RECOMMENDATION ENGINE

Q1:
How is ALS differ from the SVD/SGD/Common CFMF

ALS
(Alternating Least Square)

Loss Function:

$$\hat{r}_{ui} = q_i^T p_u.$$

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

ALS

(1) Start from randomly generated p & q
(2) Then we fix first q, and this optimization problem can be solved with least square, we can update p

(3) Same as step 2, and we can update q

(4) Repeat step 2 and step 3 until the loss function converge.

$$L = \sum_{u,i \in \mathbb{K}} (r_{ui} - p_u^T q_i)^2 + \lambda(p_u^T p_u + q_i^T q_i)$$

$$-\frac{\alpha L}{2\alpha p_{uk}} = \sum_i q_{ik}(r_{ui} - p_u^T q_i) - \lambda p_{uk} = 0$$

$$\Rightarrow \sum_i q_i(r_{ui} - p_u^T q_i) - \lambda p_u = 0$$

$$\Rightarrow (\sum_i q_i q_i^T + \lambda E) p_u = \sum_i q_i r_{ui}$$

$$\Rightarrow p_u = (\sum_i q_i q_i^T + \lambda E)^{-1} \sum_i q_i r_{ui}$$

$$\Rightarrow p_u = (Q_{u,i \in \mathbb{K}} Q_{u,i \in \mathbb{K}}^T + \lambda E)^{-1} Q_{u,i \in \mathbb{K}} R_{u,i \in \mathbb{K}}^T$$

ALS

1. Adding Biases

$$b_{ui} = \mu + b_i + b_u$$

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

$$MIN_{PQB} \sum_{u,i \in \mathbb{K}} (r_{ui} - u - b_u - b_i - p_u^T q_i)^2 + \lambda(p_u^T p_u + q_i^T q_i + b_u^2 + b_i^2)$$

ALS

2. Temporal Dynamics

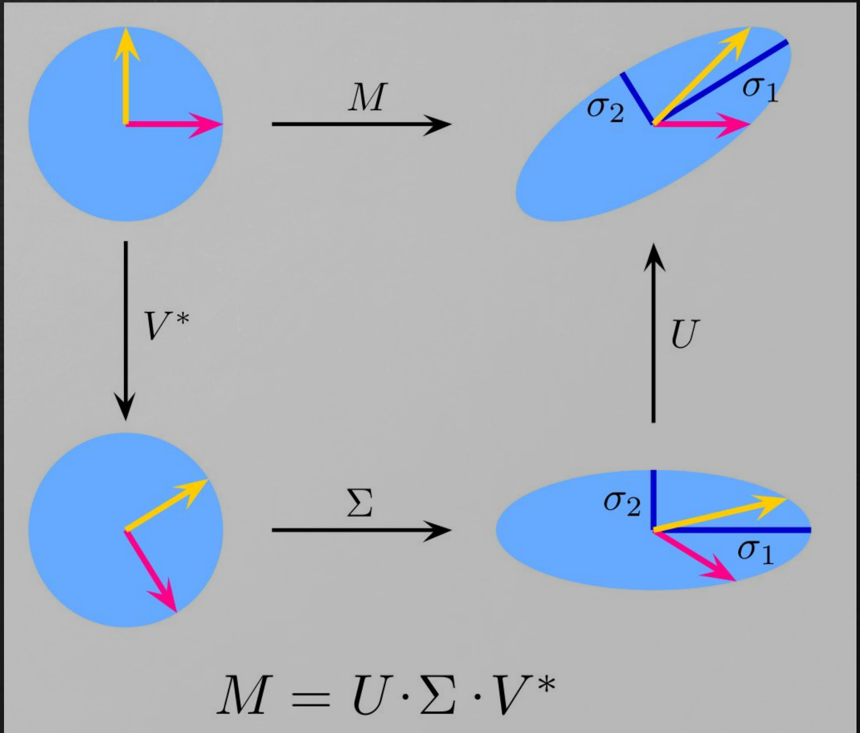
$$\hat{r}_{ui}(t) = \mu + b_i(t) + b_u(t) + q_i^T p_u(t)$$

$$MIN_{PQB} \sum_{u,i\in\mathbb{K}} (r_{ui}(t) - u - b_u(t) - b_i(t) - p_u(t)^T q_i^{\,2}) + \lambda(p_u(t)^T p_u(t) + q_i^T q_i + b_u(t)^2 + b_i(t)^2)$$

