Deep Learning Methods for Recommender Systems

Andrew Singh, Edward Goebel
10-417 Intermediate Deep Learning
Carnegie Mellon University
Pittsburgh, PA 15213
andrewsi@andrew.cmu.edu, egoebel@andrew.cmu.edu

1 Background

1.1 Introduction

Given the increasingly large amount of content available on movie streaming services and the varying preferences of users, the ability to provide relevant recommendations to users based on their preferences is becoming more and more important. Users indicate their preferences through rating movies they have watched, and a service delivers personalized recommendations to the users. The vast amount of data available in the form of user ratings makes the problem particularly appropriate for machine learning techniques.

The input to the recommender system is a $U \times M$ ratings matrix R, where U is the number of users and M is the number of movies, and $R_{i,j}$ is user i's rating of movie j on an integer scale from 1 to 5 inclusive. The problem is to predict each user's rating on their unseen movies; that is, to fill in the missing values in the matrix.

To train and evaluate our models, we use ratings data from MovieLens. We use the MovieLens 1M dataset, which consists of 1,000,209 ratings of 3,900 movies made by 6,040 users, in addition to the timestamp of each rating. Each rating is an integer value from 1 to 5 inclusive. In our content-based approach, we additionally utilize the tag and genre data included in the dataset. Tags are strings that users apply to movies, and each movie has a set of genres assigned to it. Each model is trained on the same training set of ratings and evaluated by calculating the root mean squared error of its predicted ratings on the test set.

1.2 Background

The baseline model from our midway report implements the collaborative filtering method of recommender systems, using a matrix factorization technique described by Koren et al. [1] to predict user ratings. The input to the recommender system is the $U \times M$ ratings matrix as described in the Introduction. Each movie is a vector of n features, where the value of each feature indicates how strongly the movie has that feature. Each user is also a vector of the same n features, where each value indicates the user's affinity for that feature. Then to predict a rating for a given user and movie, we simply check if the corresponding feature vectors are a good match, which amounts to taking the dot product of the vectors and normalizing the result into the rating range 1 to 5. The movie and user's hidden features are determined through a learned embedding optimized through gradient descent with a mean squared error loss function.

Our midway report baseline model was trained on a smaller dataset, the MovieLens 100K dataset. This dataset consists of 100,000 ratings from 943 users on 1682 movies, where each rating is an integer between 1 and 5 inclusive. We measured the root mean squared error for multiple learning rates and numbers of features. Table 1 summarizes our results.

10417 Class Project (2019).

Learning Rate	10-Feature	20-Feature
0.001	**1.030**	1.040
0.01	0.968	**0.957**
0.1	0.971	**0.956**

Table 1: Test Loss (RMSE) of 10-feature and 20-feature models at different learning rates on MovieLens 100K.

These results led us to believe that the model with a learning rate of 0.001 had not yet converged. To explore this hypothesis, we trained the 20-feature model for 500 epochs at various learning rates. Figure 1 shows the test loss curves for these experiments.

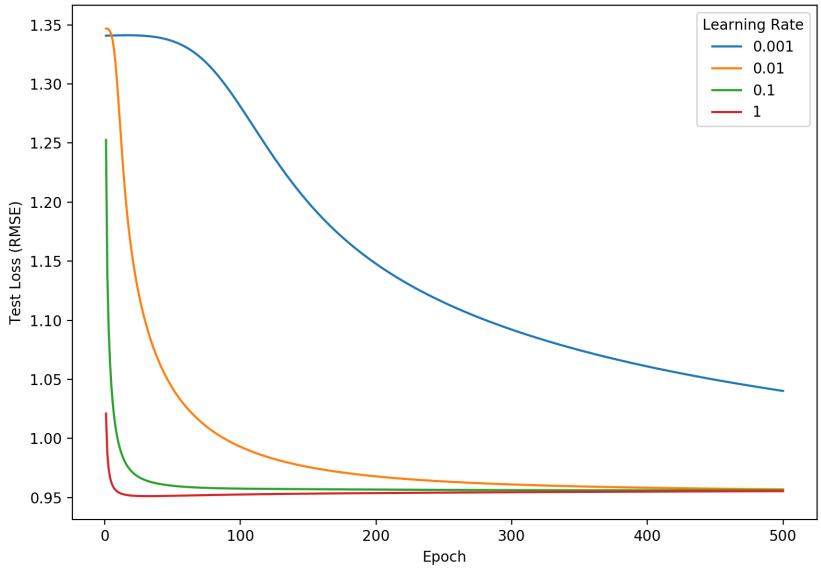

Figure 1: Performance of 20-feature model at different learning rates on MovieLens 100K.

