes that B2C marketplaces are expanding at an exponential rate and it's estimated that several trillion dollars are being exchanged annually over internet.

2.2 Frosmo Tools

Frosmo Oy is a software company based in Helsinki, Finland. Frosmo's primary focus is to provide personalization services to E-commerce companies such as product recommendations and improvement of UX/UI on their websites. Their goal is to help E-commerce companies in retail and iGaming verticals to significantly improve their key performance indicators such as average order value, user engagement and conversion rates [8].

Frosmo provide their clients with a script tag which is inserted into the

front-end HTML body of the client’s website. This script then enables Frosmo to track the data and modify the website with personalized content. One of Frosmo’s major offering is personalized product recommendations for which different strategies such as affinity based recommendations and collaborative filtering is used.

In this work, we have utilized the data tracking tools mainly based on the front-end JavaScript packages and data ingestion tools such as Kafka. Frosmo’s platform comprises of dedicated APIs and robust and scalable tools for data tracking and distribution of pre-computed personalized recommendations. For production level distribution of pre-computed recommendations we have utilized Golang based Recommendation API which act as an outward platform of providing recommendations to Frosmo’s customer websites.

2.3 Recommender Systems

An E-commerce platform facilitates the users in their shopping and hence its critical to have such functionality that enables the platform to provide users with products of interest. The problem thus is to find relevant and useful choices for the users which can be based on some form of user engagement with the platform or on active information filtering mechanism. To alleviate this issue, in comes Recommender Systems which serve users with relevant data when needed. However, in recent works as Cacheda and Paraper [9] notice, users are increasingly expecting answers to their information needs without the formulation of any explicit query. Hence, on top of being active in their mechanism, modern recommendation systems tend to include value added features of personalization and the capability of predicting the preference of a set of items for a user. This agency of recommender systems is due to the fact that their output is expected to be pre-tailored to an individual user rather than being based on a query.

The Recommender Systems Handbook [10] concludes that the primary target of recommender systems is to provide individuals with personalized and diverse suggestions, especially when there is an absence of users’ own experience in the presence of overwhelming choices and alternates. In their book Machine Learning Paradigms, Lampropoulos formulates the recommendation problem as an estimation of rating of an item that has not been encountered by the user. To estimate the rating, the system can use any indicator of associative interest such as the ratings user gave to other similar items, ratings given to the item by other users, and items’ and users’ own attributes.

In concrete terms, for the set of all users \mathcal{U} , set of all items \mathcal{I} and ratings,

a total ordered set of non-negative integers R , we define a utility function f to measure the usefulness of item i to user u

$$f : \mathcal{U} \times \mathcal{I} \longrightarrow R$$

Then, the objective of a recommender system is to maximize the following utility function by choosing an item $i^* \in \mathcal{I}$ such that i_u represents the interaction between item i and user u for each user $u \in \mathcal{U}$

$$\forall u \in \mathcal{U}, i_u^* = \arg \max_{i \in \mathcal{I}} f(u, i) \quad (2.1)$$

The utility function above outlines a user's preference to the item in terms of ratings. Such ratings are given as a value on a scale for example, from 1 to 5 $R(u, i) = \{1, 2, 3, 4, 5\}$. In real world datasets, the utility function has values based on any implicit or explicit ratings set by users on items. However, due to the large size of user-item space $\mathcal{U} \times \mathcal{I}$, R has many missing values and therefore the user-item rating matrix is usually very sparse. Eventually, the recommendation system needs to generalize the entire $\mathcal{U} \times \mathcal{I}$ space along with maximizing the utility function.

This generalization from known rating space to unknown involves an estimation of a utility function which targets to optimize certain performance criterion, such as Precision or Mean Absolute Error (MAE). Once these unknown ratings are computed, the system generates recommendations for a given user by opting for the highest rated items according to function (2.1).

2.4 Collaborative Filtering

Collaborative Filtering (CF) as aptly described by Bobadilla et al. [11] is a technique to predict how a user would rate an item based on a sparse matrix of available historical ratings of all users. Resnick and Varian [12] in their brief on recommender systems note that the term *collaborative filtering* was first coined by the developers of Tapestry [13] which is regarded as the earliest recommender system.

The CF approach is based on the simple idea that users like things that are liked by other users with a similar taste. Hence, it's assumed to be possible that a missing rating of a user for any item can be predicted by taking into account other similar users and their ratings. Gong et al. [14] note that the collaborative filtering (CF) technique has been proved to be one of the most successful techniques in recommender systems.

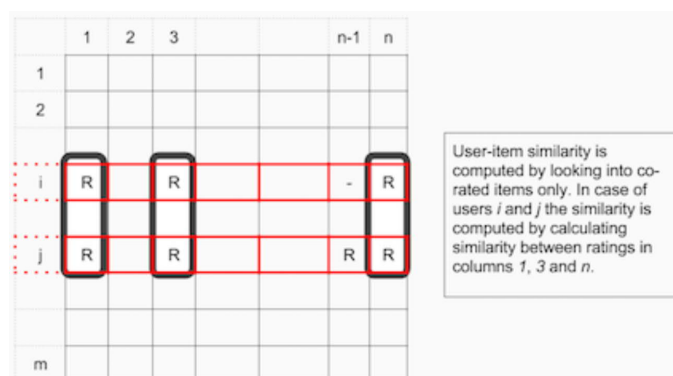


Figure 2.1: User based collaborative filtering [15]

2.4.1 Methods of Collaborative Filtering

CF methods rely entirely on the ratings in the user-item matrix without using any information about either users or items themselves. Hence, the focus is on utilizing inter-item or inter-user correlation when predicting unspecified ratings. In this context, CF methods are categorized into two general classes: model based and memory based.

Memory-based methods are the part of the traditional approach to collaborative filtering, in which the ratings of user-item combinations are predicted based on their neighborhoods. The neighborhoods can be defined in user-based or item-based ways. Memory-based techniques have the advantage that they are simple to implement but they do not work well with real world data-sets and suffer from the issue of sparsity in the data-set. On the other hand, model-based methods utilize data mining and machine learning models for alleviating these issues. Aggarwal in his book *Recommender Systems* [16] writes that model-based methods are better than memory-based methods in regards to space-efficiency, training, prediction speed and in avoidance of over-fitting. Model based methods are also apt to overcome the two most common challenges that CF approach encounters: sparsity and scalability.

2.4.2 Sparsity and Scalability

Papagelis et al. [17] notes that the *sparsity* problem occurs when available data is insufficient for identifying similar users in the neighbourhood of the principal user and it is a major issue that impacts the ability of a recommender system. Since the user base and item inventory of any major E-commerce service is huge, therefore its inevitable that a significant portion

of the interaction matrix of user and items will be empty.

Moreover, large E-commerce sites are facing a constant increase in the number of users and their inventories are also growing by the day. This scaling up of the operations and the growth in the data necessitates an increase in the demand of the computational resources needed for data processing tasks. Lampropoulos et al. [18] describe that the issue of *scalability* refers to the ability of a recommendation method to handle the growth of the data. Any approach whose efficiency subdues when the amount of data grow would be very time-consuming and scale poorly and such a method would fail to achieve the goals of a recommendation system. Thus, it is paramount that any recommendation approach intended for E-commerce data is capable of scaling up in an efficient manner.

