Variational Autoencoders for Recommender Systems

Shaelin Naidoo

218003390@stu.ukzn.ac.za

University of KwaZulu-Natal,

School of Mathematics, Statistics, and Computer Science

# Abstract

Collaborative Filtering is a popular technique in Recommender Systems. However, until very recently this field has been dominated by linear models like SVD++. Recent research such as RecVAE has highlighted the potential that variational autoencoders have for producing good quality Recommender Systems. Although literature has shown that SVD++ performs better than most DL-based solutions on Recommender Systems, some studies have shown that VAE-based solutions to Recommender System can be just as good as SVD++. Given that SVD++ is linear model, this study hypothesizes that a VAE-based solution equipped with implicit feedback has the potential of outperforming SVD++. In this project, a VAE model is developed which implements implicit feedback, and is evaluated on the Yelp and the MovieLens datasets. Recall@K and Normalized Discounted Cumulative Gain (NDCG) are calculated for the model and assessed using the performance of SVD++ on the same data as a baseline. The results obtained showed that SVD++ outperformed the proposed VAE-based solution based on both the Recall@K and Normalized Discounted Cumulative Gain metrics.

# Introduction

Recommender Systems are an integral part of the modern web, contributing hugely to the user friendliness of it. While online services such as Yelp can contain millions of individual products, services or other items, Recommender Systems improve the user experience by providing a user with a small, manageable subset of these items which are the most likely to be of interest to them. This prevents users from being inundated by information. Recommender Systems allow users to efficiently access the content that they are interested in, without needing to search through huge stores of information (Luo T, et. al. 2013). One of the most prominent approaches to Recommender Systems is Collaborative Filtering (Xian et. al. 2017).

Collaborative Filtering (CF) has been an important and successful means of producing a Recommender System (Luo T, et. al. 2013). In a nutshell, it analyses preference trends across all users. It recognizes user similarities based on ratings histories for all users. For any user, it then recommends to them items which are rated highly by similar-minded users. To be more comprehensive, CF can be split into neighborhood-based, latent factor based, graph based, and social-network based models (Luo T, et. al. 2013). For this project, the latent factor approach is of interest. When there is a small number of items compared to users, Latent factor CF can be used to model either the similarity between items based on user preference, or the similarity between users based on item preference (Luo T, et. al. 2013). User similarity models cannot make recommendations to new users, while item similarity models cannot make recommendations of new items, until the model is retrained or updated (Luo T, et. al. 2013).

Like many machine learning domains, Recommender Systems operate on a sparse dataset: i.e., a matrix where most cells are empty (Liang et. al. 2018). Machine learning algorithms that address the Collaborative Filtering problem must be able to recognize latent factors in the sparse user-item ratings matrix, which are then used to predict ratings for the items the user has not rated.

Collaborative filtering can be done with explicit or implicit feedback. Explicit feedback refers to actual ratings left by users regarding the items, where higher values express higher levels of user preference for the item. Implicit feedback can take many forms, from social networks, to search and purchase history. For all models in this paper, implicit feedback will be extracted from the explicit ratings themselves, bearing in mind that there is an important, latent piece of information within every explicit rating apart from the numeric value that indicates preference or dislike. This latent information is that the user had at some time in the past expressed enough interest in the item to provide a rating to the system. As such, it is reasonable to treat all review events, even ones with low ratings, as an indication of interest of the user for item rated. The reasoning is that the users would not leave reviews of items that fall into categories that do not interest them at all. Therefore, even a bad review shows that the broader category in which the item falls can be considered a subject of interest for the user.

Deep Neural Networks (DNNs) are a powerful machine learning technology. They can model complex non-linear relationships in data. However, they are not the dominant solution to the Collaborative Filtering problem in the real world, despite the fact that evidence for their suitability for use in this domain is mounting. Oftentimes, matrix factorization techniques such as SVD++ are preferred since they produce good results despite their simplicity. SVD++, is based on the mathematical principle of Singular Value Decomposition of Matrices (Koren, Bell & Volinsky. 2009).

