Collaborative Filtering Recommendation System for Steam with Deep and Variational Autoencoders

Daniel Liang [*1] **Shuidie Yao** [*1]

Abstract

Recommendation systems are commonly used to provide interesting and meaningful content personalized to a users preferences. While multiple approaches are used to create them, autoencoders are commonly used as the model for collaborative filtering recommendation systems. We prepare data from a Steam dataset into a matrix based on the time that users have played specific games. Additional matrices containing game names, game genres and user reviews from individual users are used as latent variables in our models. We build recommendation systems based on the formatted data, two with deep autoencoder models and two with variational autoencoder models. Finally, we examine the differences between the various aspects of the models.

1. Introduction

The goal of recommendation systems is to recommend items for customers based on their past reflection of purchased and/or used goods. To deal with a large amount of data, two primary types of algorithms are prevalent when creating recommendation systems: content-based filtering algorithms and collaborative filtering algorithms. Content-based filtering algorithms produce recommendations by looking at past positive reactions from a user to a certain item and recommending items based on similar attributes. This could take the form of recommending another popular action movie because the user had liked a similar action movie in the past. Collaborative filtering algorithms differ because they recommend based on multiple users historical preferences on a set of items to recommend items from similar users rather than rely on item attributes.

The data for collaborative filtering generally has a specific pattern to consider. A single customer usually buys a few types of items out of the hundreds and thousands of total items offered. These items tend to be popular items that other users own. Most items therefore are favored only by a few people, while top items are extremely popular in comparison. This popularity is generally shown in a form that indicates what the user thinks of each item. In the case of television shows, this would most likely be the amount of time a user spends watching each show. In a typical representation of such data—a two-dimensional matrix where rows are users, columns are items, and each element is a score of one user's preference of one item—we would see that a majority of the total shows offered would have a watch time of zero, since most users don't interact with most items.

Neural networks have been very effective in investigating large and sparse matrices. In particular, a popular method in building recommendation systems to describe features contained in these types of matrices is to use an autoencoder. Autoencoders consist of two neural networks, where one is an encoder that trains a function to bring high dimension input data to a bottleneck layer where the number of neurons is the smallest. The other neural network is a decoder that converts the encoded input back to its original dimension. The autoencoder works well by learning to extract the most relevant features in the bottleneck from the latent space, and the latent space contains the only information a decoder can use to reconstruct the input as faithfully as possible.

Variational autoencoders share roughly the same process but train different things. It assumes that the data has a certain distribution, and finds the parameters of the distribution during training. Despite the reconstruction loss, its loss function also contains a regularizer, which is the Kullback-Leibler divergence between the encoder distribution and the decoder distribution. Variational autoencoders implicitly create a probability distribution based on the data it is given.

Autoencoders have been successfully implemented for various different datasets and have produced impressive results. A deep autoencoder researched by NVIDIA on the Netflix prize dataset that outperforms previously built autoencoder models (Kuchaiev & Ginsburg, 2017). Similarly, a variational autoencoder was implemented on the Movie-

[*]Equal contribution [1]Courant Institute of Mathematical Sciences, New York University, New York City, United States. Correspondence to: Daniel Liang <dzl217@nyu.edu>, Shuidie Yao <sy2327@nyu.edu>.

Lens 20M dataset (Liang et al., 2018). The implementations discussed in this paper are based on the models proposed by these authors for deep and variational autoencoders on a Steam dataset.

2. Method

Four recommendation systems are built using one procedure but with different autoencoders. The data we use required cleaning and formatting prior to training. We train the models on Google Colaboratory. Recommendations were produced as well as data from the models' training that allows us to compare the performance of the two autoencoders. The full code can be found at https://github.com/yshuidie/steamNNRecom.

