CIS606 – Lecture 2

Woon Wei Lee, Jacob Crandall
Spring 2013, 4:15-5:30pm,
Mondays and Thursdays

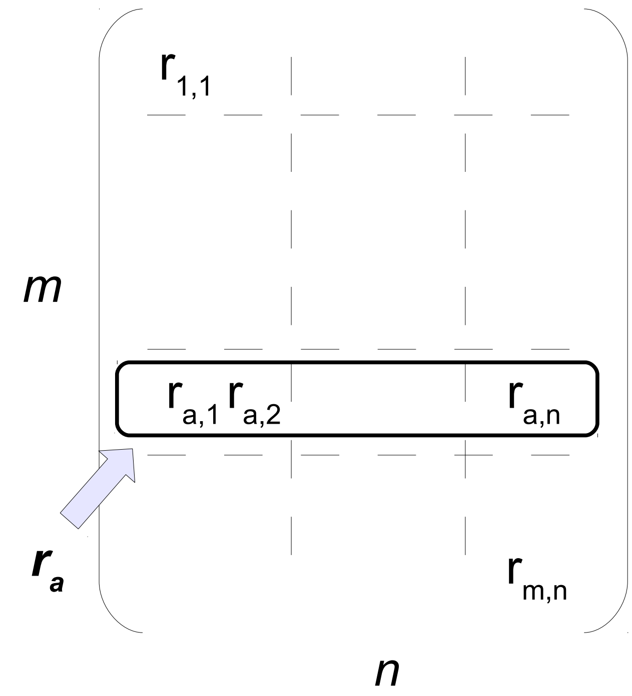
For today:

- (More about) collaborative filtering

Techniques for CF

- **Standard formulation**
  - In a typical scenario, there is:
    - A list of users, $U=\{u_1,u_2, \ldots, u_m\}$
    - A list of items, $I=\{i_1,i_2, \ldots, i_n\}$
    - Each user a has a list, I_a, of items for which ratings are available, and a corresponding rating, $r_{a,i}$ for each item in I_a
  - Matrix representation (depicted right)
    - In data mining context, common to represent in the form of an $m \times n$ matrix
- **Goal is normally one of:**
  1. To provide a *Prediction*, $P_{a,j}$ of the rating that that user would provide to item i_j

     (Given that $i_j \notin I_a$)
  2. To provide a *Recommendation* for user a. This is typically a list of *N* items with highest probability of being "liked" by the user.



Masdar INSTITUTE

Cont'd

- **Two broad classes of CF algorithms:**
  - Memory based:
    - Based on comparing “active” user with existing users in database
    - Final prediction of rating typically obtained via a weighted sum of neighbours
    - Association rules!
  - Model based
    - Based on constructing a model which describes important properties of the data
    - Two broad classes:
    1. Non-Probabilistic
        - Clustering algorithms
        - Matrix factorization
    2. Probabilistic
        - Bayesian Networks
        - EM algorithm

Masdar INSTITUTE

Example problem

Name\ Item	Star Wars	Jaws	Avatar	Alien	Chicken Little
Richard	?	2.5	?	5	?
Ahmad	4	?	?	?	?
Fauziah	?	?	3	?	?
Foo	5	2	?	4	1.5
Kok Hwa	4.5	?	3	?	3
Latiff	3	5	3	2.5	?

- **Typical sort of problem**
  - Represent using 6×5 matrix but only 15 elements – moderately sparse
  - Ratings available for some elements but not for others
  - We would like to determine value of ratings for all the cell
- **Will look at three example “solutions”**

Approach 1: Association rule mining

Name\ Item	Star Wars	Jaws	Avatar	Alien	Chicken Little
Richard	?	1	?	1	?
Ahmad	1	?	?	?	?
Fauziah	?	?	1	?	?
Foo	1	1	?	1	1
Kok Hwa	1	?	1	?	1
Latiff	1	1	1	1	1

- **To generate recommendations, can use "association rule" mining!**
  - In a later lecture we will discuss this in more detail
  - Ratings are thresholded to either "1" (watched) or "0" (not watched)
  - For e.g. we can see that
    - [Jaws → Alien], and [Alien → Jaws] with confidence of 1
    - [Star wars → chicken little] with confidence of 3/4, [chicken little → star wars] with confidence level of 1

Association rule mining (Cont'd)