2.5 Representation Learning

Representation learning is a branch of Machine Learning that focuses on the extraction of the underlying explanatory factors hidden in data. These learned representations $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^N$ from a dataset with datapoints $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ are desired to be of lower-dimensions, linearly separable and clusterable. Furthermore, the semantic explanation of the dimensions of learned manifold helps in intuitive disentanglement of latent factors present in the data.

As described in the section above, traditional approaches to Collaborative Filtering suffers from the issues of sparsity and scalability. Common approaches such as Principal Component Analysis (PCA) which strives to learn low dimensional representations of data with a linear projection to uncorrelated variables has major drawbacks. For example, the underlying assumption in PCA that the learned factors are linearly related to the directions of maximum variance in the data falls short of being true for complex datasets which are drawn from arbitrarily complicated nonlinear functions. A highly sparse user-item interaction matrix in CF is one such example of a complex dataset.

The most well-known representation learning approaches in CF are Probabilistic Matrix Factorization (PMF) [19] and Singular Value Decomposition (SVD) [20]. We will discuss these approaches and other related work briefly in next chapter. On top of the these approaches, during the past decade the use of Artificial Neural Networks (ANN) for CF is an active line of research [21]. An increasing amount of studies strive to generate effective representations by extracting remarkable latent factors with the use of deep learning while incorporating the models with side information of users and items.

2.5.1 Deep Collaborative Filtering

Yujia Zhang in their latest proposal for Large-Scale Collaborative Filtering [3] notes that the Neural Networks, with their excellent capability of learning representations of data with multiple levels of abstraction are frequently applied into recommender systems. Neural networks are also good at capturing the intricate relationships within the mixed data source consisting of user-item interactions as well as their side-information. Moreover, in their comprehensive review of deep learning methods for recommender systems, Batmaz et al. [4] observe that neural networks now attract significant attention in building recommender systems due to availability of every increasing computational resources and big data processing capabilities.

One such commonly applied deep learning approach is Autoencoders (AE). As a specific form Neural Networks, Autoencoders attempts to reconstruct the input data in the output layer in an unsupervised fashion. They do so by using the representation obtained in the learning process in the hidden layers. A typical autoencoder has a "diabolo" shape and its consists of three parts: the input, the hidden, and the output layer. During the learning phase the network uses the encoder part to progressively reduce an arbitrarily highly dimensional input over layers of progressively shrinking sizes. The last layer from the encoder part is the smallest is size and called bottleneck. Attached to the bottle neck is the decoder network with layers of progressively increasing size until the last layer of the whole network matches the size of the input layer. During the learning phase Network's weights are optimized by minimizing a loss function which captures the error between the given input and the reconstructed output. An illustration of a typical Autoencoder network is given in 2.2

The use of Autoencoders have shown promising results in and various types of autoencoders have been developed in the literature: a denoising autoencoder (DAE) [22] with the motivation of generalization attempts to reconstruct the input by using a noisy source of input, a Stacked Denoising Autoencoder (SDAE) [23] utilizes Probabilistic Matrix Factorization (PMF) with a marginalized denoising autoencoder to learn latent factors.

2.5.2 Variational Inference

In representation learning, we assume that each datapoint x is generated by unknown (latent) variables \mathbf{z} . The problem at hand then is to devise a generative model such that, given the instances of latent variables \mathbf{z} it is able to reproduce the respective datapoints x as well as possible. In this context, inference is concerned with learning the parameters of a generative model

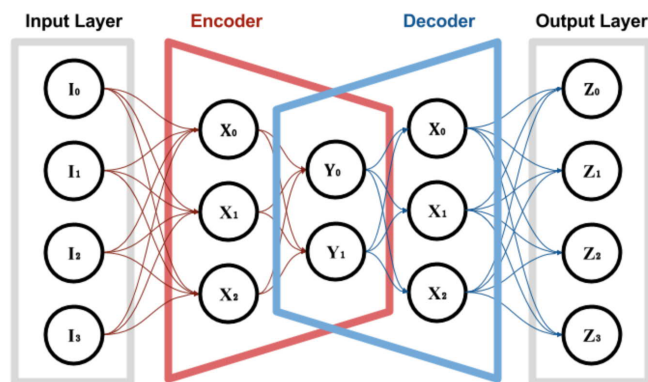


Figure 2.2: An autoencoder [24]

and finding a distribution over \mathbf{z} conditioned by datapoint x .

We start with an initial hypothesis using a prior distribution $p_\theta(\mathbf{z})$ on how the latent variables are distributed under our generative model θ . This prior distribution is then updated to the datapoint x and likelihood $p_\theta(\mathbf{x}|\mathbf{z})$. For this formulation, we can use Bayes rule to infer the posterior distribution of the latent variables \mathbf{z} :

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})}$$

Since $p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})d\mathbf{z}$ is intractable therefore the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is unavailable and we look to find an approximation $q(\mathbf{z}|\mathbf{x})$ using variational inference.

Variational inference (introduced by Jordan et al., 1999 [25] and reviewed by Blei et al., 2018 [26]) aims to minimize the distance between the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ and the approximation $q(\mathbf{z}|\mathbf{x})$. It starts by introducing a family \mathcal{Q} of parametric approximations $q_\phi(\mathbf{z}|\mathbf{x})$ to the true posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$, indexed by ϕ . Variational inference then seek to minimize the Kullback-Leibler divergence, which is a similarity measure for probability distributions

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) = \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

The KL divergance can be decomposed into:

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

Let, $\mathcal{L}(\phi, \theta; \mathbf{x}) = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{x}, \mathbf{z})} \right]$

Summarizing:

$$\log p_\theta(\mathbf{x}) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) + \mathcal{L}(\phi, \theta; \mathbf{x}) \quad (2.2)$$

Here, the term $\mathcal{L}(\phi, \theta; \mathbf{x})$ is called the Evidence (or variational) Lower Bound (ELBO). It is clear in (2.2) that $\log p_\theta(\mathbf{x})$ is fixed w.r.t. \mathbf{z} , and since $D_{KL}(q \| p) \geq 0$: therefore $\mathcal{L}(\phi, \theta; \mathbf{x})$ is a lower bound:

$$\log p_\theta(\mathbf{x}_i) \geq \mathcal{L}(\phi, \theta; \mathbf{x}_i)$$

From this its evident that the maximizing of the ELBO implies necessarily the minimizing of $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x}))$ which is our original goal - minimize the distance between the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ and the approximation $q(\mathbf{z}|\mathbf{x})$. Hence to sum over all data points, our final learning objective in variational inference becomes:

$$\max_{\theta} \sum_{i=1}^N \max_{\phi} \mathcal{L}(\phi, \theta; \mathbf{x}_i)$$

This is the basis of variational inference, to maximize the ELBO in order to find the best possible approximation to the true posterior.