2.1. Data and Preparation

We use the Steam video game dataset from the collection of recommendation system datasets offered by Julian McAuley from the University of California San Diego. The data used can be found and downloaded at http://cseweb.ucsd.edu/~jmcauley/datasets.html. On the Steam platform users can buy games, play those games and write reviews of them. Based on our experience as players along with our observations by profiling the data, we see that most players play less than half of the games they own. This shows that some players are generally interested in the games they purchase, but have preferences towards the games they have played. Additionally, players usually spend a significant amount of time on their favorite games more than the other games they own. We therefore decide to treat a user's play time for a certain game as a scoring metric to show how much a user likes that game. From the game play time of each user, we formulate a user preference matrix.

We denote the play time matrix T, where each entry T_{ij} represents the play time in minutes that user i spent on game j. We create an approximation of a probability matrix S, which represents the user preference matrix, that a player would play a game with the following properties:

- A function exists that maps values $T_{ij} \in (0, \infty)$ to $S_{ij} \in (0, 1)$
- The probability S_{ij} increases with playtime T_{ij}
- S_{ij} and T_{ij} have a non-linear relationship. S_{ij} should increase marginally as T_{ij} increases.

According to these needs, we use the tanh function to create the new probability matrix. We set the maximum playtime of any game to 50 hours, and based on the observation that $\tanh(2) \approx 0.96 \approx 1$, we construct the probability matrix such that

$$S_{ij} = \tanh(\frac{T_{ij} * 2}{50 * 60})$$

We perform several tasks in order to properly prepare the data for training. First, we filter out games that were never played by any user and users who never played any games from our Pandas dataframe. Next, due to memory limits within Google Colaboratory, we limit the number of games we'd investigate. We cap the game ID that steam uses for game indentification and consider all games with an ID lower than 10000. At a result, 354 games are considered. For each user, we extract a list of games that the user played, the names of those games, the games genres and the reviews the user made. The words in this data are transformed into integers by keras.one_hot(), so that they could be fed into the autoencoders. In addition, we reformat the game information in a new dataframe to include all the games with information about their game price, developer and publisher. This allows us to gain more immediate insight from the recommendations produced by the models.

2.2. Model

We build four models using a deep autoencoder for two models and a variational autoencoder for the other two models using the Keras framework. All the models we build have the same input of the user preference probability matrix. However, two models with differing autoencoders are created that utilize three latent variable matrices with each user's game names, game genres and reviews respectively. Every users information is separated by row in each matrix, with the ordering of rows being consistent across all matrices. The models output is a scoring matrix with the same dimension as the preference matrix S.

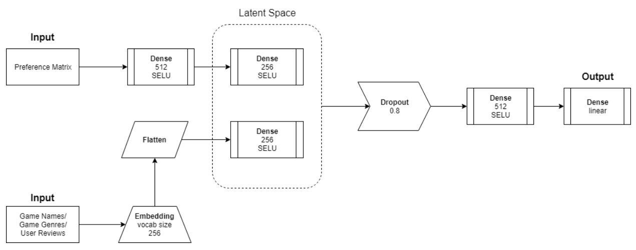


Figure 1. Deep Autoencoder model with Latent Matrices

The architecture of the deep autoencoder implemented can be seen in Figure 1. The user preference matrix passes through the encoder neural network consisting of two SELU dense layers that reduces the spatial size of the representation to 256. The other input matrices are each embedded, flattened, and then pass through a SELU dense layer that reduces the spatial size to 256. The two 256-size layers make up the latent space of the autoencoder. The data in the latent space then passes through a dropout layer to avoid overfitting, and then passes through the decoder neural network that converts the hidden representation to one intermediate dense layer of 512 units. Finally, the data is converted back to its original dimension through a linear dense layer.

