A

Report on

Recommender System for Recommending Courses to Students based on feedback ratings

Prepared for Machine Learning BITS F464

Submitted By:

RONIL PANCHOLIA	2012C6PS629P
RITWIK SAHANI	2012C6PS686P
KARAN KATYAL	2012A7PS026P

Table of Contents

1. Introduction

The goal of a Recommender System is to generate meaningful recommendations to a collection of users for items or products that might interest them. Suggestions for books on Amazon, or movies on Netflix, are real world examples of the operation of industry-strength recommender systems. The design of such recommendation engines depends on the domain and the particular characteristics of the data available.

Two basic type of Recommender Systems are :

1. Content Based Recommendations
    - The user will be recommended items similar to the ones the user preferred in the past.
2. Collaborative Filtering
    - The user will be recommended items that people with similar tastes and preferences liked in the past.

Collaborative filtering methods are based on collecting and analyzing a large amount of information on users' behaviors, activities or preferences and predicting what users will like based on their similarity to other users. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as a course without requiring an "understanding" of the course itself.

On similar lines, we thought of developing a Recommender System with Collaborative Filtering for courses offered in BITS based on interests of individual student captured through a feedback taken for every course that they have done. To normalize the feedback score, we will multiply it by the percentage of classes that a student has attended, because students who have attended more classes tend to have a better opinion of the course. We will try to find the ratings of the courses that the student has not completed and recommend the courses with highest predicted ratings.

The key feature of our Recommender System is that it predicts the ratings of courses for a student based on the courses previously rated by other like-minded students without having an understanding of the course itself.

2. Data collection

We required a feedback rating for every course by students who have completed each one of the 15 CS/IS courses we took into consideration.
This feedback ratings were integer values ranging from 1 to 10.
We collected the data by online surveys using Google forms. The form was circulated to various students of third year or higher who had done some courses related to computer science field. The data was then checked for any bogus entries which were not likely to be filled by the student himself. Data was collected in a format where the students rates all the 15 courses listed in the form and marks 0 for the courses he/she has not done.

