On the Gravity Recommendation System

Gabor Takacs
Istvan Pilaszy
Bottyan Nemeth
Domonkos Tikk

Budapest University of Technology and Economics

Presented by Matt Rodriguez
May 13th 2008

Outline

Gravity Team

- Gravity Team is from Budapest University of Technology and Economics in Hungary.
- They have been competitive in the Netflix Data Mining contest.
- The paper is from SIGKDD Workshop 2007.

CF Approaches

Memory-based versus Model-based

Memory-based approach: uses a function to determine ratings of a user/item pair. The function examines the ratings fo the nearest neighbors(NNs).

1. Neighbors of the user and/or
2. Neighbors of the item

Model-based approach: uses the ratings to learn a global model which is used to to make ratings predictions.

Outline

The Netflix Dataset

- Can be stored in a matrix
- 99% entries in the rating matrix are unknown.
- A sparse matrix representation is beneficial.

users	480,189
movies	17,770
ratings	100,480,507

100 million ratings, 8.5 billion entries missing

Storing the Netflix Dataset

Store the data per-user, each row is a list of column index, value pairs.
Store the data per-movie storage, each column as a list of (row index, value)

$$\begin{aligned} \log_2 |\text{rating scale}| &= 2.3 < 3 \\ \log_2 |\text{users}| &= 18.8 < 19 \\ \log_2 |\text{movies}| &= 14.1 < 15 \end{aligned}$$

Requires 22 bits per user element rating.
Requires 18 bits per movie element rating.

Each entry will take require 3 bytes.
Total storage is 300 MB.

Storing the Netflix Dataset

Objective

Map the movie j and rating x_{ij} to a compact representation using only 2 bytes. $(j, x_{ij}) \longrightarrow c_{ij}$

Split the matrix into 2 halves.
First half has 1-8885 movies, the second half has 8885-17770 movies.
Notice $8885 \times 5 = 44425 < 2^{16} = 65536$
$c_{ij} = 5j + x_{ij} - 1$
$j = \lfloor \frac{c_{ij}}{5} \rfloor$
$x_{ij} = 1 + (c_{ij} \mod 5)$

Total Storage is 200 MB.

Glossary

c_{ij}: compact rating, 2 bytes
x_{ij}: original rating

Outline

Matrix Model

Objective

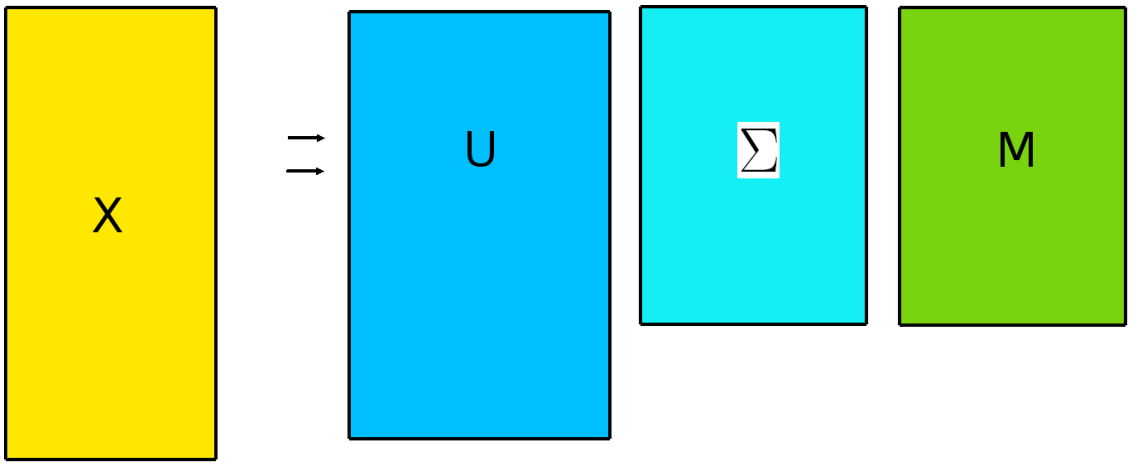
X matrix contains some known ratings, mostly unknown ratings.

Factor X into U and M to develop a model that can predict unknown ratings.

Use a technique to approximate X when not all values are known.

$X \approx UM$

Singular Value Decomposition



$X = U\Sigma M$

Σ has only values along the diagonal

U and M are orthogonal matrices.