SVD

*SINGULAR VALUE DECOMPOSITION

*A FACTORIZATION OF A REAL OR COMPLEX MATRIX THAT GENERALIZES THE EIGENDECOMPOSITION OF A SQUARE NORMAL MATRIX TO ANY MATRIX VIA AN EXTENSION OF THE POLAR DECOMPOSITION.



$$M = U \cdot \Sigma \cdot V^*$$

SVD in Recommender Engine

- SVD
- Funk SVD
- Bias SVD/regularized SVD/PMF
- SVD++
- NMF/NSVD/WRMF

....

$$A_{m*n} \approx U_{m*r} \Sigma_{r*r} V_{r*n}^T$$

rating matrix: $R \in \mathbb{R}^{N \times M}$

$$\hat{R} = PQ$$
$$\hat{r}_{ui} = p_u^T q_i$$

*In these models, no SVD(singular value decomposition) is applied.

*Common methods in these models are SGD and ALS.

SVD VS Funk SVD(ALS)

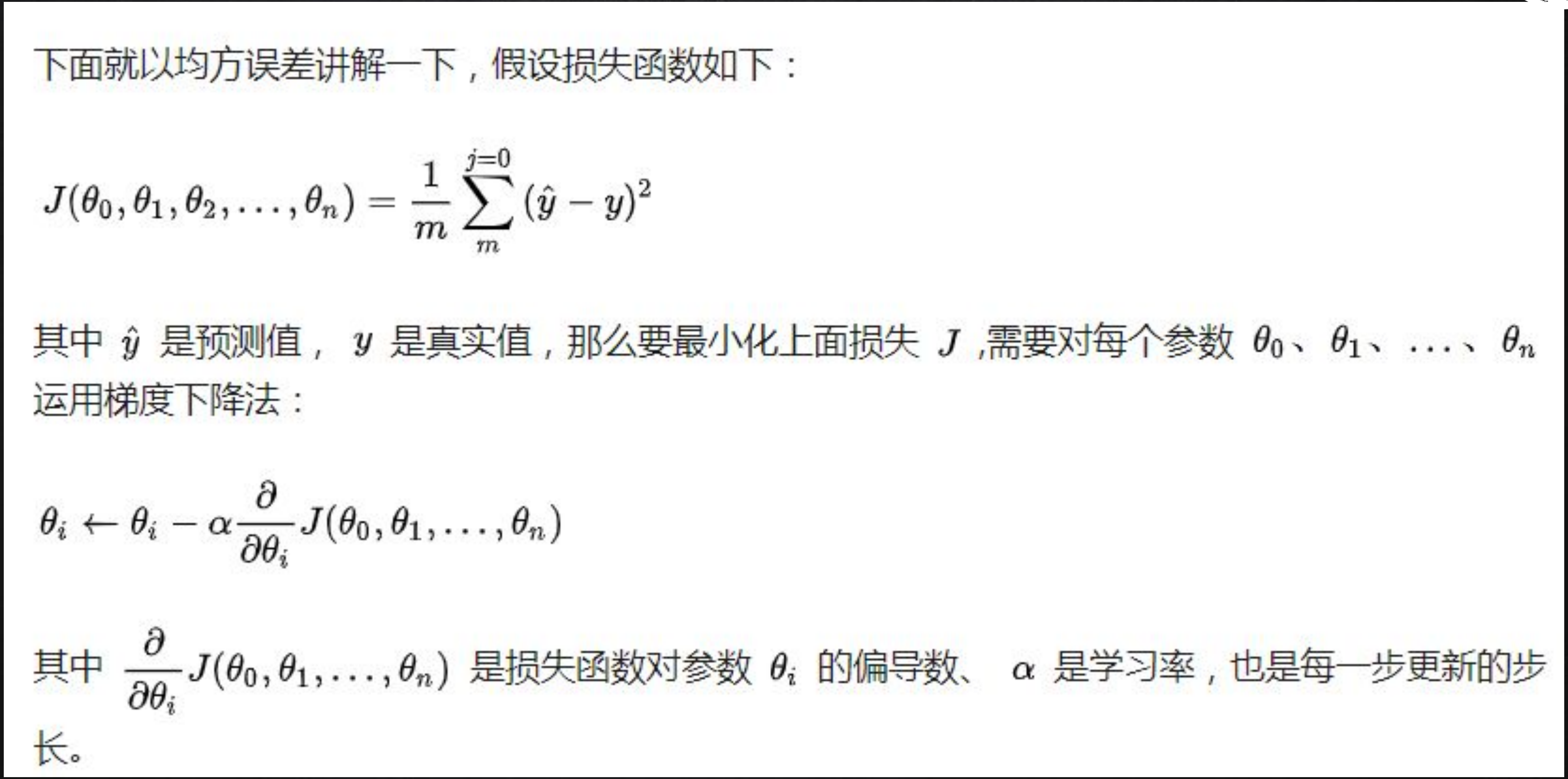
1) Sparsity: SVD need a complete matrix(no-sparse) matrix. ALS could solve the sparse matrix.