As shown in Figure 1, the 20-feature model had not converged after 500 epochs as the models with larger learning rates had.

Our preliminary model served as a benchmark for our later experiments. An average loss of 0.956 can be interpreted in this domain as the model being 0.956 off in a typical rating prediction from 1 to 5.

1.3 Related Work

The inspiration for our baseline model was the aforementioned matrix factorization method used for collaborative filtering in recommender systems described by Koren et al. [1]. The underlying assumption of collaborative filtering is "If person A likes movie X, and person B is similar to person A, then person B will like movie X." We explored collaborative filtering methods as our baseline because such methods are generally more successful than content-based methods which consider only movies a given user has watches in order to find patterns in the user's preferences which can be generalized [1]. The matrix factorization method involves using a learned embedding to convert each user and movie into a vector of n features. A prediction of the user's rating for that movie can be determined by calculating the cosine similarity between the vectors. While this method works reasonably well, we explored other methods which could improve performance.

We explored other recommender system models which attempt to improve prediction performance and take advantage of the wealth of data available. One such model by Salakhutdinov et al. [2] uses Restricted Boltzmann Machines (RBMs) to model user ratings. The model addresses limitations to preexisting approaches to collaborative filtering methods which could not handle large datasets. The researchers' model introduced a class of of RBMs to model tabular or count data, such as movie ratings. The model was trained on the Netflix dataset with over 100 million rating datapoints and was found to outperform existing singular value decomposition models as well as Netflix's own system. This research led us to continue exploring the user of deep learning techniques for recommender systems. We were also interested in models which utilize additional features of the rating data to improve predictive performance.

In addition to user ratings, the MovieLens dataset provides additional data which we hypothesized could be used to improve rating predictions. One additional piece of data is a list of tags a particular user assigns to a particular movie to describe it. Another piece of data is a timestamp of when each user rating was submitted. Utilizing this additional information is an active area of research, and we analyzed the work of Liang et al. [3] who developed a tag-aware recommender system which utilizes both tags and rating timestamps in predictions.

The TRSDL (tag-aware recommender system based on deep learning) model developed by Liang et al. [3] addresses two inherent issues with tags which exist because they can be any words assigned by users. The first issue is that different tags may be used to express similar concepts. The second issue is ambiguity due to the same tag having multiple possible meanings. TRSDL uses Word2Vec to map words to K-dimensional vector embeddings. Word2Vec reduces the complexity of text processing and captures semantic information present in the tags. The distance between such vectors can be used to assess the semantic similarities among tags. TRSDL also organizes user ratings by time and uses an LSTM to capture evolving user preferences over time. For instance, after watching a movie, one may be likely to watch its sequel, or another movie with the same director or lead actor.

In addition to the aforementioned uses of tags and rating timestamps, the TRSDL model utilizes deep neural networks to better extract latent features, leading to increased accuracy and ability to generalize to new data. Specifically, TRSDL combines two models: an item model, and a user model. The item model is a deep neural network which takes as input the average word embedding of the tags for a given movie. The user model generates users' latent features by feeding the tags for each movie a user has rated to an LSTM. The learned latent features of the items and user are used to predict the corresponding user rating for that movie. Liang et al. [3] report that this model outperforms all baseline models they compared it to.

We seek to expand upon this work in our own model which incorporates tags, rating timestamps, and deep neural network structures. Specifically, for the item inputs we use more descriptive pre-trained GloVe embeddings, as well as including a one-hot vector of the item's genres. In addition, we build an ensemble model that incorporates both matrix factorization and our content-based deep learning model to achieve greater performance than either model alone.

