ud741 Project Description

Project Instructions

Overview

In this project, you will formulate a recommendation engine framework by running the *Pearson Correlation Similarity Measure* on a given dataset of movie recommendations, and then refine your predictions using clustering algorithms.

- In Part 1, you will use your programming skills to produce a utility matrix predicting the ratings that users will give to items. You can verify your predictions by submitting them in this matrix, and by finding the Mean Squared Error of your test set, which you can verify here (see Part 1 Instructions).
- In Part 2, you will use clustering algorithms to refine your utility matrix. You will have the chance to test your predictions, and see if your score passes the threshold here (see Part 2 Instructions).

Assessment

Parts of this project will be automatically verified through the

Udacity autograder, but the majority of the project will be self-assessed using this rubric.

Part 1: Collaborative Filtering

Recommendation engines generally use two different techniques to make predictions:

1. Content-Based Systems examine properties of the items recommended. For example, if a Netflix user has watched many sci-fi movies, then recommend a movie from the "sci-fi" genre.
2. Collaborative Filtering Systems recommend items based on similarity measures between users and/or items. The items they recommend to a user are drawn from those preferred by similar users. For example, if an Amazon shopper places a toy robot in their cart, then a robot-themed Lego set is recommended because other shoppers who liked the toy robot also liked the Lego set.

We will use Collaborative Filtering to build our recommendation engine.

We will start by mathematically formulating a recommendation engine framework. In a recommendation-system application, there are two classes of entities, *users* and *items*. Users have preferences for certain items and these preferences must be discovered from the data. The data is represented as a *utility matrix*, a value that represents the rating given

by that user for that item and is given for each user-item pair. The ratings can take a value between 1 to 5, representing the number of stars that the user gave as a rating for that item.

We will assume that the matrix is sparse, meaning that most of the entries are unknown.

Here is an example of a utility matrix, depicting four users: Ann, Bob, Carl and Doug. The available movie names are HP1, HP2 and HP3 for the Harry Potter movies, and SW1, SW2 and SW3 for Star Wars episodes 1, 2 and 3.

Table 1. An example Utility Matrix for Movie Recommendation (you can fill this out here)

	HP1	HP2	HP3	SW1	SW2	SW3
Ann	4			1		
Bob	5	5	4			
Carl				4	5	
Doug		3				3

The goal of the recommendation engine is to predict the blanks in a utility matrix. For example, how *will Ann rate SW2?*

In order to make these predictions, we must first measure similarity of users or items from the rows and columns of the Utility Matrix. The data is too small to draw any reliable conclusions. Examine the preferences of Ann and Carl. They both have rated the same movie but have diametrically opposite opinions on it. We would expect that a good distance measure would set them far apart. We can use different similarity measures for this task, but in this project we will use the *Pearson*

Correlation Similarity Measure.

Let $r_{x,i}$ denote the rating given by user x to item i. If I is the set of all items that two users x and y have rated, then the Pearson Correlation Similarity Measure between the two users is given by

$$pcs(x,y) = \frac{\sum_{i \in I}(r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sum_{i \in I}(r_{x,i} - \bar{r_x})^2 (r_{y,i} - \bar{r_y})^2}$$

,

where $\bar{r_x}$ denotes the average rating given by user $x$to all items. To calculate $r_x$we only consider items that were rated by the user.

We still need to take an additional step to recommend items to users. One way of predicting the value of the utility matrix entry of a given user, *U*, and a given item, *I*, is to find the *n* number of users most similar to *U* and average their ratings for item *I*, counting only those among the *n* similar users who have rated *I*.

It is generally better to normalize the matrix first. That is, for each of the *n* users, subtract the average rating for all items from the rating of each item, i. Average the difference for those users who have rated *I*, and add this average to the average rating that *U* gives for all items. This normalization adjusts the estimate in the case that *U* tends to give very high or very low ratings.

Part 1 Instructions:

1. Fill the Utility Matrix in Table 1. Use the following code template to get started. You can verify your answer by submitting them here.
    a. Complete the

function `pcs(x, y)`. It finds the Pearson Correlation Similarity Measure between two users.
b. Complete the function `guess(u, i, top_n)`. It finds the best guess of a rating for user u and item i, if we look at the top_n users similar to user u.

2. Following is the test set given to you.. Report the Mean Squared Error on this test set. You can verify your answer by submitting them here.
    a. Ann rates SW2 with 2 stars
    b. Carl rates HP1 with 2 stars
    c. Carl rates HP2 with 2 stars
    d. Doug rates SW1 with 4 stars
    e. Doug rates SW2 with 3 stars
3. Can we improve the accuracy of our predictions if we work from similar items instead of similar users? What are the pros/cons of this approach? Explain in 4-5 sentences.
4. (Optional) Cluster similar items using Pearson Correlation Similarity Measure and report the new Utility Matrix. Test your predictions against the test set and find out if clustering items can improve your prediction.

Part 2: Clustering Users and Items

In this part of the project, you will be using the MovieLens dataset to create a utility matrix. Each of the users and items will be defined in the dataset itself. You should download the part2.py template to get started. Check the entries in the data/README for finding out how the data is given to you.

It is difficult to detect the similarity between items and users because we have little information about user-item pairs in this sparse utility matrix. Even if two items belong to the same genre, they are likely to be very few users who bought or rated both. Similarly, even if two users both like a genre or multiple genres, they may not have bought any items to indicate that.

One way of dealing with this lack of data is to cluster items and/or users. In this example, we will cluster similar movies.

	HP	SW
Ann	4	1
Bob	4.67	
Carl		4.5
Doug	3	3

After clustering the items, we can revise the utility matrix so the columns represent clusters of items and the entry for User *U* and Cluster *C* is the average rating that *U* gave to the members of Cluster *C* that *U* did not individually rate. Note that *U* may have rated none of the cluster members, in which case the entry for *U* and *C* is left blank.

Part 2 Instructions:

1. Download the MovieLens Dataset (Training Set, Test Set).
2. Cluster similar items and create a utility matrix. Use k-means clustering and then the Expectation Maximization (EM) algorithm. Try difference values of k for k-means or choosing different number of gaussian mixtures for the EM algorithm.
3. Fill the new utility matrix using the Pearson Correlation Similarity Measure.
4. Test the ratings you get with the test set. Remember that for each user, you will first have to find out which cluster they belong to and then find their ratings.
    a. Use the u.test dataset to test your modified implementation with clustering. The u.test file contains user, items pairs and ratings given by them. So you can use the mean squared error metric to verify performance of your recommender system.
    b. If you would like to challenge your recommender system further, download the predict.test dataset here (in the instructor notes).

This dataset has user, item pairs but not ratings. You have to predict ratings and submit your answers in the form of *userid, itemid, rating* rows in the text box. It will return you a score. Your score should be higher than 3.589.

5. Did using the clustering algorithms improve accuracy of your prediction? Explain briefly in 4-5 sentences. For example, if you are using k-means for clustering, graph a plot of performance vs k. If you used EM algorithm, show how the performance changed when you varied parameters of your Gaussian mixture model.
6. (Optional) Make a user interface, on the terminal or otherwise, where you will ask a user to rate 10 movies and then recommend 5 movies they should watch.