Assignment 2 - KNN Recommendation System

Team name: M2 Robo

Chan Kwan Yin
3035466978
Team leader

Lee Chun Yin
3035469140

Chiu Yu Ying
3035477630

1. Background

In the KNN (k-nearest neighbours) model, we assume that similar users will give close ratings on similar movies. To predict the rating $\hat{r}_{ij}$ of user i on movie j ($1 \leq i \leq m, 2 \leq j \leq n$), the k most similar users ($n_1, \ldots, n_k$) to user i are computed based on similar choices of ratings, and $\hat{r}_{ij}$ is estimated based on the known values $r_{n_1 j}, \ldots, r_{n_k j}$.

1.1. Similarity (Nearness) metrics

The similarity metric $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ used in the KNN model satisfies

$$\begin{cases} d(x,y) = 0 & \iff x = y \\ 0 < d(x,y) \leq 1 & \iff x \neq y \\ d(x,y) = d(y,x) & \forall x, y \end{cases} \tag{1}$$

While $d(i,j) < d(i,k)$ implies that j is more similar to i than k is, it is not necessary that the metric follows the triangular inequality. This enables the use of correlation coefficients in addition to common norm metrics.

1.1.1 Pearson correlation

Considering a user u_i rates movies with a distribution $R_i \sim (\mu_i, \sigma_i)$, the correlation between user u_i and the other user u_j would be the coefficient between the two distributions R_i and R_j:

$$\rho_{ij} = \frac{E[(R_i - \mu_i)(R_j - \mu_j)]}{\sigma_i \sigma_j}$$

To get the k similar data by considering the n movies that both user u_i and u_j have, we estimate the co-variance and variances:

$$E[(R_i - \mu_i)(R_j - \mu_j)] \approx \frac{1}{n} \sum_k (r_{ik} - \mu_i)(r_{jk} - \mu_j)$$

$$\sigma_i \approx \sqrt{\frac{1}{n} \sum_k (r_{ik} - \mu_i)}, \sigma_j \approx \sqrt{\frac{1}{n} \sum_k (r_{jk} - \mu_i)},$$

Note that the range of the Pearson Correlation Coefficient is $[-1, 1]$. To make the coefficient satisfy the conditions in Equation 1 and not have negative values, we definean extraordinarily narrow situation

$$d(i,j) = \frac{1}{2}(1 - \rho_{ij})$$

such that d has the range $[0, 2]$. A smaller value implies higher similarity between users.

1.1.2 Spearman Correlation

Contrary to Pearson's choice of mean and variance, Spearman Correlation uses the ranking of variables in the vector as the metric to eliminate effects of non-uniform distributions.

$$\rho_{ij} = \frac{\mathrm{Cov}(\mathrm{rank}_i, \mathrm{rank}_j)}{\sigma_{\mathrm{rank}_i} \sigma_{\mathrm{rank}_j}}$$

Similar to Pearson Correlation, the metric based on Spearman correlation is defined as

$$d(i,j) = \frac{1}{2}(1 - \rho_{ij})$$

1.1.3 Euclidean Distance

We take the 2-norm squared:

$$d(i,j) = \sum_{l=1}^{n} (r_{il} - r_{jl})^2$$

1.1.4 Taxicab as a similarity metric

We take the 1-norm:

$$d(i,j) = \sum_{l=1}^{n} |r_{il} - r_{jl}|$$

1.1.5 Cosine similarity

This can be thought as the cosine of the angle between the two vectors of ratings. We define

$$\rho(i,j) = \frac{1}{\|\tilde{\mathbf{r}}_{i\cdot}\| + \|\tilde{\mathbf{r}}_{j\cdot}\|} \sum_{l=1}^{n} (\tilde{r}_{il} \tilde{r}_{jl})$$

We further rescale the metric to the range $[0, 1]$ using the same strategy as in correlations:

$$d(i,j) = \frac{1}{2}(1 - \rho(i,j))$$

This metric is sensitive to the exact value (rather than just relative difference); a negative value could imply that the user does not like the data. Therefore, we propose two mechanisms for rescaling the ratings, using $\tilde{r}_{ij} = r_{ij} - \overline{\mathbf{r}_{\cdot\cdot}}$ and $\tilde{r}_{ij} = r_{ij} - 3$ respectively, where 3 is the standard rating for "neutral".

1.2. Aggregation of ratings

After the k nearest neighbour users have been selected, $\hat{r}_{ij}$ can be estimated by aggregating $\{r_{n_1 j}, r_{n_2 j}, \ldots, r_{n_k j}\}$. A naive aggregation method is to just take the arithmetic mean of these rating values without considering other factors such as the actual correlation.

Considering the case when all the similar users do not have a rating of the movie, the rating may be high due to the other less similar users. We believed that implementing the weights based on the similarity is important:

$$\hat{r}_{ij} = \mu_j + \frac{\sum_{v \in P(i)} d(i,v)(r_{vj} - \mu_j)}{\sum_{v \in P(i)} |d(i,v)|}$$

This formula increases the effect of more similar users on the rating prediction.