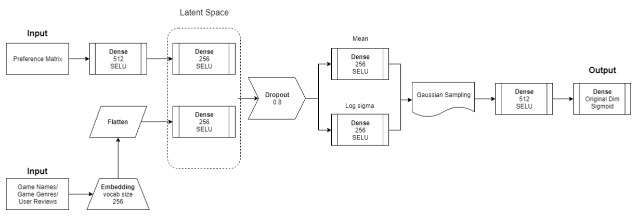


Figure 2. Variational Autoencoder model with Latent Matrices

The architecture of the variational autoencoder can be seen in Figure 2. In the variational autoencoder, we assume that the probability distribution $P(z|X)$, which is the probability of latent variable z given input X is Gaussian to learn its parameters. Instead of outputting a hidden representation z, the encoder outputs mean $\mu(X)$ and $\log \sigma(X)$. Next, we use the Gaussian reparameterization trick such that the operation

$$z = \mu(X) + \exp(\frac{\log \sigma(X)}{2}) \cdot \epsilon$$

where $\epsilon \sim N(0, 1)$, is equivalent to a Gaussian sampling because the sampling process does not have a gradient. Finally, the decoder $P(X|z)$ is similar to the decoder in the deep autoencoder with the exception that the activation function of the output layer is the non-linear sigmoid function.

2.3. Recommender

Once we have the reconstructed matrix as an output from the autoencoder, we set the entries of played games to 0 in order to not recommend games that the user had previously played. Finally, a recommender is built to present the top recommended games for a queried user based on the highest scoring games from the models output matrix.

3. Results

3.1. Loss and Mean Squared Error

The mean square loss function is used to train the deep autoencoder and the general loss function for the variational autoencoder is shown as

$$E[(X|z)] - KL[Q(z|X)|P(z)]$$

The first term for our equation, which is the construction loss, can be calculated by the binary cross entropy C, given input X and predicted probability $P(X|z)$:

$$C = \frac{1}{n} \sum_{i=1}^{n} [X_i \ln P(X_i|z) + (1 - X_i) \ln(1 - P(X_i|z))]$$

This is possible because the input for our model contains probability values between zero and one. The KL divergence term is implemented as:

$$KL = 0.5 * \sum [\exp(\log \sigma) + \mu^2 - 1 - \log \sigma]$$

Although the two types of autoencoders use different loss functions to train, the Keras fit method can calculate the mean squared error (MSE) of the variational autoencoder while training. This metric is provided for equal comparison between the two autoencoders.

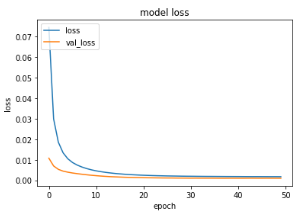
We experiment with the deep autoencoders (DAE) and the variational autoencoders (VAE) and produce four models. The MSE obtained after the final epoch of training from each experiment is shown in Table 1. In addition, plots of the loss curve from each experiment is shown in Table 2 where we see all the models converging at the end of their training. Because the DAE models were trained with the mean squared function, the loss shown in these graphs is the mean squared error of that model.

We see from Table 1 that the DAE models both with and without latent variables converge to a lower MSE than either VAE models even if those models initially had a lower MSE at the start of training. We also see that the MSE is higher for models using the latent matrices than without them. In Table 2, we see that most of the models start with high MSE then very quickly converge to low MSE. However, the VAE with latent variables shows a slower convergence, and at the end of training the validation loss does not plateau like the other models. This indicates that the latent matrices have more impact on the VAE models than the DAE models. With larger latent matrices, we can expect to see larger changes with how VAE models train than the DAE models.

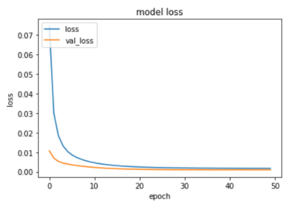
Table 1. Mean Squared Error after training

MODELS	LEARNING RATE	EPOCHS	MSE
DAE	0.00005	50	0.0018
DAE WITH LATENTS	0.00005	50	0.0026
VAE	0.0001	50	0.0056
VAE WITH LATENTS	0.00005	50	0.0082

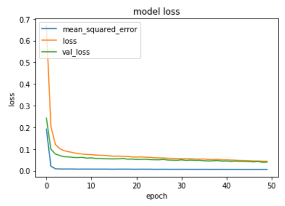
Table 2. Plots of history loss during training