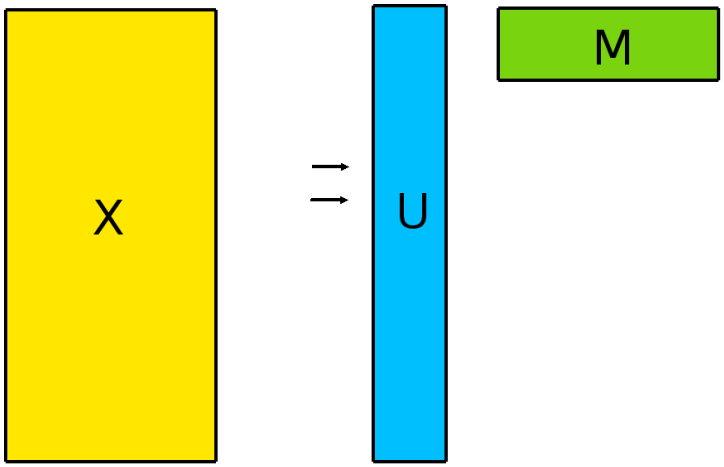
Glossary

$U \in \mathbb{R}^{I \times \min(I,J)}$

$M \in \mathbb{R}^{\min(I,J) \times J}$

$\Sigma \in \mathbb{R}^{\min(I,J) \times \min(I,J)}$

K-Rank Matrix Factorization techniques



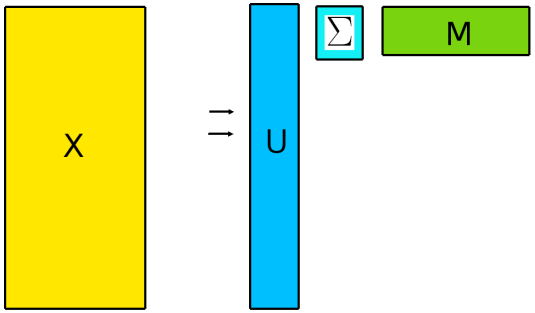
$X \approx UM$

Glossary

$X \in \{1,2,3,4,5\}^{I \times J}$
$U \in \mathbb{R}^{I \times K}$
$M \in \mathbb{R}^{K \times J}$

Originally X is described by $I \times J$ parameters.
Reduces the number of parameters to describe X to $IK + KJ$.

K-rank approximation using SVD



Least Squares Goal

Find $U, M = \arg\min_{U,M} ||X - UM||_2$
Solution $\hat{X} = UM$ approximation of X

K-rank

Choose the K largest eigenvalues.
$||\hat{X} - X|| = min||X' - X|| : X' \in \mathbb{R}^{I \times J}, \text{rank}(X') \leq K$
Minimizes the Frobenius norm
$||A|| = \sqrt{\sigma_{a \in A} a^2}$

Use Linear Algebra methods when all entries in X are known.

Minimizing Squared Error

New Least Squares Goal

Minimize SE $= \sum_{(i,j) \in R} e_{ij}^2$

$\hat{x}_{ij} = \sum_{k=1}^{K} u_{ik} m_{kj}$

$e_{ij} = x_{ij} - \hat{x}_{ij}$ for $(i,j) \in R$

SE $= \sum_{(i,j) \in R} e_{ij}^2$

(U,M) $= \arg\min_{(U,M)}$ SE

Glossary

R: non-missing values in X $\hat{x}_{ij}$: prediction

$u_{ik} \in U$

$m_{kj} \in M$

e_{ij}: error

Minimizing Squared Error by Gradient Descent

To minimize the squared error use Gradient Descent.

$$\nabla e_{ij}^2 = \left(\frac{\partial}{\partial u_{ik}}(x_{ij} - \sum_{k=1}^{K} u_{ik} m_{kj})^2, \frac{\partial}{\partial m_{kj}}(x_{ij} - \sum_{k=1}^{K} u_{ik} m_{kj})^2\right)$$

$$\nabla e_{ij}^2 = \left(2e_{ij}\frac{\partial}{\partial u_{ik}}(x_{ij} - \sum_{k=1}^{K} u_{ik} m_{kj}), 2e_{ij}\frac{\partial}{\partial m_{kj}}(x_{ij} - \sum_{k=1}^{K} u_{ik} m_{kj})\right)$$