2 Methods

2.1 Extended Baseline (Biased MF)

The baseline method in our midway report was conventional matrix factorization, without any biases. The only parameters were the feature vectors for the users and items. In addition, that model was trained on the MovieLens 100K dataset, not the MovieLens 1M. For our final report, we have extended the baseline to biased matrix factorization, where in addition to a feature vector, each user and item now has a scalar bias parameter. The rating prediction is now given by:

$$r_{ui} = \mu + b_i + b_u + {\mathbf{q}_i}^T \mathbf{p}_u \tag{1}$$

where r_{ui} is user u's rating on item i, μ is the global average rating, b_i and b_u are the item and user biases respectively, ${\mathbf{q}_i}^T \mathbf{p}_u$ is the user-item interaction: the dot product of their respective feature vectors. In addition, we increased the size of the feature vectors from 20 to 30. Models with more than 30 latent features did not appear to significantly improve performance, and more than 50 began to overfit the data and worsen performance.

2.2 Content-Based Model

From our background reading we realized that a major challenge of recommender systems is sparsity: most users have only rated very few movies, and most movies have very few ratings. To address this challenge, we concluded that our model needed another source of data in addition to the ratings matrix. Drawing upon the methods described by Liang et al. [3], we decided to utilize the tag data included in our MovieLens dataset. A tag is simply a string that a user applies to a movie's profile page on the MovieLens website. In addition to applying a new tag, users also have the option of "upvoting" a current tag, essentially applying that tag again. Our intuition was that the top n tags of the model by number of applications would provide a semantic representation of the movie.

Our content-based model draws upon the architecture proposed by Liang et al. [3]. We have an item model that computes item feature vectors, a user model that computes user feature vectors, and a bias for each item and user. The rating prediction is the same as matrix factorization; the difference is how the user and item feature vectors are learned.

2.2.1 Item Model

The item model assumes that a movie is fully described by its genres and most popular tags. The model takes in item tags and genres as input and computes the item's latent feature vector. The input is a concatenation of two components: the genre vector and tags vector. The genre vector is a one-hot encoding of the item's genres selected from a list of 18 genres. Note that an item can have multiple genres. The tags vector is a concatenation of the embeddings of the item's top 6 tags by number of tag applications. The embeddings are 50-dimensional pre-trained GloVe embeddings from GloVe's Twitter dataset. Since item tags are simply strings applied by users, they can be anything at all. We decided to use GloVe's Twitter embeddings because we believed that the tag content is better captured by a system trained on social media, where language has more flexibility, than on sources of grammatically correct English such as Wikipedia. We believe that these embeddings provide a more accurate semantic representation of the item than the Word2Vec embeddings used by Liang et al. [3].

The item model architecture is a single-layer perceptron with 250 hidden units and ReLU activation. We attempted training a two-layer network, but the model didn't perform as well, likely due to overfitting the data because of the additional parameters. Given a larger dataset, additional layers could have possibly improved performance, but for MovieLens 1M, a single hidden layer was sufficient.

2.2.2 User Model

The user model assumes that a user is fully described by the movies they have seen. The model is an LSTM that takes a sequence of item feature vectors as input and computes the user's latent feature vector. For a given user, the input to the model is the sequence of item vectors corresponding to the movies they have rated, ordered by the time they rated them. By processing input sequentially, our model can account for changes in the user's preferences over time as well as temporal patterns in their movie watching behavior.

The model architecture is a single-layer LSTM with a hidden dimension equal to that of the user and item feature vector; in our case, 30. Note that the item feature vectors that are given to the user model as input are the vectors that are computed by the item model.

2.3 Hybrid Model

The model described thus far is content-based: while it uses the ratings data to learn the parameters, these parameters are shared among all users and items. The model's assumption that an item is fully described by its genres and tags, and that a user is fully described by their movie history, doesn't directly consider each item's and user's ratings. Our collaborative filtering model, on the other hand, only considers the ratings data. After training each model independently, we created a hybrid model that incorporates both our content-based and our collaborative filtering models.

The hybrid model consists of the user model and item model described above, along with the biased matrix factorization model. With the additional factor, the rating prediction becomes

$$r_{ui} = \mu + b_i + b_u + \mathbf{q}_i{}^T \mathbf{p}_u + \mathbf{r}_i{}^T \mathbf{s}_u \tag{2}$$

where the other factors are as in equation 1, and $\mathbf{r}_i{}^T \mathbf{s}_u$ is the user-item interaction of our content-based model: the dot product of the item and user feature vectors respectively.