The feature that distinguishes it from its simpler matrix factorization ancestors, is the concept of implicit feedback: SVD++ does not only use explicit ratings in order to gauge a user’s preferences. It also uses the fact that a user rated some product at all, as implicit feedback implying preference for that product (Koren, Bell & Volinsky. 2009). This vastly increases the scope of data available to the algorithm.

While SVD++ is powerful, it is a linear model, and linear models are only able to learn linear functions. For this reason, it is hypothesized that Recommender Systems built from (non-linear) Deep Neural Networks should have better recommendation efficacy than those built from Linear models, since they are able to detect patterns that are beyond the reach of linear models. It might be the case that if Variational Autoencoders were equipped to handle implicit feedback, that they could outperform the SVD++ algorithm. This study therefore developed a Variational Autoencoder to solve the collaborative filtering problem based on implicit feedback. The VAE is trained on the Yelp dataset, an extremely sparse dataset (by Recommender System standards), and the MovieLens 10M dataset, a dataset comparable to ones used in other similar works. The VAE is trained and evaluated on both these datasets individually.

The remainder of this paper is as follows; Section 2 presents the background regarding recommender systems as well as some related works. Section 3 presents the proposed approach in this work. Section 4 presents the results of this paper. Section 5 presents the conclusions and future work.

# Background and Related Work

## 2.1. Matrix Factorization

Users and items are mapped to a latent factor space. Each item maps to a vector in the space, as does each user. The elements of an item vector express if an item contains the ‘factor’ at that index, and similarly for user vectors, each element corresponds to the user's preference/lack thereof for a particular item. The dot product of a user and item vector is a representation of the user's interest in the item, which is an estimate of the rating a user would give the item (Koren, Bell & Volinsky. 2009). This approach is similar to the mathematical operation of the Singular Value Decomposition. However, extra steps are needed to apply it to Recommender Systems, as the sparsity of user-item matrices, containing many ‘unknowns’, means that general SVD is undefined for it (Koren, Bell & Volinsky. 2009).

## 2.2. Mathematical SVD

SVD in mathematics is a way to break down an n x m matrix X into the product of U (m x n), S (n x n), and VT (n x m), where S is a diagonal matrix. The columns of U are called “left singular vectors”, and rows of VT are called “right singular vectors” (Kingma & Welling. 2019). SVD Recommender Systems use U and VTas analogies for user and item latent factor matrices.

## 2.3. SVD++

SVD++ was designed to take advantage of the large amount of data generated by users on large e-commerce sites. Since users do not always leave reviews, and the reviews left are biased toward negative reviews, it was important to find a way to understand user preferences without relying on their own reviews. This is where implicit feedback comes in. It allows the Recommender System to analyze trends in the users’ data profiles: their purchases, searches, and other aspects of online behavior which do not qualify as explicit positive or negative reviews. Implicit feedback counts interactions of all kinds as a positive. Implicit feedback significantly improves the quality of recommendations for SVD++, negating much of the hinderance that is caused by the sparseness of data (Cao et. al. 2015). Slight tweaks to the bias latent factor model allow SVD++ to also be resistant against overfitting (Cao et. al. 2015). An optimizer algorithm used in SVD++ models is Stochastic Gradient Descent (Luo T, et. al. 2013), (Cao et. al. 2015). One of the biggest problems with SVD++ is that being based on matrix factorization, it inherits its flaws of being slow to train, while also needing to be optimized again after every new user and item added (Shenbin et. al. 2020)

## 2.4. Variational Autoencoders

Variational Autoencoders (Liang et. al. 2018, Shenbin et. al. 2020) are a type of neural network that are being investigated for Recommender Systems. Autoencoder neural networks are designed to learn an encoding function and an approximate decoding function to reproduce the input from the encoding. Variational Autoencoders are a related concept, however they are designed to learn probability distributions. This allows for a generative model which excels at fitting a distribution to a data sample. VAE models are abundant in image modelling and generation but have recently become a hot topic for research into Recommender Systems. It has been theorized that log and gaussian likelihood functions may be holding back VAE Recommender Systems. Models using multinomial likelihood functions outperform the Gaussian and log likelihood models (Liang et. al. 2018). Variational autoencoders draw inspiration from variational inference and are superficially similar to autoencoder networks (Kingma & Welling. 2019).

