CSG230: Data Mining Techniques
Fall 2008

Data mining project

Copyright for this document:

This document is based on the original version from Stefan Savev, modified by Jian Wen.

Motivation:

The purpose of recommendation systems (also known as *collaborative filtering* systems) is to recommend items which a customer is likely to order. These recommendations are based on statistics of a customer's purchasing history and the histories of customers with similar interests. Examples of recommendation systems can be found at amazon.com and netflix.com. Apart from the usefulness consideration, you also have a personal motivation to work on this project because netflix offers a million dollars[1] to the person(s) who can beat their baseline system by 10%. So, don't wait any longer. Start coding!

Goal:

In this project you will implement a movie recommendation system which will work on a subset of the data provided by netflix.com for the purpose of their Netflix prize competition (http://www.netflixprize.com/).

Method (in principle):

The methods used for building recommendation systems rely on machine learning (data mining, statistical inference) techniques. This means that the program is provided with data from the past[2] from which the program should learn to predict the future. Machine learning programs typically work in the following way:

- **Algorithm Design Phase**: One designs a model which can "explain" or fit the data. Usually the designer restricts the model to a specific class of models (for example: decision tree, neural network, probabilistic models etc.) and makes assumptions during the process. Normally, the model has a set of parameters

[1] This may be a much easier way to win a million dollars than solving the P not equal to NP problem from Theoretical Computer Science. It may also be a much easier way to make a profit of a million dollars if you are a start-up company (a few start-up companies among the best participants in netflix leaderboard).

[2] in the present case those are records (movie_id, user_id, user_ranking, ...). The meaning of the record is that user with id = user_id gave user_ranking "stars" to movie with id = movie_id.

whose values are not specified but are obtained by optimizing a certain cost/loss function. The function that is optimized is application dependent but is usually the training error. In our case the criterion function is the so called root mean squared error (RMSE) which is connected to the standard deviation of the predictions.

- **Training phase**: The program preprocesses the data and estimates the parameters, if any (not all learning methods need to have parameters that are optimized; for example nearest neighbor classifier does not have a training phase).
- **Tuning phase**: In that phase one tests the predictions of the model on the tuning set[3]. If the quality of the results is satisfactory the model is run on the test set and evaluated.
- **Testing phase**: When the model achieves good results from the tuning phase, one can proceed to run the model on the test data. It is important not to use the test data to tune the model's performance. By using the test data too many times one might adjust her model to peculiarities of the test set and obtain results that will not generalize to another test set. This rule is known as "do not train on the test data" and students should not violate it.

Data:

The data on which your program will run consists of the following files[4]:

- **training_set.txt**: Contains the training data.
- **tuning_set.txt**: Contains data which you can use to test your model in the process of its development.
- **tuning_set_anwers.txt**: Contains the true rankings of the tuning set.
- **test_set.txt**: Once you are satisfied with the quality of your model (your RMSE score on the tuning set), run the model on the test set and submit the results via the web interface.
- **movie_titles.txt**: The dates and titles the movies (as provided by netflix)
- **movie_details.xml**: details such as movie director, actors, etc.
- **rmse.pl** a Perl script that reports RMSE[5]. Run the Perl script *./rmse.pl predictions tuning_set_anwers.txt* where predictions is a file you should produce from tuning_set.txt.

Those files (except movie_details.xml) are extracted from the original netflix data [6]

[3] Netflix has called those the probe set and the qualifying set respectively.

[4] The files are different than the ones you get from netflix.

[5] For definition of RMSE consult the Perl script.

[6] You can obtain the original data from http://www.netflixprize.com/teams but it is not necessary for the purpose of the project.

The file movie_details.xml is extracted from www.imdb.com and contains the genre of the movie, the name of the director, the list of actors, etc.

