SVD

May 17, 2018

Abstract

1 Eigenvalue Decomposition

Suppose there is $m \times m$ full rank symmetric matrix A, it has m different eigenvalue, and the eigenvalue is λ_i, and the corresponding uint eigenvector is x_i, then

$$\begin{aligned} Ax_1 &= \lambda_1 x_1 \\ Ax_2 &= \lambda_2 x_2 \\ &\cdots \\ Ax_m &= \lambda_m x_m \end{aligned} \tag{1}$$

then :

$$AU = U\Lambda \tag{2}$$

$$U = [x_1 \quad x_2 \cdots x_m] \tag{3}$$

$$\Lambda = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_m \end{bmatrix} \tag{4}$$

Since the symmetric matrix eigenvectors are orthogonal to each other, U is a orthogonal matrix, and the inverse matrix of the orthogonal matrix is eaqual to its transpose.

$$A = U\Lambda U^{-1} = U\Lambda U^T \tag{5}$$

2 Singular Value Decomposition

Suppose there is $m \times n$ matrix A, the aim is to find a group of orthogonal basis in n-dimensional space which is also orthogonal after A-transforms. If the orthogonal basis v is the eigenvector of the matrix $A^T A$, then the elements in v are orthogonal to each other because the matrix $A^T A$ is symmetric matrix.

$$
\begin{aligned}
{v_i}^T A^T A v_j &= {v_i}^T \lambda_j v_j \\
&= \lambda_j {v_i}^T v_j \\
&= \lambda_j v_i v_j = 0
\end{aligned} \tag{6}
$$

Then normalize the orthogonal basis after mapping:

Tips: $\begin{bmatrix} a \\ b \end{bmatrix} \cdot \begin{bmatrix} c \\ d \end{bmatrix} = [ac + bd] = \begin{bmatrix} a \\ b \end{bmatrix}^T \begin{bmatrix} c \\ d \end{bmatrix}$

$$
\begin{aligned}
Av_i \cdot Av_i &= (Av_i)^T Av_i \\
&= {v_i}^T A^T A v_i \\
&= \lambda_i v_i \cdot v_i = \lambda_i
\end{aligned} \tag{7}
$$

$\Rightarrow$

$$
|Av_i|^2 = \lambda_i \geq 0 \tag{8}
$$

So the unit vector is:

$$
u_i = \frac{Av_i}{|Av_i|} = \frac{1}{\sqrt{\lambda_i}} Av_i \tag{9}
$$

$\Rightarrow$

$$
Av_i = \sigma_i u_i \tag{10}
$$

σ_i is singular value, and $\sigma_i = \sqrt{\lambda_i}, 0 \leq i \leq k, k = Rank(A)$

Now expand the orthogonal vector $\{u_1, u_2, \cdots, u_k\}$ to $\{u_1, u_2, \cdots, u_m\}$ as a group of orthogonal vector in m-dimensional space. Meanwhile, expand $\{v_1, v_2, \cdots, v_k$ to $\{v_1, v_2, \cdots, v_n$ as a group of orthogonal vector and $\{v_{k+1}, v_{k+2}, \cdots, v_n$ in $Ax = 0$ solution space, when $i > k$, $\sigma_i = 0$

$$
\begin{aligned}
&A[v_1 v_2 \cdots v_k | v_{k+1} \cdots v_n] \\
&= [u_1 u_2 \cdots u_k | u_{k+1} \cdots u_m] \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \sigma k & \\ & 0 & & 0 \end{bmatrix}
\end{aligned} \tag{11}
$$

$$
A = \begin{bmatrix} u_1 \cdots u_k \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} {v_1}^T \\ \vdots \\ {v_k}^T \\ {v_{k+1}}^T \\ \vdots \\ {v_n}^T \end{bmatrix} \tag{12}
$$

Use the multiplication of matrix:

$$
A = \begin{bmatrix} u_1 \cdots u_k \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} {v_1}^T \\ \vdots \\ {v_k}^T \end{bmatrix} + \begin{bmatrix} u_{k+1} \cdots u_m \end{bmatrix} [0] \begin{bmatrix} {v_{k+1}}^T \\ \vdots \\ {v_n}^T \end{bmatrix} \tag{13}
$$

Then matrix A can be decomposed as:

$$A = \begin{bmatrix} u_1 \cdots u_k \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} {v_1}^T \\ \vdots \\ {v_k}^T \end{bmatrix} \tag{14}$$

$$X = \begin{bmatrix} u_1 \cdots u_k \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} = \begin{bmatrix} \sigma_1 u_1 \cdots \sigma_k u_k \end{bmatrix} \tag{15}$$

$$Y = \begin{bmatrix} {v_1}^T \\ \vdots \\ {v_k}^T \end{bmatrix} \tag{16}$$

Then $A = XY$ is the full rank decomposition of matrix A.

