CSE 258 ASSIGNMENT 2

Shichong Peng
A53317072
s5peng@ucsd.edu

Curtis Spears
A14231834
cspears@ucsd.edu

Haotian Qiu
A53306394
h1qiu@ucsd.edu

Tianchong Jiang
A92089573
tij005@ucsd.edu

ABSTRACT

Deep learning based collaborative filtering is making tremendous progress in recommendation systems. In particular, deep generative models like variational autoencoders are producing state-of-the-art results. We explore several methods from this category along with linear models on the well-established MovieLens dataset. Ranking based metrics like Recall@K and truncated normalized discounted cumulative gain (NDCG@K) are used for evaluating model performance and strong generalization are employed to better simulate real world scenarios. Contrary to the popular research interests on larger and larger datasets, we focus on how the models adapt to different data sizes and summarize an intuitive interpretation of the relation between latent prior distribution and data size.

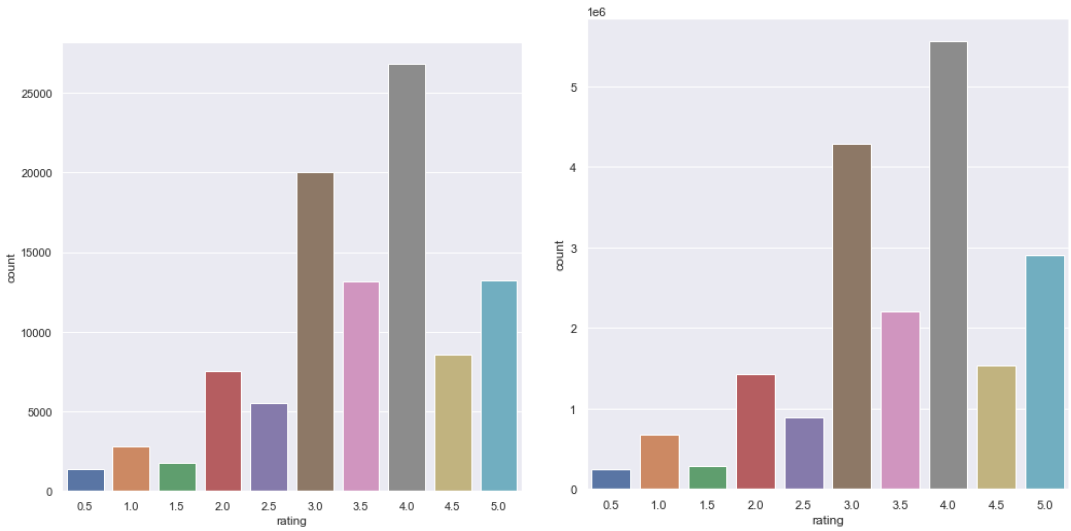
1 INTRODUCTION

Recommendation systems can be seen everywhere, from search engines like Google to movie recommendation websites like Netflix and much more. How to help users find new items that they might prefer among countless many available options is a hot research topic. Various attempts have been made including the ones that incorporate deep learning techniques. Linear models are among the first ones to be considered which achieved descent performance on the sparse data. However, those methods lack the expressive power of nonlinear models and one branch of those models is deep generative model. Variational Autoencoders (VAE) (Kingma & Welling, 2014; Rezende et al., 2014) are generative models that are originally designed for image data. However, the latent representation learnt by the model corresponds well to the user&item representation in collaborative filtering objective. Various approaches have been proposed and are shown to be effective given the sparse user-item interaction history data. As size of data grows, more methods are only targeting large scale data and neglect their adaptive ability to data of smaller scale. We posit by conducting a study on model's performance on data of different sizes, the effect of model parameters could be interpreted in a more intuitive manner and could inspire better design in the future. Therefore, we select some recent methods and evaluate them against datasets of different sizes to discover this relationship between model design and data size.

