COMP9417 Assignment 2 Report:
Movie Recommendation System

Introduction

Today there is so much online content available to people it is difficult to be aware of it all let alone decide which choose. As such, companies like Amazon and Netflix have implemented recommendation systems to ensure users are able to discover items, movies and other content that are relevant to them (Arora, 2016). Accordingly, several types of recommendation systems have developed over time, some of which will be explored in this project.

The primary goal of this project is to implement movie recommendation systems that use collaborative filtering algorithms based on:

- Memory based approach:
  - Amazon Item-Item Collaborative Filtering
  - User-User Collaborative Filtering
- Model based approach: Matrix Factorization using Stochastic Gradient Descent
- Hybrid approach: Weighting between memory and model based approaches

Features of dataset

There are 3 types of data from the downloaded dataset:

1. User data: Data for each user includes user ID
2. Movie data: Data for each movie includes movie ID
3. Ratings data: These are rating values between 0-5 for user-movie pairs. Not all user-movie pairs are used

Method

A sample of 100,000 movie ratings were downloaded from the GroupLens site at http://www.grouplens.org/node/73. The data sets were collected over various periods of time, depending on the size of the set. Preprocessing was used to convert the data files into pandas' data frame objects in Python. From the ratings data, a ratings matrix is derived where the index values were user IDs and the column values were movie IDs. The method for each recommender implemented (based on lecture slides):

1. Amazon Item-Item Collaborative Filtering: Use ratings matrix to create a movie similarity matrix by applying the cosine similarity function to movies' vector of user ratings for every movie pair. Calculate the predicted rating for all unknown ratings by multiplying the user's ratings vector with the movies similarity vector and scaling it.
2. User-User Collaborative Filtering: This is similar to the approach above, except a user similarity matrix is calculated and used instead. A user similarity vector is then multiplied by that user's ratings and a scalar value to predict a rating.
3. Matrix Factorization using Stochastic Gradient Descent (SGD): Estimate the latent features for each user and the latent features for each movie using SGD iteratively updating each parameter by the gradient of the cost curve with respect to that parameter. Once the parameters have been optimized, unknown ratings may be calculated by multiplying the user's and movie's latent feature vectors to obtain a prediction.

4. Hybrid: This involved combining the methods from the memory and model based approaches. It is based on the researched idea of switching between different models based on a criteria of the dataset (Ghazanfar and Prugel-Bennett, 2010). As memory based approaches perform worse with sparse data, the hybrid approach involves switching between the memory and model based approach when sparsity is above a specified threshold. This method was implemented for the project but not tested alongside others as it is essentially the memory and model based approaches shown.

Results

- Root Mean Squared Error (RMSE) for 100k dataset using K-folds cross-validation (see Appendix Figure 2 for standard output and results directory for predictions):
    - Amazon item-item similarity model: 2.1519 RMSE
    - User-User similarity model: 2.6561 RMSE
    - Matrix Factorisation with Stochastic Gradient Descent: 0.9914 RMSE

Furthermore, the "time python" command was used to measure the time taken on the 100k dataset with the model based approach outperforming both memory based approaches (see Figures 3). Additionally, I tuned the hyper parameter for how many iterations to run whilst optimizing user-item features using stochastic gradient descent. As seen in Figure 1 in the Appendix, the prediction error falls sharply as the number iteration increases with a negligible amount of error change after 50 iterations. As such, the default iteration number was set to 50.

Discussion

The user/item similarity memory-based approaches have a sharper decrease in prediction accuracy compared with model-based approaches as the sparseness of data increases. The reason for this is likely to do with the fact that similarity based approaches assume values (such as zero in this project) for unknown ratings. This is necessary for the model to work but inserts information bias in the process. As the sparsity increases, more information bias is inserted thus decreasing the accuracy of these approaches. On the other, matrix factorization creates a low rank approximation of the ratings matrix where there is no information bias added in.

Related work

The models implemented as part of this project are quite popular and been around for a while. The item-item similarity approach to recommenders was invented by Amazon to improve the efficiency of their item recommendations to customers (Linden, Smith and York, 2003). The matrix factorization approach using stochastic gradient descent was developed as a solution to the Netflix Prize Competition to crowdsource recommender solutions to improve the company's movie recommendations (Koren, Bell and Volinsky, 2009). Since then there have been significant developments in these recommender models. In particular, a new model called Neural Network Matrix Factorisation has been developed which usually outperforms the version implemented for this project (Dziugaite and Roy, 2016). This model concatenates estimated user and item features which is then passed into a neural network to predict ratings.