There are 9058 movies and 477293 users in our dataset. Each movie is identifiable by movie id ranging from 1 to 17770. The movie id's are not contiguous. The user id's start from 1 and are not contiguous too. Detailed descriptions of the files follow:

training_set.txt: This file is a list of records
(movie_id, user_id, date, ranking) which should be interpreted as *user_id* gave ranking stars to *movie_id* on date *date*. The ranks in the training and test sets are integers from 1 to 5 but you may predict real numbers from 1.0 to 5.0.

tuning_set.txt: This file is a list of records:
(movie_id, user_id).

tuning_set_answers: This file is a list of records
(movie_id, user_id, true_ranking).

predictions: This is a file you should produce (the name is up to you) which should be in the following format
(movie_id, user_id, your_ranking).

movie_details.xml: This file contains information extracted from www.imdb.com. Only the main page of a movie was extracted. The implication is that, for example, only the main actors will be listed. If you are interested you may extract more information from the web site or by using some of the provided interfaces (see http://www.imdb.com/interfaces). You can use this file to obtain features which may be used either for direct prediction ("if a movie has that actor ... rank it with 4 stars") or to cluster the movies. Based on the clustering you may be able to better predict users for which there is little data (i.e. smoothing). It is up to you how you will use the data. Here is how the file looks like. Note that here the file is not a real XML file, but just with the similar hierarchy.

```
<movies>
<movie>
        <id>1</id>
        <imdb_url>http://www.imdb.com/title/tt0389605/</imdb_url>
        <title>Dinosaur Planet</title>
        <cast>
                <actor>Christian Slater</actor>
                <actor>Scott Sampson</actor>
        </cast>
        <directors>
        </directors>
        <writing_credits>
        </writing_credits>
        <genre>
                <single_genre>Documentary</single_genre>
                <single_genre>Animation</single_genre>
                <single_genre>Family</single_genre>
        </genre>
        <votes_on_imdb>7.4/10</votes_on_imdb>
        <number_of_votes_on_imdb>68</number_of_votes_on_imdb>
        <awards>Won 2 Emmys.
 Another 4 nominations</awards>
        <keywords>
        </keywords>
        <has_poster>True</has_poster>
        <run_time></run_time>
        <sound></sound>
        <year_on_imdb>2003</year_on_imdb>
        <color>Color</color>
        <country>USA</country>
        <language>English</language>
        <plot>A four-episode animated series charting the adventures of four dinosaurs - each on a
different continent in the prehistoric world: a lone female Velociraptor in Asia; a young male
Daspletosaurus in North America; a South American female Saltasaur; and a young adult Pyroraptor in
Europe. Narrated by Christian Slater and hosted by paleontologist Scott Sampson.</plot>
</movie>
...
</movies>
```

Your task:

1. Your task is to **design two models** (algorithms) (or obtain a model from literature[7]) that work well on the movie recommendation task. These models should be non-trivial ones, where a trivial model is one that simply predicts averages. The model that you implement also should have RMSE score that is lower than the RMSE's produced by trivial models.

Your grade on the project will reflect the amount of novelty in your work. In other words implementation of model from literature gets less points that an algorithm that you devised[8]. If you are influenced by or modify a previous work you should acknowledge it. Before starting you may want to review some of the existing literature. Search for collaborative filtering on www.google.com or http://scholar.google.com/. The following paper http://ccs.mit.edu/papers/CCSWP165.html may provide a good starting point since it presents a system that is relatively simple.

The **first algorithm** should not use the **movie_details.xml** file and similar film-specific information**.** The **second algorithm** should use **movie_details.xml** or similar

[7] This means that you should understand the model and implement it. You should not use source code that implements a recommendation system and just run it. You can use general purpose dataming / machine learning packages, for example WEKA (http://www.cs.waikato.ac.nz/ml/weka/)

[8] If you devise some parts yourself, for example a special feature representation but use a general purpose classification or regression technique I will regard that work as novel. Similarly, if you manage to combine existing collaborative filtering methods/ideas in a new way, I will regard your work as novel.

information[9]**.** While the first task is known as collaborative filtering, the second is known as content-based collaborative filtering since it uses content information from the items (movies). In your report you may want to address the question whether using content from the movies helped your predictions.

