Assignment-06
Matrix Factorization

Ravi Kumar
14MCMI12
SCIS, University of Hyderabad

March 25, 2015

1 Introduction

Matrix Factorization is a factorization of a matrix into a product of matrices. There are many different matrix factorization, each finds use among a particular class of problems.Matrix Factorization is an example Unsupervised learning (features are learned with unlabeled input data).From an application point of view, Matrix Factorization can be used to discover latent features [Latent feature are feature that are not directly observed but are rather inferred (through a mathematical model) from other features that are observed (directly measured) underlying the interactions between two different kinds of features].
The intuition behind using matrix factorization to solve problem is that there should be some latent features that determine how a user rates an item. For example, two users would give high ratings to a certain movie if they both like the actors/actresses of the movie, or if the movie is an action movie, which is a genre preferred by both users. Hence, if we can discover these latent features, we should be able to predict a rating with respect to a certain user and a certain item, because the features associated with the user should match with the features associated with the item.

Method :

Firstly, we have a set $U$of users, and a set V of items. Let R of size $|n| \times |m|$ be the matrix that contains all the ratings that the users have assigned to the items. Also, we assume that we would like to discover K latent features. Our task, then, is to find two matrics matrices U (a $|n| \times |k| matrix$) and V (a $|k| \times |m| matrix$) such that their product(X) approximates R .

$$R \approx U_{n\times k} \times V_{m\times k}^{T} = X$$

$$\text{where } k \text{ is } \min\{m, n\}$$

Objective Function: Intialize the two matrices(U, V) with some values(between 0 and 1), calculate how 'different' their product is to E, and then try to minimize this difference iteratively. Such a method is called gradient descent, aiming at finding a local minimum of the difference.

$$J(U,V) = \sum (R_{i,j} - (U \times V^T))^2 + \lambda(||U||^2 + ||V||^2)$$

we have to minimize above function. The difference here, usually called the error between the estimated rating and the real rating, can be calculated by the following equation for each user-item pair:

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

To minimize the error, we have to know in which direction we have to modify the values of u_{ik} and v_{kj}. In other words, we need to know the gradient at the current values, and therefore we differentiate the above equation with respect to these two variables separately:

$$U_{t+1} = U_t + \alpha \frac{\partial J(U,V)}{\partial U}$$

$$V_{t+1} = V_t + \alpha \frac{\partial J(U,V)}{\partial V}$$

Most matrix factorization methods can be adapted so that the error function is only evaluated at known points. For instance, you can take the gradient descent approach to SVD (minimizing the frobenius norm) but only evaluate the error and calculate the gradient at known points.

The final Formule for updation are derived as below:

$$U_{i,k} = U_{i,k} + \alpha(2e_{i,j} * V_{k,j-\beta * U_{i,k}})$$

$$V_{i,k} = V_{i,k} + \alpha(2e_{i,j} * U_{k,j-\beta * V_{i,k}})$$

Here, α is a constant whose value determines the rate of approaching the minimum. Usually we will choose a small value for α , say 0.0002. This is because if we make too large a step towards the minimum we may run into the risk of missing the minimum and end up oscillating around the minimum. New parameter β is used to control the magnitudes of the user-feature and item-feature vectors such that U and V would give a good approximation of R without having to contain large numbers. In practice, β is set to some values in the range of 0.02.

2 Modules

2.1 Preprocessing

This module is responsible for reading the data set from file. This module takes the dataset name via command line arguments. Depending upon the these parameters the initial processing of table takes place. Among the input samples, 80% of them are chosen as training dataset at random and rest as testing.

2.1.1 Calculation/Updation

Formule for updation are derived as below:

$$U_{i,k} = U_{i,k} + \alpha(2e_{i,j} * V_{k,j-\beta * U_{i,k}})$$

$$V_{i,k} = V_{i,k} + \alpha(2e_{i,j} * U_{k,j-\beta * V_{i,k}})$$

By using this formula we update the U and V values.Each time we update U and V values and calculte the error with respect to original R matrix.Repeat this process untill we get approximate value $X(X{=}U \times V^T)$ to R.

2.2 Error Calculation

We calculated two error values RMSE and MAE. RMSE (root mean squared error), also called RMSD (root mean squared deviation), and MAE (mean absolute error) are both used to evaluate models by summarizing the differences between the actual (observed) and predicted values. MAE gives equal weight to all errors, while RMSE gives extra weight to large errors.

2.3 Contribution

My contribution to this Assignment is the implementation in each module partitialy.

