Matrix Factorization : ALS vs. SGD

Algorithms widely used in recommender systems

Hannah Do | March 26th, 2022

Algorithm

Very similar to last week's ALS, MF-SGD aims to factorize a matrix into two low-rank matrices named matrix P and matrix Q.

R = P x Q

- **R** : Rating Matrix (m x u)
- **Matrix P** : User Latent Matrix (m x k)
- **Matrix Q** : Item Latent Matrix (u x k)

Q : Product Matrix (product row = j)

	M1	M2	M3	M4	M5
F1	1.2	3.1	0.3	2.5	0.2
F2	2.4	1.5	4.4	0.4	1.1

P : User Matrix (user row = k)

	F1	F2
A	0.2	0.5
B	0.3	0.4
C	0.7	0.8
D	0.4	0.5

	M1	M2	M3	M4	M5
	1.44	1.37	2.26	0.7	0.59
	1.32	1.53	1.85	0.91	0.5
	2.76	3.37	3.73	2.07	1.02
	1.68	1.99	2.32	1.2	0.63

↔

R : Ratings Matrix

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

Rating Prediction

$$\boxed{\hat{r}_{ui}} = \mathbf{x}_u^\top \cdot \mathbf{y}_i = \sum_k x_{uk} y_{ki}$$

Assuming we have a user-item matrix, **r-hat** represents the prediction for the true rating - the dot product of the two latent vectors.

$$L = \sum_{u,i \in S} (\boxed{r_{ui}} - \boxed{\mathbf{x}_u^\top \cdot \mathbf{y}_i})^2 + \lambda_x \sum_u \|\mathbf{x}_u\|^2 + \lambda_y \sum_u \|\mathbf{y}_i\|^2$$

To reduce the difference between the r-hat and the actual rating, *L2 regularization* terms are added along with the squared difference.

This will be our loss function, which will be taken *derivatives* as a tool for minimizing functions.

Loss Minimization : ALS

User vector and Item vector takes turn (alternatives) to be taken derivatives - and the loss is minimized accordingly.

$$L = \sum_{u,i \in S} (\boxed{r_{ui}} - \boxed{\mathbf{x}_u^\top} \cdot \boxed{\mathbf{y}_i})^2 + \lambda_x \sum_u \|\mathbf{x}_u\|^2 + \lambda_y \sum_u \|\mathbf{y}_i\|^2$$

$$\frac{\partial L}{\partial \mathbf{x}_u} = -2 \sum_i (r_{ui} - \boxed{\mathbf{x}_u^\top} \cdot \mathbf{y}_i)\mathbf{y}_i^\top + 2\lambda_x \boxed{\mathbf{x}_u^\top}$$

$$0 = -(\mathbf{r}_u - \mathbf{x}_u^\top Y^\top)Y + \lambda_x \mathbf{x}_u^\top$$

$$\mathbf{x}_u^\top (Y^\top Y + \lambda_x I) = \mathbf{r}_u Y$$

$$\mathbf{x}_u^\top = \mathbf{r}_u Y (Y^\top Y + \lambda_x I)^{-1}$$

Derivative in respect to **x**

$$\frac{\partial L}{\partial \mathbf{y}_i} = -2 \sum_i (r_{iu} - \boxed{\mathbf{y}_i^\top} \cdot \mathbf{x}_u)\mathbf{x}_u^\top + 2\lambda_y \boxed{\mathbf{y}_i^\top}$$

$$0 = -(\mathbf{r}_i - \mathbf{y}_i^\top X^\top)X + \lambda_y \mathbf{y}_i^\top$$

$$\mathbf{y}_i^\top (X^\top X + \lambda_y I) = \mathbf{r}_i X$$

$$\mathbf{y}_i^\top = \mathbf{r}_i X (X^\top X + \lambda_y I)^{-1}$$

Derivative in respect to **y**

Loss Minimization : SGD

SGD : Derivative of each variable is taken, solving for the feature weights and updates for each features until convergence.

