华南理工大学

South China University of Technology

# The Experiment Report of *Machine Learning*

**SCHOOL:** SCHOOL OF SOFTWARE ENGINEERING

**SUBJECT:** SOFTWARE ENGINEERING


*Author:*
Riyang Hu
Xianzhe Wu
Canguang Li

*Supervisor:*
Mingkui Tan

*Student ID:*
201530611616
201530613078
201530611937

*Grade:*
Undergraduate


December 28, 2017

# Recommender System Based on Matrix Decomposition

***Abstract*—In this experiment we use Matrix Decomposision to construct a Recommender System ,which predict whether a user likes a movie he has not seen.**

## I. Introduction

RECOMMENDER System is a subclass of information filtering system that seeks to predict the $\boldsymbol{rating}$ or $\boldsymbol{preference}$ that a user would give to a item.

Recommender systems hava become increasingly popular in recent years,and are utilized in a variety of areas including movies,music news, books, research articles.There are also recommender systems for experts, collaborators, jokes, restaurants, garments, financial services, life insurance, romantic partners, and Twitter pages.

In this experiment, we use Matrix Decomposition to construct a simple Recommender System, and optimizes it by SGD.

## II. Methods and Theory

**Problem formulation:**

$n_u$ = number of user

$n_m$ = number of item

$n$ = number of feature

$r(i,j) = 1$ if user $j$ has rated item $i$ (0 otherwise)

$y^{(i,j)}$ = rating by user $j$ on item $i$ (if defined)

$P^{(j)}$ = feature vector for user $j$

$Q^{(i)}$ = feature vector for item $i$

For user $j$, item $i$, predicted rating :$(P^{(j)})^T(Q^{(i)})$

**Collaborative filtering optimization objective:**

Given $Q^{(1)}, ..., Q^{(n_m)}$

we can estimate $P^{(1)}, ..., P^{(n_u)}$ by minimizing:

$$\frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}((P^{(j)})^TQ^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(P_k^{(j)})^2$$

Given $P^{(1)}, ..., P^{(n_u)}$

we can estimate $Q^{(1)}, ..., Q^{(n_m)}$ by minimizing:

$$\frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}((P^{(j)})^TQ^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(Q_k^{(i)})^2$$

So we can $\boldsymbol{minimizing}$

$P^{(1)}, ..., P^{(n_u)}$ and $Q^{(1)}, ..., Q^{(n_m)}$ $\boldsymbol{simultaneously}$ **:**

$$J(P,Q) = \frac{1}{2}\sum_{r(i,j)=1}h(i,j) + \frac{\lambda}{2}(\sum_{i=1}^{n_m}f(Q^{(i)}) + \sum_{j=1}^{n_u}f(P^{(j)}))$$

$\boldsymbol{while}$ **:**

$$h(i,j) = ((P^{(j)})^TQ^{(i)} - y^{(i,j)})^2$$

$$f(X) = \sum_{k=1}^{n}(X_k)^2$$

**Collaborative filtering algorithm:**

---

*Algorithm's process* :

- $\boldsymbol{Given}$ learning rate $\alpha$ and $\lambda$
- $\boldsymbol{Initalize}$ $P^{(1)}, ..., P^{(n_u)}, Q^{(1)}, Q^{(n_m)}$ to small values.
- $\boldsymbol{Minimize}$ $J(P,Q)$ using SGD :

  for every $i = 1, .., n_u, j = 1, ..., n_m$ :

$$P_k^{(j)} -= \alpha(\sum_{i:r(i,j)=1}((P^{(j)})^TQ^{(i)} - y^{(i,j)})Q_k^{(i)} + \lambda P_k^{(j)})$$

$$Q_k^{(i)} -= \alpha(\sum_{j:r(i,j)=1}((P^{(j)})^TQ^{(i)} - y^{(i,j)})P_k^{(j)} + \lambda Q_k^{(i)})$$

- For a user with parameters $\theta$ and a movie with features $x$, predict a rating of $\theta^T x$.

---

## III. Experiments

### A. Dataset

- Utilizing MoviesLens-100k dataset.
- u.data – Consisting 10,000 comments from 943 users out of 1682 movies, At least, each user comment 20 videos, Users and movies are numbered consecutively from number 1 respectively.The data is sorte randomly.
- u1.base / u1.test are train set and validation set respectively, seperated from dataset u.data with proportion of 80% and 20%. It also make sense to train set and validation set from u1.base / u1.test to u5.base / u5.test.

### B. Implementation

- 1. Read the data set and divide it. Populate the original scoring matrix $y_{n_u,n_m}$ against the raw data, and fill 0 for null values.
- 2. Initialize the user factor matrix $P_{n_u,n}$ and the item (movie) factor matrix $Q_{n_m,n}$, where $n$ is the number of potential features.

- 3. Determine the loss function and hyperparameter learning rate $\alpha$ and the penalty factor $\lambda$.
- 4. Use the stochastic gradient descent method to decompose the sparse user score matrix, get the user factor matrix and item (movie) factor matrix:
  - 4.1 Select a sample from scoring matrix randomly;
  - 4.2 Calculate this sample's loss gradient of specific row(column) of user factor matrix and item factor matrix;
  - 4.3 Use SGD to update the specific row(column) of $P_{n_u,n}$ and $Q_{n_m,n}$;
  - 4.4 Calculate the $L_{val}$ on the validation set, comparing with the $L_{val}$ of the previous iteration to determine if it has converged.
- 5. Repeat step 4. 10000 times, get a satisfactory user factor matrix $P$ and an item factor matrix $Q$, Draw a $L_{val}$ curve with varying iterations.

TABLE I: Simulation Parameters

| | |
|---|---|
| learning rate $\alpha$ | 0.045 |
| $\lambda$ | 0.02 |
| number of features $n$ | 5 |

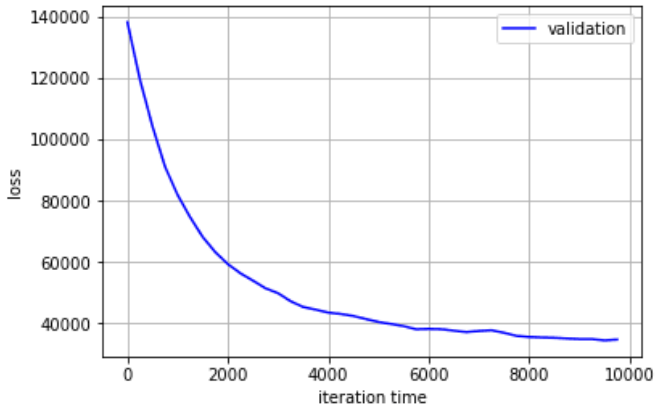


Fig. 1: $L_{val}$ curve with varying iterations.

## IV. Conclusion

So we successfully constructed a basic recommender system and got the best result as we can . In this experiment , we used familiar algorithms like SGD and combined them with new algorithm to satisfy new requirement . We all believed this process was important because we could turn our knowledge into something real and useful so we gained a lot from this experiment . And our system chose SGD as our optimizing method and Readers can also try another method to develop recommender system and compare the result between different optimizing algorithm .