Homework 1 of CS7316

METHOD

`MovieLens 1M` dataset contains 1 million ratings from 6000 users on 4000 movies. To predict the ratings of test dataset, we use the `Latent Factor Models` method as our system recommendation algorithm after considering all methods learned in class.

`Latent Factor Models` is a learning based method by breaking down the user-item rating matrix $R^{m\times n}$ into two matrix $P^{m\times k}$ and $Q^{k\times n}$, shown as Figure 1. After finding P and Q, for each item $R[i,j]$ missed in R, we can compute it as

$$R[i,j] = P[i,] * Q[,j] = \sum_{t=1}^{k} P[i,t] * Q[t,j]$$

Then, the problem becomes a optimization problem. Our goal is

$$\min_{P,Q} \sum_{(i,j)\in R} (R[i,j] - P[i,] * Q[,j])^2$$

We can use the Stochastic Gradient Descent (SGD) normally used in Machine Learning to find a good P and Q. To avoid overfitting problem, we can add regularization to the objective function as

$$\min_{P,Q} \sum_{(i,j)\in R} (R[i,j] - P[i,] * Q[,j])^2 + [\lambda_1 \sum_i ||P[i,]||^2 + \lambda_2 \sum_j ||Q[,j]||^2]$$

where λ_1 and λ_2 are hyperparameters. What's more, we can add some biases to get a better performance. Then we can compute $R[i,j]$ as

$$R[i,j] = \mu + b_i^u + b_j^t + P[i,] * Q[,j]$$

where μ is the overall mean rating, b_i^u is used for user i and b_j^t is used for item j. Then the final objective function is

$$\min_{P,Q} \sum_{(i,j)\in R} (R[i,j] - (\mu + b_i^u + b_j^t + P[i,] * Q[,j]))^2 + [\lambda_1 \sum_i ||P[i,]||^2 + \lambda_2 \sum_j ||Q[,j]||^2 + \lambda_3 \sum_i ||b_i^u||^2 + \lambda_4 \sum_j ||b_j^t||^2]$$

With these optimizations, we can find the P and Q.

IMPLEMENTATION

We use the `pytorch` to implement the training code. We split the dataset into training dataset and validation dataset. Specially, we use `col_matrix[2000:4100, 2700:]` as the validation dataset and other remaining part as training dataset. And we only use non-zero values. To get good optimization, We use `torch adam` as the optimizer. And we initialize the P and Q randomly.

The training system we used has a Nvidia GPU RTX4090 device with 24GB memory and two CPUs with 48 cores and 256GB memory.

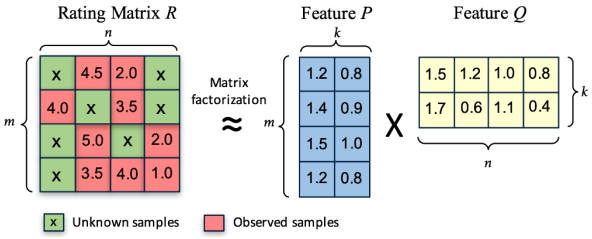

Figure 1: Latent Factor Model Example

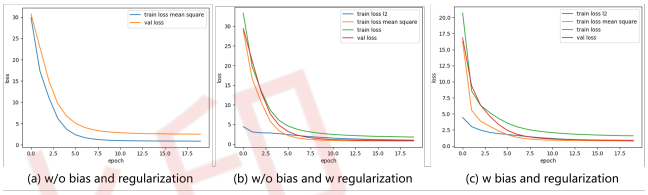

Figure 2: Training Loss: *train loss mean square* is mean square error of training dataset; *val loss* is mean square error of validation dataset; *train loss l2* is regularization loss; *train loss* is the sum of *train loss mean square* and *train loss l2*.

EVALUATION

Table 1: Training Parameters

K	30
learning rate	0.001
$\lambda_1, \lambda_2, \lambda_3, \lambda_4$	0.1, 0.1, 0.1, 0.1
batch size	16
number epoch	20

We use the parameters in Table 1 to train model and the mean square error as the evaluation metric. We train the model for 2 to 3 hours and finally we get training loss of the training process as Figure 2. From the comparison between Figure 2-(a) and Figure 2-(b), it can be observed that regardless of whether regularization is

Conference'17, July 2017, Washington, DC, USA
2024. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

used, the mean square loss during training shows no significant change. However, the mean square loss for validation becomes smaller, indicating that regularization contributes to enhancing the model's generalizability. From Figure 2-b and Figure 2-c, it is evident that using bias helps to further reduce both training and validation loss. Finally, the mean square errors on the validation dataset for the three scenarios are 1,5904 (without regularization and bias), 0.9514 (with regularization and without bias), 0.9424 (with regularization and bias), respectively.