Conclusions

It can be concluded from the results that the three models have varying levels of effectiveness when it comes to predicting user-movie ratings. In general, the user/item similarity memory-based approaches seem to have less effectiveness then the matrix factorization model-based approach.

Acknowledgements

The following sites aided in learning how to implement recommender models in Python:

- Similarity memory based model: http://www.salemmarafi.com/code/collaborative-filtering-with-python/
- Matrix factorization with SGD: http://nicolas-hug.com/blog/matrix_facto_4
- Reading in Python with pandas data frame: http://www.gregreda.com/2013/10/26/using-pandas-on-the-movielens-dataset/

References

Arora, S. (2016). Recommendation Engines: How Amazon and Netflix Are Winning the Personalization Battle. [online] Martechadvisor.com. Available at: https://www.martechadvisor.com/articles/customer-experience-2/recommendation-engines-how-amazon-and-netflix-are-winning-the-personalization-battle/ [Accessed 28 May 2018].

Linden, G., Smith, B. and York, J. (2003). Amazon.com Recommendations. [online] IEEE Computer Society, p.1. Available at: https://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf [Accessed 1 Jun. 2018].

Koren, Y., Bell, R. and Volinsky, C. (2009). MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS. [online] IEEE Computer Society, p.1. Available at: https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf [Accessed 1 Jun. 2018].

Dziugaite, G. and Roy, D. (2016). NEURAL NETWORK MATRIX FACTORIZATION. [online] ICLR. Available at: https://arxiv.org/pdf/1511.06443.pdf [Accessed 1 Jun. 2018].

Ghazanfar, M. and Prugel-Bennett, A. (2010). An Improved Switching Hybrid Recommender System Using Naive Bayes Classifier and Collaborative Filtering. [online] IMECS. Available at: http://www.iaeng.org/publication/IMECS2010/IMECS2010_pp493-502.pdf [Accessed 2 Jun. 2018].

Appendix

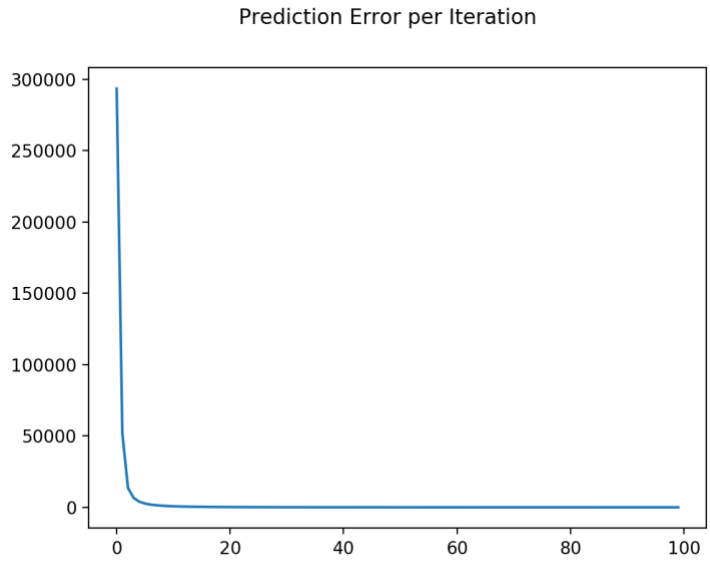

Figure 1. Prediction error falls as iterations increase

```
('Movie Movie RMSE is: ', 2.1518560879697697)
```

Figure 2.1. Item-Item Similarity RMSE

```
('User User RMSE is: ', 2.656149329379307 9)
```

Figure 2.2. User-User Similarity RMSE

```
('MF SGD RMSE is: ', 0.9913593782722781 6)
```

Figure 2.3. Matrix Factorisation RMSE

```
real    52m2.346s
user    49m35.589s
sys     0m14.800s
```

Figure 3.1. User-User Similarity time taken

```
real    83m16.685s
user    80m23.469s
sys     0m20.981s
```

Figure 3.2. Item-Item Similarity time taken

```
real    15m21.207s
user    14m3.574s
sys     0m4.578s
```

Figure 3.3 Matrix Factorisation time taken