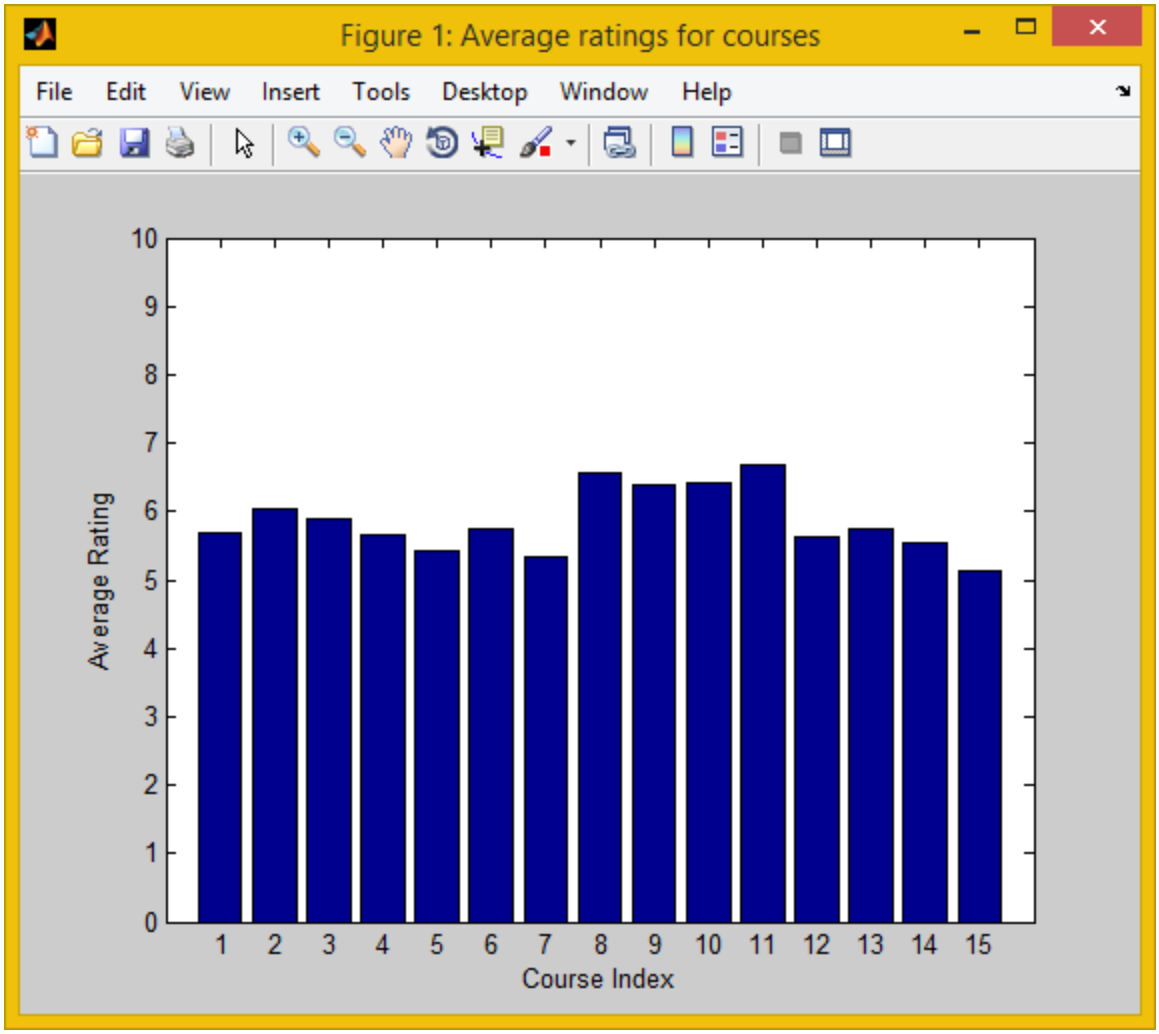

Figure 1 : Average Rating for each of the 15 courses in our data set

3. Modelling

The basic aim of Collaborative Filtering algorithms is establishing similarities between the interests of different students based on the ratings that they have provided for the courses that they have done. This allows us to recommend courses to students on the basis of likings of other “similar students”.

We will be modelling our Collaborative Filtering Recommender System with k-NN.

3.1 k-NN or k-Nearest Neighbours

k-NN (which is also called the *memory-based approach*) utilizes the entire user-item database to generate predictions directly, i.e., there is no model building.

k-NN approach can be user-based and item-based. We will be using **item-based k-NN approach**.
The item-based k-NN approach works by comparing items based on their pattern of ratings across users.

To apply k-NN in our problem, we need to define a similarity measure to measure the “distance” between each item. This is the critical step in the item based collaborative filtering algorithm.

We will be using Correlation-based similarity function to measure the similarity between two courses.

$$sim(i,j) = \frac{\sum_{u \in U} (R_{u,i} - \overline{R_i})(R_{u,j} - \overline{R_j})}{\sqrt{\sum_{u \in U} (R_{u,i} - \overline{R_i})^2} \sqrt{\sum_{u \in U} (R_{u,j} - \overline{R_j})^2}}$$

$R_{u,i}$ denotes student u’s rating for course i.

R_i is the average rating for course i.

4. Training the Model :

For applying collaborative filtering algorithm we will follow the following 3 steps :

1. Initialize $X_{(1)}, X_{(2)}, X_{(3)}$ and $\theta^{(1)}$, $\theta^{(2)}$ … $\theta^{(n)}$,where n denotes the number of students to random small values.

2. Optimize the value of X and the cost function $J(X_{(i)}, \theta^{(n)})$ using fmincg with update rule,

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i=1}^{m} ((\theta^{(j)})^T X^{(i)} - Y^{(i,j)}) X_k^{(i)} \quad \text{(for k=0)}$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i=1}^{m} ((\theta^{(j)})^T X^{(i)} - Y^{(i,j)}) X_k^{(i)} + \lambda \theta_k^{(j)}) \quad \text{(for k≠0)}$$

3. For a student with parameters , θ and a course with learned features X, predict a feedback rating of $\theta^T X$.

5. Prediction of Ratings :

Given values of $C^{(1)}$, $C^{(2)}$,… $C^{(m)}$ (and course rating) , we can estimate $\theta^{(1)}$, $\theta^{(2)}$ .. $\theta^{(n)}$

Given values of $\theta^{(1)}$, $\theta^{(2)}$ … $\theta^{(n)}$, we can estimate $C^{(1)}$, $C^{(2)}$,… $C^{(m)}$.