Variational inference modelling is good for producing statistical models of this nature, but with large datasets, the number of parameters becomes unwieldy (Liang et. al. 2018). They replace the individual parameters with an inference model/function. This model must be able to reproduce/approximate the parameters for any inputs.

Autoencoder networks are designed to be able to compress a datapoint to a feature set, and then reproduce the input datapoint given the compressed feature set (NG. 2011). Combining the intuition of the ideas of replacing input data points with a probability distribution and building a neural network that can ‘translate’ between a learned feature set and externally useful data points, we get the Variational Autoencoder, which is able to learn the probability distribution from the input data, and apply it to produce meaningful, true samples (Liang et. al. 2018)

VAEs are trained using the ELBO function, which seeks to minimize the Kullback-Leiber divergence. There is a common technique known as the reparameterization trick which makes sampling from arbitrary distributions trivial, by expressing the sample in terms of the standard distribution and those factor values that scale and translate the standard distribution to produce the actual distribution of the data (Liang et. al. 2018).

## 2.6. VAE Recommender Systems

Currently, some of the leading research in the usage of VAEs for Recommender Systems are systems such as RecVAE (Shenbin et. al. 2020), a VAE model of a Recommender System which includes implicit feedback. It proposes several improvements to the baseline MultVAE (VAE with multinomial likelihood function. MultVAE was shown by Liang et. al. To exhibit better results than VAEs with other common likelihood functions). There is also EnsVAE, a framework for developing hybrid Recommender Systems.

EnsVAE Recommender Systems combine multiple sub-recommenders into a unified model (Drif, Zerrad, & Cherifi. 2020). Another form of Hybrid VAE Recommender System was developed by Gupta et. al., to work on the MovieLens dataset. It augments the ratings matrix with data from the movie embeddings. These embeddings are the result of dimension-compression of movie features by one VAE. The resulting dataset (ratings matrix and embeddings) is used to train a second VAE (Gupta, Raghuprasad, & Kumar. 2018).

Liang et. al. proposed a technique of defining VAE (MultVAE) as an extension of the Denoising Autoencoder, a sub-order of autoencoders which can be described as specializing in producing outputs more useful than the input data (Vincent et. al. 2010). DAEs are known to be less prone to overfitting. It is noted that multinomial likelihood functions produce better results than log and Gaussian likelihood functions (Liang et. al. 2018). They suggest that some of the traditional features of generative models are not necessary for the Recommendation Problem and can be traded off for performance. For example, they suggest removing the ability to perform ancestral sampling, since this feature does not carry much usefulness when generating recommendations.

They make use of SGD for optimization. They modify the default ELBO function to include a parameter beta that controls the strength of regularization (the KL term in the loss function) and apply an annealing optimizer to tune beta. They use a dropout layer in order to prevent overfitting. Adam optimizer is also employed for training of the VAE model (Liang et. al. 2018).

They use the metrics of Recall@K and Normalized Discounted Cumulative Gain (NDCG) to evaluate the system and find it to outperform other common Recommender Systems such as matrix factorization.

RecVAE, was designed by Shenbin et. al. with the objective of superseding the original MultVAE designed by Liang et al (Shenbin et. al. 2020). Shenbin et. al. note that one of the key steps to the success of MultVAE is that it inherits denoising functionality due to the design being based on a DAE.

RecVAE improves on MultVAE in the several ways. Firstly, the architecture of the encoder network is improved. Bernoulli noise is introduced so that he denoising encoder can be trained. The encoder borrows the intuition of Densely connected layers from Convolutional Neural Networks, as well as implementing Swish Activation functions and Layer Normalization (Shenbin et. al. 2020).