2 DATASET

We explore the MovieLens dataset which records the 5-star rating history from the movie recommendation service MovieLens. Among various datasets collected over time, two datasets of different sizes are studied and compared:

1. MovieLens-Small (ML-Small): 100,000 ratings from 600 users to 9,000 movies



(a) ML-Small (b) ML-20M

Figure 1: Rating histogram comparison between ML-Small and ML-20M. As shown, the distribution of counts for each rating is similar across the two datasets.

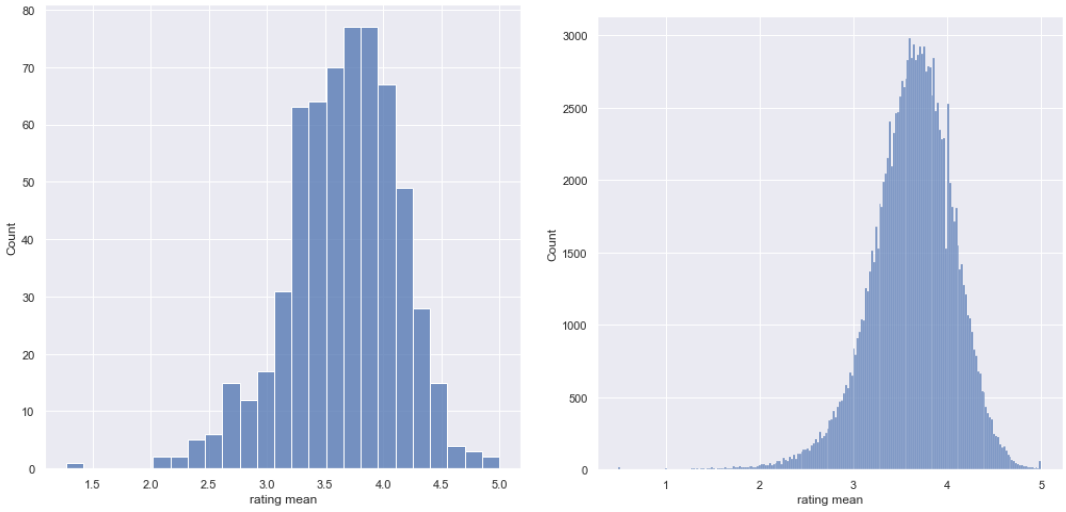
2. MovieLens-20M (ML-20M): 20 million ratings from 138,000 users to 27,000 movies

Figure 1 shows the counts for each rating (minimum 0.5 and maximum 5.0) for ML-small and ML-20M. As shown, the distribution of rating count is similar for the two datasets with most users giving rating of 4.0. We also observe that the next two popular ratings are 3.0 and 5.0 with fewer ratings of 3.5 and 4.5. This hints that users' opinion on movies are relatively clear. Figure 2 shows the distribution of the mean rating by each user for ML-small and ML-20M. We can see from the figure that the mean rating roughly follows a normal distribution which is centered around 3.75. This indicates that users tend to choose movies that they like or interested and their watching history is an accurate reflection of their tastes.

3 Predictive Task

3.1 Setup

Given a user's rating history of some movies, we want to build a model that predicts whether the user would give an unseen movie high rating (≥ 4.0). We evaluate the models performance under strong generalization (Marlin, 2004) where the user group is disjoint among training, validation and testing sets. This is a more challenging setting than weak generalization where only the user-item pair is disjoint in the three sets (training, validation and testing). In addition, this setting is closer to real world scenarios where the recommendation system is used for suggesting new items for users instead of memorizing the items that users have interacted with. The models are trained using the training set and the held-out sets (validation and testing sets) are further divided into two parts. The first part of the held-out sets or the "fold-in" set is used for obtaining necessary initial representation of the user to avoid the cold-start problem. The model then try to predict the second part of the held-out set where the evaluation is based on. In the experiments, the fold-in set consists of 80% of the records in the held-out sets which is selected randomly, i.e. if a user in the held-out set watches 50 movies, 40 of them are randomly selected into the fold-in set and the remaining 10 movies are used for prediction evaluation. Only users who have watched at least 5 movies are kept in the data. The rating records are binarized using the threshold 3.5 (if the rating is > 3.5, then the label is 1 and 0 otherwise) which is an ideal cut-off given the data distribution shown in Figure 1. User id, movie id and the binarized rating label are used as data features. The training, validation and testing dataset are the same for all the experiments.