Initially, we guess θ and estimate X , then we use that value to estimate θ again.

Therefore, iterating between θ and X .

$\theta \rightarrow X \rightarrow \theta \rightarrow X \rightarrow \theta \rightarrow ...$

This can be done by taking an objective function which can minimize both θ and X simultaneously.

So, our final Collaborative Filtering optimization algorithm (including Regularization) is :

$$\underset{\theta}{min}\ \frac{1}{2}\sum_{j=1}^{m}\sum_{i}((\theta^{(j)})^{T}x^{(i)} - y^{(i,j)})^{2} + \frac{\lambda}{2}\sum_{j=1}^{m}\sum_{k=1}^{n}((\theta_{k}^{(j)})^{2}$$

6. Testing Metrics

We have divided the given data set into 3 categories training data, testing data and cross validation data.

Training set - It is a set of data to discover a potential predictive relationship.It is almost 70% of the data collected.

Testing set - It is a set of data used to assess strength and utility of a predictive relationship. It consists of almost 15% of the data collected.

Cross validation set - It is a model validation technique for assessing how the results of a statistical analyzer will generalize to an independent dataset. We obtain a matrix that would do a similar task. It consists of the remaining 15% of the data collected.

We will be evaluating our Recommender System on three metrics.

1. Root Mean Squared Error
2. Precision and Recall
3. F1-Score

6.1 Root Mean Squared Error

As RMSE, the Root Mean Squared Error is the most widely used evaluation metric for Recommender Systems, we will be using RMSE to evaluate our Recommender System.

RMS errors are calculated using the formula :

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(\overline{y_t} - y_t)^{2}}{n}}$$

For our recommender system, **RMS Error = 1.0176.**

6.2 Precision and Recall

To use these metrics, recommender system must convert its ratings scale into a binary {Do not recommend,Recommend} scale. Items for which the prediction is to recommend are shown to the user, other items — are not shown. Since our dataset was small, we have done this transition for testing purposes and calculated Precision and Recall.

We have calculated the following values on 30% of our dataset and trained it on the other 70%.

	Recommended	Not Recommended
Preferred	True Positives = 75	False Negatives = 27
Not Preferred	False Positives = 6	True Negatives = 180

Table 1: Classification of the possible result of a recommendation of a course

Precision is the fraction of all recommended items that are relevant.

$$Precision = \frac{\#True\ Positives}{\#True\ Positives + \#\ False\ Positives} = 0.925$$

Recall is the fraction of all relevant items that were recommended.

$$Recall = \frac{\#True\ Positives}{\#True\ Positives + \#\ False\ Negatives} = 0.735$$

6.3 F1 - Score

The F-Score or F-measure is a measure of a statistic test's accuracy. It considers both precision and recall measures of the test to compute the score. We could interpret it as a weighted average of the precision and recall, where the best F1 score has its value at 1 and worst score at the value 0.

$$F_1 = \frac{2 \,.\, Precision \,.\, Recall}{Precision + Recall} = 0.819$$

7. Addressing Overfitting

Overfitting generally occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.

Options to handle overfitting **:**

- Reduce number of Features
- Regularization

7.1 Reducing the number of Features

Using different number of features will decide on how many values, the similarity between courses are being compared. Taking very less features will lead to Underfitting and poor prediction of further data. However, taking too many features will cause Overfitting and lead to high generalization errors.