Secondly, they introduce a composite prior that builds over MultVAE's Gaussian prior. This involves the use of the latent code distribution from the previous epoch to supplement the Gaussian prior (Shenbin et. al. 2020).

Thirdly they use a new technique for KL divergence scaling. Instead of slowly increasing beta by annealing, they suggest that a constant beta works just as well. They perform scaling of the KL term in the objective function on a user-by-user basis, choosing beta based on the amount of implicit feedback available for a user (Shenbin et. al. 2020).

And fourthly, they introduce a different training schedule to account for the fact that the Encoder network is much more complex than the decoder and needs more work to train. RecVAE alternates between updating the encoder and decoder. It performs multiple updates to the encoder for every update to decoder. The approach for training taken by RecVAE also allows corrupted inputs to be used since it incorporates the denoising functionality.

However, it is suggested that corrupted inputs should only be used when training the encoder, as this leads to better generalization, but not the decoder. They also note that the decoder should ideally be treated as a decoder in a vanilla VAE (with no denoising) instead of like a decoder in a denoising VAE (Shenbin et. al. 2020). They use the Adam optimizer for training, and a batch size of 500 (Shenbin et. al. 2020). This study was built on the work of MultVAE and uses the same evaluation metrics. It was repeatedly ranked above MultVAE in performance and performed favorable against most common Recommendation System models, including all VAE-based models (Shenbin et. al. 2020).

Gupta et. al. Have developed a hybrid model for Collaborative Filtering that includes an aspect of content-based filtering. Their work was in the sub-domain of movie recommendation. They reduce the feature set that describe a movie by running it through one VAE. Their model contains a layer which then combines these embeddings together with the user ratings for each movie (Gupta, Raghuprasad, & Kumar. 2018). Besides for this, the hybrid is similar to the standard VAE architecture. It was proven to outperform the standard VAE

(Gupta, Raghuprasad, & Kumar. 2018), but no comparison is made by Gupta et. al. Between their hybrid model and other specialized VAEs such as MultVAE and RecVAE.

# Proposed Approach

The study proposed VAE-based recommender Systems solution that uses the concept of implicit feedback.

## 3.1. VAE design

The variational autoencoder network consists of an encoder, decoder and sampling layer. The encoder consists of 5 dense layers, each using the tanh activation. Each dense layer is followed by layer normalization.

The encoder outputs are the mean and log variance of data. As training proceeds, these layers’ weights are expected to converge to the necessary weights required to modify the standard normal distribution. That is, they are used to perform the reparameterization trick on the normal distribution to produce the resultant sampling distribution which accurately captures the essence of the data.

Diagram

Description automatically generated

*Figure 1: Graph of the VAE used in this paper*

The decoder consists of two dense layers using ReLU activation, as well as an output that makes use of swish activation.

The loss function is the sum of the KL Divergence and the reconstruction loss. Reconstruction is measured as the binary cross entropy between input and output samples. Given that the data is sparse this value tends to be very small. Therefore, it is multiplied by the size of user vectors in order to give weight to its contribution towards the total loss.

KL annealing is performed whereby the contribution of the KL divergence to the total loss is multiplied by a constant beta which initially takes a value of 0 but is slowly increased up to 1. This technique results in a decrease in the final training loss value.

Two training phases are allowed for the VAE. Firstly, there is a primary training phase where the model learns from the main training set, using the validation set for validation. A secondary training phase is also in place where the model learns from the smaller, test-training set, using the same validation set for validation.

# Experimental Evaluation

Two datasets were used to evaluate, the Yelp dataset (<https://www.yelp.com/dataset>) and the MovieLens dataset (<https://grouplens.org/datasets/movielens/10m/>).

## 4.1. Pre-processing

Yelp reviews were stripped from the json file they are stored in. The only used data are the tuples of (user id, business id, stars) obtained by parsing the reviews json file. In the case of the MovieLens 10M dataset, the corresponding fields are extracted from the data file.