2.5.3 Variational Autoencoder

The Variational Autoencoder (VAE) proposed by Kingma and Welling [5] is a generative model which uses variational inference for an analytical approximation of posterior distribution of the latent variables. Since the true posterior distribution $p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})d\mathbf{z}$ is intractable hence the approximation $q_\phi(\mathbf{z}|\mathbf{x})$ is introduced whose parameters ϕ are learned simultaneously with the parameters θ .

Since the training of an autoencoder relies on forwarding data and backpropagation, we can not have latent variable \mathbf{z} be a random variable sampled from $q(\mathbf{z}|\mathbf{x})$ as it will prevent gradients from flowing in backpropagation. The key contribution of Kingma and Welling [5] is an alternate estimator called Stochastic Gradient Variational Bayes (SGVB), which can circumvent this issue by employing a *reparameterization trick*. This trick involves generating a sample $\epsilon \sim \mathcal{N}(0, I)$ from a Normal distribution and by using this transformation:

$$\hat{\mathbf{z}} = \boldsymbol{\mu}_\phi + \boldsymbol{\sigma}_\phi \cdot \epsilon$$

a sample is obtained from $\mathcal{N}(\boldsymbol{\mu}_\phi, \boldsymbol{\sigma}_\phi)$. Using this we can now obtain gradient of the expectation w.r.t. $q_\phi(\mathbf{z}|\mathbf{x})$ of any function $f(\mathbf{z})$ as:

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[f(\mathbf{z})] = \nabla_\phi \mathbb{E}_{p(\epsilon)}[f(\hat{\mathbf{z}})] = \mathbb{E}_{p(\epsilon)}[\nabla_\phi f(\hat{\mathbf{z}})]$$

This allows us to estimate the ELBO using minibatches, the gradients are computed with automatic differentiation and the values of parameters ϕ and θ are updated using some gradient ascent method such as SGD, RMSProp or Adam. Figure 2.3 illustrates the architecture of a Variational Autoencoder.

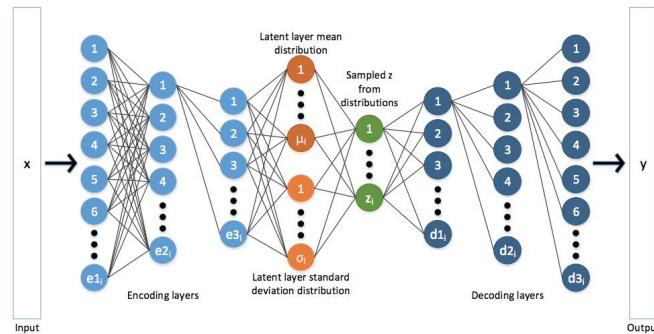


Figure 2.3: A Variational Autoencoder with fully connected layers [27]

Chapter 3

Related work

In this chapter we present a brief overview of methods related to our work.

3.1 Probabilistic Matrix Factorization

Salakhutdinov and Mnih [19] introduced the Probabilistic Matrix Factorization (PMF) approach for Collaborative Filtering problem which performs well on sparse datasets and scales linearly.

PMF approach factorize the sparse data matrix with observed ratings \mathbf{R} into low-dimensional factor matrices \mathbf{U} and \mathbf{V} . It assumes that ratings \mathbf{R}_{ij} of user i for item j are normally distributed around mean $\mathbf{U}_i^\top \mathbf{V}_j$ with a common variance σ^2 . Letting \mathbf{I}_{ij} as an indicator function such that it is equal to 1 if user i has rated the item j or 0 otherwise, the likelihood is given as:

$$p(R|U, V, \sigma^2) = \prod_{i=1}^N \prod_{j=1}^M [\mathcal{N}(R_{ij}|U_i^\top V_j, \sigma^2)]^{I_{ij}} \quad (3.1)$$

Based on this probabilistic assumption, the log-likelihood in PMF then becomes a sum of terms where each term is dependent on observable rating of specific item j by a user i having $\mathbf{I}_{ij} = 1$. This enables gradient descent updates of \mathbf{U}_i and \mathbf{V}_j .

3.2 Matrix Factorizing Variational Autoencoder

Van Balen [28] in their Master's thesis extend on Kingma and Welling [5] and propose Matrix Factorizing Variational Autoencoder (MFVAE) by applying

VAE to Matrix Factorization problem.

MFVAE attempts to learn intractable posterior distribution $p(\mathbf{U}, \mathbf{V}|\mathcal{D})$ with variational approximation $\mathbf{Q}(\mathbf{U}, \mathbf{V})$ over latent user \mathbf{U} and item \mathbf{V} factors by conditioning the variational posterior on the observable data itself:

$$Q_\phi(\mathbf{U}, \mathbf{V}|\mathcal{D}) = \prod_i q_{\phi_u}(\mathbf{u}_i|r_i) \prod_j q_{\phi_v}(\mathbf{v}_j|r_{.j}) \quad (3.2)$$

Where, each q_ϕ is dependent on the observable rating r_i . Thus, the Evidence Lower Bound (ELBO) for MFVAE becomes a sum over the KL divergences of each latent factor and the expected log probability of the ratings:

$$\begin{aligned} \mathcal{L} = & - \sum_i D_{KL} [Q_u(\mathbf{u}_i|r_i) || p(\mathbf{u}_i)] \\ & - \sum_j D_{KL} [Q_v(\mathbf{v}_j|r_{.j}) || p(\mathbf{v}_j)] \\ & + \sum_{i,j \in \mathcal{D}} \mathbb{E}_{Q_u, Q_v} [\log p(r_{ij}|\mathbf{u}_i, \mathbf{v}_j)] \end{aligned}$$

3.3 Neural Collaborative Filtering

He et al. [21] propose *Neural network-based Collaborative Filtering* (NCF) framework which uses a neural architecture to learn an arbitrary function targeted to replace the latent features' inner product in matrix factorization.

NCF attempts to learn the user-item latent factors by employing a multilayer perceptron (MLP). The input layer of network takes in the one-hot encoded user and item feature vectors. It is then connected to a dense embedding layer where sparse user and item representation are projected to a dense vector. This embedding is then fed to a multilayer neural architecture with the output layer giving out the predicted score as following:

$$\hat{r}_{ui} = f(\mathbf{U}^T \cdot \mathbf{v}_u^{user}, \mathbf{V}^T \cdot \mathbf{v}_i^{item} | \mathbf{U}, \mathbf{V}, \theta) \quad (3.3)$$

Where, \hat{r}_{ui} is the predicted rating of user u for item i , \mathbf{v} denotes the one-hot encoded user and item identifier vectors, whereas \mathbf{U} and \mathbf{V} are user and item latent factors respectively; and θ denotes the parameters of the multi-layer neural network represented by $f(\cdot)$.

The NCF method conveniently fuses the neural interpretation of the matrix factorization problem by leveraging the nonlinearity of MLP with the linearity of matrix factorization to enhance recommendations. The whole

network can then be trained with gradient descent like updates to optimize a loss function such as cross-entropy loss for implicit feedback:

$$\mathcal{L} = - \sum_{(u,i) \in R \cup R^-} r_{ui} \log \hat{r}_{ui} + (1 - r_{ui}) \log (1 - \hat{r}_{ui})$$

Chapter 4

Method

In this chapter we present the methodology used to conduct the experiments in this thesis. We outline the model and the algorithm in detail and explain the design choices and the reasoning behind them.