(a) ML-Small (b) ML-20M

Figure 2: Mean rating per user histogram for ML-Small and ML-20M. As shown, the distribution of user mean rating is similar across the two datasets with peak concentrated around 3.75.

3.2 Evaluation Metric

Simple accuracy evaluation may not reflect the relative relation among different items, therefore ranking-based metrics Recall@K and truncated normalized discounted cumulative gain (NDCG@K) are used. In the held-out sets, fold-in sets are used for user representation generation and the rest are compared against the model prediction. Suppose the model prediction at rank k is denoted as r_k (the ranking is obtained by sorting the raw prediction scores of the models), the set of held-out movies that the given user u gives high rating is denoted as I_u, $|I_u|$ denotes the total number of movies that u have watched in the held-out set and $\mathbb{1}[r_k \in I_u]$ denotes whether the prediction is within the ground truth set. The Recall@K for user u is defined as:

$$Recall@K(u) = \frac{\sum_{k=1}^{K} \mathbb{1}[r_k \in I_u]}{min(K, |I_u|)} \tag{1}$$

The final score for the held-out set is computed as the mean score of all users in the set with value in $[0, 1]$. Recall@K treats all top-K predictions as equally weighted, whereas NDCG@K assigns monotonically decreasing importance to lower ranked predictions. Truncated discounted cumulative gain(DCG@K) for a given user u is defined as:

$$DCG@K(u) = \sum_{k=1}^{K} \frac{2^{\mathbb{1}[r_k \in I_u]} - 1}{log(k+1)} \tag{2}$$

NDCG@K(u) is DCG@K(u) normalized to range $[0, 1]$ by dividing the largest score of all gains. The NDCG@K for the held-out set is computed as the mean value of all users in the set.

4 Model

4.1 Latent Factor Model (LFM)

The baseline model we will use for this prediction task is the simple Latent Factor Model introduced in class. Latent Factor Model assumes that there exists a low-dimensional representation of users and items where user-item affinity can be modeled accurately. For this task, we will predict the ratings of items of a certain user with the formula:

$$rating = \alpha + \beta_u + \beta_i \tag{3}$$

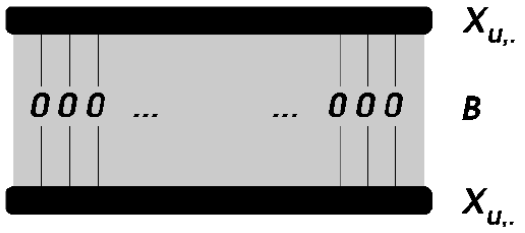


Figure 3: The self-similarity of each item is constrained to zero between input and output layers.(excerpted from (Steck, 2019)).

where α is a latent factor, β_u is the user bias which could be interpreted as the amount that this user tend to rate above the mean, β_i is the item bias which could be interpreted as the amount of higher rating that this item tend to receive. The objective function is defined as:

$$argmin_{\alpha,\beta} \sum_{u,g} (\alpha + \beta_u + \beta_i - T_{u,i})^2 + \lambda(\sum_u \beta_u^2 + \sum_i \beta_i^2) \tag{4}$$

Where λ is the regularization factor and $T_{u,i}$ is the ground truth label of user u to movie i. The prediction scores are then ranked to determine which movies that the user is more likely to watch. The model is learnt by gradient descent with learning rate 0.0001, λ of 5.4 and a converging threshold of 10^{-7}.

