Music Recommender System using Autorec Method for Implicit Feedback

Muhammad Faishal Irawan, Z K A Baizal
School of Computing, Informatics, Telkom University, Bandung, Indonesia
Email: 1faishalirawan@student.telkomuniversity.ac.id, 2baizal@telkomuniversity.ac.id
Correspondence Author Email: baizal@telkomuniversity.ac.id

Abstract—As the number of music and users in music streaming services increases, the role of music recommender systems is getting important to make it easier for users to find music that matches their tastes. The collaborative filtering paradigm is the most commonly used paradigm in developing recommender systems. Many studies have proven that deep learning is able to improve the performance of matrix factorization. One such method in deep learning that has been adapted for use in Recommender Systems is Autorec, which is a variation of the Autoencoder technique. Autorec shows that it performs better than the baseline matrix factorization using Movielens and Netflix datasets. Therefore, in this study we propose the use of Autorec to develop a recommender system for music. The experimental results show that Autorec performs better than Singular Value Decomposition (SVD), with an RMSE difference of 0.7.

Keywords: Recommender System; Autoencoder; Deep Learning; Music Recommender System; Autorec

1. INTRODUCTION

Music streaming services such as Spotify, Apple Music, and YouTube Music are increasingly used by many users around the world, along with the amount of music that continues to grow. The increase in consumable content will make it difficult for users to find suitable music to listen to [1]. This makes the role of recommender systems increasingly important in music search [2]. With the existence of a music recommender system, the problem of finding products that suit user tastes and information overload can be overcome. Paradigms for building recommender systems are collaborative filtering, content-based filtering and knowledge-based filtering [3].

Collaborative filtering (CF) is the most commonly used paradigm in recommender systems [4]. The idea of this paradigm is to predict a user's rating of an item based on other users who have similar tastes. One of the methods used in CF is matrix factorization. Matrix factorization is a classic method that has been used and won the Netflix Prize competition, so it has been proven to give good results [5]. This method is a latent factors method or a method that seeks to find hidden factors and predict ratings by characterizing items and users on many factors inferred from rating patterns. Previous studies have proven that the recommendation results of matrix factorization approaches can be improved with the use of deep learning [6].

In recent years, deep learning has become the dominant approach for many ongoing tasks in the field of machine learning [7]. Deep learning has also started to enter the realm of recommender systems. Where previously, deep learning has given excellent results on other machine learning problems such as computer vision [6] and speech recognition [8]. This research seeks to use a deep learning approach with an autoencoder architecture that has been customized for a recommender system called Autorec.

Compared to traditional models such as matrix factorization that can only use one data source such as ratings, the models using autoencoder can utilize multiple data sources such as ratings, audio, visual, and video. Additionally, autoencoders may offer better recommendation outcomes than traditional models due to their improved grasp of user preferences and item characteristics [9].

Sedhain et al. [10] proposed the use of autoencoder to perform collaborative filtering named Autorec. The model got better results than the state-of-the-art CF technique in that year. We examined the model's application in the field of music by constructing a music recommendation system and comparing the outcome to that of a conventional matrix factorization method.

The music domain itself is interesting to use in this research because the dataset contains implicit feedback. Implicit feedback is a type of feedback that does not directly indicate the user's preference for an item, as opposed to explicit feedback. For example, in explicit feedback users can rate an item between 1 and 5, 1 means the user dislikes the item and 5 means the user really likes an item. Meanwhile, implicit feedback has various forms, such as how long the user views an item, how many times the user clicks on an item, and how many times the user listens to a piece of music. In this research, we use the frequency information of each user in listening to each music in making recommendations.

The primary sources for this research include one of the initial studies that applied autoencoders in the area of recommendation systems. This method is called AutoRec [10], this method was proposed by Sedhain et al. in 2015. Using the Movielens and Netflix datasets, the study showed that AutoRec managed to provide better results than other matrix factorization methods and Restricted Boltzmann machine for collaborative filtering methods using the root-mean-square error (RMSE) evaluation metric. Movielens and Netflix datasets are among the datasets that use explicit feedback.

