Report on PRML Data Contest

(By- cs18s038,cs18d020)

Best MSE on Kaggle : .738

Datasets we used for the prediction:

Train, test, validation, genome_scores

Approach1: Using baseline as μ

The mean rating of all movies was calculated and the same was used as constant rating for all prediction. Test MSE (Leaderboard score) = 0.98

Approach2: Using baseline as $\mu + b_u + b_i$

b_u for every user *u* in train set was calculated and b_i for every movie *i* in the train set was calculated. For test set, the predicted rating is calculated as follows

- If user *u* and movie *i* present in the train set, then predicted the rating is $\mu + b_u + b_i$.

- If user *u* is present in the train set and movie *i* not present, then predicted rating is $\mu + b_u$.

- If user *u* and movie *i* not present in the train set, then predicted the rating is μ. Test MSE (Leaderboard score) = 1.01

Approach3: Using Neighborhood Models

Each movie was represented as a feature vector of 1128 dimensions using genome_scores dataset. For user *u* and movie *i* in test set, we found k nearest movies to movie *i* that are rated by user u (k = 3, 7, 16, 32). A simple average was taken and also weighted average was taken using cosine similarity and similarity defined using Euclidean distances. The average MSE using all these approaches was nearly same. Test MSE (Leaderboard score) = 0.92

Approch4: Modified Latent Factor model

Details of the best model is below:

My model uses the latent factor model as base and with using genome_score we manage to come up with good prediction.

Steps involved:

Data preprocessing of genome_scores:

We made the matrix of movieId and attributeId. We got rows of relevance corresponding to each movie.

Then we calculate the cosine similarity between each pair of movie on the basis of relevance and made similarity matrix of movies.

Training Process:

1. Using matrix factorization method, we created two matrix of size P*F and Q*M. Where 'P' is the number of users in the training set, Q is the number of movies in training set and F(hyperparameter) is the number of hidden features to be discovered. And initialized with some random values.
2. Also initialized array of average rating given by each user 'bu' and average rating got by each movie 'bd' by some random values.
3. Equations used for updating corresponding user feature vector and corresponding movie feature vector:

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + \alpha(2e_{ij}q_{kj} - \beta p_{ik})$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + \alpha(2e_{ij}p_{ik} - \beta q_{kj})$$

$$bu'_i = bu_i + \alpha \times (e_{ij} - \beta bu_i)$$

$$bd'_j = bd_j + \alpha \times (e_{ij} - \beta bd_j)$$

Where are α is learning rate and β is regularization hyperparameter and e is the error

4. We have used the stochastic gradient descent to train it.
5. Ater each epoch we have used validation data to control the overfitting the model

Testing or Prediction Process:

6. As four cases are possible while predicting the rating for a movie and user.

A) when both the user and movie is present in training data.

Predicted rating= (np.dot(p[u],q[m])+mu[0] + bu[u] + bm[m])

P[u] is a feature vector of user,

q[m] is a feature vector of movie m

bu[u] is an average rating given by user u,

bm[m] is an average rating of movie m

Mu is an overall average rating

B) when the user is present but movie is not present.

In this case we are using genome_scores for finding movie average rating. We find k similar movies using above similarity matrix and taking b[m] of each movie which are present in training set from k movies and taking average of them and assigning this to current b[m] and then make prediction.

rating = mu +bu[u]+mean

C)when movie is present but the user is not present.

rating = mu +bm[m]

D)when both are not present.

Again handling the movie as in case B)

rating = mu +mean

Experimental results:

Hyper parameters involved in the model

α = learning rate

β = regularization

best_similar_movies

K = number of hidden factors/features

Epochs

(**validation data is also added in training data)**

Experiment no.	Learning rate	regularization	Best_similar_movies	k	epochs	MSEon Kaggle
1	.005	.02	50	10	15	.752
2	.005	.02	50	50	12	.749
3	.005	.02	50	100	12	.75
4	.005	.02	50	70	15	.740
5	.005	.02	50	70	12	.738