Advanced Topics in Recommendation Systems – EX1

Authors:

Maya Vilenko - 315339747

Lior Tondovski – 307870766

Ilan Vasilevsky - 322545682

Part 0 – Baseline model:

For Every user and item, we give the average of all the training data reviews as the prediction.

The main components are:

1. Calculation of evaluation measures function - returns a dictionary of all the desired evaluation measures based on a dataset.
2. Fit function that calculates the average of all reviews.

Evaluation Measures on validation set:

MAE	RMSE	R^2	MPR
0.956	1.127	0	0.0008

Part 1 – Stochastic Gradient Descent:

1. Objective function's components notations and definitions:

$D - The\ matrix\ of\ all\ users - items\ ratings$

$M - The\ set\ of\ users$

$N - The\ set\ of\ items$

$U_m - The\ user\ learnable\ vector$

$V_n - The\ item\ learnable\ vector$

$b_m - The\ user\ bias$

$b_n - The\ item\ bias$

$\lambda_m - The\ regularization\ parameter\ of\ U_m$

$\lambda_n - The\ regularization\ parameter\ of\ V_m$

$\lambda_{bm} - The\ regularization\ parameter\ of\ b_m$

$\lambda_{bn} - The\ regularization\ parameter\ of\ b_n$

$R_{mn} - The\ rating\ that\ user\ m\ gave\ to\ item\ n$

$\hat{R}_{mn} - The\ prediction\ of\ the\ rating\ that\ user\ m\ gave\ to\ item\ n$

$\mu - The\ average\ rating\ in\ D$

$\hat{R}_{mn} = \mu + b_m + b_n + {U_m}^T V_n$

$\mathrm{Lr} - The\ Learning\ rate$

The objective function:

$$\frac{1}{||D||} * \sum_{(m,n,R_{mn})\epsilon D} (R_{mn} - \hat{R}_{mn})^2 + \lambda_m \sum_{m=1}^{M} ||U_m||^2 + \lambda_n \sum_{n=1}^{N} ||U_n||^2 + \lambda_{bn} \sum_{n=1}^{N} {b_n}^2 + \lambda_{bm} \sum_{m=1}^{M} {b_m}^2$$

2. The update step for each parameter:

1. $U_m = U_m + \mathrm{Lr} * [(R_{mn} - \hat{R}_{mn})\mathrm{V_n} - \lambda_m U_m]$
2. $V_n = V_n + \mathrm{Lr} * [(R_{mn} - \hat{R}_{mn})U_m - \lambda_m V_n]$
3. $b_m = b_m + \mathrm{Lr} * [(R_{mn} - \hat{R}_{mn}) - \lambda_{bn} * b_m]$
4. $b_n = b_n + \mathrm{Lr} * [(R_{mn} - \hat{R}_{mn}) - \lambda_{bn} * b_n]$

3. For epoch in epochs:

```
For each $(m, n, R_{mn})_\epsilon D_{train}$:
    Calculate prediction ($\hat{R}_{mn} = \mu + b_m + b_n + {U_m}^T V_n$)
    Update the learnable parameters:
    1. $U_m = U_m + \mathrm{Lr} * [(R_{mn} - \hat{R}_{mn})\mathrm{V_n} - \lambda_m U_m]$
    2. $V_n = V_n + \mathrm{Lr} * [(R_{mn} - \hat{R}_{mn})U_m - \lambda_m V_n]$
    3. $b_m = b_m + \mathrm{Lr} * [(R_{mn} - \hat{R}_{mn}) - \lambda_{bn} * b_m]$
    4. $b_n = b_n + \mathrm{Lr} * [(R_{mn} - \hat{R}_{mn}) - \lambda_{bn} * b_n]$
for each $(m, n, R_{mn})_\epsilon D_{validation}$:
    calculate the RMSE or other relevant metric over all samples in the validation dataset
```

Note: It is possible to add an early stopping criterion as well. If you chose to do so: check whether the stopping criterion is met- if it is, then the training loop should be stopped, and otherwise, the training loop should continue to the next epoch. In our implementation the early stopping criterion is checking whether the difference in the RMSE in the last two epochs is lower than 0.001.

4. The hyperparameters that can be tuned are:

- Learning Rate
- User and Item learnable vector size
- The early stopping criterion
- The number of epochs
- The value of the regularization's parameters

5. In our validation set, the RMSE was calculated after each epoch (another relevant measure may be selected in this case). RMSE decreasing throughout the epochs indicates that the model is learning and going towards a local minima. In addition, a sanity check can also be performed by calculating the RMSE of a naive model that always predicts the average rating in the dataset and checking whether our SGD model's performance better.