2. Technical Details

The training data set provided by Netflix consists of more than 100 million ratings with 17770 movies and 480189 users. Such a huge data set would consume a significant amount of training time and memory ($O(m^2 n)$, since a correlation matrix between users is to be constructed), which is not possible for our hardware available. Therefore, only a subset of data is used for evaluation. To be specific, only first 10000 movies and first 1000 users that appear in the data set are considered.

2.1. Data preprocessing

The first 1000 movies are loaded into a numpy array with columns of Movie ID, User ID and Rating. The movie IDs and user IDs are reordered from 0 for the ease of indexing. Approximately 80% data are then reformatted into a rating matrix $R \in \mathbb{R}^{m \times n}$ for training, where r_{ij} is the rating of user i on movie j; the rest are retained for performance evaluation. The retained and missing data are imputed with the mean rating for the corresponding movie.

2.2. Hyperparameter selection

In this KNN model, there are three hyperparameter to be selected, namely

- Value of k
- Similarity metric
- Aggregation function

2.2.1 Choice of k

A large value of k would reduce accuracy as users with lower similarity are selected. On the other hand, a small value of k would be biased over the choice of the most similar user. For the initial experiment, we only tested with $k = 10$.

As a baseline model, we also set $k = m$, i.e. to evaluate the RMSE by taking the plain arithmetic mean of all users.

2.2.2 Choice of metric

Since the ratings are discrete in nature, little difference is expected between the different metrics, with the exception of the cosine metric, which is sensitive to the exact value.

2.2.3 Choice of Aggregation function

For every metric, we compute both the unweighted and weighted aggregates and compare the results.

2.3. Predictive test set score (RMSE)

The model is evaluated by computing the RMSE between predicted and actual rating values:

$$\text{RMSE} = \sqrt{\sum_{(i,j) \in E} \frac{(\hat{r}_{ij} - r_{ij})^2}{|E|}}$$

where E is the set of retained evaluation data.

3. Model performance

The following table exhaustively lists our test results. The RMSE of the baseline is 1.1977149830409672.

Euclidean		
k	**Aggregation function**	**RMSE**
10	Unweighted mean	1.1743212065844748
10	Weighted	1.174185480169039

Taxicab		
k	**Aggregation function**	**RMSE**
10	Unweighted mean	1.1761011327320081c
10	Weighted	1.1758026291697672

Cosine with 3-corrected ratings		
k	**Aggregation function**	**RMSE**
10	Unweighted mean	1.1807121450228732
10	Weighted	1.1800525348636057

Cosine with mean-corrected ratings		
k	**Aggregation function**	**RMSE**
10	Unweighted mean	1.1064212349492457
10	Weighted	1.104635214204885

Pearson		
k	**Aggregation function**	**RMSE**
10	Unweighted mean	1.1977283650557216
10	Weighted	1.1977314956919407

Spearman		
k	**Aggregation function**	**RMSE**
10	Unweighted mean	1.1981137810682911
10	Weighted	1.1981166551209421

3.1. Discussion

The results are mostly very similar to or even worse than the baseline value.

3.1.1 Cosine metric

Among all metrics, the cosine metric with mean-corrected ratings appears to have the best result. This is probably because the cosine metric is effective in characterizing how different a user is from the ”normal” user; for example, if a movie is rated with an average of 4, the contrast between rating 3 and 5 is much more significant than that between 1 and 3, as is reflected by the fact that the former is different by 0.707 while the latter is 0.242.

3.1.2 Aggregation method

It is observed that weighted average performs similarly to or even worse than the unweighted mean.

For the weighted method, all metrics did not have better performance than that of baseline. One possible reason for this is related to the lack of data for similar users. When most of the similar users did not have a rating of the movie we are predicting, the formula of weighted average may not include enough information and therefore led to less accurate predictions.

4. Future work

We noticed several problems in our design and plan to resolve them in the future weeks.

4.1. Do not use imputed values for prediction

Currently, missing data are imputed using the corresponding mean value of ratings on the same movie. These imputed values do not actually represent opinions of similar users, and they should not be considered. It is proposed that we only aggregate real ratings from neighbours, and only fallback to mean rating if all K neighbours did not rate the movie.

4.2. Top Q optimization

Hong and Tsamis proposed considering only the neighbourhood that intersects with the users with the top $Q < m$ number of ratings [1]. They suggested that this improves performance as the distance matrix can be reduced from $O(m^2)$ to $O(Qm)$. We noticed that this also reduces the chance of the aforementioned problem where a substantial proportion of the top K neighbours did not the movie of which a rating is to be predicted.

References

[1] Ted Hong and Dimitris Tsamis. Use of knn for the netflix prize. *CS229 Projects*, 2006. 4