After the autoencoder method for recommender systems was proposed, many researchers developed this method to improve its performance. The DeepRec method [11] was one of them, it was developed by Kuchaiev et
al. of NVIDIA team in 2017. The method extends AutoRec by using a deeper network, Scaled Exponential Linear Unit (SELU) activation function, high dropout rate, and iterative output re-feeding. Meanwhile, another type of autoencoder called variational autoencoders emerged that are developed specifically in the CF area, such as Mult-VAE [12], this method was created by Liang et al. in 2018. It introduces a Bayesian inference approach in recommending items and was developed specifically for implicit feedback data. Other commonly used autoencoder types include the denoising autoencoder [13], this method was built by Wu et al. in 2016. It introduces noise in the input rating to improve the recommendation of the system.

Mult-VAE is an autoencoder that has a problem formulation close to this research. The research specializes in creating a variational autoencoder to solve collaborative filtering problems using implicit feedback. While Autorec's research only uses explicit feedback datasets. Therefore, in this research we tried to explore the use of Autorec, specifically U-Autorec, in different domain from the original research and demonstrate how it performs in that particular domain. We chose music domain because the nature of most music dataset in recommender system is in the form implicit feedback. This poses an intriguing problem as the Autorec was only tested in the movie domain that have an explicit feedback type. The baseline that we are going to use, in accordance to the previous research, is matrix factorization type, in this case Singular Value Decomposition (SVD).

2. RESEARCH METHODOLOGY

2.1 Research Stages
The flow of music recommender system development can be seen in figure 1. First, we acquire the publicly available Million Song Dataset (MSD) The Echo Nest Taste Profile Subset and preprocess it using the method that will be explained in the subsequent section. Following that, we are going to build and train the Autorec model for recommending music. Finally, the model will be evaluated using root-mean-square error and compared with baseline method.

2.2 Autoencoder
An autoencoder (AE) is a type of neural network architecture that is typically used for unsupervised learning tasks, such as generating new data, reducing the number of features, and encoding data efficiently [14]. This method is better in learning the representation of latent features in recognizing images [15], speech [16], and in computer vision [17], among others. AE is considered an effective method to obtain nonlinear features [18]. Autoencoders are used to transform input into a more compact and informative representation, and then decode it back, so that the reconstructed input is as similar as possible to the original one [19].

AE is divided into an encoder and a decoder, the encoder consists of the input and hidden layer. Meanwhile, the decoder consists of that same hidden layer and an output layer. The input goes into the encoder where it’s feature will be reduced, then the decoder tries to reconstruct the original input using that reduced form.

The encoder uses the function \( f \) to convert the input data \( x \) that’s high-dimensional into a hidden representation \( h \) that has a lower dimension. The formula for this process can be found in equation 1. Where \( s_f \) is an activation function, meanwhile \( W \) representing a matrix of weight, and \( b \) representing the vector of bias.

\[
h = f(x) = s_f(Wx + b) \quad (1)
\]

The decoder utilizes another function \( g \) to convert the hidden representation \( h \) back to a reconstruction of \( x' \). The formula for this process can be observed in equation 2. With \( s_g \) is the activation function, \( W' \), \( b' \) representing the weight matrix and bias vector respectively.

\[
x' = g(h) = s_g(Wh + b') \quad (2)
\]

The selection of \( s_f \) and \( s_g \) are non-linear functions such as sigmoid, hyperbolic tangent (TanH), or rectified linear unit (ReLU). This grants the autoencoder the ability to recognize valuable features.
We can reduce the difference between $x$ and $x'$ in a regression or classification task with different function for each of them. In regression task, we can use square error function that can be found in equation 3. Meanwhile, in classification task, we can use the cross-entropy error function that can be seen in equation 4.

$$SE(x, x') = ||x - x'||^2$$ (3)

$$CE(x, x') = \sum_{i=1}^{n}(x_i \log x_i' + (1 - x_i) \log (1 - x_i'))$$ (4)

### 2.3 AutoRec

AutoRec is a CF model based on autoencoder [10]. AutoRec uses the vector of user rating $r^{(u)}$ or vector of item rating $r^{(i)}$ as an input to the encoder and returns the reconstructed rating in the decoder. Based on its input AutoRec has two types: item-based and user-based AutoRec (I-AutoRec and U-AutoRec respectively). For example, if there is an input $r^{(i)}$, the reconstruction can be seen in equation 5.

$$h(r^{(i)}, \theta) = f(W \cdot g(V \cdot r^{(i)} + \mu) + b)$$ (5)

AutoRec employs a typical autoencoder structure similar to the one depicted in figure 3. The goal of the model is comparable to the loss function of the autoencoder.

