

CPSC 558 Final Project: Movie Recommendation

Bo Song, Fan Yang and Xinyu Weng

December 12, 2015

Contents

1	Introduction	1
2	Algorithm	1
2.1	Movie ratings dataset	1
2.2	Collaborative filtering learning algorithm	2
2.2.1	Collaborative filtering cost function	2
2.2.2	Collaborative filtering gradient	2
2.2.3	Regularized cost function	2
2.2.4	Regularized gradient	3
2.3	Learning movie recommendation	3
3	Implementation	3

1 Introduction

Facing increasingly large numbers of movies in the world nowadays, we want to build a recommendation system, which can help us find out movies that fit our taste best.

For this project, we build a movie recommendation system, which can predicate the users preference and recommend movies to them. The basic theory is that, when we know the similarity between two items, we can use the existing data to forecast the unknown preferences.

In this movie recommendation system, it will compute the similarity between the movies you did see and the movie you didnt see. If theres a high similarity between these movies, the algorithm will infer that youll like this movie.

2 Algorithm

2.1 Movie ratings dataset

The matrix Y (a num movies num users matrix) stores the ratings $y^{(i,j)}$ (from 1 to 5). The matrix R is an binary-valued indicator matrix, where $R(i, j) = 1$ if user j gave a rating to movie i , and $R(i, j) = 0$ otherwise. The objective of collaborative filtering is to predict movie ratings for the movies that users have not yet rated, that is, the entries with $R(i, j) = 0$. This will allow us to recommend the movies with the highest predicted ratings to the user.

We define two matrices, X and $Theta$ here:

$$X = \begin{bmatrix} -(x^{(1)})^T - \\ -(x^{(2)})^T - \\ \dots \\ -(x^{(n_m)})^T - \end{bmatrix}$$

$$Theta = \begin{bmatrix} -(\theta^{(1)})^T - \\ -(\theta^{(2)})^T - \\ \dots \\ -(\theta^{(n_u)})^T - \end{bmatrix}$$

where n_m is the number of movies, n_u is the number of users.

The i -th row of X corresponds to the feature vector $x^{(i)}$ for the i -th movie, and the j -th row of $Theta$ corresponds to one parameter vector $\theta^{(j)}$, for the j -th user. Both $x^{(i)}$ and $\theta^{(j)}$ are n -dimensional vectors. For the purposes of

this exercise, you will use $n = 100$, and therefore, $x^{(i)} \in \mathbb{R}^{100}$ and $\theta^{(j)} \in \mathbb{R}^{100}$. Correspondingly, X is a $n_m \times 100$ matrix and Θ is a $n_u \times 100$ matrix.

2.2 Collaborative filtering learning algorithm

The collaborative filtering algorithm in the setting of movie recommendations considers a set of n -dimensional parameter vectors $x^{(1)}, \dots, x^{(n_m)}$ and $\theta^{(1)}, \dots, \theta^{(n_u)}$, where the model predicts the rating for movie i by user j as $y^{(i,j)} = (\theta^{(j)})^T x^{(i)}$. Given a dataset that consists of a set of ratings produced by some users on some movies, we will learn the parameter vectors $x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}$ that produce the best fit (minimizes the squared error).

2.2.1 Collaborative filtering cost function

The collaborative filtering cost function (without regularization) is given by

$$J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j): r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2$$

The cost for user j and movie i will be accumulating only if $R(i, j) = 1$.

2.2.2 Collaborative filtering gradient

The gradients of the cost function is given by:

$$\frac{\partial J}{\partial x_k^{(i)}} = \sum_{j: r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)}$$

$$\frac{\partial J}{\partial \theta_k^{(j)}} = \sum_{i: r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)}$$

The function returns the gradient for both sets of variables by unrolling them into a single vector.

2.2.3 Regularized cost function

The cost function for collaborative filtering with regularization is given by

$$J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \left(\frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2 \right) + \left(\frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2 \right)$$

The regularization will be added to the original computations of the cost function, J .