3 Results

We evaluate our models through their average Root Mean Square Error (RMSE) on the validation and test datasets. The RMSE for a model is calculated as

$$RMSE = \frac{1}{N} \sum_{1}^{N} \sqrt{(y_{ui} - r_{ui})^2} \tag{3}$$

where y_{ui} is the true rating of user u on movie i, r_{ui} is the rating predicted by the model, and N is the number of ratings in the dataset.

3.1 Biased MF

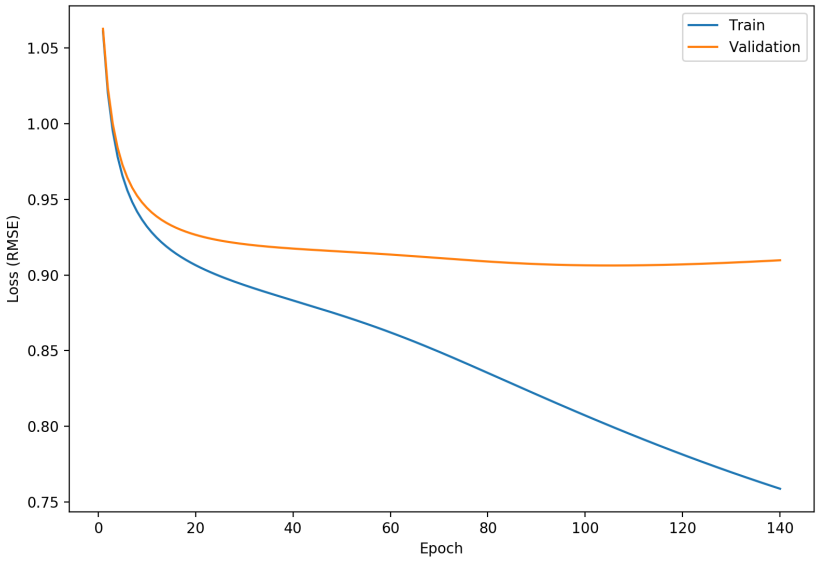

Figure 2: Performance of Biased Matrix Factorization Model on MovieLens 1M.

Final Test RMSE: **0.9063**

This result is fairly unsurprising, as we expected the enhanced baseline model to outperform the 0.956 RMSE achieved by our midway report baseline. Compared to that model, this one has biases, more factors, and a larger dataset.

3.2 Content-Based Model

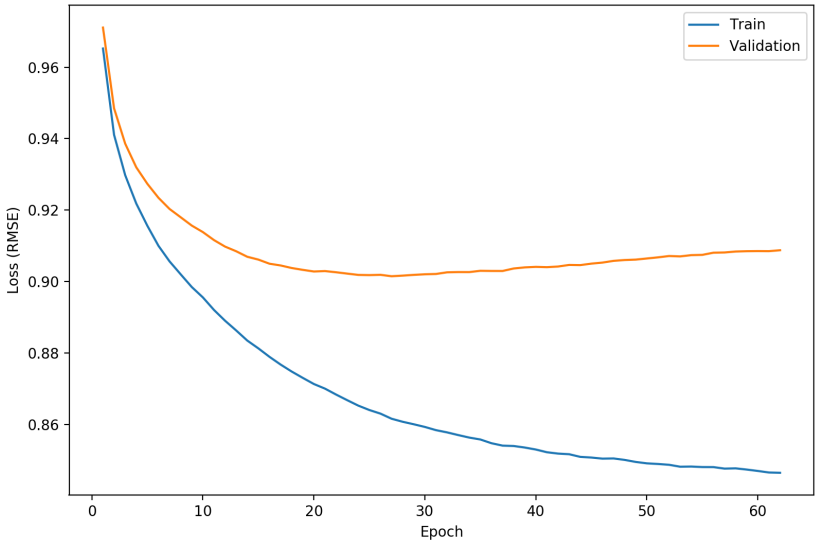

Figure 3: Performance of Content-Based Model on MovieLens 1M.

Final Test RMSE: **0.9015**

We found the results of the content-based model alone to be interesting. Generally, content-based models don't perform as well as collaborative filtering models and thus aren't as widely used. Our content-based approach actually slightly outperforms our Biased Matrix Factorization model. This result shows that the inductive biases made by the content-based model are accurate to a degree.