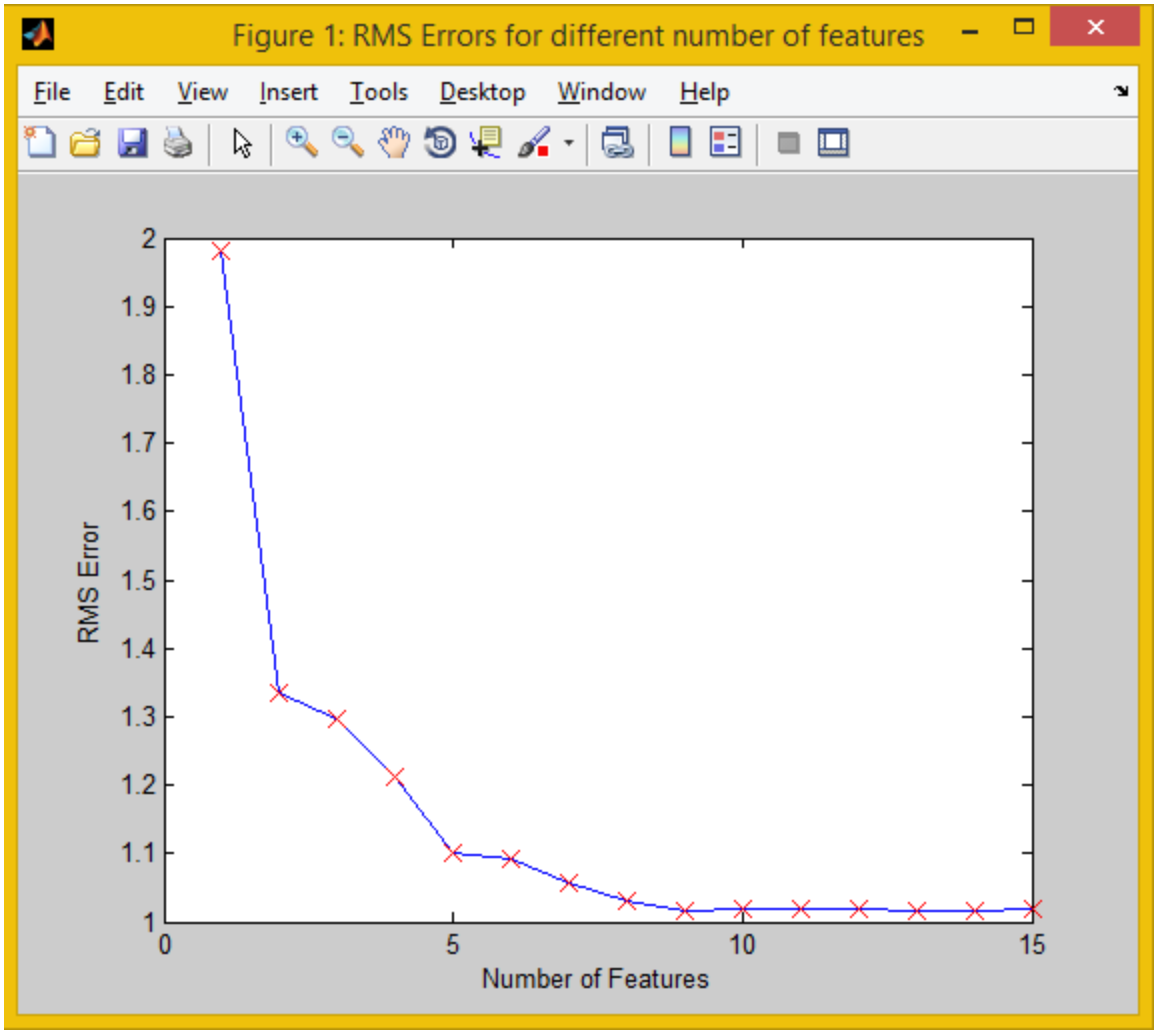

Figure 2 : RMS Errors for different Number of Features

The above figure shows the RMS Errors for different number of features on our dataset of 80 students. As we increase the number of features, the error decreases.

As it is clear from the graph in Figure 2, after **number of features = 9**, the decrement in RMS error is not significant enough and increasing number of features further would make our model suffer with the **Curse of Dimensionality** and involve expensive computations. Thus, we will use number of features as 9. So, we cannot reduce the number of features any more. To handle the overfitting because of 9 features, we will need to use Regularization technique.

7.2 Regularization

Regularization, refers to a process of introducing additional information in order to solve an ill-posed problem or to prevent overfitting. Regularization methods are used for model selection, in particular to prevent overfitting by penalizing models with extreme parameter values.