Assuming each user and item has a bias term, the resulting **r-hat** and **loss function** will be as the following :

$$\boxed{\hat{r}_{ui}} = \boxed{\mu + b_u + b_i} + \mathbf{x}_u^{\top} \cdot \mathbf{y}_i$$

$$L = \sum_{u,i} \left(\boxed{r_{ui}} - \left(\boxed{\mu + b_u + b_i} + \mathbf{x}_u^{\top} \cdot \mathbf{y}_i\right)\right)^2 + \lambda_{xb} \sum_u \boxed{\|b_u\|}^2$$

$$+ \lambda_{yb} \sum_i \boxed{\|b_i\|}^2 + \lambda_{xf} \sum_u \|\mathbf{x}_u\|^2 + \lambda_{yf} \sum_u \|\mathbf{y}_i\|^2$$

Loss Minimization : SGD

$$L = \sum_{u,i} (\boxed{r_{ui}} - (\mu + \boxed{b_u + b_i + \mathbf{x}_u^\top \cdot \mathbf{y}_i}))^2 + \lambda_{xb} \sum_u \|b_u\|^2$$

$$+ \lambda_{yb} \sum_i \|b_i\|^2 + \lambda_{xf} \sum_u \|\mathbf{x}_u\|^2 + \lambda_{yf} \sum_u \|\mathbf{y}_i\|^2$$

To reduce the loss function, the update for the user bias is as the following, where **η is the learning rate** and *e* as the error in the prediction.

$$\boxed{b_u} \leftarrow b_u - \eta \frac{\partial L}{\partial b_u}$$

$$\frac{\partial L}{\boxed{\partial b_u}} = 2(r_{ui} - (\mu + b_u + b_i + \mathbf{x}_u^\top \cdot \mathbf{y}_i))(-1) + 2\lambda_{xb} b_u$$

$$\frac{\partial L}{\partial b_u} = 2(e_{ui})(-1) + 2\lambda_{xb} b_u$$

$$\frac{\partial L}{\partial b_u} = -e_{ui} + \lambda_{xb} b_u$$

And for each features, the updates or derivates end up as the following :

$$\boxed{b_u} \leftarrow b_u + \eta\,(e_{ui} - \lambda_{xb} b_u)$$

$$b_i \leftarrow b_i + \eta\,(e_{ui} - \lambda_{yb} b_i)$$

$$\mathbf{x}_u \leftarrow \mathbf{x}_u + \eta\,(e_{ui}\mathbf{y}_i - \lambda_{xf} \mathbf{x}_u)$$

$$\mathbf{y}_i \leftarrow \mathbf{y}_i + \eta\,(e_{ui}\mathbf{x}_u - \lambda_{yf} \mathbf{y}_i)$$

ALS vs. SGD

Which is better?

- SGD is useful when there is redundancy in data since it uses sampling to evaluate the loss minimization.
- When using a new data, SGD takes one more step using the new sample, without starting the optimization all over again.
- SGD is generally faster and more accurate than ALS except in cases of sparse data in which ALS performs better.

References

- Explicit Matrix Factorization, ALS SGD and All that Jazz, Insight https://blog.insightdatascience.com/explicit-matrix-factorization-als-sgd-and-all-that-jazz-b00e4d9b21ea
- How does Netflix recommend movies? Matrix Factorization, Serrano.Academy, Youtube (Sep 8, 2018) https://www.youtube.com/watch?v=ZspR5PZemcs&ab_channel=Serrano.Academy
- Stochastic Gradient Descent, Clearly Explained!!! StatQuest (Youtube) https://www.youtube.com/watch?v=vMh0zPT0tLI&ab_channel=StatQuestwithJoshStarmer
- Recommender : An Analysis of Collaborative Filtering Techniques, Christopher R. Aberger http://cs229.stanford.edu/proj2014/Christopher%20Aberger,%20Recommender.pdf