Computer Architecture

Project 1: Parallel SGD

1 Project Description

The project aims to design a matrix factorization-based collaborative filtering recommendation algorithm for movie recommendation systems. Based on movie ratings data, this algorithm can be used to implement a complete collaborative filtering recommendation system to help users discover movies they might like.

1.1 How to predict user ratings for movies?

In this project, you will design an algorithm based on the movie ratings dataset provided by http://grouplens.org/datasets/movielens/. In the dataset, ratings range from 1 to 5, where higher scores indicate that users like the movie more. Your task is to design an algorithm to predict ratings for movies that users have not yet watched.

2 Matrix Factorization-based Collaborative Filtering Recommendation Algorithm

The data obtained can be represented as a user-item rating matrix, as shown in the diagram below:

	D1	D2	D3	D4
U1	5	3	-	1
U2	4	-	-	1
U3	1	1	-	5
U4	1	-	-	4
U5	-	1	5	4

In the matrix, the ratings (1-5) of 4 items (D1, D2, D3, D4) by 5 users (U1, U2, U3, U4, U5) are described, where "-" indicates no rating. The objective is to predict the missing ratings and recommend items to users based on these predictions.

How to predict the missing ratings? For the missing ratings, they can be transformed into a regression problem based on machine learning, which aims to predict continuous values. For matrix factorization, the following equation can be used, where R is the rating matrix similar to the one shown above, assumed to be NM dimensions (N represents the number of rows, M represents the number of columns), and it can be factorized into matrices P and Q. The P matrix has dimensions NK, and the Q matrix also has dimensions M*K.

$$R \approx P \times Q^T = \hat{R}$$

Intuitively, the P matrix represents the relationships between N users and K topics, while the Q matrix represents the relationships between K topics and M items. The specific meaning of the K topics can be provided by the designer, as it is essentially a form of soft clustering. In algorithms, K is a adjustable parameter, typically ranging from 10 to 100.

For each element r_{ij} in R, we have

$$\hat{r}_{ij} = p_i q_j^T$$

The variable r_{ij} represents the element in the i-th row and j-th column of the predicted rating matrix. It corresponds to the i-th row vector p_i in matrix P and the j-th row vector q_j in matrix Q.

Therefore, the matrix factorization model maps users and items to their corresponding latent factor matrices P and Q. Each user corresponds to a vector, and each item corresponds to a vector. The relationship (preference) between the user and item is then modeled as the inner product of these two vectors. These vectors can contain certain meanings, for example, representing the user's interests as a 4-dimensional vector, representing:

- Preference for the script
- Preference for special effects
- Requirements for actors
- Preference for directors

Similarly, the characteristics of the item are represented as a 4-dimensional vector:

- Script
- Special effects
- Actors
- Directors

Finally, the preference degree of a user for an item is naturally the product of these two vectors. The following equation represents the error between predicted ratings and actual scores:

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - p_i q_j^T)^2,$$

Taking into account the issue of scale variations among users' ratings, wherein some users consistently assign higher scores than others, or where certain users possess a more lenient rating scale, similarly, some items are consistently rated highly. This is a commonly encountered problem, thus necessitating the inclusion of several components when constructing the objective function: the average value of all ratings *u*, the user bias score b1, and the item bias score b2.

$$\hat{r}_{ij} = \mu + b1_i + b2_j + p_i q_j^T,$$

Here, b1 and b2 represent the baselines obtained from rating deviations of user i and item j, respectively. The mean rating, denoted as *u*, can be directly calculated along with b1 and b2 through statistical analysis.

$$\mu = \frac{1}{|R|} \sum_{i,j \in R} r_{ij};$$

$$b1_i = \frac{1}{|R_i|} \sum_{j \in R_i} (r_{ij} - \mu);$$

$$b2_j = \frac{1}{|R_j|} \sum_{i \in R_j} (r_{ij} - \mu).$$

Here, R is the rating matrix, R_i is the vector in R for user i, and R_j is the vector in R for item j. In order to prevent the occurrence of ill-conditioned matrices P and Q, and to enhance their rationality and interpretability, we need to introduce regularization terms. Consequently, the overall loss function can be expressed as follows:

$$\min Loss(P,Q) = \sum_{(i,j)\in R} (r_{ij} - \mu - b1_i - b2_j - \mathrm{p}_i \mathrm{q}_j^T)^2 + \lambda(b1_i^2 + b2_j^2 + \|\mathrm{p}_i\|^2 + \|\mathrm{q}_j\|^2)$$

Here $\lambda(b1_i^2 + b2_j^2 + \|\mathrm{p}_i\|^2 + \|\mathrm{q}_j\|^2)$ is the added regular term, which serves as the regularization factor (typically manually determined). With the objective function defined, we can proceed to optimize it using Stochastic Gradient Descent (SGD).

Taking the gradient of the objective function yields the iterative updates for each element:

$$\begin{aligned}
e_{ij} &= r_{ij} - \hat{r}_{ij} \\
b1_i' &= b1_i + \gamma(e_{ij} - \lambda b1_i) \\
b2_j' &= b2_j + \gamma(e_{ij} - \lambda b2_j) \\
\mathrm{p}_i' &= \mathrm{p}_i + \gamma(e_{ij}\mathrm{q}_j - \lambda \mathrm{p}_i) \\
\mathrm{q}_j' &= \mathrm{q}_j + \gamma(e_{ij}\mathrm{p}_i - \lambda \mathrm{q}_j)
\end{aligned}$$

3 Evaluation

3.1 Movie Rating Prediction

In this section, the evaluation criterion is the root mean squared error (RMSE) between the recommended predicted rating values by the participants and the actual rating values. In the well-divided test set, if the participant's algorithm predicts the user's predicted rating for a movie as Y, and the user's actual rating for this movie is X, then the obtained RMSE = sqrt(sum(for all items(X-Y)^2)/n). This part accounts for 30% of the total score. Matlab algorithm reference: Intel Q8200 at 3.0GHz, 8G RAM, it takes 60s to complete the matrix recovery of 1000*1700 under the termination iteration condition of converging to an acceptable RMSE.

3.2 Acceleration Ratio of Parallel Algorithms

In this section, the evaluation criterion is the acceleration ratio obtained from optimizing parallel algorithms. The acceleration ratio is defined as R = T0 / TP, where TP is the time used by the parallel algorithm, and T0 is the time used by the unoptimized algorithm. This part accounts for 70% of the total score.

4 Code

Participants are required to use Python for implementation.

5 Submission

Before the specified deadline.