4.2 Embarrassingly Shallow AutoEncoder (EASE)

The Embarrassingly Shallow AutoEncoder (EASE) (Steck, 2019) is an efficient model with closed-form solution. In most cases, a recommender model will take the "interaction matrix" as input, for example, a matrix $X \in \mathbb{R}^{|U| \times |I|}$ where $|U|$ represents number of users and $|I|$ is the number of items. Doing computations on this large matrix is very expensive, therefore EASE took a different route by taking the item-item weight matrix $B^{|I| \times |I|}$. The size of matrix B is generally much smaller than X because the number of users could be several magnitudes larger than that of items. The matrix B will hold the similarity between items, and then during the prediction, we can calculate a score for each of the movie for a specific user. However, it is very important that we do not allow the self-similarity to be 1 because then the weight matrix will become the identity matrix. Therefore, we mandate the self-similarity of each item to be zero, as the figure 3 shows:

Since all self-similarity is constrained to zero, the diagonal of the matrix B, $diag(B) = 0$. Accordingly, the optimization task of EASE model is:

$$\begin{aligned} \min_B \quad & \|X - XB\|_F^2 + \lambda \cdot \|B\|_F^2 \\ \text{s.t.} \quad & diag(B) = 0 \end{aligned} \tag{5}$$

Where the $\|X\|_F$ denotes the Frobenius norm of matrix X. The author chose the Frobenius norm over the 2-Norm because the Frobenius norms allows a closed form solution. To include the zero self-similarity constraint in the optimization task, we form the Lagrangian:

$$L = \|X - XB\|_F^2 + \lambda \cdot \|B\|_F^2 + 2 \cdot \gamma^\top \cdot diag(B) \tag{6}$$

Where $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_{|I|})$ is the vector of Lagrangian multipliers. By minimizing this Lagrangian, we get an estimate of the weight matrix B:

$$\hat{B} = (X^\top X + \lambda I)^{-1} \cdot (X^\top X - diagMat(\gamma)) \tag{7}$$

Where $diagMat(\cdot)$ denotes the diagonal matrix and I is the identity matrix. We define a matrix $\hat{P}$ for sufficiently large λ:

$$\hat{P} \doteq (X^\top X + \lambda I)^{-1}$$

By substituting $\hat{P}$ into equation 5 and simplifying:

$$\hat{B} = I - \hat{P} \cdot diagMat(\tilde{\gamma}) \tag{8}$$

The closed-form solution of weight matrix B can be written as:

$$\hat{B}_{i,j} \begin{cases} 0, & \text{if } i = j \\ -\frac{\hat{P}_{i,j}}{\hat{P}_{j,j}}, & \text{otherwise} \end{cases} \tag{9}$$

With input X and regularization factor λ, we can conveniently calculate matrix $\hat{P}$ and then drive weight matrix B. Finally, we can predict the score $S_{u,j}$ for item $j \in I$ for a user $u \in U$ by a simple dot product of two vectors:

$$S_{u,j} = X_{u,\cdot} \cdot B_{\cdot,j} \tag{10}$$

Where $X_{u,\cdot}$ is the row vector in the interaction matrix X for user u and $B_{\cdot,j}$ denotes j^{th} column of item-item weight matrix B. For each user the model computes a score for all items that the user may or may have not interacted with. The top-K recommendation could be derived by sorting all scores and return first K items. In the experiment, we used the regularization factor $\lambda = 0.5$.

4.3 Multinomial Variational Autoencoders (Mult-VAE)

Liang et al. (2018) extended VAE to collaborative filtering for implicit feedback. Different from the latent factor model, VAE is a non-linear probabilistic mode which contains more expressive power. More explicitly, the authors proposed Mult-VAE which replaces the Gaussian and logistic likelihoods in normal VAE to multinomial conditional likelihood. Suppose the the binary rating history of a user u is denoted as $x_u = [x_{u1}, \ldots, x_{uI}]$ where I is the total number of movies in the dataset. The model starts by sampling a K-dimensional latent vector from standard Gaussian prior $z_u \sim \mathcal{N}(0, I_K)$. Then a learnable non-linear function $f_\theta(\cdot) \in \mathbb{R}^I$ parameterized by θ is applied to z_u to get the probability distribution $\pi(z_u)$ over all items. The binary rating history x_u is then sampled from the multinomial distribution $x_u \sim Mult(N_u, \pi(z_u))$ given the total number of high-rated movies $N_u = \sum_i x_{ui}$. Therefore, the conditional log-likelihood for user u is:

$$log\ p_\theta(x_u|z_u) = \sum_i x_{ui}\ log\ \pi_i(z_u) \tag{11}$$

This can be learnt by optimizing the commonly known evidence lower bound (ELBO) of the log marginal likelihood:

$$log\ p(x_u; \theta) \geq \mathbb{E}_{q_\phi(z_u|x_u)}[log\ p_\theta(z_u|x_u)] - KL(q_\phi(z_u|x_u)||p(z_u)) \equiv\ \mathcal{L}(\theta, \phi) \tag{12}$$

The Mult-VAE model further reinterprets this lower bound by adding a trade-off factor β which controls the degree of fitting to the data and closeness between the approximate posterior and the prior. The adjusted loss function:

$$\mathcal{L}_\beta(\theta, \phi) \equiv\ \mathbb{E}_{q_\phi(z_u|x_u)}[log\ p_\theta(z_u|x_u)] - \beta \cdot KL(q_\phi(z_u|x_u)||p(z_u)) \tag{13}$$

where

$$q_\phi(z_u|x_u) = \mathcal{N}(\mu_\phi(x_u), diag\{\sigma^2_\phi(x_u)\}) \tag{14}$$

The amortized variational distribution $q_\phi(\cdot)$ parameterized by ϕ is used to approximate the intractable posterior $p(z_u|x_u)$ and μ and σ are free variational parameters. In the experiments, we select a single layer network with 600 latent dimension, 200 z dimensions and β equals to 0.3 for both datasets. Network weights are optimized using Adam (Kingma & Ba, 2015) with learning rate 0.0005.

4.4 RecVAE

The RecVAE model (Shenbin et al., 2020) is based on the Mult-VAE but with improvements. RecVAE has a new architecture for the encoder network. It introduced a novel composite prior distribution for the latent code z which helps to stabilize training and performance. It made the hyperparameter for the Killback-Leibler term β in the objective function user specific and dependent on the amount of data for the given user $\beta = \beta(\mathbf{x}_u)$. This change is reflected in the equation below:

$$\mathcal{L}(\theta, \phi) = \mathrm{E}_{q_\phi(z|\bar{x})}\mathrm{E}_{q(\bar{x}|x)} \left[\log p_\theta(x \mid z) - \ -\beta'(x)\mathrm{KL}\left(q_\phi(z \mid \bar{x}) \| p\left(z \mid \phi_{\text{old}}, x\right)\right)\right] \tag{15}$$

where

$$p\left(z \mid \phi_{\text{old}}, x\right) = \alpha N(z \mid 0, 1) + (1 - \alpha) q_{\phi_{\text{old}}}(z \mid x) \tag{16}$$

RecVae alternates updates for the encoder and decoder which allows multiple updates of the encoder (which is more complex) for every update of the decoder. It also uses corrupted inputs only for training the encoder, while the decoder is trained on clean input only. This is beneficial for the final model training due to the difference in complexity between the encoder and decoder. The models used for both datasets uses a latent dimension is 200, a hidden dimension is 600, and a learning rate of 0.0005. β is no longer a constant and is calculated with:

$$\beta' = \beta'\left(x_u\right) = \gamma \left|\mathrm{X}_u^o\right| = \gamma \sum_i x_{ui} \tag{17}$$

where γ is a constant hyperparameter shared across all users and we set it to 0.005 in the experiments. A constant γ showed better performance than modifying it for each user. X_u^o represents the set of items the user has positively interacted with (in the observed data).

