PRML Data Contest Documentation

(Team Goodfellas)
Vishwesh and Rahul
CH16B024 and AE16B005

Introduction:

There are several methods to solve the movielens data contest. In our documentation, we explore different methods used to tackle the problem, and comment on what worked the best and what didn’t work.

Models which didn't work

We tried a lot of models and a major portion of them didn't work.

1) Neighbourhood model: the score improvement that was given by this model was not worth the computation effort it was taking
2) Neighbourhood model on regression instead of baseline: this also didn't give as much results as the algorithm was simple over fitting and it's hypothesis didn't make sense as the neighbourhood model on baseline did
3) Neighbourhood model on ensemble of regression and baseline
4) Neighbourhood model on ensemble of latent factor model
5) Latent factor on ensemble of regression and baseline
6) Latent factor on regression
7) Latent factor on neighbourhood model
8) User regression and regression on movies ensemble: This probably happened because user regression is a local and smaller version of movie regression. Hence we didn't get a good improvement from before.

These model didn't give much improvement , we also tried other models which we didn't include in the end because we got better results using other methods

1) Neural networks regression
2) Latent factor
3) Baseline

Models which worked:

1) Repeated matrix reconstruction
2) Regression on movies
3) Regression on users

We realized from our tries that ensembles works well only if your models are independent of each other and if it tackles the problem in a different way. Hence developing models using different methods helped us in reducing our MSE error.

Repeated Matrix Reconstruction[1]

Consider the ratings matrix X, with rows as User Ids and columns as Movie Ids. A natural way to tackle this problem is to group users and movies into genres, like we did in latent factor model. Based on this similar idea, we believe that the matrix X consists of a lower rank, hence low rank approximation is well suited for problem such as this. Here, the author attempts to decompose the matrix using SVD into smaller rank matrices
$X = USV.T$, which is analogous to user groups and movie genres

Pseudo-code:

```
Preprocessing step
    1) First generate a rating matrix X from both train and test file
    2) Mean shift the rating matrix X w.r.t rated entries
    3) Make the non-rated entries as 0. These are the entries which we will procure by
       USV.T

Repeated matrix reconstruction
    1) Using truncated SVD, get the new matrix X with n-principle components. This is
       got by doing SVD and selecting the top n eigenvectors of V, U and n eigenvalues
       S and take the product as ,
                       Xnew = U1S1V1.T
    2) Since this approximation is not guaranteed to keep the original rated entries as
       same, we reset those values to the original rating, and repeat step 1 again, but
       without any change in the non-rated entries
    3) If there is truly a lower rank matrix, this algorithm should converge
    4) Mean shift it back to get the ratings
    5) There are movies which no user rated in train, we fill those values using
       regression values mentioned below.
```

Based on the results of research paper[1] which we followed, we ensemble models with number of PCs-15 and with number of PCs-20

Just using the model, we got **MSE = 0.76**

Regression on movies:

In our algorithms for baseline, neighbourhood models or latent factors , we are not using the information about the movies. In essence of collaborative filtering , we are just trying to group movies into genres and users into groups for most of the algorithms. For instance, just using the information that the movie has Tom Hanks or it is made by Steven Spielberg we can surely say that the movie will perform above average. Here we take advantage of the different informations that is provided for each movies given in the genome.csv. We also included the genres as features. We ignored the tags.csv here because based on the data provided in the movies, most of the input felt like noise.

Psuedocode:

1) Build the movie matrix by taking a movie and building its features from genome.csv.
2) Build Y vector as the average of all users for each movies. For movies not rated , put ratings as -1
3) Build SVR model using the rated movies from movie matrix and Y vector.

Just using the model, we got **MSE = 0.79**

Regression on users:

We build on the top of regression on movies. We now also consider the effect of users. Since we don't have any user specific knowledge, we build a separate model for each user. For each user, we find the movies rated by that user and using the features that we procured earlier, we build a user specific training matrix and build a model for that user. The procedure is same for training.

Since we encountered a lot of users in test but not in train, we replaced those values with regression on movies.

Psuedocode:

1) For each UserId,_create a matrix of movies rated by that user with each row containing the 1128 features of the movie.
2) Pass this matrix through a regressor, We used Kernel Ridge Regressor from sklearn. The kernel used was 'rbf'. Regularization was set to auto.
3) Since each of the model required a lot of memory to store, we decided not to store the models.
4) Instead, for each userId, after fitting the model immediately predicted on the test dataset for all the entries containing that userId.
5) The ratings which remained zero after step 4 are replaced by the ratings from movie based regression model.

Just using the model, we got **MSE = 0.76**

End model

The final prediction is the ensemble of user regression and repeated matrix reconstruction .Based on the leaderboard scores, we improved our scores by varying the weights associated to each model.

Final Prediction = (2*Pred_repeated_matrix_reconstruction + User_regression)/3

Final MSE = 0.728

References:

1) http://cs229.stanford.edu/proj2006/KleemanDenuitHenderson-MatrixFactorizationForCollaborativePrediction.pdf
2) http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/
3) **Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model** by Yehuda Koren
4) **The BellKor solution to the Netflix Prize** by Robert M. Bell, Yehuda Koren and Chris Volinsky.