Name\ Item1	Star Wars	Jaws	Avatar	Alien	Chicken Little
Richard	?	1	?	1	?
Ahmad	1	?	?	?	?
Fauziah	?	?	1	?	?
Foo	1	1	?	1	1
Kok Hwa	1	?	1	?	1
Latiff	1	1	1	1	1

- **To generate item recommendations:**
  - Simply merge the outputs of all active association rules for items that are already in a user's collection
  - For e.g., in the case of Fauziah, we can see that:
    - [Avatar->chicken little] with confidence of 2/3
    - [Avatar->star wars] with confidence 2/3
  - Therefore, we would recommend {chicken little,star wars} to Fauziah.
  - Shortcomings:
    - A user which is new may not have watched many movies – does not mean that he “dislikes” those movies
    - Unable to generate recommendations for first time users

Masdar INSTITUTE

Approach 2: Nearest neighbours

- **Idea: find the most similar individuals, watch what they're watching!**
  - For each potential neighbour, only consider items which are rated for both individuals
  - So, for e.g. to compare Latiff and Richard, only movies “Jaws” and “Alien” are considered
  - Let's denote Richard as u_1 and Latiff as u_6, then this gives us:

    $\boldsymbol{r_1}$=[2.5,5]

    $\boldsymbol{r_6}$=[5,2.5]
  - Any standard distance vector can be used, for e.g. Euclidean, Cosine, etc..
- **Two of the commonly used measures are:**

1. Cosine
2. Pearson correlation

Name\ Item	Jaws	Alien
Richard	2.5	5
Latiff	5	2.5

Nearest neighbours (Cont'd)

Cosine Similarity

$$Simil(i,j) = \frac{r_i \cdot r_j}{\|r_i\| * \|r_j\|}$$

$$= \frac{2.5*5+5*2.5}{2.5^2+5^2}$$

$$= 0.8$$

Name\ Item	Jaws	Alien
Richard	2.5	5
Latiff	5	2.5

Pearson correlation

$$Simil(i,j) = \frac{\sum_{k \in I} (r_{i,k} - \overline{r_i})(r_{j,k} - \overline{r_j})}{\sqrt{\sum_{k \in I} (r_{i,k} - \overline{r_i})^2} \sqrt{\sum_{k \in I} (r_{j,k} - \overline{r_j})^2}}$$

$$= \frac{(2.5-3.75)*(5-3.75)+(5-3.75)(2.5-3.75)}{\sqrt{(5-3.75)^2+(2.5-3.75)^2} * \sqrt{(2.5-3.75)^2+(5-3.75)^2}}$$

$$= \frac{-3.125}{3.125} = -1$$

Recommendation/Prediction

- **To provide simple recommendations:**
  - Simply collect top rated items from each of the closest users
  - Merge into recommendation set and prune accordingly (e.g. top-N most frequently occurring based on user requirements, etc)
- **Prediction – more interesting challenge**
  - For the "active" user, provide predicted rating for a previously unrated item, let's say i_h.

  1. *k*-NN – pick *k* most similar users, return average rating for item (where available):

$$p_{a,h} = \frac{1}{k} \sum_{u=1}^{k} r_{u,h}$$

- In the case of Richard, let's say we want to predict the rating for "Star Wars", there is only one value to consider:

$$p_{a,h} = \frac{1}{k} \sum_{u=1}^{k} r_{u,h}$$
$$= 3$$

Recommendation/Prediction

2. Weighted averaging
   - *k*-NN technique has a problem in that each user has a different rating scale (some are "tougher" than others!)
   - A more commonly used scheme returns a predicted value is based on mean value for the user, and weighted average of the deviation:

$$p_{a,h} = \overline{r_a} + \kappa \sum_{u=1}^{n} w(a,u) * (r_{u,h} - \overline{r_u})$$

   (κ is a normalizing constant which ensures that all the weights sum to one)

