	Matrix	Titanic	DieHard	ForrestGump	Wall-E
John	5	1		2	2
Lucy	1	5	2	5	5
Eric	2		3	5	4
Diane	4	3	5	3	

Table 1: Example ratings

0.1 Idea of *similarity*

Let's say the users of a website like Imdb [TODO FIXME] rate the movies on a scale of 1 to 5, where a rating of 5 implies the user really liked the movie and a rating of 1 implies the opposite. This data can be stored in the from of a *utility matrix* where each row of the matrix corresponds to a user and each column corresponds to a movie. Consider an example rating set shown in Table [TODO FIXME] that can be represented in the form of utility matrix *utility matrix* U as shown in equation [TODO FIXME]. This 4x5 matrix stores the ratings given by 4 users to a set of 5 movies. As expected, not every user would have rated each of the 5 movies. The user-movie combination for which a rating does not exist is indicated by a 0.

$$U = \begin{bmatrix} 5 & 1 & 0 & 2 & 0 \\ 1 & 5 & 2 & 5 & 5 \\ 2 & 0 & 3 & 5 & 4 \\ 4 & 3 & 5 & 3 & 0 \end{bmatrix} \tag{1}$$

EXERCISE: Look at the ratings of user *Eric* in Table 1. Also look at the ratings given by other users in the table. Can you guess which user(s) are similar to user *Eric*?

0.2 Similarity weight computation

The example data set in Table 1 is very small in size for which finding users that are *similar* to the user of interest might be possible. In real data sets, we need define a measure of *similarity* that can be used to find the items/users that are *similar* to each other. The computation of of the similarity weights is one of the most critical aspects of building *neighborhood-based* recommendation systems, since it can have a significant impact on the performance and accuracy of a system.

Cosine Similarity: We apply the traditional notion of Cosine Vector (CV) similarity to find the similarity between two users u and v. If $\mathbf{x_u}$ is vector of ratings r_{ui} of a user u, then the cosine similarity of users u and v can be

	Matrix	Titanic	DieHard	ForrestGump	Wall-E
Eric	-1.5	0	-0.5	1.5	0.5
Diane	0.25	-0.75	1.25	-0.75	0

Table 2: Mean centered ratings

calculated using the equation [TODO FIXME].

$$CV(u,v) = cos(\mathbf{x}_u, \mathbf{x}_v) = \frac{\sum_{i \in I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I_u} r_{ui}^2 \sum_{j \in I_v} r_{vj}^2}} \tag{2}$$

where I_u represents the set of ratings for the movies that have been rated by both users.

EXERCISE2: In the example data set of Table [TODO FIXME], find the users that are most similar to user Eric. You can use the function *CosSimVecMatrix* provided to generate the similarity of user *Eric* with other users.

Hopefully that was easy and your have the answer - users that seem most similar to *Eric* are *Lucy* and *Diane*. That was easy. Let's look at the ratings of *Diane* who seems to be *similar* to *Eric*. If we calculate the means of ratings of *Eric* and *Diane*, 3.5 and 3.75, and subtract the respective means from their respective ratings, the new ratings look like as shown in Table 2.

Positive values in Table 2 represent a preference for the movie by the user since it's rating is more than the average, whereas a negative value implies a disliking for the movie. On analyzing these *mean-centered ratings*, *Eric* and *Diane* don't seem to be so *similar* anymore. In fact, they seem to have quite the opposite taste.

Pearson Correlation Similarity: A major flaw in using *Cosine Similarity* to find the *similarity* between users is the fact that *Cosine Similarity* does not take into account the differences in the mean and variance of the ratings made by the users. *Pearson Correlation* (PC) similarity is a popular measure that can be used to compare the ratings of users since it removes the effects of mean and variance in ratings while calculating the *similarity* between users. The PC similarity between users u and v can be calculated using the equation [TODO FIXME].

$$PC(u,v) = PC(\mathbf{x}_u, \mathbf{x}_v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r_u})(r_{vi} - \bar{r_v})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r_u})^2 \sum_{i \in I_{uv}} (r_{vj} - \bar{r_v})^2}} \tag{3}$$

where $\bar{r_u}$ [TODO FIXME] represents the mean of the ratings given by user u. Note that PC accounts for variance in ratings only for the ratings of the movies I_{uv} that have been rated by both users u and v. The sign of similarity weight

calculated using PC indicates whether the correlation between the two users is direct or inverse, and the magnitude of similarity weight (ranging from 0 to 1) represents the strength of correlation. These will be used to predict unknown ratings for a user using *neighborhood based* methods later.