2) Result:The result of SVD must be two unitary matrices and one diagonal matrix, the results of ALS are just 2 latent factor matrices without any restrictions.

3) Complexity:SVD need a much higher storage space, very high computational complexity, and very long model training time.

4) Application

GD (Gradient Descent)



下面就以均方误差讲解一下，假设损失函数如下：

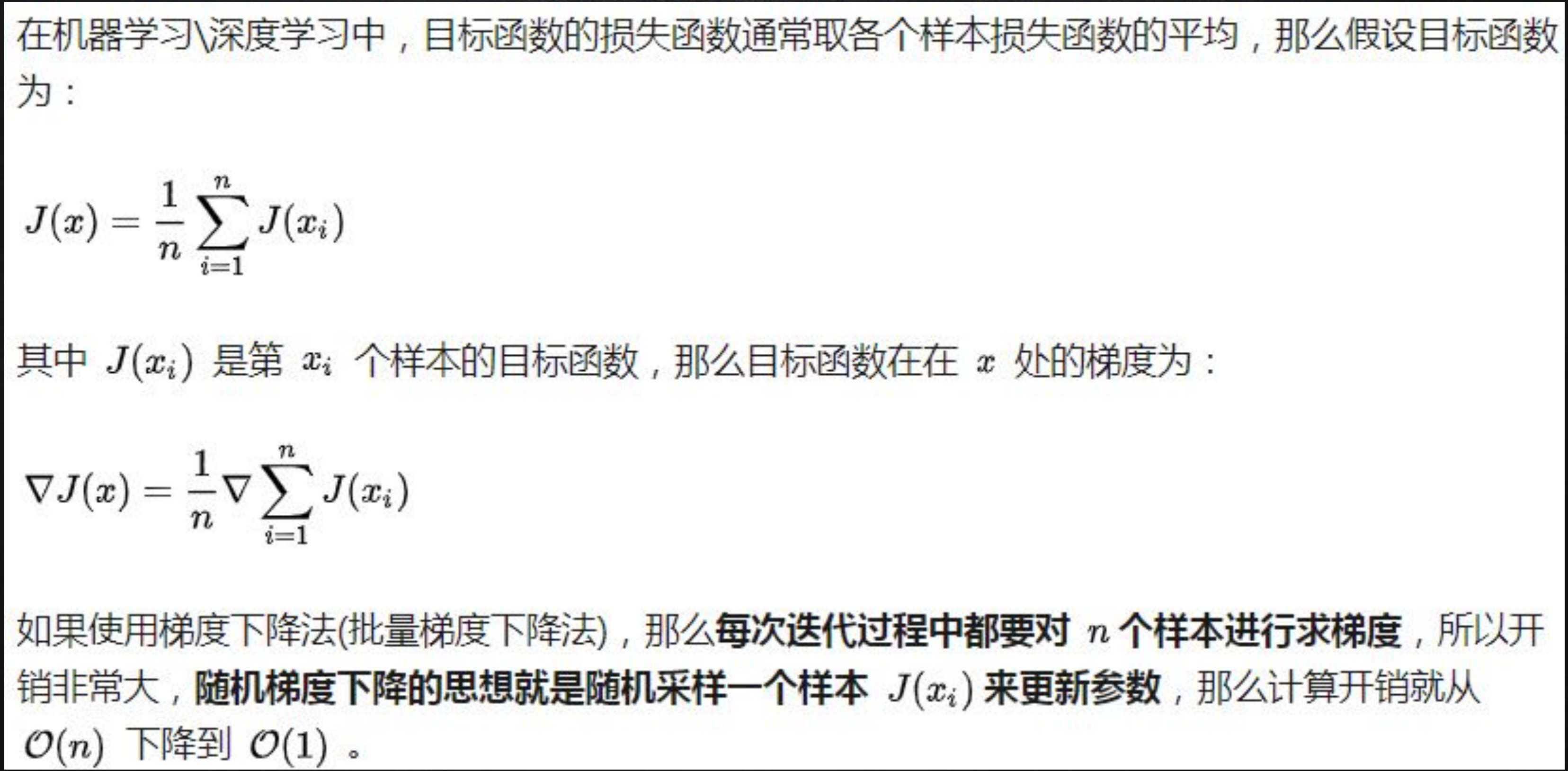
$$J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n) = \frac{1}{m} \sum_{m}^{j=0} (\hat{y} - y)^2$$

其中 $\hat{y}$ 是预测值， y 是真实值，那么要最小化上面损失 J ,需要对每个参数 θ_0、θ_1、$\ldots$、θ_n 运用梯度下降法：

$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, \ldots, \theta_n)$$

其中 $\frac{\partial}{\partial \theta_i} J(\theta_0, \theta_1, \ldots, \theta_n)$ 是损失函数对参数 θ_i 的偏导数、 α 是学习率，也是每一步更新的步长。

SGD (Stochastic Gradient Descent)



在机器学习\深度学习中，目标函数的损失函数通常取各个样本损失函数的平均，那么假设目标函数为：

$$J(x) = \frac{1}{n}\sum_{i=1}^{n} J(x_i)$$

其中 $J(x_i)$ 是第 x_i 个样本的目标函数，那么目标函数在在 x 处的梯度为：

$$\nabla J(x) = \frac{1}{n}\nabla\sum_{i=1}^{n} J(x_i)$$

如果使用梯度下降法(批量梯度下降法)，那么**每次迭代过程中都要对 n 个样本进行求梯度**，所以开销非常大，**随机梯度下降的思想就是随机采样一个样本 $J(x_i)$ 来更新参数**，那么计算开销就从 $\mathcal{O}(n)$ 下降到 $\mathcal{O}(1)$ 。

SGD vs ALS

1. Convergence of Formula: For SGD, the convergence is highly sensitive to the learning rate .

2. Dataset property: SGD has the upper hand regarding dealing with missing data. Whenever dealing with implicit datasets, which are usually not sparse, SGD is not practical.

3. Speed: SGD appears to be faster in terms of the time complexity for one iteration, but typically it needs more iterations than ALS to achieve a good enough model.