4.5 Enhanced VAE

Kim & Suh (2019) enhanced Mult-VAE by introducing flexible priors and gating mechanisms. The ELBO in Equation 12 can be rewritten as the following:

$$\begin{aligned} \mathcal{L}(\theta, \phi, \lambda) = \mathrm{E}_{x\sim q(x)}[\mathrm{E}_{q_\phi(z|x)}[log\ p_\theta(x|z)]] \\ +\mathrm{E}_{x\sim q(x)}[\mathrm{H}[q_\phi(z|x)]] \\ -\mathrm{E}_{z\sim q(z)}[-log\ p_\lambda(z)] \end{aligned} \tag{18}$$

where the loss is separated into the reconstruction error, expectation of variational posterior entropy and the cross entropy between the aggregated posterior $q(z) = \frac{1}{N}\sum_{u=1}^{N} q_\phi(z|x_u)$ and the prior. The last term enforces the posterior to be close to the simple prior which might lead to unwanted regularization effect. Therefore flexible prior like VampPrior (variational mixture of posteriors prior)(Tomczak & Welling, 2018) which conditions on K learnable inputs can be introduced:

$$p_\lambda(z) = \frac{1}{K}\sum_{k=1}^{K} q_\phi(z|u_k) \tag{19}$$

A hierarchical stochastic unit which replaces the latent variable z by two variables z_1 and z_2 is also adopted to capture more complex latent structures. More concretely, the variational inference is changed to:

$$q_\phi(z_1|x, z_2)q_\psi(z_2|x) \tag{20}$$

and the decoder part changed to:

$$p_\theta(x|z_1, z_2)p_\lambda(z_1|z_2)p(z_2) \tag{21}$$

and $p(z_2) = \frac{1}{K}\sum_{k=1}^{K} q_\phi(z_2|u_k)$ is the VampPrior.

The gating mechanism replaces linear units by element-wise product of two outputs:

$$h_l(X) = (X * W + b) \otimes \sigma(X * V + c) \tag{22}$$

where W, V, b, c are learnable parameters, $\otimes$ is element-wise product and σ is the sigmoid function. This allows better gradient flow to earlier layers and increases the model capacity.

To model user preference given only user's rating history is a challenging task. Simple models like latent factor model and EASE are easy to implement but cannot learn richer latent user item representations which limits their performance. Deep generative models using variational

Model	NDCG@100	NDCG@20	NDCG@10	Recall@50	Recall@20	Recall@10
LFM	0.0669	0.0555	0.0418	0.0763	0.0611	0.0450
EASE	0.1374	0.0466	0.0145	0.4139	0.2953	0.2541
Mult-VAE	0.3448	0.2732	0.2603	0.4376	0.3322	0.2915
RecVAE	0.2662	0.2006	0.2037	0.3353	0.2203	0.2068
Ours	**0.3627**	**0.3030**	**0.3030**	**0.4403**	**0.3437**	**0.3134**

Table 1: Comparison of NDCG@100, NDCG@20, NDCG@10, Recall@50, Recall@20, Recall@10 on ML-Small dataset among latent factor model, EASE, Mult-VAE, RecVAE and our adopted model. The higher the score, the better the performance. As shown, our adopted method beats the baselines on all the score which shows better modeling of user preference.

autoencoders overcome this issue and are shown to be producing state-of-the-art results. However, models like Mult-VAE are limited by over-simplified assumption on the posterior (loss enforcing it to be similar the simple prior). RecVAE tries to improve upon the encoder network and modified prior which shows improvements compared to Mult-VAE. However, the KL divergence term between the posterior and the prior still produces an unintended strong regularization effect.

Therefore, We build our final model based on this method and experimented with different model architecture for the best performance. We found that even with the gating mechanism, deeper networks tend to show worse performance especially on the smaller ML-Small dataset. In addition, large latent and prior dimension can also significantly worsen the performance on small dataset. The reason behind is common to a lot of the deep learning based method where the model overfits to the training data and loses generalizability. Therefore, our final model architecture uses a single layer network with 150 latent dimension and 50 for z_1&z_2 dimensions for the ML-Small dataset. For ML-20M dataset, our model is a two-layer network with 600 latent dimension and z_1&z_2 dimensions equal to 200. In model for both datasets, K is 1000, β is 0.3 and network weights are optimized using Adam with learning rate 0.0005.