$$\nabla e_{ij}^2 = \left(2e_{ij}(0 - \frac{\partial}{\partial u_{ik}}\sum_{k=1}^{K} u_{ik} m_{kj}), 2e_{ij}(0 - \frac{\partial}{\partial m_{kj}}\sum_{k=1}^{K} u_{ik} m_{kj})\right)$$

$$\nabla e_{ij}^2 = (-2e_{ij} m_{kj}, -2e_{ij} u_{ik})$$

Gradient

$$\frac{\partial}{\partial u_{ik}} e_{ij}^2 = -2e_{ij} m_{kj}$$

$$\frac{\partial}{\partial m_{kj}} e_{ij}^2 = -2e_{ij} u_{ik}$$

Gradient Descent with Regularization

Minimize the error

Update the Weights in the opposite direction of the gradient

Update Rules

$$u'_{ik} = u_{ik} + \eta 2 e_{ij} m_{kj}$$
$$m'_{kj} = m_{kj} + \eta 2 e_{ij} u_{ik}$$

$$u'_{ik} = u_{ik} + \eta (2 e_{ij} m_{kj} - \lambda u_{ik})$$
$$m'_{kj} = m_{kj} + \eta (2 e_{ij} u_{ik} - \lambda m_{kj})$$

Glossary

η: learning rate

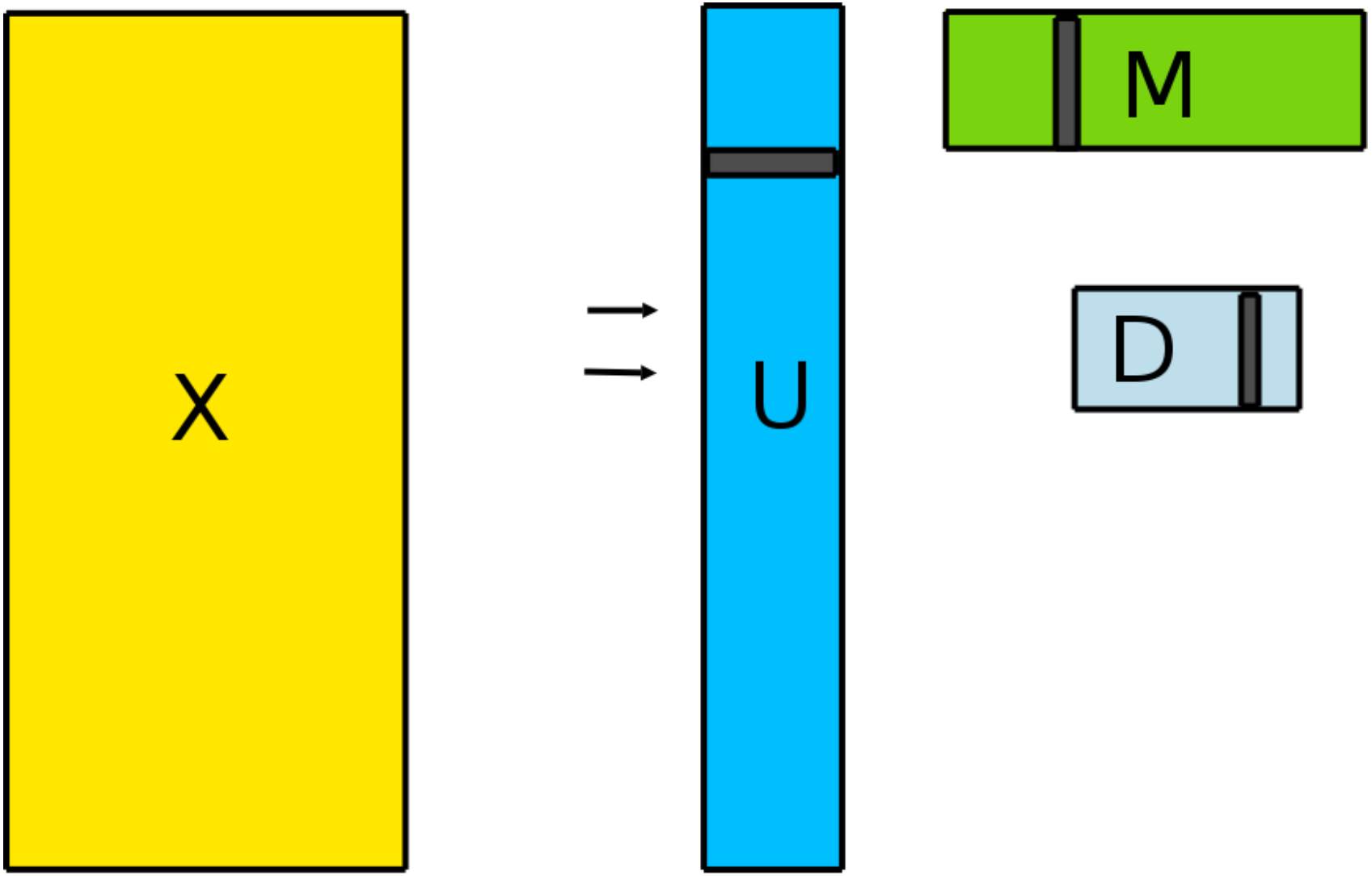
λ: regularization

Gradient Descent Algorithm

1. Initialize the entries in U and M randomly.
   Set η and λ to some small positive value.
2. Loop until the termination condition is met
   1. Iterate over each training element of X. For x_{ij}:
      1. compute e_{ij}
      2. compute the gradient of e_{ij}^2
      3. update the jth row of U and the jth column of M according to equations.
   2. Calculate the RMSE on the probe subset.

The loop terminates when RMSE does not decrease after two iterations.

Incorporating date information