4. Parallel: Conducting several SGD updates in parallel directly might raise an overwriting issue as the updates for the ratings in the same row or the same column of matrix involve the same variables.

Q.5
Explore: Data Type

Data types

	User X Item	(user–item pair relationship)
✘	Explicit Feedback	Yes
✘	Implicit Feedback	No

Algorithm to use: RankALS , ImplicitALS and more ...

RankALS Objective Functions

Prediction based methods

$$f_P(\Theta) = \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} c_{ui} (\hat{r}_{ui} - r_{ui})^2,$$

$$\frac{\partial f_I(P,Q)}{\partial p_u} = \sum_{i \in \mathcal{I}} c_{ui} \left(q_i^T p_u - r_{ui} \right) q_i =$$

$$\underbrace{\left(\sum_{i \in \mathcal{I}} c_{ui} q_i q_i^T \right)}_{\bar{A}_u} p_u - \underbrace{\sum_{i \in \mathcal{I}} c_{ui} r_{ui} q_i^T}_{\bar{b}_u} = \bar{A}_u p_u - \bar{b}_u.$$

Ranking based methods

$$f_R(\Theta) = \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} c_{ui} \sum_{j \in \mathcal{I}} s_j \left[(\hat{r}_{ui} - \hat{r}_{uj}) - (r_{ui} - r_{uj}) \right]^2,$$

$$\frac{\partial f_R(P,Q)}{\partial p_u} =$$

$$\sum_{i \in \mathcal{I}} c_{ui} \sum_{j \in \mathcal{I}} s_j \left[(q_i - q_j)^T p_u - (r_{ui} - r_{uj}) \right] (q_i - q_j)$$

Runtime

RankALS : $O(TF^2 + (U + I)F^3)$

ImplicitALS: almost the same

Experiment

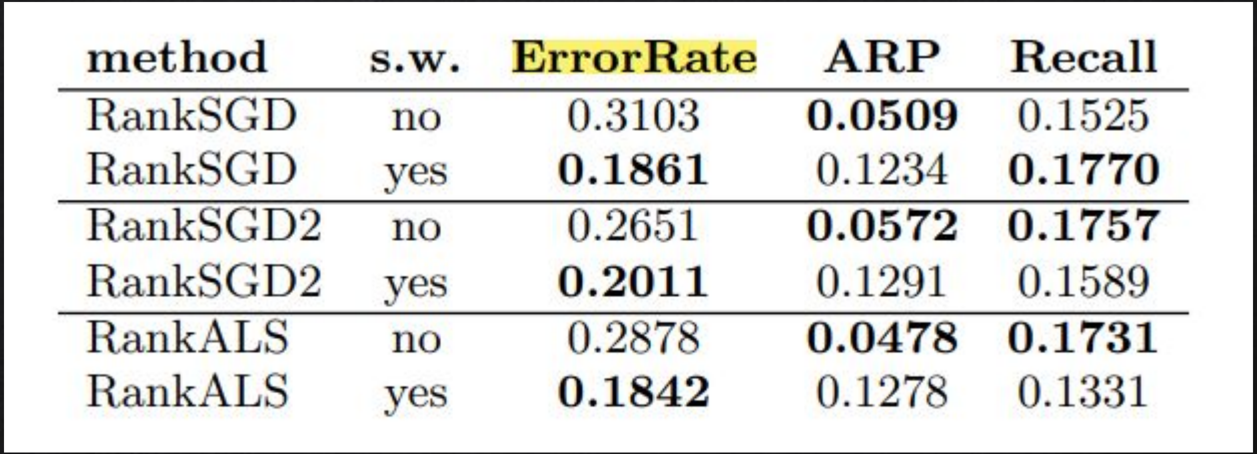
The Netflix training set consists of 100,480,507 ratings ranging from 1 to 5 from 480,189 users on 17,770 items

Methods in comparison:

1. Pop: Item popularity based, non-personalized baseline method.
2. ImplicitALS: uses an MF model, a prediction based objective function and an ALS optimizer.
3. RankSGD: uses an MF model, a ranking based objective function and an SGD optimizer
4. RankSGD2: uses an asymmetric factor (NSVD1) model, a ranking based objective function and an SGD optimizer.
5. RankALS: uses an MF model, a ranking based objective function, and an ALS optimizer.