For either dataset, A more interactive subset of users and businesses is picked. This involves screening out users and businesses which are involved in fewer than 5 reviews events. The 1-5 rating scale is replaced by the implicit form where all ratings have a positive value of 1.

|  |  |  |
| --- | --- | --- |
| Dataset | Yelp | ML-10M |
| Sparsity % | 0.012 | 1.403 |

*Table 1: Sparsity value for each dataset used in this paper. This is a measure of the percentage of cells in the rating matrix which actually have known values*

The bulk of all review events are used for a primary training phase. This allows the model to learn the latent embeddings in the dataset, using most of the available user vectors. However, reviews belonging to 10000 randomly picked users are held out to form a validation set. Additionally, reviews of a different 10000 randomly selected users are held out to form the testing set. This means there are 10000 user-vectors in the testing set. Due to the cold start problem, the model cannot be expected to draw inferences about these testing users without knowing anything about them. Therefore, from this set of testing users, 80% of review events are actually used for training again, in order to overcome the cold start problem. The model will only be measured in its ability to reproduce the remaining 20% of reviews that have never been seen by it before. This is in line with the previous works like MultVAE (Liang et. al., 2018), and RecVAE (Shenbin et. al., 2020). Each of the data subsets are converted to sparse matrix format.

Diagram

Description automatically generated

*Figure 2.1: Representation of the dataset split for a dataset with n users, in line with MultVAE (Liang et. al., 2018)*

*Table

Description automatically generated*

*Figure 2.2: representation of a user vector from the training, validation and testing set, for a dataset with m items (businesses for Yelp, movies for MovieLens) in it.*

## 4.2. SVD++ baseline

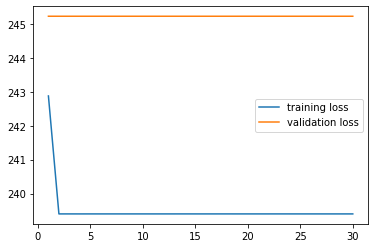
In this study, a VAE was developed and trained on implicit feedback. It was evaluated and benchmarked against the SVD++ algorithm. The SVD++ model’s training data consists of the training, validation, and testing-training review events. It is necessary for SVD++ to have seen at least some review events from the test users for it to make predictions about them (i.e., educated predictions, not simple mean values as the Surprise Library provides when a user is not known to it).

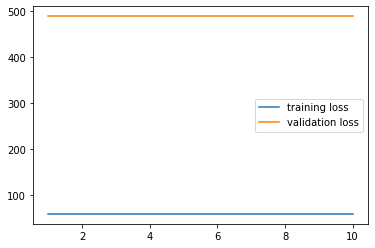
## 4.3. Training the Models

The learning curves for the Yelp dataset are shown in Figures 3.1.a and 3.1.b. The VAE trained for 30 epochs in the primary training phase. During this training, there was a tiny decrease in loss up to the end of the second epoch. However, the loss value then stagnated until the end of the training period. During the 10 epoch secondary training phase, a similar pattern emerges.

The model appeared to quickly become trapped in a local minimum when training on Yelp data. However, the implementation of a mechanism to reduce the learning rate during execution, whenever too many epochs pass without a decrease in validation loss, did not affect the loss curve at all. Increasing the intermediate dimension of the encoder and decoder (from 512 up to at least 4096) similarly had no effect on the loss curve.

Additionally, it seems to be a trend that validation loss is much higher than training loss. This probably stems from the cold start problem, as in this work, the model was never exposed to partial vectors from validation users the way it was exposed to partial vectors from testing users during evaluation.



*Figure 3.1.a: Yelp Primary Training Phase Loss Curves*

*Figure 3.1.b: Yelp Secondary Training Phase Loss Curves*

By comparison, training the same model on a comparable sample from the MovieLens dataset yields loss curves that show a greater degree of learning. The major difference between these two datasets (Yelp and MovieLens) is the sparsity. The sparsity of the MovieLens data (in the form of the matrix of users and businesses) is 1.403%. By comparison, the sparsity of the Yelp data is 0.012%. This is a difference in sparsity by a factor of around 117.The learning curves for the MovieLens dataset set are shown in Figures 3.2.a, and 3.2.b.

