5th-week-1

Matrix Factorization

Wang Jianfang(王建芳 in Chinese)

Agenda

- MF
- Fitting

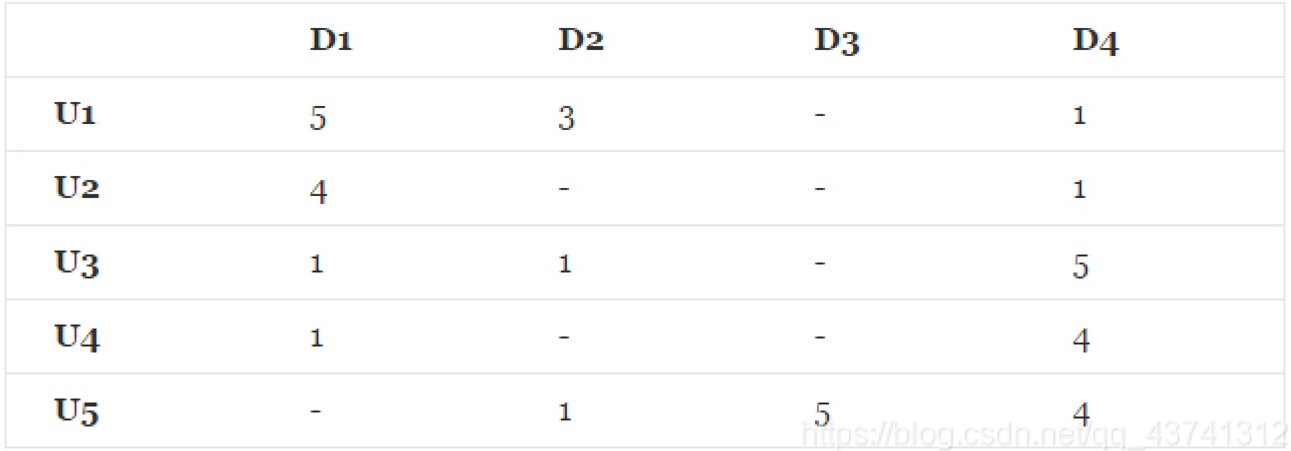
Matrix Factorization

- At present, the most used in the recommendation system is the matrix factorization method, which has achieved outstanding results in the netflix prize recommendation system contest. Taking the user-item rating matrix as an example, matrix factorization is to predict the missing value in the scoring matrix, and then recommend it to the user in some way according to the predicted value. Today ," user-item rating matrix R (M ×N)" is used to explain the principle and python implementation of matrix factorization.

Matrix Factorization

- R(5, 4) is rating matrix :("-" means the user does not rate) .

where the rating matrix R (n, m) are *n* row and *m* columns, *n* represents user number, *m* is item value.



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

- So, how do you predict the rating of an unrated item (how do you get the rating of a user with a score of 0) based on the current matrix R (5,4)?
- This problem can be solved by the idea of matrix factorization, which can actually be regarded as a supervised machine learning problem (regression problem).

- In the process of matrix factorization, the matrix R can be approximately expressed as the product of matrix P and matrix Q:

$$R_{m\times n} \approx P_{m\times k} \times Q_{k\times n} = \hat{R}_{m\times n}$$

- The matrix P(n,k) represents the relationship matrix between N users and K features, which is an intermediate variable, the transpose of the matrix Q(k,m) is the matrix Q(m,k), and the matrix Q(m,k) represents the relationship matrix between M items and K features. The k value here is controlled by itself, and the best k value can be obtained by cross-validation. In order to get the approximate R(n,m), we have to solve for the matrices P and Q, how do we solve for them?

Methods

- S1:

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj}$$

- S2:For the left-hand side of 1, which is the *r_hat* value to the *i* row, the *j* column, how do we measure it.we can give the following formula, which is the loss function, the square loss, and the final goal, which is the sum of the minimum values of *e*(*i*,*j*) for each of the elements (non-missing values)

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2$$

Methods

- S3:Modified P and Q components were obtained by using gradient descent method:
  - To solve the negative gradient of the loss function:

$$\frac{\partial}{\partial p_{i,k}} e_{i,j}^2 = -2\left(r_{i,j} - \sum_{k=1}^{K} p_{i,k} q_{k,j}\right) q_{k,j} = -2 e_{i,j} q_{k,j}$$

$$\frac{\partial}{\partial q_{k,j}} e_{i,j}^2 = -2\left(r_{i,j} - \sum_{k=1}^{K} p_{i,k} q_{k,j}\right) p_{i,k} = -2 e_{i,j} p_{i,k}$$

  - Update variables according to the direction of the negative gradient:

$$p_{i,k}' = p_{i,k} - \alpha \frac{\partial}{\partial p_{i,k}} e_{i,j}^2 = p_{i,k} + 2\alpha e_{i,j} q_{k,j}$$

$$q_{k,j}' = q_{k,j} - \alpha \frac{\partial}{\partial q_{k,j}} e_{i,j}^2 = q_{k,j} + 2\alpha e_{i,j} p_{i,k}$$

Methods

- S4： Constantly iteration until finally the algorithm convergence (until the sum (e ^ 2) < = threshold, at the end of the gradient descent condition: f (x) less than their true value and the predicted values setting threshold value) .
- S5:in order to prevent a overfitting, increase the regularization item.