   - Back to our friend Richard and "Star Wars", in this case, the predicted rating now becomes:

$$p_{a,h} = \overline{r_a} + \kappa \sum_{u=1}^{n} w(a,u) * (r_{u,h} - \overline{r_u})$$

$$= 3.75 + (3 - \frac{3+5+3+2.5}{4}) = 3.375$$

Item based collaborative filtering

- **The nearest neighbour calculations which we have discussed are *user-based***
  - *i.e.* all distances/similarities are calculated between pairs of users at a time
  - This is fine for established users who have an existing profile, however, some applications require much faster updating
- **For example, “pandora.com” creates instant virtual radio stations based on songs/artistes which anyone can enter.**
  - Creating these recommendations on-the-fly would require calculating distances to all users in the database (possibly millions of users) – impractical!
  - Besides, even sites like Amazon and Ebay would benefit from faster incorporation of latest purchases, etc, into recommendations
- **Traditionally user-based approaches were the first to appear, but in recent times item-based CF has gained in popularity**
  - Almost identical calculations, but are performed on a per-item basis
  - Allows for inter-item similarities to be calculated in advance, and stored in efficient data structures to facilitate search
  - In simplest case, recommendations can be returned “instantly” - simply by returning the most similar items

(Cont'd)

- **For Cosine similarity, formula is identical (except now indexing against *items* and not users**

$$Simil(i,j)=\frac{r_i \cdot r_j}{\|r_i\| * \|r_j\|}$$

- **For the Pearson correlation, similarly:**

$$Simil(i,j)=\frac{\sum_{k\in U}(r_{k,i}-\overline{r_i})(r_{k,j}-\overline{r_j})}{\sqrt{\sum_{k\in U}(r_{k,i}-\overline{r_i})^2}\sqrt{\sum_{k\in U}(r_{k,j}-\overline{r_j})^2}}$$

- **To perform prediction..**
  - *k*NN approach is identical
  - Weighted average formula is slightly modified:

$$p_{a,h}=\kappa\sum_{j=1}^{n} w(h,j) * r_{a,j}$$

Matrix Factorization for CF

- **The general idea:**
  - We know that matrix R is (i) huge (ii) very sparse (iii) lots of unknown/missing values (iv) probably rank-deficient
    - In short – horrendous to work with!
  - Question: is it possible to represent the matrix R as a product of two separate matrices:

$$R = P.Q$$

    - However, we would like to do it such that P and Q have more desirable properties!
  - There are a number of matrix factorization schemes have been deployed in the past, examples:
    - Independent Component Analysis
      - Factorize matrices as shown above, such that the rows of Q have minimal statistical dependency
    - Sparse Component Analysis
      - Factorize matrix so that components are as sparse as possible (!)
    - SVD (Singular Value Decomposition)
      - Factorize matrices such:

$$R = U.S.V^T$$

      - Where the columns of U and V are orthogonal basis vectors in *m* and *n*-dimensions respectively

Singular Value Decomposition

$$\underset{n \times m}{R} = \underset{n \times n}{U} \; \underset{n \times m}{S} \; \underset{m \times m}{V^T}$$