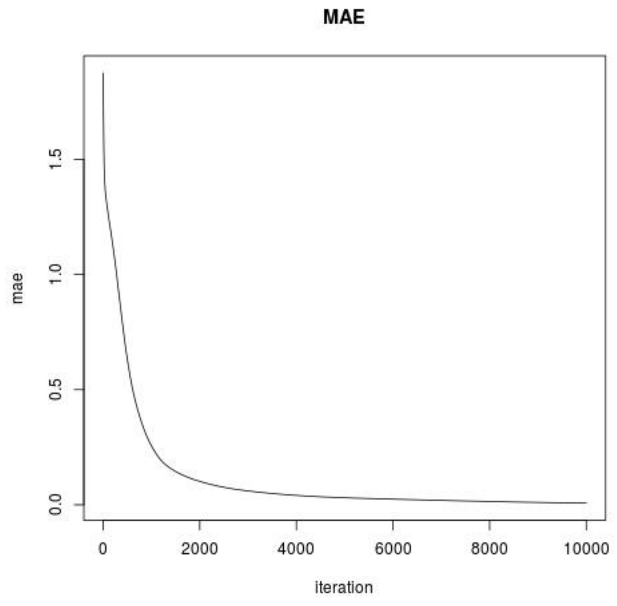
2.4 Experimental Result

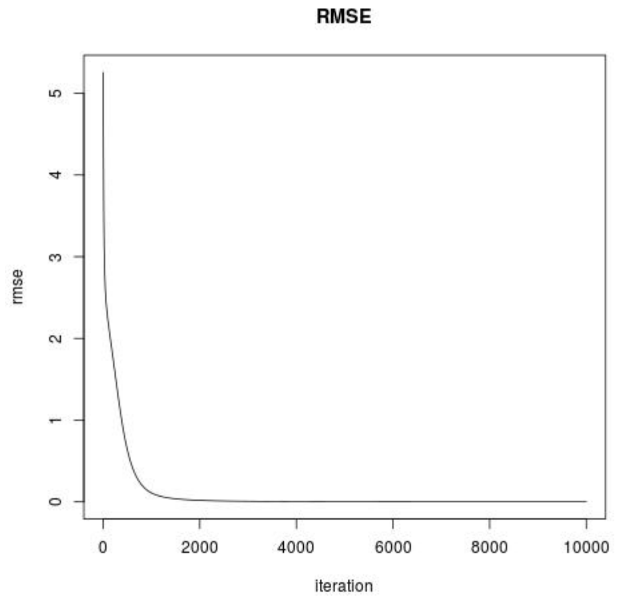
This code is working completely fine.According to problem definition the training data set & test data set are divided randomly.so the accuracy rate is also depend upon the type of data set that is involved and division. The given data set was around uniformly distributed across different classes.

2.5 Conclusion

we can be concluded that Matrix Factorization performs reasonably well on small as well as large dataset.
For Small dataset :
RMSE =3.45142 , MAE = 3.1142
For Big Dataset :
RMSE =3.262 MAE =2.938

Figure 1: Small Data set





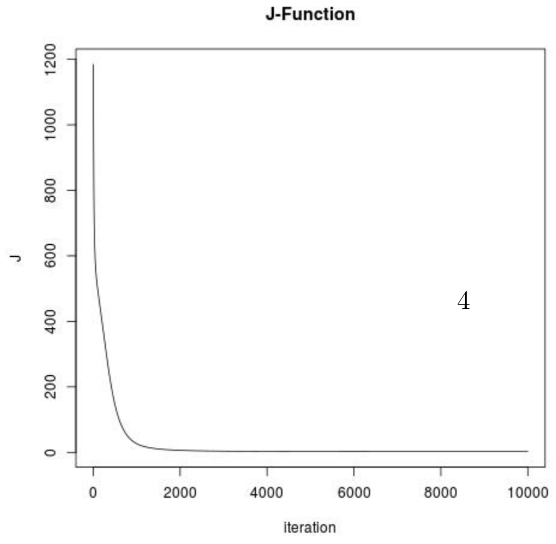
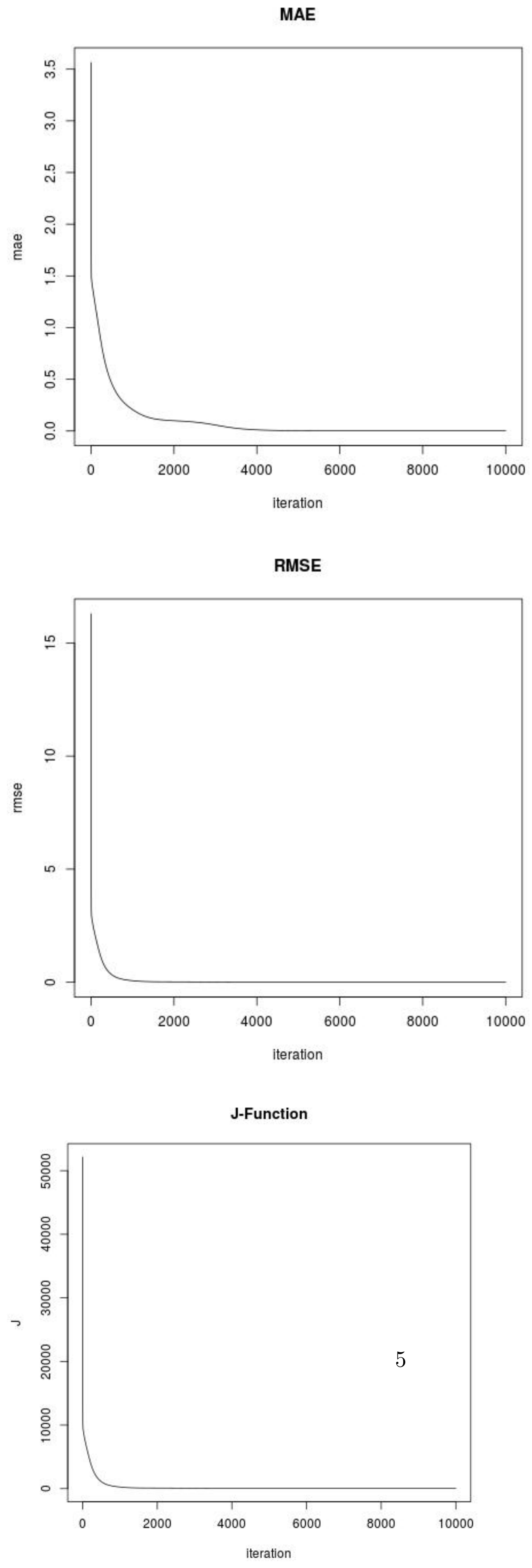


Figure 2: Big Data set


MAE
mae
iteration
RMSE
rmse
iteration
J-Function
J
iteration