Predictions using date information

Dates are represented as numbers from 1 to L.
$L = \{1, 2, 3, \ldots, 1095\}$ if ratings spanned 3 years.
l = is the date of the (i, j)th rating.

$d_{kl} \in D$
$D \in \mathbb{R}^{K \times L}$

Prediction

$$\hat{x}_{ij} = \sum_{k=1}^{K} u_{ik} m_{kj} d_{kl}$$

Note the Matrix dimensions

$U \in \mathbb{R}^{I \times K}$
$M \in \mathbb{R}^{K \times J}$
$D \in \mathbb{R}^{K \times L}$

Minimizing Error in the New Model

$$\hat{x}_{ij} = \sum_{k=1}^{K} u_{ik} m_{kj} d_{kl}$$
$$e_{ij}^2 = (x_{ij} - \hat{x}_{ij})^2$$

Gradient

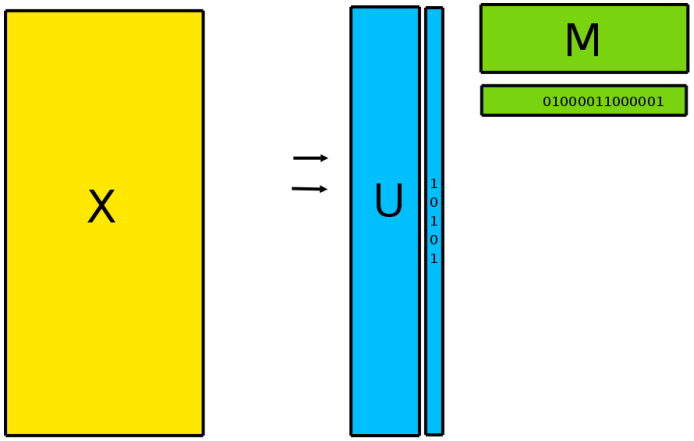
$$\frac{\partial}{\partial u_{ik}} e_{ij}^2 = -2 e_{ij} m_{kj} d_{kl}$$
$$\frac{\partial}{\partial m_{kj}} e_{ij}^2 = -2 e_{ij} u_{ik} d_{kl}$$
$$\frac{\partial}{\partial d_{kl}} e_{ij}^2 = -2 e_{ij} u_{ik} m_{kj}$$

Updating the weights using date features

Update Rules with regularization

$$u'_{ik} = u_{ik} + \eta\left(\frac{\partial}{\partial u_{ik}} e_{ij}^2 - \lambda u_{ik}\right)$$
$$m'_{kj} = m_{kj} + \eta\left(\frac{\partial}{\partial m_{kj}} e_{ij}^2 - \lambda m_{kj}\right)$$
$$d'_{kl} = d_{kl} + \eta\left(\frac{\partial}{\partial d_{kl}} e_{ij}^2 - \lambda d_{kl}\right)$$

Incorporating Content information



Constant values in one or other matrix

Fixed Feature values

These values are not features that are not updated by the gradient descent.