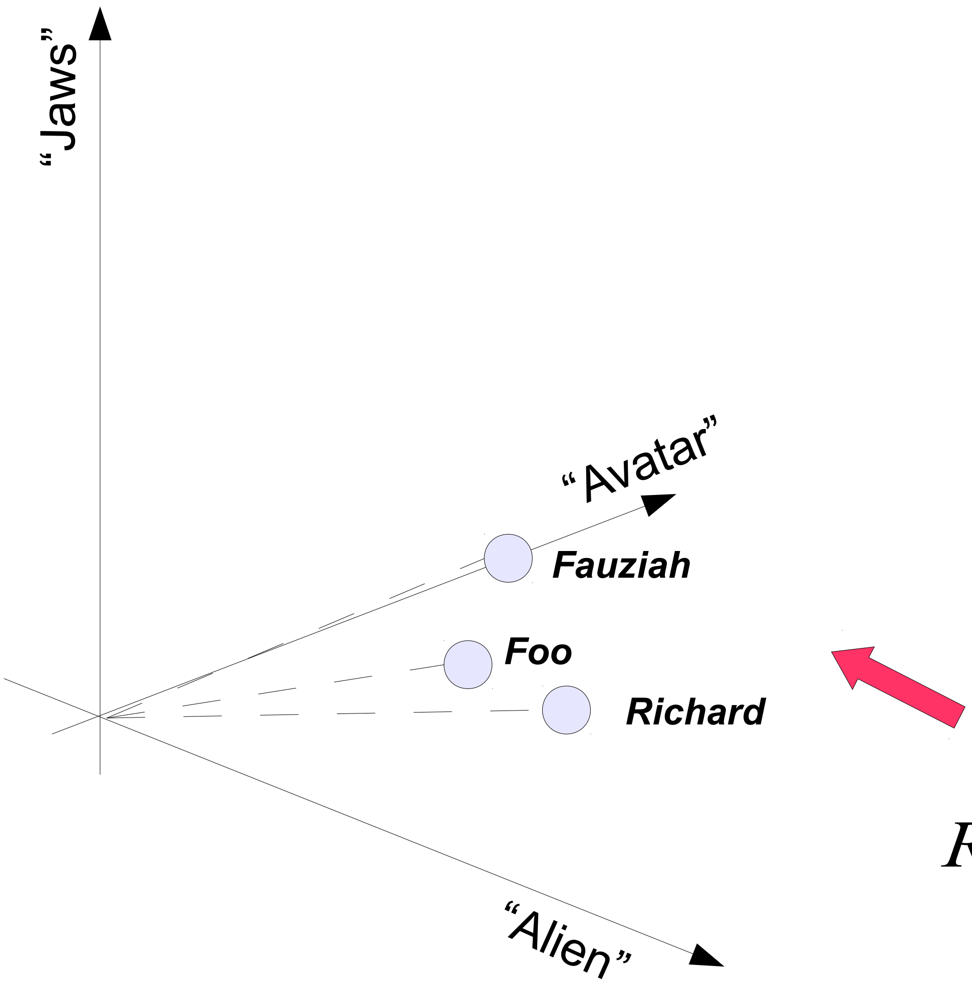
- **SVD is very useful for subspace project/dimensionality reduction**
  - Matrix S contains singular values. Setting non-significant singular values to zero constrains data to subspace of full dimensionality.
  - SVD is a common mathematical operation;
  - Numerous libraries exist (libLAPACK - opensource)
  - Efficient algorithms to compute SVD
- **However..**
  - Applying SVD to a 1000,000 x 500,000 matrix is going to kill your PC!
  - Besides, how do we deal with missing data?
  - Also, incrementally editing the matrix would be very difficult.
- **For collaborative filtering, an incremental simplification is available:**
  - First proposed by Simon Funk, used in the *NetFlix Challenge*
  - (3rd Place!)

Matrix Factorization for CF

$$\underset{n \times m}{\left(R \right)} = \underset{n \times k}{\left(P \right)} \underset{k \times m}{\left(Q \right)}$$

