Building Recommendation Systems

1. Access movie recommendation system:
   http://www.movielens.org/
   username: cenlkang@gmail.com, password: butterfly (Create your own account)

   self rate a list of movies <so the system may match me to similar users later>
   exam the movies recommended to me

2. How to build recommendation system?
   - User based approach
   - Item based approach

3. Example data

   Training Data:

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6
User1	3	2	NA	5	4	4
User2	2	3	4	NA	NA	4
User3	NA	2	5	4	5	3
User4	3	NA	4	4	3	NA

   Test Data:

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6
User A	3	?	3	4	?	4

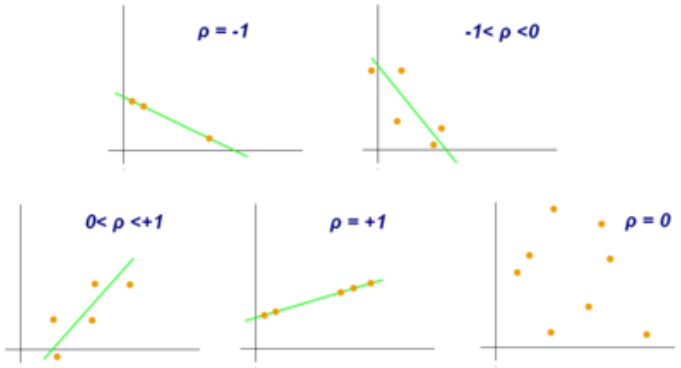
4. User based approach
   - Compute how similar people are in their taste of movie

   - How to represent each person? What does the data look like?
     - http://www.grouplens.org/ (Data Sets → ML 100k)
     - README, u.item, u.user, u.data

   - Compute similarity between pair-wise people
     - Similarity measures
       - Euclidean distance

$$d = |\mathbf{x} - \mathbf{y}| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} .$$

       - Pearson correlation
         - measure of the linear dependence (correlation) between two variables X and Y. It has a value between +1 and −1 inclusive, where 1 is total positive linear correlation, 0 is no linear correlation, and −1 is total negative linear correlation.



- Pearson's correlation coefficient between two variables is defined as the covariance of the two variables divided by the product of their standard deviations:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y},$$

The above formula defines the *population* correlation coefficient, commonly represented by the Greek letter ρ (rho). Substituting estimates of the covariance and variances based on a sample gives the *sample correlation coefficient*, commonly denoted r :

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Or the following used in the programming:

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{\left(\sum X^2 - \frac{\left(\sum X\right)^2}{N}\right)\left(\sum Y^2 - \frac{\left(\sum Y\right)^2}{N}\right)}}$$

- Load (movie) preference data
- Best matching users
  - The best matching user, or
  - The best K matched user?
    - Weighted average of the rating
  - Dealing with missing ratings in training data
- Compare the results from Euclidean distance and Pearson Coefficient
- Get the recommendations

- Project Questions:
  - Can Jaccard coefficient, Manhattan distance be used to compute similarity in these cases? Write python functions to compute each of these similarity values. Which top 5 matches are returned in each case?

- Currently, all, except for those with negative similarity value, user's ratings are used for computing the ranking. How can this be modified to improve the way ranking maybe computed? (only the top n most similar user's rating should be used)

- Python notes
  - 2 spaces per indentation
  - nested dictionary
  - setdefault usage

    DATA_SOURCE = (('movieA', '4'), ('movieA', '3.4'), ('movieB', '3'), ('movieB', '5'), ('movieB', '4.4'),)

    version1: test for empty key explicitly in code:

```
newdata = {}
for k, v in DATA_SOURCE:
  if newdata.has_key(k):
    newdata[k].append(v)
  else:
    newdata[k] = [v]
print newdata
```

    version2: use setdefault to test for empty key implicitly

```
newdata = {}
for k, v in DATA_SOURCE:
  newdata.setdefault(k,[]).append(v)
print newdata
```

    result from the two versions are the same:
    {'movieB': ['3', '5', '4.4'], 'movieA': ['4', '3.4']}