4.1 Model

In this work we employ Mult-VAE: a partially regularized variational autoencoder with multinomial likelihood. The main reason to use a variational autoencoder (VAE) in our approach is that we want to apply recent advances in deep learning research and measure how well they perform on collaborative filtering as compare to traditional matrix factorization approaches. Hence, the use of VAE, as a non-linear probabilistic models enables us to go beyond the limited modeling capacity of linear factor models. We borrow extensively from the approach proposed by Liang et al., 2018 [6] as the idea of this thesis is to evaluate their particular process of using VAE for collaborative filtering on the real world e-commerce datasets. We base our model on their mechanism and try to reproduce the results on our datasets as their findings have revealed that this approach improves the recommendation quality immensely.

Under this system, we learn from explicit user feedback where user has clicked an item. We use $u \in \{1, \dots, \mathcal{U}\}$ to index users and $i \in \{1, \dots, \mathcal{I}\}$ to index items, hence the user-item interaction matrix becomes $X \in \mathbb{N}^{\mathcal{U} \times \mathcal{I}}$. The bag-of-words vector containing number of clicks by user u for each item is $\mathbf{x}_u = [x_{u1}, \dots, x_{u\mathcal{I}}]^\top \in \mathbb{N}^{\mathcal{I}}$. We also binarize the click matrix for simplicity.

The generative process is the same where for each user u we sample a K -dimensional latent representation \mathbf{z}_u from a standard Gaussian prior. A non-linear function $f_\theta(\cdot) \in \mathbb{R}^{\mathcal{I}}$ is used to transform this latent representation \mathbf{z}_u to produce a probability distribution over I items $\pi(\mathbf{z}_u)$. We assume that

the click history \mathbf{x}_u is drawn from this produced probability distribution $\pi(\mathbf{z}_u)$. Our model thus becomes:

$$\mathbf{z}_u \sim \mathcal{N}(0, \mathbf{I}_K), \quad \pi(\mathbf{z}_u) \propto \exp\{f_\theta(\mathbf{z}_u)\}, \quad \mathbf{x}_u \sim \text{Mult}(N_u, \pi(\mathbf{z}_u)) \quad (4.1)$$

Here, the non-linear function f is an MLP with parameters θ . We normalize the output of f_θ by a softmax function which produces the probability vector $\pi(\mathbf{z}_u)$ over the entire item set. Hence the observed vector \mathbf{x}_u containing number of clicks by user u for each item is assumed to be sampled from a multinomial distribution with probability $\pi(\mathbf{z}_u)$ given the total number of clicks $N_u = \sum_i \mathbf{x}_{ui}$ from user u . Essentially then the log-likelihood for a user u conditioned on the latent representation \mathbf{z}_u becomes:

$$\log p_\theta(\mathbf{x}_u | \mathbf{z}_u) \stackrel{c}{=} \sum_i x_{ui} \log \pi_i(\mathbf{z}_u) \quad (4.2)$$

This multinomial likelihood as reasoned by Liang et al., 2018 [6] is well suited to model click data, mainly due to the process by which model distributes the probability mass to entries in \mathbf{x}_u . Since the model has a limited budget of probability mass ($\pi(\mathbf{z}_u)$ must sum to 1) the items in \mathbf{x}_u would compete for this limited budget and therefore the items who are more likely to be clicked should be assigned more probability mass by the model. This probability mass reward mechanism is well suited to perform better [4] in a setting of the top-N recommender system in which such models are commonly evaluated.

4.2 Amortized Variational Inference

In order to learn the model in (4.1) we want to estimate the parameters θ of our non-linear transformation function f . To do so we will use variational inference (as described in 2.5.2) to approximate the intractable posterior distribution $p(\mathbf{z}_u | \mathbf{x}_u)$ with a variational distribution $q(\mathbf{z}_u)$. In our case, as setup by Liang et al., 2018 [6] we put $q(\mathbf{z}_u)$ to be a diagonal Gaussian distribution with free variational parameters $\{\boldsymbol{\mu}_u, \boldsymbol{\sigma}_u^2\}$

$$q(\mathbf{z}_u) = \mathcal{N}(\boldsymbol{\mu}_u, \text{diag}\{\boldsymbol{\sigma}_u^2\})$$

The objective of variational inference is to minimize the Kullback-Lieber divergence between the approximated $q(\mathbf{z}_u)$ and the intractable $p(\mathbf{z}_u | \mathbf{x}_u)$. However, for this setting of variational inference the number of parameters $\{\boldsymbol{\mu}_u, \boldsymbol{\sigma}_u^2\}$ (which we need to optimize) grow linearly with the size of the

dataset. This is far from ideal for massive datasets and can create performance bottlenecks. It comes one of the benefits of using a Variational Autoencoder (described in 2.5.3) where we employ a data dependent inference model parameterized by ϕ to replace these individual variables of $\{\boldsymbol{\mu}_u, \boldsymbol{\sigma}_u^2\}$.

$$q_\phi(\mathbf{z}_u | \mathbf{x}_u) = \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}_u), \text{diag}\{\boldsymbol{\sigma}_\phi^2(\mathbf{x}_u)\})$$

This entails that the inference model based on parameters ϕ takes data \mathbf{x}_u as input and outputs the parameters of the variational distribution $q_\phi(\mathbf{z}_u | \mathbf{x}_u)$. Notice that this solves the core issue of high number of variational parameters, as now the variational parameters ϕ are constant w.r.t. to the input data and hence the variational inference is *amortized*. Moreover, Liang et al., 2018 [6] also note that the mechanism of amortized inference fits well with idea of collaborative filtering as it flexibly reuses the inference model to fit it to new queries, thus essentially manipulating patterns from the past behaviour (data) to infer similarities.

4.3 Evidence Lower Bound (ELBO)

We now turn our attention to the objective we seek to maximize while learning the Variational Autoencoder. To be exact, we want to find the best possible approximation to the true intractable posterior. As described in detail in 2.5.2 and asserted in (2.2), we use the lower bound to the log marginal likelihood of the data also called as ELBO (evidence lower bound) as a standard objective.

In our current context, we want to maximize ELBO for each user u and then eventually average it out over the whole dataset.

$$\begin{aligned} \log p(\mathbf{x}_u; \theta) &\geq \mathbb{E}_{q_\phi(\mathbf{z}_u | \mathbf{x}_u)} [\log p_\theta(\mathbf{x}_u | \mathbf{z}_u)] - \text{KL}(q_\phi(\mathbf{z}_u | \mathbf{x}_u) || p(\mathbf{z}_u)) \\ &\equiv \mathcal{L}(\mathbf{x}_u; \theta, \phi) \end{aligned} \quad (4.3)$$

Here the ELBO $\mathcal{L}(\mathbf{x}_u; \theta, \phi)$ is parameterized by both θ , the parameters of our non-linear transformation function f_θ and by ϕ , the parameters of the amortized inference model. Moreover, the ELBO here is made out of two terms, the first is the reconstruction error and the second regularization KL divergence term.