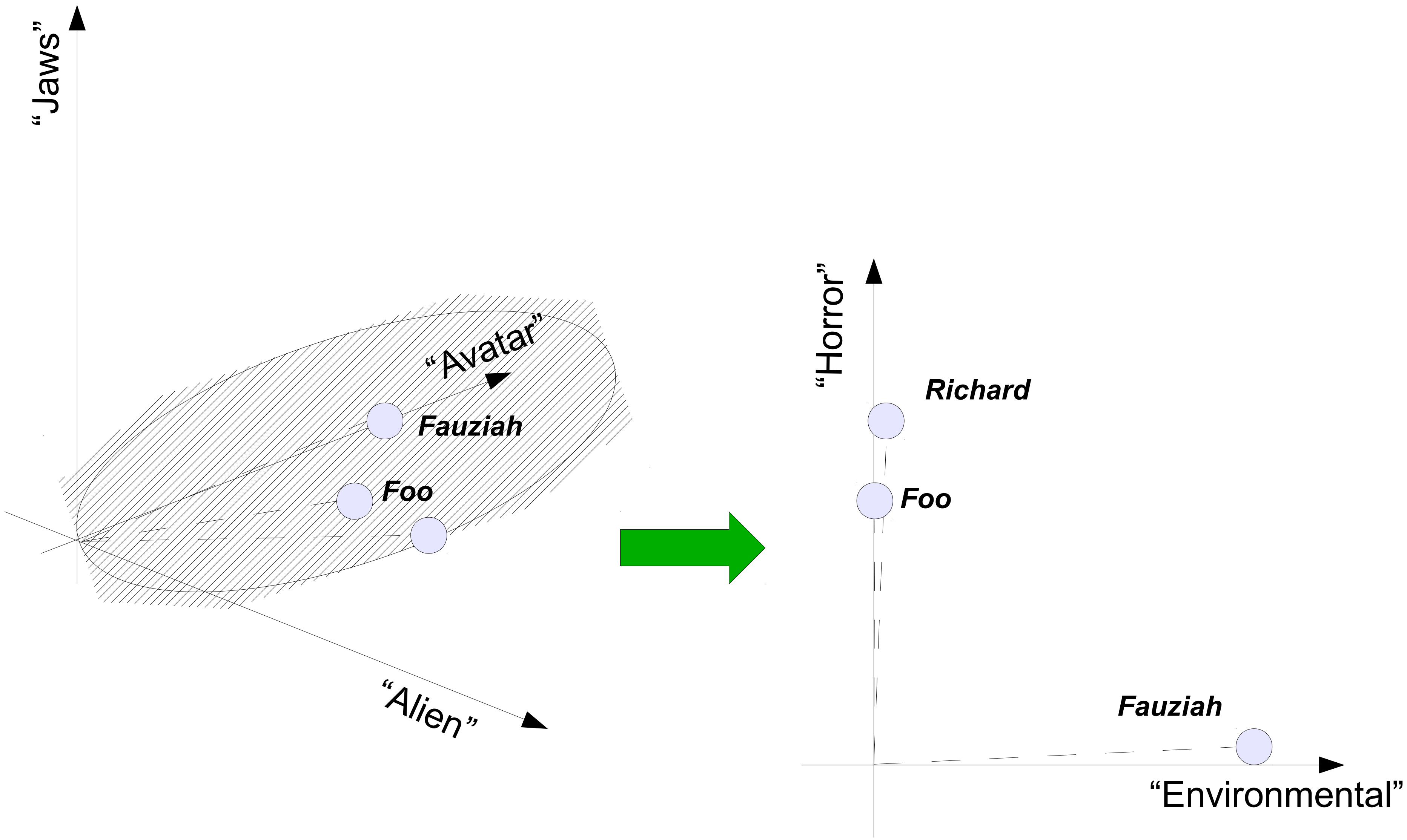
- **(Back to context of CF) Proposed method:**
  - Factorize matrix into two denser matrices P and Q
  - P is known as the *user features* matrix
  - Q is known as the *item features* matrix
  - In most cases k << n and k << m
- **Rationale: In "item-space" there are *m* dimensions**
  - 100s of thousands dimension, or more → "real" data lies in a subspace of this space
  - Selection of a small value of *k* forces the algorithm to search for the subspace on which the ratings lie

Cont'd



$$R = \begin{bmatrix} ? & 2.5 & ? & 5 & ? \\ 4 & ? & ? & ? & ? \\ ? & ? & 3 & ? & ? \\ 5 & 2 & ? & 4 & 1.5 \\ 4.5 & ? & 3 & ? & 3 \\ 3 & 5 & 3 & 2.5 & ? \end{bmatrix}$$

(Cont'd)



“Jaws”
“Avatar”
Fauziah
Foo
“Alien”
“Horror”
Richard
Foo
Fauziah
“Environmental”

Incremental Matrix Factorization

Optimization problem

$$R \approx PQ = \hat{R}$$

$$\hat{r}_{ij} = \sum_{h=1}^{k} p_{ih} q_{hj} = \boldsymbol{p_i q_j}$$

$$e_{ij} = \frac{1}{2}(r_{ij} - \hat{r}_{ij})^2$$

$$SSE = \sum_{i,\, j=1}^{i=n,\, j=m} e_{ij}$$

$$(P_{opt}, Q_{opt}) = \underset{P,Q}{argmin}\, SSE$$

$$\frac{\partial e_{ij}}{\partial p_{ih}} = (\hat{r}_{ij} - r_{ij}).\, q_{hj}$$

$$\frac{\partial e_{ij}}{\partial q_{hj}} = (\hat{r}_{ij} - r_{ij}).\, p_{ih}$$

Update equations

$$p_{ih}(t+1) = p_{ih}(t) + \eta.(\hat{r}_{ij} - r_{ij}).\, q_{hj}$$

$$q_{hj}(t+1) = p_{hj}(t) + \eta.(\hat{r}_{ij} - r_{ij}).\, q_{ih}$$

- **To factorize the matrices:**
  1. Take, as input, matrix R, with elements r_{ij},
  2. Create component matrices P and Q, by initializing randomly
  3. Loop over all element of R which has been rated
  4. Iterate until convergence