![Autoencoder Architecture](image.png)

**Figure 2.** Autoencoder architecture

We utilized U-Autorec in building a music recommendation system. The dataset includes $M$ users and $N$ items. Let $r_m \in \mathbb{R}^N$ be the preference vector for user $m$ which includes his preference scores for each of the $N$ items. The U-AutoRec decoder maps the $r_m$ into a representation vector $z \in \mathbb{R}^d$, where $d \ll N$ via $z = gt(r_m)$. The encoder $f(z)$ is used to reconstruct $h(r_m; \theta) = f(g(r_m))$, where $\theta$ is a model parameter [19]. The objective function of U-AutoRec is stated in equation 6.

$$\arg\min_\theta \sum_{i=1}^{N} ||r^{(i)} - h(r^{(i)}; \theta)||_O^2 + \lambda \cdot reg$$ (6)

In equation 6, $||r^{(i)}||_O^2$ implies that the loss is computed based on the user's observed preferences only. During prediction, the preference vector is reconstructed for each item.

### 3. RESULT AND DISCUSSION

#### 3.1 Dataset and Data Preprocessing

The dataset that will be used in modeling is the Million Song Dataset (MSD) The Echo Nest Taste Profile Subset which is publicly accessible. This dataset contains 2,000,000 music listening records from 76,353 users. Due to limited computational resources, we limited the number of records to the first 500,000 records. The dataset columns consist of `userid`, `songid`, and `listen_count` as shown in table 1.

<table>
<thead>
<tr>
<th>userid</th>
<th>songid</th>
<th>listen_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>b80344d063b5ccb321...</td>
<td>SOAKIMP12A8C130995</td>
<td>2</td>
</tr>
<tr>
<td>b80344d063b5ccb321...</td>
<td>SOBBDMDR12A8C13253B</td>
<td>5</td>
</tr>
<tr>
<td>b80344d063b5ccb321...</td>
<td>SOBXHDL12A81C204C0</td>
<td>13</td>
</tr>
</tbody>
</table>

In the dataset, a user listens to a piece of music at least once. On average, a user listens to the same music 3 times and 796 times at most. A user listens to 26 pieces of music on average. The `listen_count` distribution is shown in figure 3.
To facilitate modeling, user id and song id data are converted into indexes starting from 0. Since Autorec expects explicit feedback, we convert implicit feedback listen_count data into explicit feedback using binning technique. The value of 0-1 is converted into 1, 1-2 into 2, and so on until 10-796 (maximum listen_count) becomes 10. We assume that more than 10 total listens indicate that a user really likes the music. The preprocessed dataset can be seen in table 2. Finally, the data is shuffled and split to prevent overfitting. We used 90% of the data as train set and 10% of the data as test set. We use the test set for hyperparameter tuning [10].

**Table 2.** Dataset after preprocessing

<table>
<thead>
<tr>
<th>userid</th>
<th>songid</th>
<th>listen_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

3.2 Metric

We use the root-mean-square error (RMSE) evaluation metric to test the model. RMSE is a metric that is often used in many recommender system literature [20]. RMSE is a measure often used to calculate the difference between the value predicted by the model and the observed value.

$$RMSE = \sqrt{\frac{1}{n} \sum_{u,i} (p_{u,i} - r_{u,i})^2}$$  \hspace{1cm} (7)

The RMSE formula is presented in equation 7, $p(u,i)$ represents predicted rating of the user $u$ for item $i$, $r(u,i)$ represents the user’s real rating, whereas $n$ represents the count of all the user’s rating to each item. Lower RMSE means a more precise recommendations in predicting user ratings.

Since predicting a rating of 0 for a vector of user items makes little sense, we mask the loss function MSE and the metric RMSE [10]. Masking here means that we only consider errors where the actual label or rating is not 0.

3.3 Building and Training Autorec

Before experiment, we reproduced the original model to ensure that our implementation is comparable. We found that the RMSE of the model was similar to the original of 0.878. After that, we experimented with the activation function on the neural network. We also tried the experiment with the default rating of 0 and the average rating of a user on all items.

We found that there was no significant change in RMSE despite the combination of sigmoid and linear activation functions and default rating. However, the activation function sigmoid in the first layer and identity in the second layer had the lowest RMSE of 1.4078. The results of this activation function experiment are almost similar to Autorec’s research which got the best results with the identity function in the first layer and sigmoid in the second layer.