Under this perspective, Liang et al., 2018 [6] contributes their core idea (which essentially translates their approach into being the most optimal in practice): they introduce a free regularization parameter β which controls the strength of regularization done through the second KL term in the ELBO

$$\mathcal{L}_\beta(\mathbf{x}_u; \theta, \phi) \equiv \mathbb{E}_{q_\phi(\mathbf{z}_u | \mathbf{x}_u)} [\log p_\theta(\mathbf{x}_u | \mathbf{z}_u)] - \beta \cdot \text{KL}(q_\phi(\mathbf{z}_u | \mathbf{x}_u) \| p(\mathbf{z}_u)) \quad (4.4)$$

The KL regularization term basically ensures how close the approximate distribution remains close to the prior. By introducing $\beta < 0$ as a control parameter for it, Liang et al., 2018 [6] are no longer optimizing for the marginal log likelihood of data but essentially controlling the impact of prior. Another important reasoning behind using $\beta < 0$ is that the ultimate goal of this model is to be able to recommend the most relevant items to users and not to generate and reconstruct the perceived similarities based on ancestral sampling of user interaction behaviour. Therefore the proposal to keep $\beta < 0$ which weakens the dependency on the prior have shown to contribute immensely to improved performance against recommender system validation metrics. We will dwell more upon on how we have settled on selecting the optimal value of β in the experimental setup section in 5.3.

4.4 Training Variational Autoencoder

The neural architecture of an autoencoder involves an encoder (inference model) and a decoder (generator). Figure 4.1 as presented by Liang et al., 2018 [6] summarize the process of variational autoencoder in which we use the observed data \mathbf{x}_u as input and employ the inference model to get the latent variable \mathbf{z} . Then we use the generator to reconstruct the input while optimizing the whole process based on the objective function (parameters θ and ϕ).

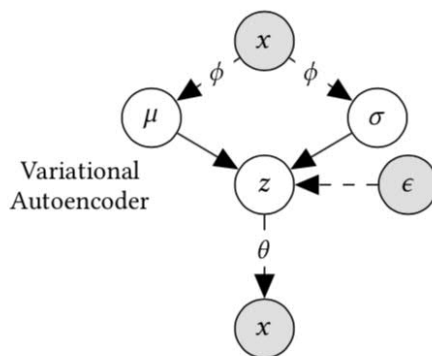


Figure 4.1: Variational Autoencoder architecture, Liang et al., 2018 [6]

The standard way of learning in Variational Autoencoders as described in 2.5.3 is to maximize the ELBO which is the objective function. However, in our case, equation (4.4) lays down the objective function for a user u as a function of θ , ϕ and the free regularization parameter β . For a given dataset, we obtain the overall objective function by averaging the objective function over the whole user base. Since the training of an autoencoder involves back propagation therefore we can not trivially sample \mathbf{z} from $q(\mathbf{z}|\mathbf{x})$ as it will prevent from taking gradients w.r.t ϕ during back propagation. We resolve this issue by using the *reparameterization trick* introduced by Kingma and Welling [5]: sample $\epsilon \sim \mathcal{N}(0, I)$ from a Gaussian distribution and reparameterize \mathbf{z} by using this transformation: $\hat{\mathbf{z}} = \boldsymbol{\mu}_\phi + \boldsymbol{\sigma}_\phi \cdot \epsilon$. This allow us to isolate the sampling process and we can take gradients w.r.t ϕ in back propagation.

The overall training algorithm for this **Mult-VAE** model which is partially regularized and has a multinomial likelihood is the same as summarized by Liang et al., 2018 [6] and as is presented in Algorithm 1.

Algorithm 1: Training VAE-SGD for collaborative filtering, Liang et al., 2018 [6]

Input: Click matrix $X \in \mathbb{R}^{\mathcal{U} \times \mathcal{I}}$
Initialize θ and ϕ randomly; **while** *not converge* **do**
 sample a batch from users \mathcal{U} ;
 forall $u \in \mathcal{U}$ **do**
 Sample $\epsilon \sim \mathcal{N}(0, \mathbf{I}_K)$ and compute \mathbf{z}_u using reparameterization trick;
 Compute gradients $\nabla_\theta \mathcal{L}$ and $\nabla_\phi \mathcal{L}$ with \mathbf{z}_u ;
 end
 For the whole batch, take mean of noisy gradients;
 Take stochastic gradient steps and update θ and ϕ ;
end
return θ, ϕ ;

Chapter 5

Empirical Analysis

In this chapter we describe in detail the experimental setup in which we empirically analyse and compare our models. We define the attributes of the datasets at hand and dwell upon the evaluation metrics used to determine the results. We finally interpret the results and report our findings.

5.1 Datasets

In our empirical analysis, we consider two real-life medium to large scale datasets from E-commerce online stores. For the sake of anonymity we will call them *Dataset-X* and *Dataset-Y*. These datasets comprises of user visits¹ to the products on the website. Both data sets are extracted by using Frosmo Oy tools of data tracking (as described in 2.2) and consist of anonymized user ids and item ids in the form of tuples: $(userid, itemid)$.

The datasets are primarily in the from of user-item interaction where each row has an entry for the user and item identifier. For the purposes of this work we have anonymized these identifiers. We also transform this tabular user-item interaction data into binarized user-item click matrices where the rows consists of the users and the columns marks the items. As true with most real-life datasets, the transformed user-item matrices has high sparsity and its one of the issues our work intend to resolve. The highlevel attributes of the datasets after preprocessing are presented in Table 5.1.

Dataset-X: This dataset is extracted from an online electronics store based out of Helsinki, Finland. User interactions in this dataset span from the period of 1st of January 2020 until 15th of March 2020. We have removed the users who have interacted with less than 5 products.

¹A direct click on a product or an indirect landing to a product page from an external source is considered as a *visit*

	Dataset-X	Dataset-Y
# of unique users	132,561	136,269
# of unique items	17,486	18,994
# of interactions	~ 4.0M	~ 6.0M
% of interactions	0.17%	0.23%
# of held-out validation users	10,000	10,000
# of held-out test users	10,000	10,000

Table 5.1: **Dataset attributes:** % of interactions refer to the user-item click matrix density.

Dataset-Y: This dataset is extracted from an online children clothing store based out of Vantaa, Finland. User interactions in this dataset span from the period of 1st of January 2020 until 30th of March 2020. Similar to *Dataset-X*, we have removed the users who have interacted with less than 5 products.

5.2 Evaluation Metrics

To empirically evaluate the models we employ two ranking based evaluation metrics: normalized discounted cumulative gain (NDCG@K) and Recall@K. We use both of these metrics for validation and testing purposes. The reason to use ranking metrics is to make sure that our model is performing well on the objective of recommending relevant items with appropriate ranking. Thus essentially we treat our model as a recommendation system and evaluate it based on the measure of ranking quality and relevance.

5.2.1 Normalized Discounted Cumulative Gain (NDCG)

Normalized discounted cumulative gain essentially measures the ranking quality of a recommendation list. As noted by, Järvelin et al. in their paper [29] on cumulated gain based evaluation methods, the underlying idea behind NDCG is two fold. Firstly, we assume that highly relevant items are more valuable to us and secondly, the lower an item is in the ranked list the less valuable it becomes because the likelihood of user engagement with a lower ranked item is also low. The technique to compute discounted cumulative gain is to employ a monotonically increasing discount to amplify the value of higher ranked items in contrast to the lower ranked items.