3.3 Hybrid Model

To speed up training of the hybrid model, we initialized the parameters of the component models to their already-trained values. The two models were essentially pre-trained individually, and then fine-tuned in this training phase. As a result, the model converged very quickly. A possible area of future exploration is whether the model would achieve better performance by training from scratch, without any pre-training of component models.

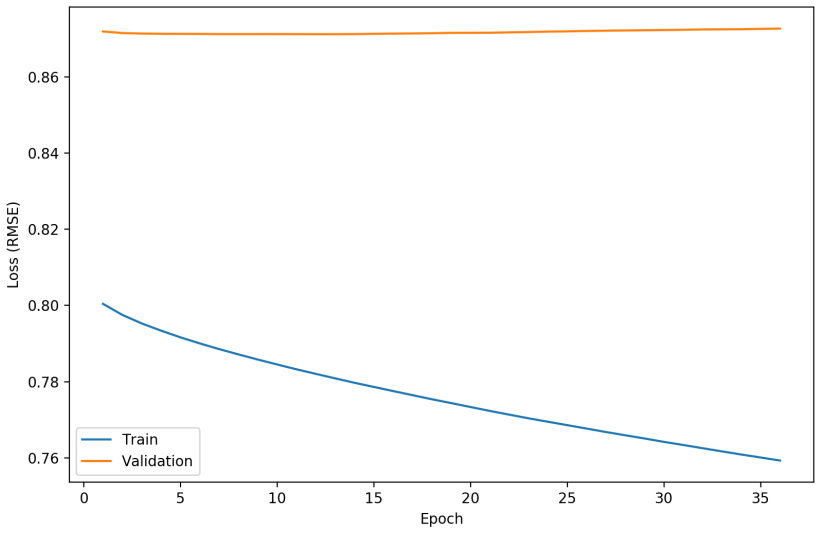

Figure 4: Performance of Content-Based Model on MovieLens 1M.

Final Test RMSE: **0.8713**

After the result of the content-based model, this hybrid model result is in line with our expectations. Because the Biased MF model and the content-based model each capture different features of the users and items, we expect that combining the two models will yield an improvement in performance over either of the models alone.

3.4 Summary of Results

Model	**Test RMSE**
Biased MF	0.9063
Content-Based	0.9015
Hybrid	**0.8713**

Table 2: Test RMSE of various models on MovieLens 1M.

4 Discussion and Analysis

The standard deviation of the ratings in the test set is 1.1175. This value can be interpreted as the RMSE of a model that predicts the mean every time. We can see that the Biased MF model improves upon this approach significantly, and our content-based approach actually manages to slightly outperform Biased MF alone. The major result is our hybrid approach, which shows that combining collaborative filtering with a content-based method yields greater performance than a model that implements either one of these approaches alone. The significant performance improvement of the hybrid model over either of the other models indicates that there are potentially benefits to a model which considers both the rating data of individual items and users (content-based features), as well as an item's genre and tags and a user's movie history (collaborative filtering features) when making predictions.

4.1 Future Improvements

We have considered improvements that could be made to our model to improve performance. One potential improvement would be to train the model on a larger dataset. MovieLens has a dataset with 20 million ratings, and while training a model using this dataset would require more computational resources, doing so could enhance the model's ability to generalize.

After incorporating tags and timestamps into our model and noting the improvements over the baseline model, we believe that incorporating additional available information into our model could improve the model. For instance, some MovieLens datasets also include gender, age, occupation, and zip code information for users. Such information could potentially improve user representations in our model as well as enable the model to better generalize relationships among users by incorporating this explicit demographic information.

An interesting prospect to consider when evaluating recommender systems is the limit on performance. Surely, there is inherent variability in user preferences which result in unavoidable prediction error. However, with the introduction of more data, more data features such as tags and the demographic features we discussed, and the development of more sophisticated models, performance of recommender systems will continue to improve.

References

[1] Y. Koren, R. Bell and C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30-37.

[2] Salakhutdinov, R., Mnih, A., and Hinton, G. E. (2007). Restricted Boltzmann machines for collaborative filtering. In ICML 2007.

[3] Liang, N.; Zheng, H.T.; Chen, J.Y.; Sangaiah, A.K.; Zhao, C.Z. TRSDL: Tag-Aware Recommender System Based on Deep Learning–Intelligent Computing Systems. Appl. Sci. 2018, 8, 799. doi:10.3390/app8050799.