In the model training process, around the 150th epoch, the RMSE stagnated in the train and test data as shown in figure 4. In addition, it can be seen that there has been a slight overfitting in the training process as shown by the gap between the RMSE of train and validation. By trying other parameters, we have not been able to overcome this.

![Figure 4. Training and validation/test set RMSE](image)
Table 3 shows the best model setting for the U-Autorec. Referring to [10], the missing data rating is set to 3, whereas in this study we found that the default rating of 0 gives better results. For the Singular Value Decomposition (SVD) baseline, we used grid search to find the best regularization strength (λ) and latent dimension (k) with various set of λ and k values that ranges from 0-1000 for the λ and 10-500 for the k.

Table 3. Model setting

<table>
<thead>
<tr>
<th>Settings/parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>0.0001</td>
</tr>
<tr>
<td>Batch size</td>
<td>256</td>
</tr>
<tr>
<td>Epochs</td>
<td>500</td>
</tr>
<tr>
<td>L2 Regularization</td>
<td>0.0001</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>Activation functions</td>
<td>Sigmoid and Identity</td>
</tr>
<tr>
<td>Default rating</td>
<td>0</td>
</tr>
</tbody>
</table>

3.4 Experiment Result

Based on the experiment, the best regularization strength and latent dimensions are 0.1 and 500 respectively. The final results of the comparison can be seen in table 4. U-Autorec produces an RMSE of 1.408 which is lower than Singular Value Decomposition (SVD). The lower RMSE shows that U-Autorec is better at giving music recommendation than the SVD counterparts.

Table 4. U-Autorec and SVD RMSE comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-Autorec</td>
<td>1.408</td>
</tr>
<tr>
<td>SVD</td>
<td>2.165</td>
</tr>
</tbody>
</table>

The results of the experiment showed that the U-Autorec model performed well, with the best RMSE value of 1.408, which is lower than the baseline SVD. The activation function sigmoid in the first layer and identity in the second layer played a crucial role in achieving this low RMSE value. Additionally, the use of default rating 0 instead of the average of all movie rating for a given user as missing data rating gave better results.

One of the key findings from the experiment was the observation of overfitting in the model training process. Around the 150th epoch, the RMSE stagnated for both train and test data, as shown in figure 4. Overfitting is a common challenge in training machine learning models, and it occurs when the model becomes too complex and starts to memorize the training data, leading to a loss of generalization ability. To overcome overfitting, techniques such as early stopping, adding more data, or using regularization can be used. In this experiment, adding more data was not an option because of the limitation of the computing resources, and early stopping and increasing the regularization strength was not effective in preventing overfitting.

One reason why U-Autorec can outperform SVD is due to its ability to capture non-linear relationships in the data. U-Autorec is a deep learning-based method that uses a neural network to model the underlying relationships in the data. This allows it to learn complex and non-linear patterns in the data that traditional methods like SVD might miss.

Additionally, U-Autorec can also handle large amounts of sparse data more effectively than SVD. This is because the neural network in U-Autorec can learn to fill in the missing values in the data through the use of the autoencoder architecture. This makes it more suitable for recommendation systems where there is a lot of missing data or where the data is highly sparse.

In comparison, SVD is a linear method that relies on matrix factorization to model the relationships in the data. It works best when the data is dense and the relationships are linear. When dealing with sparse data or non-linear relationships, SVD can struggle to provide accurate recommendations.

In conclusion, while both U-Autorec and SVD have their strengths and weaknesses, U-Autorec tends to perform better in recommendation systems due to its ability to handle non-linear relationships and sparse data effectively.

4. CONCLUSION

This research aims to develop a music recommender system using the Collaborative Filtering (CF) paradigm with the autoencoder method, specifically U-AutoRec using music data containing implicit feedback, i.e., the frequency of music listening. We want to try to see how U-Autorec performs with such data because in its initial creation U-Autorec used explicit feedback. Then, the performance of the system is compared with the baseline matrix factorization (SVD) using the RMSE testing metric. Empirically, U-Autorec performed better than SVD with an RMSE of 1.408, which is about 0.7 less than SVD's 2.165. From this experiment, we notice that autoencoder can learn the hidden representation better than SVD in the music domain, especially on the MSD dataset. The
difference of this dataset compared to Autorec’s research dataset is in the difference of rating range and rating distribution. Due to time constraints, the baseline used only includes one matrix factorization method i.e., SVD. Future research can expand the baseline using other collaborative filtering methods and implement I-Autorec.

REFERENCES


