South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Haipeng Deng

Supervisor:
Mingkui Tan

Student ID:
201730686193

Grade:
Undergraduate

November 11, 2019

Recommender System Based on Matrix Decomposition

***Abstract*—This report introduces my work in lab5, where I built a recommending system based on matrix factorization and optimized it using the strategy of stochastic gradient descent.**

I. Introduction

THE recommender system has been developed as an independent discipline for nearly twenty years but not yet has a precise definition. Several strategies have been proposed in the way to constructing a more accurate and more useful model, such as Content-based Algorithm, Item-based Collaborative Filtering Algorithm and Matrix Factorization Algorithm. In this lab, we will work on a Matrix Factorization model, which put the observed samples in a sparse matrix and try to decompose it through introducing a new parameter K, which tries to identify the potential features of each user and each item. The main goals of the lab are completely generating a movie recommender system and cultivate engineering ability under a small dataset.

II. Methods and Theory

Before constructing a recommender system using matrix factorization, there are some concepts we need to introduce. A matrix $R \in \mathbb{R}^{m*n}$ is a sparse rating matrix for m users and n items(movies in this case). An observed samples $r_{u,i}$ (means the rate of user u towards item(movie) i) will be represented as a specific value in R. In the matrix factorization method we use, we will try to decompose R into two matrices $P \in \mathbb{R}^{m*k}$ and $Q \in \mathbb{R}^{k*n}$ where k denotes potential features for each user and movie.

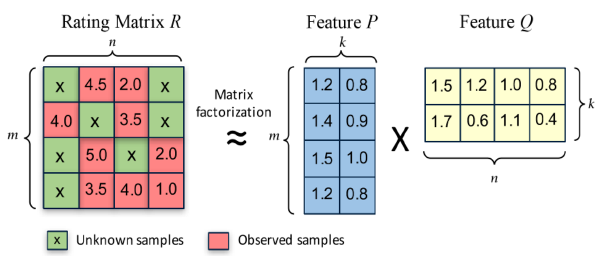

Fig. 1. An illustration of matrix factorization (m=4, n=4, k=2).

After confirming the basic skeleton of our model, the next step is to determine how we evaluate the goodness of it and the way we optimize it. A naive loss function is squared error loss:

$$L(r_{u,i}, \hat{r}_{u,i}) = (r_{u,i} - \hat{r}_{u,i})^2 \tag{1}$$

Besides, there are two ways to minimize the loss on our lab guidance alternate least squares optimization(ALS) and stochastic gradient descent method(SGD). I chose SGD as my strategy to optimize, because it is a simple and scalable to large-scale datasets way compared to ALS.
In stochastic gradient descent, we define the loss function as below:

$$L = \sum_{u,i \in \Omega} (r_{u,i} - p_u^T q_i)^2 + \lambda_p ||p_u||^2 + \lambda_q ||q_i||^2 \tag{2}$$

$r_{u,i}$ denotes the actual rating of user u for item i. Ω denotes the set of observed samples from rating matrix R. λ_p and λ_q are regularization parameters to avoid overtting. In order to minimize the loss and update our matrices P and Q, we randomly pick an observed sample $r_{u,i}$ in the sparse matrix R and there are two other important formulas which we use to calculate gradients:

$$\frac{\partial l}{\partial p_u} = E_{u,i}(-q_i) + \lambda_p p_u \tag{3}$$

$$\frac{\partial l}{\partial q_i} = E_{u,i}(-p_u) + \lambda_q q_i \tag{4}$$

In the equations above, p_u denotes the uth user's column vector while q_i denotes the ith movie's column vector. They both have shape k * 1
The following step is to update the feature matrices P and Q with learning rate α making use of the gradients we computed:

$$p_u = p_u + \alpha \frac{\partial l}{\partial p_u} \tag{5}$$

$$q_i = q_i + \alpha \frac{\partial l}{\partial q_i} \tag{6}$$

We will repeat the steps above until the loss converge.

III. Experiments

A. Dataset

In this lab we Utilize the 'MovieLens-100k dataset' which Consists 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 sorted randomly.

B. Implementation

Firstly, I decided to use u1.base as my train data and u1.test as my test data, then I read the data set to generate my matrix R and fill 0 for null values. According to the principle that $k << min(m,n)$, I finally chose $k = 50, \lambda_p = 0.5, \lambda_q = 0.5 \alpha = 0.01$ as my input hyperparameters. To perform stochastic gradient descent, I randomly selected an observed sample $r_{u,i}$ from observed set, then calculated the gradient $\frac{\partial l}{\partial p_u}$, $\frac{\partial l}{\partial q_i}$ to the loss function. Next, I updated the feature matrices P and Q with learning rate $alpha$ and gradient using formula (5) (6). I repeated the above processes until convergence.

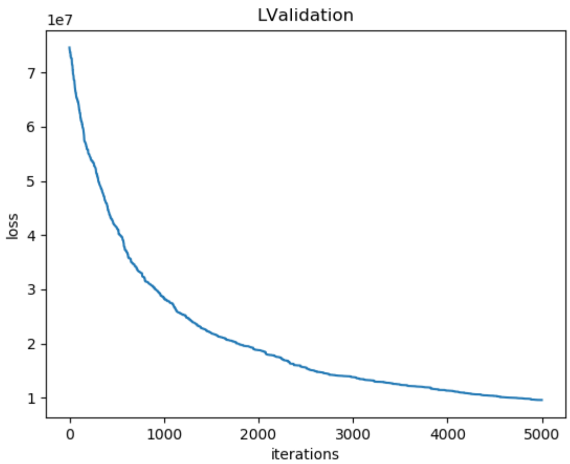

Fig. 2. The change of Lvalidation.

IV. Conclusion

To summarize, I feel this lab very inspiring and beneficial for me. Instead of reading and deriving the theory and formulas in books and slices, it offers us a practical way to practise machine learning.