3 SVD in Recommender System

The data matrix of *user and item* is huge. And there are many blanks in this matrix, so it is also an extrmely sparse matrix. We defined this scoring matrix as $R_{U \times I}$

Table 1: user/item matrix

user/item	1	2	3	4	$\cdots$
1	5	4	4.5	?	$\cdots$
2	?	4.5	?	4.5	$\cdots$
3	4.5	?	4.4	4	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

The scoring matrix $R_{U \times I}$ can be decompesed as two matrices$P$ and Q:

$$R_{U \times I} = P_{U \times k} Q_{k \times I} \tag{17}$$

U is the number of users, I is the number of items, k is the rank of matrix R. Assuming that the known score is r_{ui}, the error between the true value and the predicted value is:

$$e_{ui} = r_{ui} - \widehat{r}_{ui} \tag{18}$$

The total error squared sum is:

$$SSE = \sum_{u,i} {e_{ui}}^2 = \sum_{u,i} (r_{ui} - \sum_{k=1}^{K} p_{uk} q_{ki}) \tag{19}$$

3.1 Basic SVD with Gradient Descent

$$\begin{aligned}\frac{\partial}{\partial p_{uk}} SSE &= \frac{\partial}{\partial p_{uk}} \sum_{u,i} (e_{ui})^2 \\ &= 2e_{ui} \frac{\partial}{\partial p_{uk}} e_{ui} \\ &= 2e_{ui} \frac{\partial}{\partial p_{uk}} (r_{ui} - \sum_{k=1}^{K} p_{uk} q_{ki}) \\ &= 2e_{ui} q_{ki}\end{aligned} \tag{20}$$

Explanation for the equation:
there is no summary symbol in the second step because only the equation which has u has the derivative result, other equations' derivative results are zero.

$$\begin{aligned}p_{uk} &:= p_{uk} - \eta(-e_{ui} q_{ki}) = p_{uk} + \eta e_{ui} q_{ki} \\ q_{ki} &:= q_{ki} - \eta(-e_{ui} p_{uk}) = q_{ki} + \eta e_{ui} p_{uk}\end{aligned} \tag{21}$$

There are two different ways to update the two arguments:

- **Batch Gradient Descent Algorithm**
  Update p,q after calculating all the predictions of known scores
- **Random Gradient Descent Algorithm**
  Update p,q after calculating one e_{ui}

We choose random gradient descent algorithm because we have to minimize the equation SSE, so we have to search in the direction of negative gradient, this may cause the gradient descent stop in the local optimal solution if we use bath gradient descent algorithm.

3.2 Adding Biases

The equation 17 complains the interactions between users and items that produce the different rating values. However, much of the ovserved variation in rating values is due to effects associated with either users or items, known as biases or intercepts.

A first-order approximation of the bias involved in ranting r_{ui} is as follows:

$$b_{ui} = \mu + b_i + b_u \tag{22}$$

b_{ui} is the estimation of the rank the user may gives. μ is the mean of all the item ranks, b_u is the user's mean rank and b_i is the mean rank that the item gets.

$$\hat{r_{ui}} = \mu + b_i + b_u + {q_i}^T p_u \tag{23}$$

3.3 Gradient Descent with Biases

As users ratings of the merchandise not only depends on a relationship between users and products, but also depends on the unique properties of the users and products.

$$\begin{aligned} SSE &= \frac{1}{2}(\sum_{u,i} {e_{ui}}^2 + \lambda \sum_u |p_u| + \lambda \sum_i |q_i| + \lambda \sum_u {b_u}^2 + \lambda \sum_u {b_i}^2) \\ &= \frac{1}{2}\sum_{u,i}(r_{ui} - \mu - b_u - b_i - \sum_{k=1}^{K} p_{uk} q_{ki})^2 + \frac{1}{2}(\lambda \sum_u |p_u| + \lambda \sum_i |q_i| + \lambda \sum_u {b_u}^2 + \lambda \sum_u {b_i}^2) \end{aligned} \tag{24}$$

$$\begin{aligned} \frac{\partial}{\partial b_u} SSE &= -e_{ui} + \lambda b_u \\ \frac{\partial}{\partial b_i} SSE &= -e_{ui} + \lambda b_i \end{aligned} \tag{25}$$