Experiment

performance indicators: ErrorRate, ARP, Recall



method	s.w.	ErrorRate	ARP	Recall
RankSGD	no	0.3103	**0.0509**	0.1525
RankSGD	yes	**0.1861**	0.1234	**0.1770**
RankSGD2	no	0.2651	**0.0572**	**0.1757**
RankSGD2	yes	**0.2011**	0.1291	0.1589
RankALS	no	0.2878	**0.0478**	**0.1731**
RankALS	yes	**0.1842**	0.1278	0.1331

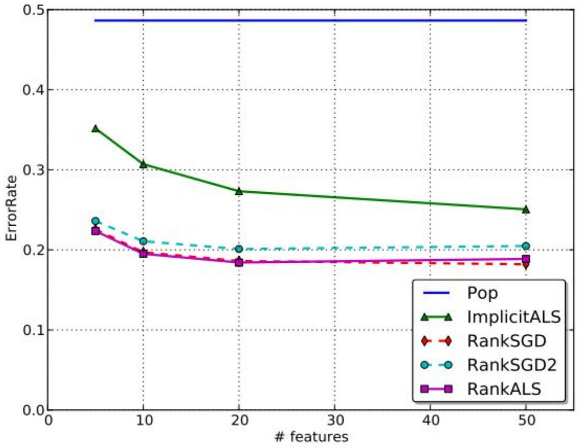
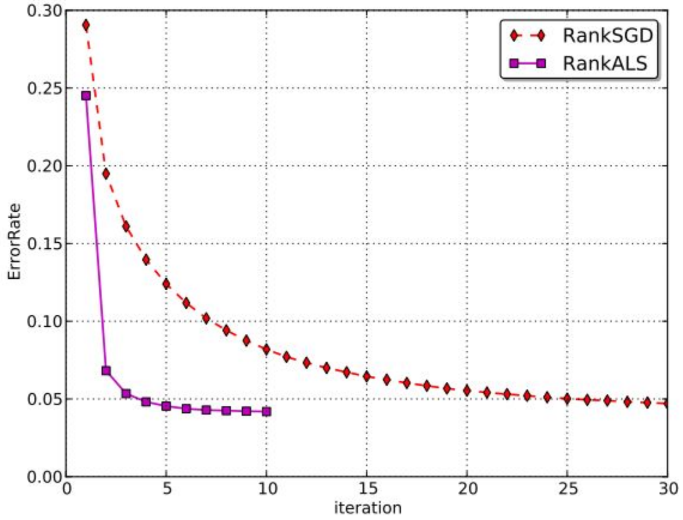
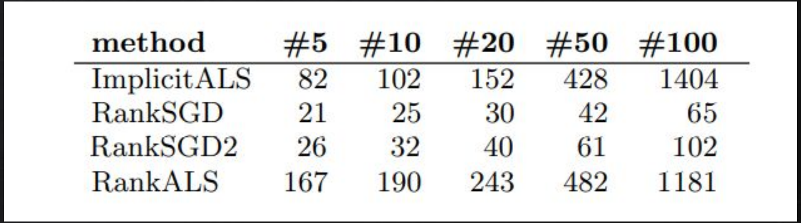


Figure 4: ErrorRate on the Netflix data set





method	#5	#10	#20	#50	#100
ImplicitALS	82	102	152	428	1404
RankSGD	21	25	30	42	65
RankSGD2	26	32	40	61	102
RankALS	167	190	243	482	1181