Formally we defined discounted cumulative gain for a user u as:

$$\text{DCG}@K_u := \sum_{i=1}^K \frac{2^{\mathbb{I}[\omega(i) \in I_u]} - 1}{\log_2(i + 1)}$$

Here \mathbb{I} is an indicator function, which is 1 if item $\omega(i)$ at rank i belongs to the held-out set of items I_u , otherwise the indicator is 0. The denominator in the expression entails a increasing value w.r.t to the rank and thus discounting the gain value in numerator and hence computing the discounted cumulative gain.

We then compute $\text{NDCG}@K$ by dividing the $\text{DCG}@K$ by the ideal $\text{iDCG}@K$ (in which all items in held-out set I_u have top ranks) which naturally also linearly normalized this value to $[0, 1]$. To report the metric we take a mean across all the recommendations for a user u to arrive at a final $\text{NDCG}@K$ score.

5.2.2 Recall

Recall can be simply described as the fraction of relevant items recommended by the system. It essentially measures the ability of the recommender system to find the relevant results. However, unlike $\text{NDCG}@K$ it does not take into consideration the ranking of the items but still is a very meaningful metric in the context of a recommendation system where we need to evaluate the relevance of the recommended items.

Formally, in our case $\text{Recall}@K$ for a user u is defined as:

$$\text{Recall}@K_u := \frac{\sum_{i=1}^K \mathbb{I}[\omega(i) \in I_u]}{\min(K, |I_u|)}$$

Here the denominator is the minimum between the K and the length of the held-out items list. This helps in normalizing the value of $\text{Recall}@K$ to $[0, 1]$.

5.3 Experimental Setup

5.3.1 Baselines

We outline our baseline results using two matrix factorization approaches: Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF). Both of these methods are currently employed by Fromso Oy for collaborative filtering problem and we want to compare the results of our Mult-VAE model with these baselines in order to establish if Mult-VAE performs better than the current setup in Fromso Oy.

5.3.1.1 Singular Value Decomposition

Matrix factorization methods strive to model the user and item interactions as an inner product to a joint latent factor space of a lower dimensionality [30]. Singular value decomposition (SVD) is a matrix factorization model which basically solves the key problem of finding a lower dimensional feature space of the user-item rating matrix.

For collaborative filtering, SVD [31] tends to transform the ratings matrix A into factors UWV^T where U represents the user interaction to each latent feature. V represents the quantity of each latent feature related to each item and K is a matrix representing the importance of each latent feature in the overall determination of the user-item rating. The W matrix is a diagonal matrix containing the singular values of A .

This factorization enables us to recompute any particular rating for an item by any user using the U , W , and V matrices as follows:

$$A = UWV^T$$

$$A_{ij} = \sum_k \sum_l U_{ik} W_{kl} (V^T)_{lj}$$

$$A_{ij} = \sum_k \sum_l U_{ik} W_{kl} V_{jl}$$

here U_{ik} represent the user i 's interaction with the feature k , W_{kl} as the relevance of latent feature k , and V_{jl} as the determination of latent feature l present in item j .

In this work, use SVD on collaborative filtering problem for both datasets (described in 5.1) and evaluates the model on NDCG@100, Recall@20 and Recall@50 metrics (described in 5.2) to establish the baselines.

5.3.1.2 Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) is a group of algorithms in multivariate analysis where a matrix A is factored into two matrices with the property that all three matrices have no negative elements. Lee and Seung [32] introduced this approach which as compared to other matrix factorization approaches take advantage of the fact that most real world data is non-negative like images or videos and maintain such constraints in factorization.

A collaborative filtering method based on Non-negative matrix factorization is very similar to SVD described above. The prediction \hat{r}_{ui} is set as:

$$\hat{r}_{ui} = q_i^T p_u$$

Stochastic gradient descent (SGD) is then used to update the factors f of user u and item i as follows:

$$p_{uf} \leftarrow p_{uf} \cdot \frac{\sum_{i \in I_u} q_{if} \cdot r_{ui}}{\sum_{i \in I_u} q_{if} \cdot \hat{r}_{ui} + \lambda_u |I_u| p_{uf}}$$

$$q_{if} \leftarrow q_{if} \cdot \frac{\sum_{u \in U_i} p_{uf} \cdot r_{ui}}{\sum_{u \in U_i} p_{uf} \cdot \hat{r}_{ui} + \lambda_i |U_i| q_{if}}$$

here, λ_i and λ_u represent the regularization parameters [33].

5.3.2 Data Preprocessing

We preprocess each dataset by removing all users whose click history contains interactions with less than five items. The reasoning behind this is to remove the noise from the datasets. Furthermore, we do a random shuffle of each dataset and split all users into training, validation and test sets. Both the validation and test sets for each dataset comprises of 10,000 held-out users as listed in the last row of Table 5.1. These train/validation/test sets are split in such a way that each is disjoint from one other and all of the three combined makes the whole dataset. We don't perform any further preprocessing on the datasets.

5.3.3 Hyperparameter Optimization

During our experiments we run a finite random grid search on a various ranges of different hyperparameters. For all the hyperparameter tuning we using the validation metric $\text{NDCG}@100$ as the criteria for the best performance. However, due to the large size of the datasets we were not able to perform a wide search operation but instead relied on the heuristics and intuitions provided by Liang et al., 2018 [6]. For the regularization parameter β (as described in the section 4.3) we gradually anneal the KL term with increasing value of β starting from 0 and aiming for 1. We found that the peak performance against the validation metric $\text{NDCG}@100$ occurs at $\beta = 0.23$ hence we cap the annealing of the KL divergence term in the objective function (4.4) to $\beta \leq 0.23$ as it performs the best and doesn't further explore it in subsequent search.

We also included informative results from each search into the next for example we found that similar to [6], the dimensions of the hidden layer as 600 is the best and did not delve into it further. We used Adam [34] as the optimizer and found the learning rate of 0.001 to be the best. We did not apply any weight decay to the optimization process. Moreover, we also observed that the 1-hidden-layer architecture of the MLP generative model $f_{\theta}(\cdot) [I \rightarrow 600 \rightarrow 200 \rightarrow 600 \rightarrow I]$ with the latent layer dimension of 200

is the best performing one and adding more layers and going deeper doesn't impact the performance.

5.3.4 Training

Since we are operating under the assumption of strong generalization therefore we train the models on the full click history of the users in the training data. However, for each held-out user, we randomly pick 20% of the items from their click-vector to report the metrics and the remaining 80% of the items are used to reconstruct the click vector representation from the learned model. We do so in order to avoid weak generalization where items from the user's click vector from training sets can leak into the evaluation process. As shown in Table 5.1 we use the same number of held-out users for both testing and validation sets.