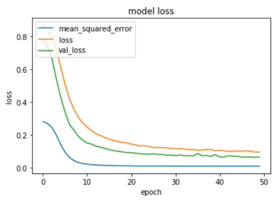
DAE



DAE with latents



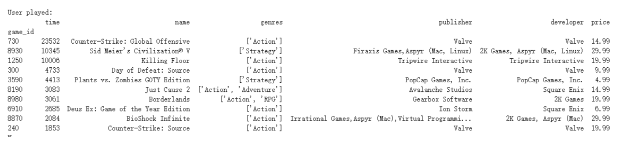
VAE



VAE with latents

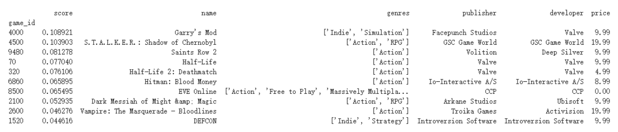
3.2. Recommendations

We review the information of user A from the dataset and the recommendations produced for user A to analyze the models' output. The history of played games from user A can be seen in Figure 3, while Figures 4-7 contain recommendations by the various models. The highest recommended game by each of the four models is the game "Garry's Mod". Additionally, the game "Saint Rows 2" appears in all four recommendations, while other games like "Half Life" and "Star Trek Online" appear in most of the lists. This indicates our models have similarities in feature extraction. We therefore look at the confidence score indicated by the output of the matrix. We see that DAE has the lowest score, indicating a weak recommendation, while the VAEs have a drastically higher score.



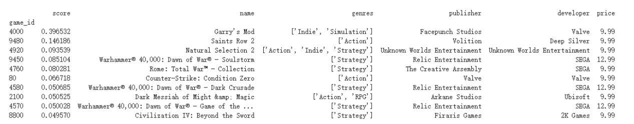
```
User played:
          time                           name                      genres                                         publisher                    developer  price
game_id
730      23532  Counter-Strike: Global Offensive                ['Action']                                             Valve                        Valve  14.99
8930     10345         Sid Meier's Civilization® V             ['Strategy']              Firaxis Games,Aspyr (Mac, Linux)  2K Games, Aspyr (Mac, Linux)  29.99
1250     10006                     Killing Floor                ['Action']                              Tripwire Interactive         Tripwire Interactive  19.99
300       4733              Day of Defeat: Source                ['Action']                                             Valve                        Valve   9.99
3590      4413    Plants vs. Zombies GOTY Edition             ['Strategy']                                 PopCap Games, Inc.           PopCap Games, Inc.   4.99
8190      3083                      Just Cause 2  ['Action', 'Adventure']                                  Avalanche Studios                  Square Enix  14.99
8980      3061                       Borderlands        ['Action', 'RPG']                                   Gearbox Software                     2K Games  19.99
6910      2685  Deus Ex: Game of the Year Edition               ['Action']                                          Ion Storm                  Square Enix   6.99
8870      2084                  BioShock Infinite               ['Action']  Irrational Games,Aspyr (Mac),Virtual Programmi...         2K Games, Aspyr (Mac)  29.99
240       1853              Counter-Strike: Source               ['Action']                                              Valve                        Valve  19.99
```