2. The next task is to **implement your algorithms** and tune them. When you are ready run them on the test set and **submit (upload) your results (at least one submission for each algorithm)** via a web interface. I will send you an email with instruction when I am ready.

3. **Prepare a report** that describes what you did. The report will preferably be organized in the following way:

- **Introduction**: Describe the problem of collaborative filtering. Describe some existing approaches. Successful unification and classification of existing approaches will be favored in the grading.
- **Theoretical Discussion**: Describe sufficiently well two methods (either your own[10] or from literature) for collaborative filtering:
  - one of the methods should work without using information (i.e. director name, actors, etc.) about the movies (i.e. should not use movie_details.xml)
  - the second method should use movie_details.xml
  - Try to provide some theoretical justification why you expect those methods to work. Discuss any modeling assumptions as well as practical "hacks" you made in your model.
- **Results:** Report the root mean squared error (RMSE) on the tuning and test sets for both methods. Report the values of the main parameters of your models that you used. If you have studied how the RMSE depends on a particular parameter you may provide extra tables and plots. Investigation of parameter effects will be favored in grading.
- **Discussion and error analysis**: Try to interpret the results of your model. Did content based analysis help? On what sets of movies and users did your model make the most errors? Which rankings are most difficult to predict? What about the users who ranked only one movie? Are some assumptions that you made in your model not satisfied in practice. Might this have hurt the model's performance? Automatic clustering of the movies might be interesting. This is neither a required nor an exhaustive list of questions. In the process of formulating and testing your model you may ask yourself other questions. You can discuss them here.
- **Appendices:**

[9] If you are not satisfied with movie_details.xml file you can go to www.imdb.org and extract all the information you think is relevant for your task.

[10] As I mentioned above devising an algorithm yourself will be favored in grading to implementing an existing method.

    - Provide an appendix with the relevant portions of your code. Input-output routines should not be included.
    - Provide a list of any external tools that you have used.
    - If you use an existing technique, for example decision tree, Support Vector Machine, EM-algorithm, LSA etc., provide an appendix that shortly (less than 1 page) describes that technique.

- **References:** If you are using information or tools from other sources apart from your own you should reference them. Avoid copying pieces of text from books or papers. If you have to do it, paraphrase and reference it.

Length of report: Your report should not exceed 15 pages. Only content but not length of report will be considered in grading. The report should be typed. The report should be submitted in hard copy on the due date.

Submit your code via the web interface. Put your code into a tar.gz or zip file and upload it via the web interface. Put a read me file that says what I should do in order to run the code and reproduce your results. There are no restrictions on the use of languages (i.e.Perl, C++, etc.) and tools (i.e. Matlab, WEKA, etc.) as well as operating systems (Windows, Linux, MAC, etc.). If I decide to reproduce your results and I cannot do it (because I lack software that you have) you should be ready upon request to demonstrate that your software can run and produce the reported results.

Want to implement more methods: In the course of doing the project you may find yourself implementing and experimenting with more that one method. You can report those results only if they are interesting, i.e. the methods are substantially different that other methods on which you report *or* produce good results. Remember you should have at least one algorithm in the two categories mentioned (using and not using content information from the movies). Extra credit may be given if you experiment with more methods.

Check list for deliverables:

- **Two files with results**
- **Report**
- **Source code**
- **Readme file that tells us how to run your code**

Grading:

The grading will be based solely on the report you provide. **Try to make the report representative of your efforts**. Even if you cannot obtain good results in terms of RMSE you should report what you did[11]. The submission of result files and source code are for the purpose of checking on the truthfulness of your report and not for grading purposes. Naturally, grading a piece of written text is subjective but points will be assigned as follows:

Report Section	**Max. points**
Introduction	**10**
Theoretical discussion of method 1 (non-content based method)	**20**
Theoretical discussion of method 2 (content based method)	**20**
Results on method 1	**20**
Results on method 2	**20**
Discussion	**30**
Appendices and references	**10**
Total	**130**

Extra credit (up to 20% from the regular credit) may be given for especially good sections of the report or experimentation with more algorithms than required**.**