Keep all the features, but reduce magnitude/values of parameters. Works well when we have a lot of features, each of which contributes a bit to predicting as is happening in our case which can be seen in Figure 1 above.

The Minimization

$$min \left| Y_i - f(X_i) \right|^2$$ may be obtained with zero errors.

But the function may not be unique.

Regularization :

$$min_{f \in H} \sum_{i=1}^{n} \left| Y_i - f(X_i) \right|^2 + \lambda ||f||_H^2$$

Regularization with smoothness penalty is preferred for uniqueness and smoothness of the curve.

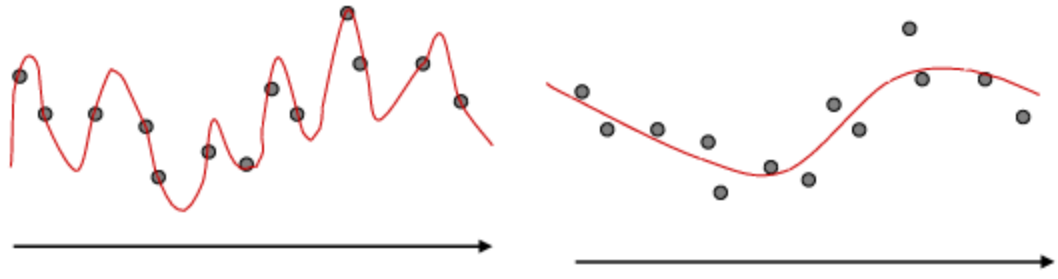

Figure 3 : Non- regularized and regularized curves

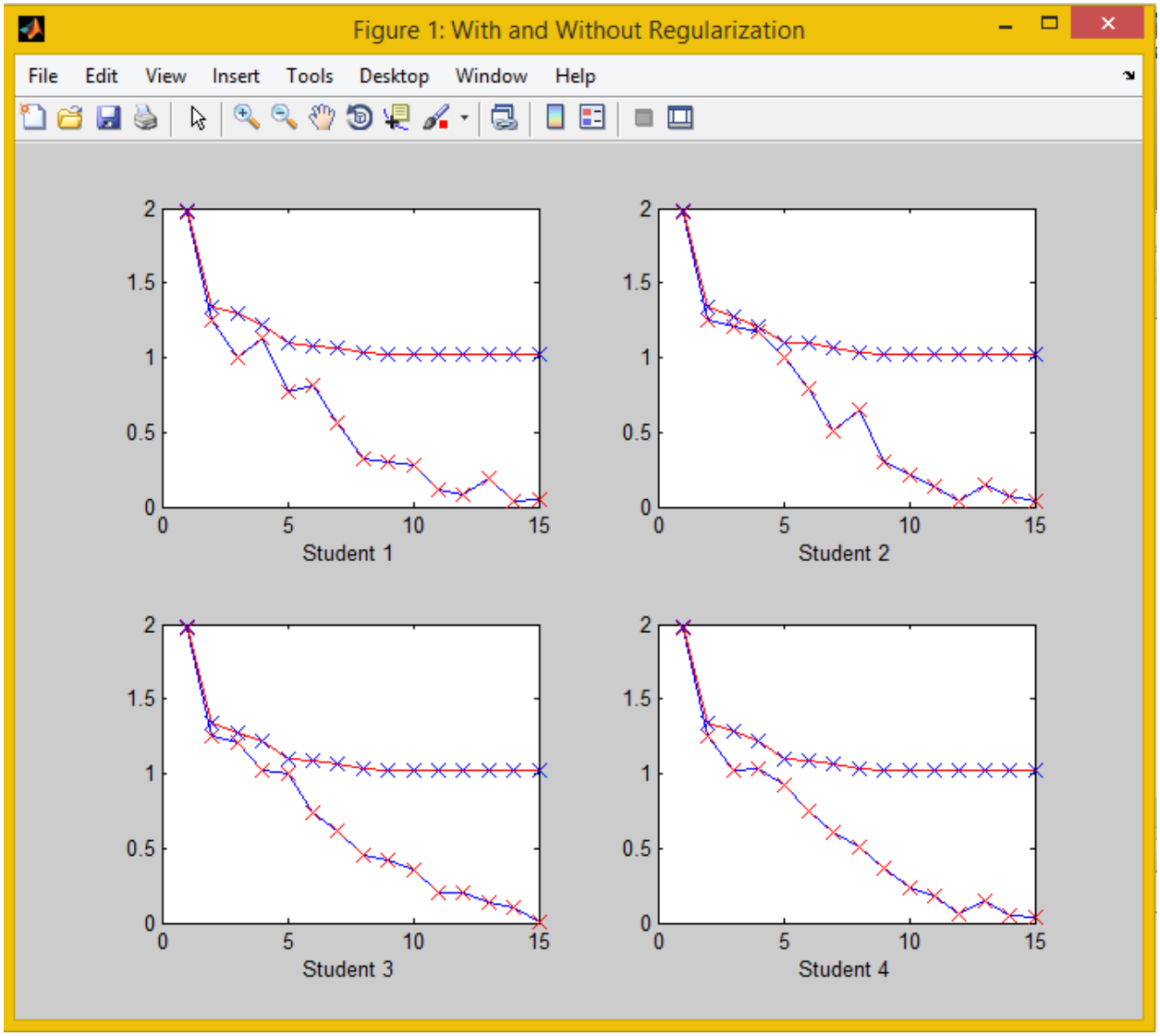

Figure 4 : Curves With and Without Regularization

The above figure shows four different plots taken with four different values of new student ratings. The red curve denotes the values without Regularization (with regularization parameter, $\lambda = 0$) and the blue curve denotes the values with Regularization (regularization parameter, $\lambda = 3.2$).

It is clear that the regularized curve is much smoother than the non regularized curve and is more general when multiple students are considered.

So, it means that the **Regularized curve provides a Lower Generalization Error** than Non-Regularized curve, hence solving the problem of overfitting.

Summary

Through working on this project, we have gained a useful insight in the process of implementation of Recommender Systems.

The data set that we had did not have sufficient students who made entries (only 80 students). If the feedback ratings for courses are recorded every semester for every course through some formal measure such as the feedback form by Instruction Division, we will easily get a large dataset with students from all years which will help our model in predicting ratings with an higher accuracy.

Some of the useful values we obtained by doing some analysis are :

