The “Netflix” Challenge – Predicting User preferences using:
-Average prediction
-Item-based Collaborative filtering
-Maximum Margin Matrix Factorization

- Somdutt Patnaik
- Nandan Dixit

Problem Statement

- Description – To predict whether someone will enjoy a movie based on how much they liked or disliked other movies.
- Given – A very large sparse training matrix containing anonymous user ratings identified by a User-ID and names of movies they are rating. The dimensionality of this matrix is 480189 x 17770 and it has only about 100 million entries
- Goal – Fill in the Sparse matrix with values asked to be predicted (specific indexes)

Our First shot at predicting – The Average method

- The Average method does a simple column-wise analysis of the dataset gathering an average rating per Movie and predicting these average value for Users who haven't rated a Movie
- Validation method – Cross validated RMSE

  We achieved a cross-validation RMSE using the RMSE script provided by Netflix of 1.05

  The Netflix “Cinematch” Algorithm achieved an RMSE of 0.9525

  So we now had a baseline to start!

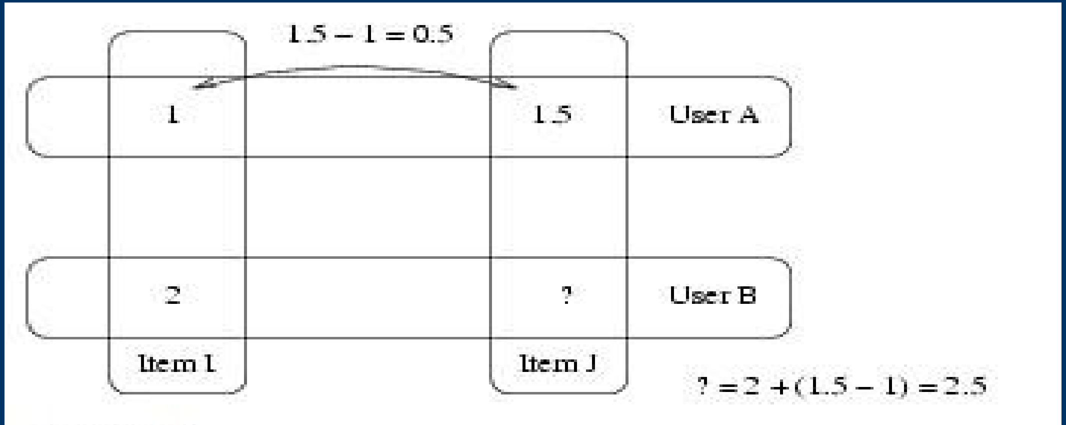
Collaborative Filtering

This method basically performs the following steps:

1. Look for users who share the same rating patterns with the active user (the user who the prediction is for).
2. Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user.

'Slope One' Item based Collaborative Filtering

1. Build an item-item matrix determining relationships between pairs of items
2. Using the matrix, and the data on the current user, infer his/her taste



Example:

1. Joe gave a 1 to Dion and an 1.5 to Lohan.
2. Jill gave a 2 to Dion.
3. How do you think Jill rated Lohan?
4. The Slope One answer is to say 2.5 (1.5-1+2=2.5).

Methodology for IBCF...

- To drastically reduce overfitting, improve performance and ease implementation, the Slope One family of easily implemented Item-based Rating-Based Collaborative Filtering algorithms was proposed.
- Essentially, instead of using linear regression from one item's ratings to another item's ratings ($f(x) = ax + b$), it uses a simpler form of regression with a single free parameter ($f(x) = x + b$).
- The free parameter is then simply the average difference between the two items' ratings. It was shown to be much more accurate than linear regression, and it takes half the storage or less.

Moving on to MMMF...

- Low Rank Factorization:
  - Factorize the given matrix into at most k-rank factors:

    $X = UV'$ where $U \in \mathbf{R}^{n \times k}$ and $V \in R^{m \times k}$
  - We are basically reducing the dimensionality of X to get more specific regions for predictions.
  - So if we fix U and learn V' then predicting each column of Y is a separate linear prediction problem
    - Here each row of U functions as a “feature vector” and each column of V' is a linear predictor, predicting the entries in the corresponding column of Y based on the “features” in U.

MMMF continued...

- Srebro et. al. “Maximum Margin Matrix Factorization” NIPS 2005 suggest that low-rank factorization regularizes the problem and is a non-convex optimization problem if we want to minimize a loss function and use a cost function for maximizing the margin as in SVMs. Instead use low-norm factorization and have no bounds on the dimensionality of the factors U and V'. This makes the problem a convex optimization problem.

MMMF Optimization objectives

- For binary labels $Y \in \{-1,+1\}$ the following objectives can be put down for the SDP to be solved:
  - Hard-margin:

$$\text{minimize } |X|_{tr}$$
$$\text{subject to } Y_{ia} X_{ia} \geq 1 \text{ for all } ia \in S$$

i.e.,

$$\text{minimize } |X|_{tr} + c \sum \max(0, 1 - Y_{ia} X_{ia})$$

MMMF optimization objectives

- Soft Margin factorization: done by introducing slack variables for the cost functions
- For our case labels are not binary and hence we need to have the following constraint:

  $\theta_{Y_{ia}} + 1 \leq X_{ia} \leq \theta_{Y_{ia}+1} - 1$

  and a set of constraints:

  $X_{ia} \geq \theta_r$ for $r < Y_{ia}$ and $X_{ia} \leq \theta_r$ for $r \geq Y_{ia}$
- where there are *R* ratings and we need to learn a threshold θ_r for each level.
- We could learn a single threshold matrix or a threshold matrix for each user (row) for capturing more variations in user behavior

Further Steps...

- We now have an optimal method to solve the problem. We are implementing it and hopefully we will have a very good RMSE
- As mentioned in the Srebro paper discussion, we intend to introduce natural parameters on scales of say 1 to 5 to capture characteristics of movies such as
  - Violence
  - Romance
  - Scenery
  - Music score

  to the U “feature” vector to have richer predictions that we will report in the paper.