6. We determined an early stopping criterion based on the RMSE which is calculated over all the samples in the validation set. If difference between the last two epochs' RMSE is less than 0.001 then we break the training loop and return the trained model which can be used on the test set.

7. The chose to implement a class of SGD matrix factorization. The main components are:

> 1. The init function - A function that determines the number epochs, the learning rate, and the dimensions of the learnable vectors, etc.
>
> 2. Run epoch function - iterates over all samples in the training data and updates the learnable vectors and parameters after each iteration.
>
> 3. Calculation of evaluation measures function – inherits from simple model.
>
> 4. Fit function that first initiates the learnable vectors and parameters, iterates over all epochs, calls the run epoch function, and calculates evaluation measures right after, and then checks whether the early stopping criteria is met.

8. What is the RMSE, MAE, R^2 and MPR of your model based on the validation set?

After 5 epochs:

MAE	RMSE	R^2	MPR
0.736	0.944	0.299	0.0008

<u>Part 2 – Alternating Least Squares:</u>

1. The objective function is the same as the SGD's:

$$\frac{1}{||D||} * \sum_{(m,n,R_{mn})\epsilon D} (R_{mn} - \hat{R}_{mn})^2 + \lambda_m \sum_{m=1}^{M} ||U_m||^2 + \lambda_n \sum_{n=1}^{N} ||U_n||^2 + \lambda_{bn} \sum_{n=1}^{N} {b_n}^2 + \lambda_{bm} \sum_{m=1}^{M} {b_m}^2$$

2. The update step for each parameter:

> Let denote some additional marks:
>
> $D_m - All\ the\ items\ that\ user\ m\ ranked$
>
> $D_n - All\ the\ users\ that\ ranked\ item\ n$
>
> Updating steps:
>
> 1. $U_m = (\sum_{D_m} V_n * V_n^T + \lambda_m I)^{-1} * \sum_{D_m}(R_{mn} - \mu - b_m - b_n) V_n$
>
> 2. $V_n = (\sum_{D_n} U_m * U_m^T + \lambda_n I)^{-1} * \sum_{D_n}(R_{mn} - \mu - b_m - b_n) U_m$
>
> 3. $b_m = (|D_m| + \lambda_{bm})^{-1} * \sum_{D_m}(R_{mn} - \mu - b_n - U_m^T V_n)$

4. $b_n = (|D_n| + \lambda_{bn})^{-1} * \sum_{D_m}(R_{mn} - \mu - b_n - U_m^T V_n)$

```
3. for epoch in epochs:
        For user in M:
                For items that user ranked:
                        Update the user vector and the user item bias parameters
        For item in N:
                For users that ranked item:
                        Update the item vector and the items bias parameters
        for each $(m, n, R_{mn})_\epsilon D_{validation}$:
                calculate the RMSE or other relevant metric over all samples in the validation dataset
```

Note: It is possible to add an early stopping criterion as well. If you chose to do so: check whether the stopping criterion is met- if it is, then the training loop should be stopped, and otherwise, the training loop should continue to the next epoch. In our implementation the early stopping criterion is checking whether the difference in the RMSE in the last two epochs is lower than 0.001.

4. The hyper-parameters are:

- User and Item learnable vector size
- The early stopping criterion
- The number of epochs
- The value of the regularization's parameters

5. In our validation set, the RMSE was calculated after each epoch (another relevant measure may be selected in this case). RMSE decreasing throughout the epochs indicates that the model is learning and going towards a local minima. In addition, a sanity check can also be performed by calculating the RMSE of a naive model that always predicts the average rating in the dataset and checking whether our ALS model's performance better.

6. ALS matrix factorization is implemented as a class that inherits from SGD matrix factorization. The only thing we needed to code from scratch was the run epoch function, which updates the parameters according to the ALS algorithm.

7. What is the RMSE, MAE, R^2 and MPR of your model based on the validation set?

After 2 epochs:

MAE	RMSE	R^2	MPR
0.86	1.093	0.06	0.0008

8. Compare the ALS and SGD solutions in terms of implementation, training, and quality.

While the ALS solution converged with fewer epochs, we have noticed a significant improvement in the SGD solution within all evaluation measures. Furthermore, we can see that the ALS solution is not far from the simple solution (using the mean rating). We assume that the latent space vector initialization in this specific dataset might lead to different local minima for both solutions.

During the project, we felt that the implementation of the ALS model was more complicated than the SGD, as seen in the significantly larger code needed for it.