2.2.4 Regularized gradient

The gradients for the regularized cost function is given by:

$$\begin{aligned} \frac{\partial J}{\partial x_k^{(j)}} &= \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(i)} + \lambda x_k^{(j)} \\ \frac{\partial J}{\partial \theta_k^{(j)}} &= \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \end{aligned}$$

2.3 Learning movie recommendation

After finishing to implement the collaborative filtering cost function and gradient, we can now use additional data from the user to train the algorithm to make movie recommendations.

After the additional ratings have been added to the dataset, the program will proceed to train the collaborative filtering model. This will learn the parameters X and θ . To predict the rating of movie i for user j , $(\theta^{(j)})^T x^{(i)}$ will need to be computed. Then the program will to compute the ratings for all the movies and users.

3 Implementation

We implement the collaborative filtering algorithm and apply it to a dataset of movie ratings (MovieLens 100K from GroupLens Research. The website is: <http://grouplens.org/datasets/movielens/>). The dataset consists of ratings on a scale of 1 to 5. The dataset has 943 users, and ratings on 1682 different movies.

The collaborative filtering algorithm will learn a model from the dataset (the rating matrix). The rating matrix is decomposed into two matrices, user preference matrix and movie feature matrix. We extract ten features from each movie. Using the collaborative filtering algorithm, we can get the parameters

in these two matrices that can be used to minimize the cost function. Then, the recommendation system can predict the preference of users based on their previous ratings on movies and recommend suitable movies to them.

The front end is a web page written in HTML, JavaScript and AJAX. The link is: <http://sunny-song.com/cs458/>. It will ask the user to enter ratings for several random selected movies first(Shown in Figure 1).

Movie Recommendation

CS 558 Automated Decision Systems Project

Bo Song, Fan Yang, Xinyu Weng

Submit











- Like Water For Chocolate (Como agua para chocolate) (1992) Rate Me 
- Grateful Dead (1995) Rate Me 
- Die Hard (1988) Rate Me 
- Return of Martin Guerre, The (Retour de Martin Guerre, Le) (1982) Rate Me 
- Twelfth Night (1996) Rate Me 
- Price Above Rubies, A (1998) Rate Me 
- Orlando (1993) Rate Me 
- Local Hero (1983) Rate Me 
- Dragonheart (1996) Rate Me 
- April Fool's Day (1986) Rate Me 

Figure 1: User entering ratings for random selected movies

The backend is written in Matlab, which implements the collaborative filtering algorithm. The learning model is stored a file, and PHP use commands to run the Matlab program, read the learning model from the file and return the result to the front end(Shown in Figure 2).

The details of our implementations are shown as follows:

Environment Requirement

Apache, PHP, Octave

How to use it

Upload all the files to your web server. Type `http://path/to/your/server/index.html` in your browser.

Or you can visit the demo page `http://www.sunny-song.com/cs458`. We have set up all the environment for you.

It may take 5 to 10 seconds for the algorithm to learn model and predict result.

Movie Recommendation

CS 558 Automated Decision Systems Project

Bo Song, Fan Yang, Xinyu Weng

- Schindler's List (1993) 4
- Shawshank Redemption, The (1994) 4
 - Casablanca (1942) 3.5
 - Godfather, The (1972) 3.5
 - Titanic (1997) 3.5
 - Good Will Hunting (1997) 3.5
 - Close Shave, A (1995) 3.5
- Silence of the Lambs, The (1991) 3.5
 - Usual Suspects, The (1995) 3.5
 - Star Wars (1977) 3.5

Figure 2: Movie recommendations for the users

Github

<https://github.com/zjusbo/Movie-Recommendation-System>

File Description

- Index.html - Entry page.
- Movie.js - Javascript code, handle user input and send request to apache server.
- Predict.php - Handle request from movie.js, encapsulate request and send it to octave program.
- Main.m - Octave program, machine learning code. Predict user preference.
- Ex8_movies.mat, movie_idx.txt - Movie data set.
- cofiCostFunc.m, fmincg.m, normalizeRatings.m, loadMovieList.m - Machine learning algorithm.