CSE 6240 - Spring 2015
Web Search & Text Mining
Homework 6

Due: 03/22/2015 23:59

1 Problem Description

In Homework 5, we implemented memory-based collaborative filtering. The goal of this homework is to implement a model-based CF method. Specifically, please implement a matrix factorization based CF method using gradient descent.

Assume that the rating matrix $M \in \mathbb{R}^{m \times n}$ (m and n are the number of users and items, respectively.) is a low-rank matrix. Intuitively, there is only a small number of factors (e.g, genre, director, main actor/actress, released year, etc.) that determine the user preference. Define r as the number of factors, we need to learn a user profile matrix $U \in \mathbb{R}^{m \times r}$ and an item profile matrix $V \in \mathbb{R}^{n \times r}$ from the observed ratings. A rating can be approximated by an inner product of two r-dimensional vectors, one for the user profile and the other for the item profile. Mathematically, a rating of user u on movie i can be approximated by

$$M_{u,i} \approx \sum_{k=1}^{r} U_{u,k} V_{i,k}. \tag{1}$$

We want to learn U and V by minimizing reconstruction error over all observed ratings $\mathcal{O}$. In other words, we want to minimize the following objective function:

$$E(U,V) = \sum_{(u,i)\in\mathcal{O}} (M_{u,i} - U_u^T V_i)^2 = \sum_{(u,i)\in\mathcal{O}} (M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k})^2 \tag{2}$$

where U_u is the uth row of U and V_i is the ith row of V.

As U and V are interrelated in (2), there is no closed form solution. We update each element of U and V using gradient descent:

$$U_{u,k} \leftarrow U_{u,k} - \mu \frac{\partial E(U,V)}{\partial U_{u,k}}, \qquad V_{i,k} \leftarrow V_{i,k} - \mu \frac{\partial E(U,V)}{\partial V_{i,k}}, \tag{3}$$

where μ is a user provided parameter determining the updating rate (a.k.a. learning rate). $\frac{\partial E(U,V)}{\partial U_{u,k}}$ and $\frac{\partial E(U,V)}{\partial V_{i,k}}$ are partial derivatives of $E(U,V)$ with respect to $U_{v,k}$ and $V_{j,k}$, respectively.

2 Requirements

2.1 Equation Derivation (20%)

Derive the update equations in (3) by specifying the partial derivatives.

2.2 Equation Derivation (10%)

To avoid overfitting, it is common to add regularization terms that penalize large values in U and V. Derive the update equations in (3) for the regularized objective function below:

$$E(U,V) = \sum_{(u,i)\in\mathcal{O}} (M_{u,i} - \sum_{k=1}^{r} U_{u,k}V_{i,k})^2 + \lambda(\sum_{u,k} U_{u,k}^2 + \sum_{i,k} V_{i,k}^2)$$

where λ is a user provided parameter controlling the effect of regularization.

2.3 Implementation (70%)

To use gradient descent to optimize an objective function, you need to update the variables repeatedly. To start, you may need a random initial guess. To stop, you may need to set stopping criteria, based on either reconstruction error or the maximum number of iterations. Usually, it is necessary to try several different values for μ and λ to find the best model. To evaluate the model, we use RMSE (Root Mean Squared Error) that, for testing ratings $\mathcal{T}$, can be computed as follows:

$$\sum_{(u,i)\in\mathcal{T}} (M_{u,i} - f(u,i))^2$$

where $M_{u,i} \in \mathcal{T}$ is the ground truth rating and $f(u,i)$ is the estimated rating.

2.3.1 Code (40%)

Given a set of ratings (*ratings.csv*), you need to implement two functions: one for the basic regularized MF and one for the cross-validation experiments based on regularized MF.

Basic Regularized MF For the basic regularized MF function, it should have the following input:

- $\mathcal{O}$: the set of training ratings (ratings.csv)
- $\mathcal{T}$: the set of testing ratings (toBeRated.csv)
- r: the number of latent factors (dimensions)
- μ: learning rate
- λ: regularization parameter

and output:

- result.csv, where each line is the rating correspoding to the (userID, movieID) pair in the toBeRated.csv file.

Cross-Validation Regularized MF For the cross-validation function, it should have the following input:

- $\mathcal{M}$: the set of ratings (ratings.csv)
- r: the number of latent factors (dimensions)
- μ: learning rate
- λ: regularization parameter
- D: the number of folds

and output:

- RMSE on the testing ratings averaged over M folds

2.3.2 Report (30%)

RMSE for CV Please report the results based on 10-fold cross-validation. Specifically, please fill in the following Table 1, 2, 3 and 4 (please round RMSE to 4 decimals).

RMSE	$\lambda = 0.05$	$\lambda = 0.1$	$\lambda = 0.5$
$\mu = 0.0001$			
$\mu = 0.0005$			
$\mu = 0.001$			

Table 1: RMSE for $r = 1$

RMSE	$\lambda = 0.05$	$\lambda = 0.1$	$\lambda = 0.5$
$\mu = 0.0001$			
$\mu = 0.0005$			
$\mu = 0.001$			

Table 2: RMSE for $r = 3$

RMSE	$\lambda = 0.05$	$\lambda = 0.1$	$\lambda = 0.5$
$\mu = 0.0001$			
$\mu = 0.0005$			
$\mu = 0.001$			

Table 3: RMSE for $r = 5$

Analysis Based on the RMSE results you have obtained, please answer the following questions:

1. What do you observe when you vary r? Why?
2. Which model is the best? Please describe the best model in Table 4 and explain your choice.
3. Suppose you are using regularized MF in real systems, how will you choose parameters? Why?

2.4 Submission

The folder structure of your submission should be as below.

```
<your gtid>-<your gt account>-hw6
|-- hw6_answer.pdf
|-- Recommender.py/Recommender.cpp/Recommender.java
|-- Recommender_CV.py/Recommender_CV.cpp/Recommender_CV.java
|-- Readme.txt
```

1. hw6_answer.pdf should contain all the tables and answers to the questions.
2. Recommender.py/Recommender.cpp/Recommender.java should be the function for basic regularized MF. We should be able to run your code using, for example, python Recommender.py ratings.csv toBeRated.csv 5 0.001 0.1
3. Recommender_CV.py/Recommender_CV.cpp/Recommender_CV.java should be the function for Cross-Validation MF. We should be able to run your code using, for example, python Recommender_CV.py ratings.csv 5 0.001 0.1 10

μ	
λ	
r	
RMSE	

Table 4: The Best Model

4. Readme.txt should show how to run your code to obtain the result.csv. (Don't provide the result.csv file, but show how we can create it using your code).

5. C++/Python/Java/Matlab is preferred. If you use other languages, please be prepared to show how to run your code in person.