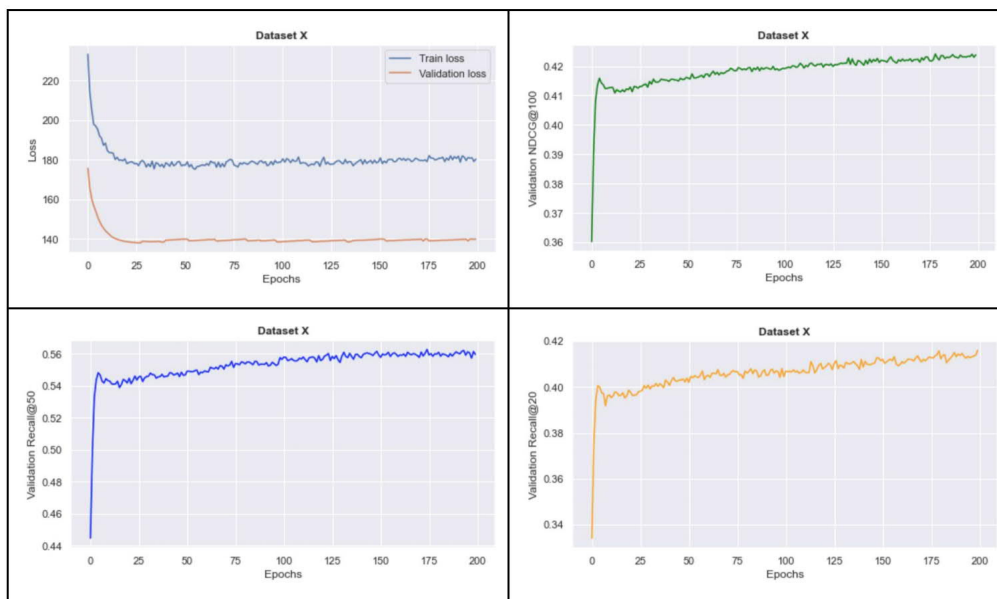
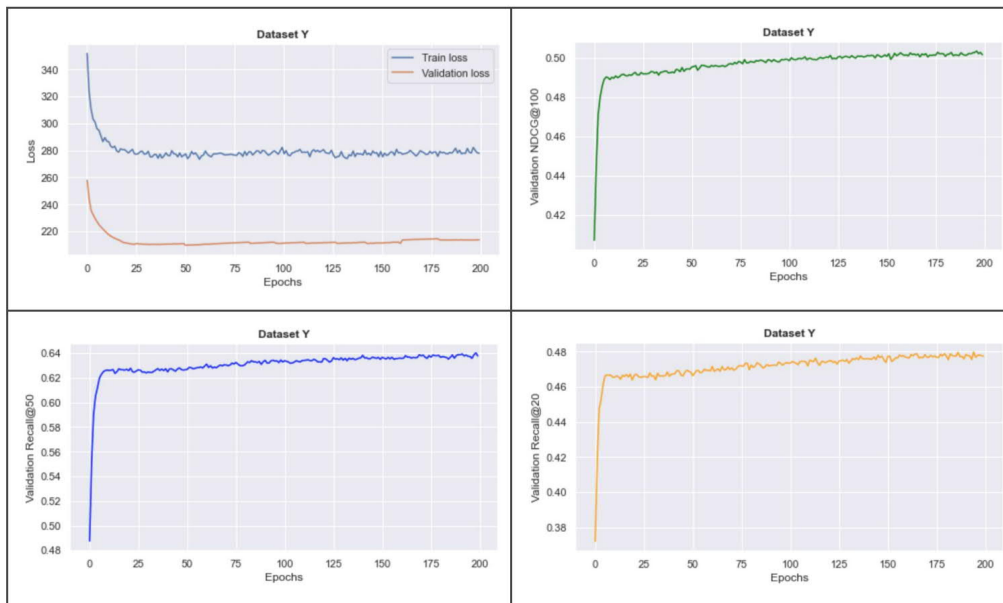


Figure 5.1: **Dataset-X**: Training progress

In the model training process, we constrain the linear annealing of KL term with $\beta \leq 0.23$ to 200,000 gradient updates. We used Adam [34] as the optimizer and perform 200 epochs with the batch size of 500 for each dataset. During training epochs we use the held-out validation set to evaluate the models and keep the best one based on $NDCG@100$ metric. For Dataset-X

Figure 5.2: **Dataset-Y**: Training progress

and Dataset-Y figure 5.1 and 5.2 respectively show the training progress and the reported validation metrics for each epoch.

5.4 Results and analysis

We now present the quantitative results based on the best kept model during the training phase. The best model is the one which has the highest value of the normalized discounted cumulative gain ($NDCG@100$) metric during the training epochs. Hence essentially we prefer a model based on our purposed method which performs the best in terms of ranking the items better for the userbase. However, we employ all three evaluation metrics to compare the results with the various baseline methods on both datasets.

Given a trained model in the form of 4.1, we make predictions based on user click history \mathbf{x} and multinomial probability $f_{\theta}(z)$. The latent representation \mathbf{z} is obtained as the mean of the variation distribution of the learned model in the form of $z = g_{\phi}(x)$. Once we have the learned model, we use the held-out training set and perform the process of using the 80% fold-in click vector to reconstruct the necessary representations and evaluate the model on how well the rest of the 20% is ranked and recalled. We then take a simple mean of the metrics based on the predictions across all test set users

Dataset-X			
	NDCG@100	Recall@50	Recall@20
Multi-VAE	0.474	0.613	0.421
SVD	0.397	0.495	0.366
NMF	0.405	0.541	0.377

Dataset-Y			
	NDCG@100	Recall@50	Recall@20
Multi-VAE	0.533	0.665	0.504
SVD	0.331	0.485	0.394
NMF	0.472	0.583	0.414

Table 5.2: Comparison of Multi-VAE method with various baselines

and report the conclusive findings for both Dataset-X and Dataset-Y.

In Table 5.2 we summarize the quantitative comparison between the baselines and our proposed method of partially regularized variational autoencoder with multinomial likelihood. We analyse these results in the light of two fundamental aspects of our approach; the multinomial likelihood and the probabilistic non-linear nature of the variational autoencoder. We argue in section 4.1 that the multinomial likelihood is a good choice for the top- N ranking models as it distributes the probability mass by the likelihood of the item getting visited hence it gives excellent empirical results on the NDCG metric when modelling click data.

Moreover, we also argue that the traditional matrix factorization approaches such as in our baselines: SVD and NMF employ inner matrix product to model user-item interactions, this limits their power of capturing non-linearity in the data and also makes them suffer with sparsity and scalability issues. However, as evident with the results, the variational autoencoder outperforms the baseline methods significantly. This reinforces our idea that non-linear probabilistic models succeed well in capturing the patterns present in the sparse data, specially for the collaborative filtering problem. Lastly, we also attribute the success of variational autoencoder to the inherit stronger modelling assumptions it possesses. Since the model learns the latent representations of the click history of user, therefore it is better able to generate results when there is sparsity involved. In the light of these results, we can attribute this behaviour of Multi-VAE performing better on the users with fewer interactions to its strong generative power of eliciting value from the learned latent representations.