Figure 3. User A game history



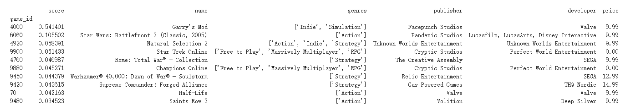
```
            score                               name                                            genres           publisher            developer  price
game_id
4000     0.108921                        Garry's Mod                          ['Indie', 'Simulation']   Facepunch Studios                Valve   9.99
4500     0.103903  S.T.A.L.K.E.R.: Shadow of Chernobyl                           ['Action', 'RPG']      GSC Game World       GSC Game World  19.99
9480     0.081278                       Saints Row 2                                    ['Action']            Volition          Deep Silver   9.99
70       0.077040                          Half-Life                                    ['Action']               Valve                Valve   9.99
320      0.076106              Half-Life 2: Deathmatch                                  ['Action']               Valve                Valve   4.99
6860     0.065895                 Hitman: Blood Money                                   ['Action']  Io-Interactive A/S   Io-Interactive A/S   8.99
8500     0.065495                          EVE Online  ['Action', 'Free to Play', 'Massively Multipla...                 CCP                  CCP   0.00
2100     0.052935       Dark Messiah of Might &amp; Magic                           ['Action', 'RPG']      Arkane Studios              Ubisoft   9.99
2600     0.046276  Vampire: The Masquerade - Bloodlines                             ['Action']        Troika Games           Activision  19.99
1520     0.044616                             DEFCON                        ['Indie', 'Strategy']  Introversion Software  Introversion Software   9.99
```

Figure 4. User A Recommendations by Deep Autoencoder



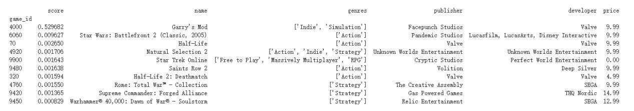
```
            score                                           name                            genres                     publisher                     developer  price
game_id
4000     0.396532                                    Garry's Mod            ['Indie', 'Simulation']             Facepunch Studios                         Valve   9.99
9480     0.146186                                   Saints Row 2                        ['Action']                      Volition                   Deep Silver   9.99
4920     0.093539                            Natural Selection 2  ['Action', 'Indie', 'Strategy']  Unknown Worlds Entertainment  Unknown Worlds Entertainment   9.99
9450     0.085104     Warhammer® 40,000: Dawn of War® - Soulstorm                     ['Strategy']          Relic Entertainment                          SEGA  12.99
4760     0.080281                  Rome: Total War™ - Collection                     ['Strategy']          The Creative Assembly                         SEGA   9.99
80       0.066718                 Counter-Strike: Condition Zero                        ['Action']                         Valve                         Valve   9.99
4580     0.050685  Warhammer® 40,000: Dawn of War® - Dark Crusade                     ['Strategy']          Relic Entertainment                          SEGA  12.99
2100     0.050525               Dark Messiah of Might &amp; Magic                ['Action', 'RPG']                Arkane Studios                       Ubisoft   9.99
4570     0.050028  Warhammer® 40,000: Dawn of War® - Game of the ...                  ['Strategy']          Relic Entertainment                          SEGA  12.99
8800     0.049570                Civilization IV: Beyond the Sword                     ['Strategy']                 Firaxis Games                      2K Games   9.99
```

Figure 5. User A Recommendation by Deep Autoencoder with Latent Matrices



```
            score                                        name                                              genres                     publisher                               developer  price
game_id
4000     0.541401                                 Garry's Mod                            ['Indie', 'Simulation']             Facepunch Studios                                   Valve   9.99
6060     0.105502        Star Wars: Battlefront 2 (Classic, 2005)                                     ['Action']             Pandemic Studios  Lucasfilm, LucasArts, Disney Interactive   9.99
4920     0.058391                         Natural Selection 2                  ['Action', 'Indie', 'Strategy']  Unknown Worlds Entertainment             Unknown Worlds Entertainment   9.99
9900     0.051433                            Star Trek Online  ['Free to Play', 'Massively Multiplayer', 'RPG']               Cryptic Studios              Perfect World Entertainment   0.00
4760     0.046987               Rome: Total War™ - Collection                                   ['Strategy']         The Creative Assembly                                    SEGA   9.99
9880     0.045271                            Champions Online  ['Free to Play', 'Massively Multiplayer', 'RPG']               Cryptic Studios              Perfect World Entertainment   0.00
9450     0.044379  Warhammer® 40,000: Dawn of War® - Soulstorm                                   ['Strategy']           Relic Entertainment                                    SEGA  12.99
9420     0.043615        Supreme Commander: Forged Alliance                                   ['Strategy']             Gas Powered Games                              THQ Nordic  14.99
70       0.042163                                   Half-Life                                     ['Action']                         Valve                                   Valve   9.99
9480     0.034523                                Saints Row 2                                     ['Action']                      Volition                             Deep Silver   9.99
```