Then implies the equation to update b_u and b_i:

$$\begin{aligned} b_u &:= b_u + \eta(e_{ui} - \lambda b_u) \\ b_i &:= b_i + \eta(e_{ui} - \lambda b_i) \end{aligned} \tag{26}$$

The details of the algorithm is as follows:

Firstly, generate two random value matrix p and q, then use gradient descent to minimize the error and update b_i, b_u, p, and the mean-square error Rmse.

If Rmse increase, it means that the result just strides the local optimal point. So the train process has been finished.

The return result is the matrix p and q.

The explaination of some variables:

- iteration_times: the iteration times that gradient descent runs
- b_u: mean rank of a user gives
- b_i: mean rank that a product gets
- e: error
- eta: gradient descent step length, when the result is close local optimal result, we should decrease the step length
- l: regularization parameter

```
Rmse = 0
mLastRmse = 100000
rui = 0
for iteration in range(0, iteration_times):
  Rmse = 0
  nRateNum = 0
  for user in range(0, bu.shape[1]):
    for item in range(0, bi.shape[1]):
      if rank_information[user][item] != 0:
        rui = mean+bu[user]+bi[item]+InnerProduct(p[user],q[item])
        if rui>MaxRank:
          rui = MaxRank
        if rui<MinRank:
          rui = MinRank
        e = rank_information[user][item]-rui

        \\update bu,bi,p,q
        bu[user] += eta*(e-l*bu[user])
        bi[item] += eta*(e-l*bi[item])
        for rank in range(0, matrix_rank):
          p[user][rank] += eta*(e*q[rank][item]-l*p[user][rank])
          q[rank][item] += eta*(e*p[user][rank]-l*q[rank][item])
        Rmse += e*e
        nRateNum++
    Rmse = sqrt(Rmse/nRateNum)
    if(Rmse>mLastRmse)
      break
    mLastRmse = Rmse
    eta *= 0.9
```

RSVD

4 Experiment

4.1 Dataset

MovieLens 100K Dataset
Stable benchmark dataset. 100,000 ratings from 1000 users on 1700 movies. Released 4/1998.

4.2 Result

$$
\begin{aligned}
Rmse &= \frac{1}{n}\sum(tain\_\hat{r_{ui}} - train\_r_{ui})^2 \\
mse &= \frac{1}{n}\sum(tain\_\hat{r_{ui}} - test\_r_{ui})^2 \\
sse &= \sum(tain\_\hat{r_{ui}} - test\_r_{ui})^2
\end{aligned}
\tag{27}
$$

matrix_rank= 600, eta = 0.1, lambda = 0.01

Table 2: recommendation result without noise

iteration	Rmse	mse	sse
0	0.9818	3.0087	905250.0317
1	0.9566	0.8326	69317.6155
2	0.9510	0.8395	70469.0158
3	0.9471	0.8478	71874.9919
4	0.9436	0.8553	73152.9617
5	0.9404	0.8620	74312.8893
6	0.9376	0.8681	75364.0547
7	0.9351	0.8736	76314.9248
8	0.9328	0.8785	77173.8627
9	0.9308	0.8829	77948.5383

Add Laplace noise $d \sim (0, 1)$ to the original data:

Table 3: recommendation result without noise

iteration	Rmse	mse	sse
0	0.9820	3.0120	907200.4728
1	0.9567	0.8325	69309.9256
2	0.9510	0.8397	70506.2410
3	0.9471	0.8480	71911.5554
4	0.9436	0.8555	73188.9633
5	0.9405	0.8623	74348.5918
6	0.9377	0.8683	75399.7170
7	0.9351	0.8738	76350.8106
8	0.9329	0.8787	77209.8048
9	0.9308	0.8831	77984.3414

4.2.1 Result Analysis

As the iteration time increase, Rmse is dereasing, which means the SVD model is more close to the trainning dataset. However, mse and sse are increasing, which means the SVD model may become overfit apparently. According to the results above, the results of data without noise and data with noise are similar. However, as for recommendation system, there is no need to care about the accurency of the predicted value. The key of the prediction model is the predicted item sets.