Chapter 6

Conclusion

In this thesis, we set out to evaluate a probabilistic deep learning based approach on the collaborative filtering problem. We specifically wanted to test a partially regularized variational autoencoder on real-life E-commerce datasets with a multinomial likelihood. At our disposal, we had the approach laid out by Liang et al., 2018 [6] in their collaborative filtering paper and we also had access to the real-life datasets collected by using Frosmo Oy tools. The task was then to reproduce the excellent empirical results showed by Liang et al., 2018 [6] using our datasets.

We started out with research on the E-commerce domain and its background and what kind of role recommendations play in the online retail business. We then moved on to examine the collaborative filtering technique and established how it has been the most powerful approach used by various kinds of recommendation systems. We also extended on the issues of sparsity and scalability that the collaborative filtering method commonly face. We then turned our attention to the domain of representation learning and described autoencoders and their generative modelling. We reflected upon the idea of variational inference and how it can help approximate the intractable posterior distribution. This path lead us to the formal definition of variational autoencoder and the reparameterization trick which enables the deep modelling architecture with back-propagation.

We moved on to present a brief overview of the related methods to our approach and outlined matrix factorization and neural collaborative filtering methods. We then dived deep into our own approach and laid down the methodology we used to conduct the experiments in detail. We described the model and how the multinomial likelihood is well suited for our case. We discussed the role of amortized inference in our architecture and how partially regularizing a term in the ELBO results in better performance. We formally presented our Mult-VAE model and the training algorithm.

After laying down the theoretic understating of our approach we moved on to perform experiments and provided an empirical analysis based on the two datasets at our disposal. We described the attributes of both datasets and discussed in detail the preprocessing steps performed to ready the data for experiments. We outlined the evaluation metrics in detail which we use to validate our model and then described the baseline methods briefly with which we want to compare the results. We setup the training of the models and presented the methodology for hyperparameter optimization and the annealing of the KL term. We illustrated the training progress based on the evaluation metrics and kept the best model for testing.

Finally we implemented the best approach and computed results. We compared our metrics with the baselines and established that our Mult-VAE model performs significantly better than the baselines. Lastly we managed to ascertain that the approach proposed by Liang et al., 2018 [6] can indeed be generalized and employed meaningfully to achieve satisfactory results on real-life E-commerce datasets.

6.1 Future work

The work done in this thesis has substantial potential for extension. Various recent advancements in collaborative filtering research can be assessed for the feasibility to improve the overall performance of our approach. More work needs to be done to solidify the scope with diversified evaluation techniques. Furthermore, it would be justified to have a separate development endeavor to productize our model for industry deployment.

One of the core areas the future work might focus on is to include the side information of the products and items in the latent representations of the user-item interaction. In their recent paper Deng et al., 2019 [35] have focused on incorporating user and item side information in the generative process of the autoencoder and have shown convincing results on benchmark datasets. Gupta et al., 2018 [36] also propose a similar approach where they use the item embeddings to augment the encoding capability of the autoencoder and consequently empower the representation learning approach for collaborative filtering.

Strictly remaining in the scope of our work, one obvious addition we can work on in future is to analyse our model using different likelihood functions and examine in detail the impact of the regularization parameter β on the annealing process. Another challenge that needs focus is how to integrate the element of surprise and serendipity in the final recommendations for a user. Its a difficult problem to quantify the serendipity and fuse it in the

evaluation procedure in order for make the model learn for it, however its worthy of attention and a continued effort should be done on this topic.

Moreover, since this work is done as a research project in industry and a precursor to a real world productize service, we plan to implement it on scale in production. This work in future will involve building the necessary scaffolding of data ingestion and processing pipelines around the model. Taking a project from the experimentation phase to production environment also mandates setting up of maintenance protocols and how the live project would evolve with service level changes and explicit user feedback.

Bibliography

- [1] United nations conference on trade and development: Covid-19 and e-commerce. https://unctad.org/system/files/official-document/dtlstictinf2020d1_en.pdf. Accessed: 2020-10-26.
- [2] Statista: Ikea worldwide visits. <https://www.statista.com/statistics/241832/number-of-visits-to-ikeacom-worldwide>. Accessed: 2020-10-26.
- [3] Yujia Zhang, Jun Wu, and Haishuai Wang. Neural binary representation learning for large-scale collaborative filtering. *IEEE Access*, 7:60752–60763, 2019.
- [4] Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kaleli. A review on deep learning for recommender systems: challenges and remedies. *Artificial Intelligence Review*, 52(1):1–37, 2019.
- [5] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [6] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, pages 689–698, 2018.
- [7] Nitish Singh, Hadi Alhorr, and Boris Bartikowski. Global e-commerce: A portal bridging the world markets. *Journal of Electronic Commerce Research*, 11, 01 2010.
- [8] Frosmo solutions. <https://frosmo.com/solution/>. Accessed: 2020-10-15.
- [9] Fidel Cacheda and Javier Parapar. Information retrieval and recommender systems. *J. UCS*, 21(13):1706–1707, 2015.

- [10] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.
- [11] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.
- [12] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.
- [13] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.
- [14] SongJie Gong, HongWu Ye, and HengSong Tan. Combining memory-based and model-based collaborative filtering in recommender system. In *2009 Pacific-Asia Conference on Circuits, Communications and Systems*, pages 690–693. IEEE, 2009.
- [15] Implementing your own recommender systems in python. <https://blog.cambridgespark.com/nowadays-recommender-systems-are-used-to-personalize-your-experience-on-the-web-telling-you-what-120f39b89c3c>. Accessed: 2020-10-26.
- [16] Charu C Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.
- [17] Manos Papagelis, Dimitris Plexousakis, and Themistoklis Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. In *International conference on trust management*, pages 224–239. Springer, 2005.
- [18] Aristomenis S Lampropoulos and George A Tsihrintzis. Machine learning paradigms. *Applications In Recommender Systems. Switzerland: Springer International Publishing*, 2015.
- [19] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.
- [20] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science, 2000.

- [21] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.
- [22] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.
- [23] Sheng Li, Jaya Kawale, and Yun Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 811–820, 2015.
- [24] An introduction to neural networks and autoencoders. <https://www.alanzucconi.com/2018/03/14/an-introduction-to-autoencoders>. Accessed: 2020-06-13.
- [25] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.
- [26] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.
- [27] Understanding variational autoencoders. <https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>. Accessed: 2020-07-14.
- [28] Mart van Baalen. Deep matrix factorization for recommendation. *Master's thesis, University of Amsterdam, the Netherlands*, 2016.
- [29] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.
- [30] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
- [31] Michael H Pryor. The effects of singular value decomposition on collaborative filtering. *Dartmouth College Undergraduate Theses*, 1998.
- [32] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

- [33] Dheeraj kumar Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay. Role of matrix factorization model in collaborative filtering algorithm: A survey. *CoRR*, *abs/1503.07475*, 2015.
- [34] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [35] Xiaoyi Deng, Fuzhen Zhuang, and Zhiguo Zhu. Neural variational collaborative filtering with side information for top-k recommendation. *International Journal of Machine Learning and Cybernetics*, 10(11):3273–3284, 2019.
- [36] Kilol Gupta, Mukund Yelahanka Raghuprasad, and Pankhuri Kumar. A hybrid variational autoencoder for collaborative filtering. *arXiv preprint arXiv:1808.01006*, 2018.