Figure 6. User A Recommendation by Variational Autoencoder



```
            score                                        name                                              genres                     publisher                               developer  price
game_id
4000     0.529682                                 Garry's Mod                            ['Indie', 'Simulation']             Facepunch Studios                                   Valve   9.99
6060     0.009627        Star Wars: Battlefront 2 (Classic, 2005)                                     ['Action']             Pandemic Studios  Lucasfilm, LucasArts, Disney Interactive   9.99
70       0.002650                                   Half-Life                                     ['Action']                         Valve                                   Valve   9.99
4920     0.001706                         Natural Selection 2                  ['Action', 'Indie', 'Strategy']  Unknown Worlds Entertainment             Unknown Worlds Entertainment   9.99
9900     0.001643                            Star Trek Online  ['Free to Play', 'Massively Multiplayer', 'RPG']               Cryptic Studios              Perfect World Entertainment   0.00
9480     0.001638                                Saints Row 2                                     ['Action']                      Volition                             Deep Silver   9.99
320      0.001594                     Half-Life 2: Deathmatch                                     ['Action']                         Valve                                   Valve   4.99
4760     0.001550               Rome: Total War™ - Collection                                   ['Strategy']         The Creative Assembly                                    SEGA   9.99
9420     0.001365        Supreme Commander: Forged Alliance                                   ['Strategy']             Gas Powered Games                              THQ Nordic  14.99
9450     0.000829  Warhammer® 40,000: Dawn of War® - Soulstorm                                   ['Strategy']           Relic Entertainment                                    SEGA  12.99
```

Figure 7. User A Recommendation by Variational Autoencoder with Latent Matrices

4. Future Work

While we have implemented various high-performance autoencoders based on contemporary research, there are other types of implementations that we can consider in the future. Currently our VAE models assume a standard Gaussian prior. Another implementation consists of a high performing VAE model that assumes user-dependent priors defined as functions of user review text instead (Karamanolakis et al., 2018), which had modeled the trends within their dataset with higher accuracy. In addition, while observations were made on the experiments, more developed performance metrics could be implemented for a more direct comparison between the two types of autoencoders.

5. Conclusion

We create models of deep and variational autoencoders on a Steam dataset to recommend games to users. Both autoencoders implemented have success in recommendations that fit the types of games users had already played in the past. While the mean squared error of deep autoencoder models is lower than that of variational autoencoder models, we see a greater effect with latent matrices on the latter models than the former. Additionally, we note that the variational autoencoder models output notably higher confidence scores for recommendations, indicating that a user has a higher probability of enjoying the recommended game. With a more diverse and expansive dataset and greater computational power for running the models, we expect to see an improvement in accuracy and confidence with the recommendations.

References

Apurva Pathak, Kshitiz Gupta, J. M. Generating and personalizing bundle recommendations on steam, 2017.

Doersch, C. Tutorial on variational autoencoders, 2016.

Kang, W. C. and McAuley, J. Self-attentive sequential recommendation, 2018.

Karamanolakis, G., Cherian, K. R., Narayan, A. R., Yuan, J., Tang, D., and Jebara, T. Item recommendation with variational autoencoders and heterogenous priors. 2018. doi: 10.1145/3270323.327032.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes, 2013.

Kuchaiev, O. and Ginsburg, B. Training deep autoencoders for collaborative filtering, 2017.

Liang, D., Krishnan, R. G., Hoffman, M. D., and Jebara, T. Variational autoencoders for collaborative filtering, 2018.

Moussawi, A. Towards large scale training of autoencoders for collaborative filtering, 2018.

Wan, M. and McAuley, J. Item recommendation on monotonic behavior chains, 2018.