Practice Recommendation Systems

In this practice students will implement, in pairs, collaborative filtering algorithm and apply it to a data set scores of movies[1], that we will denominate dataset. Ratings are on a scale of 1 to 5. The data come from $n_u = 943$ users, and a total $n_m = 1682$ film. Students will have a script that will support running, in parts, in each section of the practice, and help them to see if your code gives correct results.

1. Input data

This section describes the script ex2_cofi.m that will be used. The first part of this script loads the dataset, which is contained in ex2_movies.mat, generating the Y and R matrices in Matlab. The number of rows of the matrix Y is NUM_MOVIES and the number of columns is num_users. The elements of the matrix Y are the scores $y^{(i,j)}$ (from 1 to 5). The matrix R is a binary indicator matrix, where $R(i,j) = 1$ if the user j scores i film, and $R(i,j) = 0$ otherwise. The purpose of collaborative filtering is to predict, for each user, the scores of movies that has not yet rated, ie $y^{(i,j)}$ values such that $R(i,j) = 0$. After the prediction, we recommend each user movies with the biggest estimated scores.

To better understand the Y matrix, the script ex2_cofi.m calculates the average score of the first film (Toy Story) and the screen shows the value obtained.
In this part of the practice matrices X and Theta is also used:

$$\texttt{X} = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(n_m)})^T — \end{bmatrix}, \quad \texttt{Theta} = \begin{bmatrix} — (\theta^{(1)})^T — \\ — (\theta^{(2)})^T — \\ \vdots \\ — (\theta^{(n_u)})^T — \end{bmatrix}$$

The *i*th row of X containing the feature vector $x^{(i)}$ of the *i*th film. The *j*th row vector Theta contains parameters vector $\theta^{(j)}$ of the user *j*. Both vectors$x^{(i)}$ and $\theta^{(j)}$ have dimension n. In this exercise we will use n = 100, and thus $x^{(i)} \in \mathbb{R}^{100}\theta^{(j)} \in \mathbb{R}^{100}$. Therefore, X is a matrix $n_m \times 100$ and Theta is a matrix $n_u \times 100$.

2 Learning algorithm for collaborative filtering

Now we begin implementing the learning algorithm for collaborative filtering. We start implementing the cost function (without adjustment).
The collaborative filtering algorithm is based on two sets of n-dimensional vectors $x^{(1)}, \ldots, x^{(n_m)}$ and $\theta^{(1)}, \ldots, \theta^{(n_u)}$, and from which the model predicts, for user j and film i the score $y^{(i,j)} = \left(\theta^{(j)}\right)^T x^{(i)}$. Given a dataset containing a set with some values $y^{(i,j)}$, the algorithm must learn vector parameters $x^{(1)}, \ldots, x^{(n_m)}$ and $\theta^{(1)}, \ldots, \theta^{(n_u)}$ that best fit values $y^{(i,j)}$ provided (minimize mean square error).

---

[1] Movielens 100k Dataset GroupLens Research.

You must complete the code in *cofiCostFunc.m* to calculate the cost function and gradient for collaborative filtering. Parameters are *X* and *theta* function, which is what the algorithm seeks to learn. To use the function *fmincg* minimizing the cost function is set to the parameters introduced into a single vector params. As you will see, X and Theta are extracted from *params*. First we will see how to implement cost and subsequently gradient

2.1 Cost Function collaborative filtering

The cost function of collaborative filtering (without adjustment) is as follows

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

Now it is when you change *cofiCostFunc.m* to provide this cost in the variable *J*. Note that you should only accumulate the cost to the user *j* and film *i* if $R(i,j) = 1$
After completing the feature, you can use the ex2_cofi script, which runs the cost function. The obtained value should be 22.22.

2.2 Gradient collaborative filtering

Now you must implement the gradient (no regularization). Specifically, you must complete the code cofiCostFunc.m to return X_grad and Theta_grad variables. Note that X_grad should be an array of the same size as X, and Theta_grad must have the same size as Theta. The X_grad and Theta_grad gradients are given by

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

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

Note that the function returns the gradient as a vector. After completing the code gradients through the script ex2_cofi, you can check this implementation. If correct, you will see that the numerical and analytical gradient are very similar.

2.3 regularized cost function

The cost function for collaborative filtering with regularization is given by

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

regularization must now add the code you've implemented to calculate the cost function J. When you have it ready, you can check if it is correct by ex2_cofi script, which runs the cost function. The value that should appear to be close to 31.34.

2.4 regularized Gradiente

Now you must implement the gradient of the regularized cost function. Must add the contributions of regularization terms X_grad and Theta_grad as indicated in formulas gradients function regularized cost

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

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

After completing the code, you can check if it works correctly by the script ex2_cofi.

3 Run the algorithm

After implementing the cost function and gradient, we will train the algorithm and generate recommendations to him. In the next part of the script ex2_cofi, you can enter your own ratings of some films, and when the algorithm is executed, will give personalized recommendations. There is a set of films and scored in ex2_cofi, but only as an example. Modifies the films scored in ex2_cofi according to your own preferences. You can rate the movies you want in the 1682 dataset. The list of films and their number within the dataset are listed in the file movie_idx.txt.

Optional 4 Challenge: set the regularization parameter

Enter a code to evaluate different values of the parameter lambda. You can use the following values for lambda: [0, 0.03, 0.1, 0.3, 1, 3, 10, 30, 100]. For evaluation, divide the array of input data and two submatrices: Ytrain and Ytest where Ytest contains approximately 10% of the scores of Y, and Ytrain the remaining 90%. Ytrain be used to learn X and Theta with each lambda, and Ytest to evaluate the performance of the X and Theta learned. As a performance must use the square root of the mean square error:

$$RMSE = \sqrt{\sum_{(i,j):r(i,j)=1} \frac{(y^{(i,j)} - \hat{y}^{(i,j)})^2}{C}}$$

where C is the number of scores in Ytest (to evaluate only the films are considered in Ytest)