Chart, line chart

Description automatically generated

*Figure 3.2.a: ML-10M Primary Training Phase Loss Curves*

Chart, line chart

Description automatically generated

*Figure 3.2.b: ML-10M Secondary Training Phase Loss Curves*

The extreme sparsity of the Yelp data likely places the task of inferring relations within it beyond the capabilities of the VAE used in this paper. A quick analysis of several generated samples from the trained VAE reveals that said generated samples tend to consist entirely of negative predictions (although this does not hold true over the entire dataset, as further results indicate that some samples do generate outputs with some positive values). That is, despite the fact that filtering during pre-processing increases the density of review events by cutting out the large number of low-interaction users, samples which have been generated using the model usually have zero positive review events. This is in contrast to when the model is trained on the MovieLens dataset, as in that scenario there are several non-zero predictions made. This is more evidence that the high degree of sparsity in the Yelp data has rendered this particular VAE incapable of reproducing positive review events. There is in general a low likelihood of the same business being rated positively over all user samples. As a result, the model quickly learns that the zero prediction is a safe assumption when averaged out over all training data.

This can explain the lower overall loss values on the Yelp dataset vs the MovieLens dataset, since the difference between input samples and zero-vectors of the same shape as any input sample is comparatively small when the number of non-zero elements in the input sample itself makes up a negligible fraction of the sample.

The phenomenon known as “posterior collapse” (He et. al. 2019), may explain why this happens. It is known that it is common for VAE models to become stuck in a local optimum early on in the training phase and thus fail to estimate the true posterior. Training the VAE on KL divergence alone leads to loss values very close to the zero-value witnessed in the basic VAE used as a baseline by He et. al, lending evidence to the suggestion that posterior collapse is the root of the issue with the model in this paper.

## 4.4. Results

The model is evaluated on the metrics of Recall@20, Recall@50 and Non-Discounted Cumulative Gain (NDCG). It is then compared against the baseline metrics provided by an SVD++ model.

|  |  |  |
| --- | --- | --- |
| Yelp Dataset | | |
|  | VAE | SVD++ |
| Recall@20 | 0.0002 | 0.0116 |
| Recall@50 | 0.0004 | 0.0120 |
| NDCG@100 | 0.0003 | 0.0178 |
| MovieLens 10M Dataset | | |
| Recall@20 | 0.1329 | 0.4686 |
| Recall@50 | 0.2711 | 0.7297 |
| NDCG@100 | 0.3170 | 0.7975 |

*Table 2: Recall and NDCG results for the VAE and the SVD++ baseline on both Yelp and MovieLens 10M datasets*

As expected, given the poor learning during training on Yelp data, the VAE developed in this paper does not contend well in terms of Recall@K and NDGC. It trails behind SVD++ by 2 orders of magnitude.

In terms of the MovieLens 10M dataset, the performance does become much better, although not quite state-of-the-art, and it is still outperformed by SVD++, although the gap is much smaller.

# Conclusions and Future work

In this work, a Variational Autoencoder is developed to solve the collaborative filtering problem based on implicit feedback from the Yelp dataset. The VAE is benchmarked against a baseline of SVD++, and it is found that it did not contend well on high-sparsity data. When formulated and evaluated on the MovieLens 10M Dataset, the VAE produces better results, but was still surpassed by SVD++.

It is likely that the VAE is failing due to posterior collapse. He et. al. stipulate that by preventing a lagging inference network, posterior collapse can be averted. It is proposed that the inference network (encoder) and generator network (decoder) are optimized individually. The inference network is “aggressively optimized” in an inner loop during the training process, such that for each update to the decoder weights, there are several updates to the encoder. This process maintains most of the standard techniques for VAE training, while achieving similar results to several other more complicated techniques. While currently the training times for the VAE are fairly quick, the aggressive optimization of the inference network will increase the time needed to train.