5 RESULTS

We test all the models on ML-Small and ML-20M dataset using the setup and evaluation metric mentioned in Section 3. Table 1 shows the results on ML-Small dataset and we can see that our adopted method is consistently beating all other methods on the metrics. On the NDCG@100, we beat the second best method Mult-VAE by almost 0.2 which is a huge performance gain. Also, we have notice that RecVAE suffers from overfitting to the training data and produces worse results. Simple one layer Mult-VAE and our model variant (by reducing the number of layers and latent dimensions) show better performance suggesting that on smaller dataset like ML-Small, simpler assumption on the latent space (indicated by the network depth, prior and latent dimensions) is preferred. Table 2 shows the results on ML-20M dataset and it can be seen that our method is producing better NDCG results than other baselines. RecVAE is producing higher recall rates for $K = 50$ and $K = 20$ but the lead over our model is marginal. All Mult-VAE based methods are outperforming the rest models indicating richer latent space feature representation is crucial in large scale datasets like ML-20M. In addition, more complicated prior like the composite prior in RecVAE and hierarchical VampPrior in our adopted method proves to be effective which supports the existing finding on generative modeling for collaborative filtering. Comparing the relative performance of the methods from the larger ML-20M dataset to the smaller ML-Small dataset, we can see that hyperparameters like network depth, prior and latent dimensions plays a big role in performance. This could be interpreted as preventing overfitting and provide sufficient latent variables for fitting the features in the data. A more intuitive interpretation is those hyperparameter reflects the assumption of the prior of the data, i.e. if a dataset is small then a simpler prior is preferred since less modes are present, whereas a larger dataset would require more flexible prior to better capture the complex feature manifold.

Model	NDCG@100	NDCG@20	NDCG@10	Recall@50	Recall@20	Recall@10
LFM	0.1045	0.0978	0.0512	0.1163	0.0911	0.0750
EASE	0.3893	0.3090	0.2929	0.4884	0.3683	0.3129
Mult-VAE	0.4231	0.3347	0.3193	0.5301	0.3924	0.3307
RecVAE	0.4434	0.3565	0.3435	**0.5560**	**0.4145**	0.3552
Ours	**0.4452**	**0.3596**	**0.3485**	0.5520	0.4141	**0.3577**

Table 2: Comparison of NDCG@100, NDCG@20, NDCG@10, Recall@50, Recall@20, Recall@10 on ML-20M dataset among latent factor model, EASE, Mult-VAE, RecVAE and our adopted model. The higher the score, the better the performance. As shown, RecVAE and our adopted method shows competitive performance and outperform the rest baselines.

6 Related Work

The MovieLens dataset is the user rating history collected by the movie recommendation service MovieLens. The most studied subsets of MoiveLens are MovieLens-100K and MovieLens-20M which includes 100 thousands and 20 million rating history correspondingly. The data contains user-movie rating pairs along with some content data like timestamp and movie genres. Group of the proposed methods in literature use only the interaction history (like our setting) and some other methods (Musto et al., 2016; Suglia et al., 2017; Oord et al., 2013) take advantage of the content data as well. A similar dataset is the Netflix Prize dataset (Bennett & Lanning, 2007) which includes 100 million ratings from 480 thousands users over 17 thousand moives. The Netflix dataset is evaluated using root mean square error (RMSE) of the prediction rating against the ground truth rating on the hidden testing set. Another dataset that is popular for predictive tasks is the Million Song dataset (Bertin-Mahieux et al., 2011). The data includes play counts of 1 million songs for more than 40 thousands users which are interpreted as implicit preferences.

Traditional methods like memory based collaborative filtering through threshold cannot capture the representation of the data whereas deep learning does. Zhang et al. (2017) provides a comprehensive list of deep learning based methods. Deep Learning is widely used in recommendation systems due to its advantage in non-linear transformation, representation learning, and sequence modelling. Some common deep learning techniques are Multilayer Perceptron(MLP) (He et al., 2017a;b; Alashkar et al., 2017), Variant Auto-encoder(VAE), Convolutional Neural Network(CNN) (Gong & Zhang, 2016; Kim et al., 2016; Seo et al., 2017), and Recurrent Neural Network(RNN) (Zheng et al., 2017; Bansal et al., 2016; Dai et al., 2016). Current state-of-the-art methods mostly use variational autoencoders for modeling user preference given sparse interaction history. As introduced in Section 4, RecVAE (Shenbin et al., 2020) and the adopted enhanced VAE (Kim & Suh, 2019) are the top performing methods on MovieLens-20M and Netflix datasets. Our study confirms the findings in those works especially for the enhance VAE by showing the adaptive power of various data sizes.