EXERCISE3: Calculate the Pearson Correlation (PC) similarity of *Eric* with other users in Table 1. You can use the function *PCSimVecMatrix* provided with the lab. Which user(s) would you say is *similar* to *Eric*. You can do a similar analysis for comparing the Cosine similarities and Pearson Correlation similarities of other users.

0.3 User based recommendation

The Recommender Systems exist so that they can make recommendations to the user. As highlighted in section [TODO FIXME], collaborative filtering relies on finding users similar to a user of interest based on the ratings given by them. We now have two important measures of finding similarity between users, namely Cosine Similarity and Pearson Correlation similarity. It follows that these similarity measures can now be used to find the neighbors nearest to a user, and subsequently their ratings can be *combined* to give us predicted ratings for a number of movies that have actually not been rated by the user of interest. It is natural that the movies whose predicted ratings are higher are the potential candidates that the recommender system can recommend to the user.

Thus, the rating r_{ui} of a user u for a new item i, can be predicted using the ratings given to i by users most similar to u. Let w_{uv} denote the similarity weight between users u and v, and N(u) be the set of k neighbors of u, then the predicted rating r_{ui} can be calculated as:

$$r_{ui} = \frac{\sum_{v \in N_i(u)} r_{vi}}{|N_i(u)|} \tag{4}$$

Note that the formula considers the only neighbors v of user u who have rated the movie i, $N_i(u)$ representing the set of such neighbors.

EXERCISE4: Carefully look at the formula in the equation [TODO FIXME]. Can you find a flaw in the formula? What does it use the similarity weights of user *u* with different users *v* for? Explain qualitatively, how the formula is not making full use of the similarity weights?

If you have an intuitive answer to EXERCISE4 [TODO FIXME], equation [TODO FIXME] that is more sophisticated in its calculation of the predicted rating should make sense.

$$r_{ui} = \frac{\sum_{v \in N_i(u)} w_{uv} r_{vi}}{\sum_{v \in N_i(u)} |w_{uv}|} \tag{5}$$

We now take a small detour to look at the concept of *Rating Normalization* and will then come back to learn how to really generate a recommendation for a user.

1 Rating normalization

Various users rate the movies differently, each having his/her own personal scale even when they have to give a rating on the same scale of 1 to 5. For example, while a rating of 4 by one user might imply a strong preference for a movie, the same rating by another user whose average rating across movies is 4, might not give us any interesting information to use. To tackle this two normalization techniques are: Mean-Centering and Z-score normalization.

1.1 Mean-Centering

The idea of mean-centering is quite simple. It determines whether a rating is positive or negative by comparing it to the mean of the ratings for that user. Once we have the ratings for nearest neighbors, they can be *normalized* by subtracting their corresponding means and then used to generate the predicted rating for a user using the following relation.

$$r_{ui} = \bar{r_u} + \frac{\sum_{v \in N_i(u)} w_{uv}(r_{vi} - \bar{r_v})}{\sum_{v \in N_i(u)} |w_{uv}|} \tag{6}$$

where $\bar{r_u}$ [TODO FIXME] is the mean of ratings of user u.

1.2 Z-score normalization

While mean-centering takes removes the user bias by removing the mean from their ratings, it still does not consider the spread of ratings given by a user. For example, consider two users both of whom have an average rating of 2.8. Further user X shows a lot of variation in his ratings from 1 to 5, whereas user Y has most ratings very close to 2.8. In this case, a rating of 4.5 by Y would imply an exceptional liking for the movie for user Y. The same cannot be said for user X. Z-score normalization takes care of this by normalizing the ratings so the ratings of a user have a mean of zero and a variance of 1. You can explore the MATLAB function *zscore*.

EXERCISE5: Consider the Table ?? which has a new user Kim added to our earlier toy example. We are interested in generating a recommendation for Kim. Can you guess which movie we should recommend just by looking at the table? Now write a small code in

	Matrix	Titanic	DieHard	ForrestGump	Wall-E
John	5	1		2	2
Lucy	1	5	2	5	5
Eric	2		3	5	4
Diane	4	3	5	3	
Kim	1	4		5	

Table 3: Example ratings

MATLAB to generate those predictions following the steps mentioned below:

- **Calculate the PC similarity of Kim with other users using the provided MATLAB function *PCSimVecMatrix***
- **Find the nearest neighbors of Kim using a threshold**
- **Use mean-centering to normalize the ratings of the neighbors**
- **Generate a recommendation for Kim using the normalized ratings of neighbors and similarity weights.**

Carefully read the specification of the provided MATLAB function *PCSimVecMatrix*. Can you use it to skip one of the steps mentioned in the EXERCISE above.