Regularization Parameter $\lambda = 3.2$

Number of features for least error = 9

Root Mean Squared Error = 1.0176

Precision = 0.925

Recall = 0.735

F1-Score = 0.819

As it can be seen in Figure 1, the average ratings for all courses are between 5 and 7 but the actual ratings are usually extreme, the variance is high and thus we get a relatively large error. This has also lead to prediction of most ratings near this range.

As a further exploration, we can add more courses to our data set from all the disciplines in BITS and make a more general Course Recommender for all students.

We can also try using some other approaches to build a Recommender System and compare accuracy of those methods. Some of these approaches could be based on Singular Valued Decomposition, K-Means Clustering, Naive Bayes etc.

References

We have use some content from the following references :

1. **Recommender Systems - Wikipedia**

http://en.wikipedia.org/wiki/Recommender_system

2. **Fmincg function**

http://doc.fabiocigliano.net/Progetto_OCR/html/fmincg_8m.html

3. **Evaluating Recommendation Systems**

by Guy Shani and Asela Gunawardana

http://research.microsoft.com/pubs/115396/EvaluationMetrics.TR.pdf

4. **Recommender Systems Handbook**

by Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kantor

http://www.cs.bme.hu/nagyadat/Recommender_systems_handbook.pdf

5. **Content-based Recommender Systems: State of the Art and Trends**

http://facweb.cs.depaul.edu/mobasher/classes/ect584/Papers/ContentBasedRS.pdf

6. **Empirical analysis of predictive algorithms for collaborative filtering**

http://dl.acm.org/citation.cfm?id=2074100

7. **Nearest Neighbor Pattern Classification**

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1053964

8. **A collaborative filtering algorithm and evaluation metric that accurately model the user experience**

http://dl.acm.org/citation.cfm?id=1009050