7 Conclusion

In this study of popular methods for recommendation, we explore their performances on two subsets of MovieLens data. We adopt ranking based metrics (Recall@K and NDCG@K) to evaluate the effectiveness of recommending new items among the full set. In addition, we assess them in a more realistic strong generalization setting where user groups of training and held-out sets are disjoint. Variational autoencoder based methods are shown to be effective in approximating latent user and item representations. By comparing the model performance from large data size to smaller data size, we find a strong relation between the model's assumption on the latent variable prior and the dataset size, i.e. simpler latent prior works better for smaller dataset and more flexible prior would fit larger dataset with more complicated structures. This would inspire more work in latent prior design of VAEs to further extend the capability of modern recommendation systems.

REFERENCES

Taleb Alashkar, Songyao Jiang, Shuyang Wang, and Yun Fu. Examples-rules guided deep neural network for makeup recommendation. In *AAAI*, 2017.

Trapit Bansal, David Belanger, and A. McCallum. Ask the gru: Multi-task learning for deep text recommendations. *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016.

James Bennett and Stan Lanning. The netflix prize. 2007.

T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *ISMIR*, 2011.

Hanjun Dai, Y. Wang, R. Trivedi, and L. Song. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv: Learning*, 2016.

Yuyun Gong and Qi Zhang. Hashtag recommendation using attention-based convolutional neural network. In *IJCAI*, 2016.

X. He, Lizi Liao, Hanwang Zhang, L. Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. *Proceedings of the 26th International Conference on World Wide Web*, 2017a.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. *CoRR*, abs/1708.05031, 2017b. URL `http://arxiv.org/abs/1708.05031`.

D. Kim and Bongwon Suh. Enhancing vaes for collaborative filtering: flexible priors & gating mechanisms. *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019.

D. Kim, Chanyoung Park, Jinoh Oh, S. Lee, and H. Yu. Convolutional matrix factorization for document context-aware recommendation. *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Diederik P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

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

Benjamin M Marlin. Collaborative filtering: A machine learning perspective. 2004.

C. Musto, Claudio Greco, Alessandro Suglia, and G. Semeraro. Ask me any rating: A content-based recommender system based on recurrent neural networks. In *IIR*, 2016.

A. Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *NIPS*, 2013.

Danilo Jimenez Rezende, S. Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Sungyong Seo, Jing Huang, H. Yang, and Y. Liu. Representation learning of users and items for review rating prediction using attention-based convolutional neural network. 2017.

Ilya Shenbin, Anton M. Alekseev, E. Tutubalina, Valentin Malykh, and S. Nikolenko. Recvae: A new variational autoencoder for top-n recommendations with implicit feedback. *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020.

H. Steck. Embarrassingly shallow autoencoders for sparse data. *The World Wide Web Conference*, 2019.

Alessandro Suglia, Claudio Greco, C. Musto, M. Degemmis, P. Lops, and G. Semeraro. A deep architecture for content-based recommendations exploiting recurrent neural networks. *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, 2017.

J. Tomczak and M. Welling. Vae with a vampprior. In *AISTATS*, 2018.

Shuai Zhang, Lina Yao, and Aixin Sun. Deep learning based recommender system: A survey and new perspectives. *CoRR*, abs/1707.07435, 2017. URL http://arxiv.org/abs/1707.07435.

Lei Zheng, Vahid Noroozi, and Philip S. Yu. Joint deep modeling of users and items using reviews for recommendation. *CoRR*, abs/1701.04783, 2017. URL http://arxiv.org/abs/1701.04783.