Add regular term loss function to solve

- In general, in order to have better generalization ability, regular terms are added to the loss function to constrain the parameters. The loss function of the regular L2 norm is as follows:

$$e_{ij}^2 = (r_{ij} - \textstyle\sum_{k=1}^{K} p_{ik} q_{kj})^2 + \underline{\tfrac{\beta}{2} \textstyle\sum_{k=1}^{K} (||P||^2 + ||Q||^2)}$$

- For those with unclear regularization, the formula can be reduced to:

$$E_{i,j}^2 = \left(r_{i,j} - \sum_{k=1}^{K} p_{i,k} q_{k,j}\right)^2 + \frac{\beta}{2} \sum_{k=1}^{K} \left(p_{i,k}^2 + q_{k,j}^2\right)$$

Add regular term loss function to solve

- Modified P and Q components were obtained by using gradient descent method:

- To solve the negative gradient of the loss function:

$$\frac{\partial}{\partial p_{i,k}} E_{i,j}^2 = -2\left(r_{i,j} - \sum_{k=1}^{K} p_{i,k} q_{k,j}\right) q_{k,j} + \beta p_{i,k} = -2e_{i,j} q_{k,j} + \beta p_{i,k}$$

$$\frac{\partial}{\partial q_{k,j}} E_{i,j}^2 = -2\left(r_{i,j} - \sum_{k=1}^{K} p_{i,k} q_{k,j}\right) p_{i,k} + \beta q_{k,j} = -2e_{i,j} p_{i,k} + \beta q_{k,j}$$

- Update variables according to the direction of the negative gradient:

$$p_{i,k}' = p_{i,k} - \alpha\left(\frac{\partial}{\partial p_{i,k}} e_{i,j}^2 + \beta p_{i,k}\right) = p_{i,k} + \alpha\left(2e_{i,j} q_{k,j} - \beta p_{i,k}\right)$$

$$q_{k,j}' = q_{k,j} - \alpha\left(\frac{\partial}{\partial q_{k,j}} e_{i,j}^2 + \beta q_{k,j}\right) = q_{k,j} + \alpha\left(2e_{i,j} p_{i,k} - \beta q_{k,j}\right)$$

Prediction

- By using the above process, we can get the matrix sum, so that user *i* can rate item *j*:

$$\sum_{k=1}^{K} p_{i,k} q_{k,j}$$

Python Code

Fitting

- Over Fitting
  - The model trained by the algorithm expresses too much noise relationship between data.
  - Just too much detail!
- Under fitting
  - the model trained by the algorithm can not completely describe the data relationship.
  - It's too general!

Exam

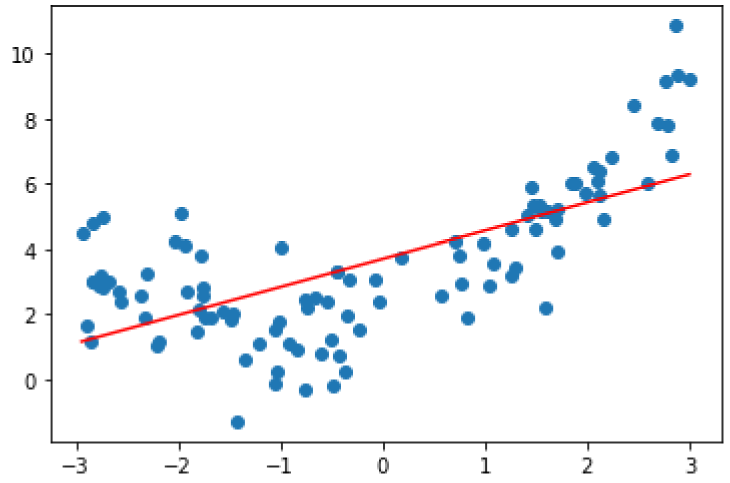
- making code by GD
- $y=x^4+x^2+2$
- to my email:972659357@qq.com

Homework

- change the distance formulas.

Linear regression

- import numpy as np
- import matplotlib.pyplot as plt
- x=np.random.uniform(-3,3,size=100) #生成x特征 -3到3 100个
- X=x.reshape(-1,1)#将x编程100行1列的矩阵
- y=0.5*x**2+x+2+np.random.normal(0,1,size=100)#模拟的是标记y 对应的是x的二次函数
-
- #使用线性回归看下score
- from sklearn.linear_model import LinearRegression
- reg=LinearRegression()
- reg.fit(X,y)
- reg.score(X,y)
-
- #将预测值y_pre画图 对比真实y
- y_pre=reg.predict(X)
- plt.scatter(x,y)
- plt.plot(np.sort(x),y_pre[np.argsort(x)],color='r')
-
- #查看MSE
- from sklearn.metrics import mean_squared_error



polynomial regression

Python Code

Python Code

Python Code

Python Code

Python Code

Python Code

Python Code