In future work, it would be beneficial to measure the effect of applying the aggressive inference network optimization described by He et. al. during the training step of this work.

Additionally, it may prove useful to unify the separate training phases and by including the 80% secondary training partial vectors for test users into the main training set, as well as to introduce similar partial vectors from the validation users to the training set.

# Acknowledgements

Sincere thanks are extended to Dr. Edgar Jembere (at the University of KwaZulu-Natal) for all guidance offered during this project, as well as to the administrators of the HIPPO High Speed Computing Facility at the University of KwaZulu-Natal for providing the resources to perform much of the computations in this work.

# References

1. B. Croft; D. Metzler; T. Strohman (2010). Search Engines: Information Retrieval in Practice. Addison Wesley
2. Cao, J., Hu, H., Luo, T., Wang, J., Huang, M., Wang, K., ... & Zhang, X. (2015, October). Distributed design and implementation of SVD++ algorithm for e-commerce personalized recommender system. In 13th National Conference on Embedded System Technology (pp. 30-44). Springer, Singapore.
3. Drif, A., Zerrad, H. E., & Cherifi, H. (2020). EnsVAE: Ensemble Variational Autoencoders for Recommendations. IEEE Access, 8, 188335-188351.
4. Gower, S. (2014). Netflix prize and SVD.
5. Gupta, K., Raghuprasad, M. Y., & Kumar, P. (2018). A hybrid variational autoencoder for collaborative filtering. arXiv preprint arXiv:1808.01006.
6. Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems (TOIS), 20(4), 422-446.
7. Kingma, D. P., & Welling, M. (2019). An introduction to variational autoencoders. arXiv preprint arXiv:1906.02691.
8. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.
9. Liang, D., Krishnan, R. G., Hoffman, M. D., & Jebara, T. (2018, April). Variational autoencoders for collaborative filtering. In Proceedings of the 2018 world wide web conference (pp. 689-698).
10. Luo, K., Yang, H., Wu, G., & Sanner, S. (2020, July). Deep Critiquing for VAE-Based Recommender Systems. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 1269-1278).
11. Luo, T., Chen, S., Xu, G., & Zhou, J. (2013). Collaborative filtering. In Trust-based Collective View Prediction (pp. 25-51). Springer, New York, NY.
12. Ng, A. (2011). Sparse autoencoder. CS294A Lecture notes, 72(2011), 1-19.
13. Sajjadi, M. S., Bachem, O., Lucic, M., Bousquet, O., & Gelly, S. (2018). Assessing generative models via precision and recall. arXiv preprint arXiv:1806.00035.
14. Shenbin, I., Alekseev, A., Tutubalina, E., Malykh, V., & Nikolenko, S. I. (2020, January). RecVAE: A new variational autoencoder for Top-N recommendations with implicit feedback. In Proceedings of the 13th International Conference on Web Search and Data Mining (pp. 528-536).
15. Vahdat, A., & Kautz, J. (2020). Nvae: A deep hierarchical variational autoencoder. arXiv preprint arXiv:2007.03898.
16. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P. A., & Bottou, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research, 11(12).
17. Wall, M. E., Rechtsteiner, A., & Rocha, L. M. (2003). Singular value decomposition and principal component analysis. In A practical approach to microarray data analysis (pp. 91-109). Springer, Boston, MA.
18. Xian, Z., Li, Q., Li, G., & Li, L. (2017). New collaborative filtering algorithms based on SVD++ and differential privacy. Mathematical Problems in Engineering, 2017.
19. Xue, H. J., Dai, X., Zhang, J., Huang, S., & Chen, J. (2017, August). Deep Matrix Factorization Models for Recommender Systems. In IJCAI (Vol. 17, pp. 3203-3209).
20. He, J., Spokoyny, D., Neubig, G., & Berg-Kirkpatrick, T. (2019). Lagging inference networks and posterior collapse in variational autoencoders. arXiv preprint arXiv:1901.05534.