The *k*th row of M can be 1 or 0 depending on whether to include the term “Season”.

The *k*th column of U can be 1 or 0 depending on whether the user is Male.

Rounding the values

Problem

The model predicts real numbers.

Ratings are only between 1 and 5.

Solution

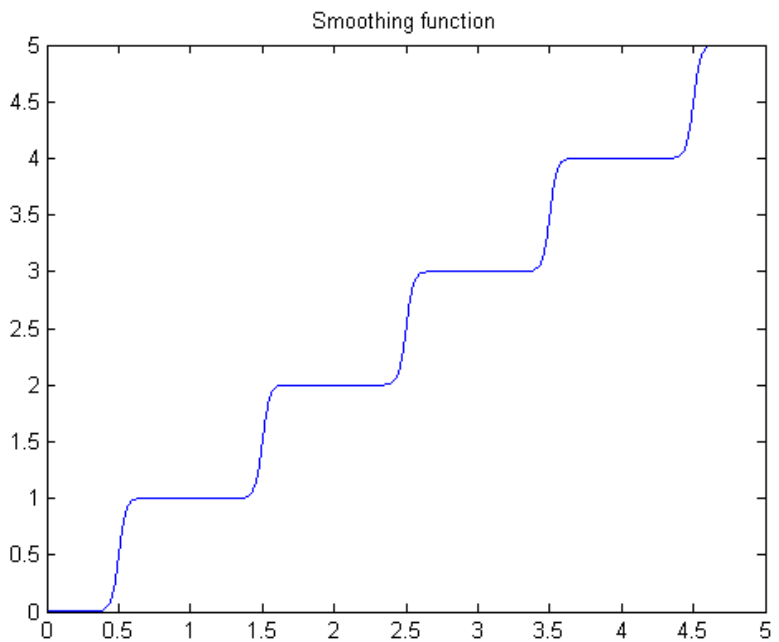
Apply smoothing to move the numbers toward a neighboring integer.

Sophisticated Rounding

Smoothing function

$0 < \rho < 1$

$\hat{x}'_{ij} = \rho \hat{x}_{ij} + (1 - \rho) round(\hat{x}_{ij})$



Analysis

The error of approximation X - UM has a normal distribution, with $\mu = 0$ and σ between .7 and 1.

Since many errors are greater than .5, standard rounding does not help.

$\varepsilon = x_{ij} - \hat{x}_{ij}$
$\varepsilon' = x_{ij} - \hat{x}'_{ij}$
$\varphi(x) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

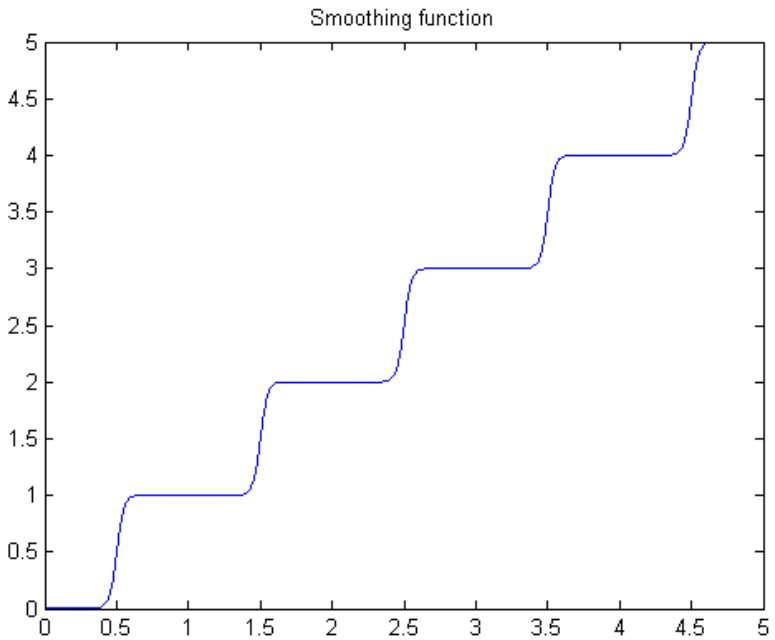
Glossary

ε : error
ε': error after rounding
φ: pdf of ε

Sophisticated Rounding

Another smoothing function

$$sr(x) = \lfloor x \rfloor + \frac{\tanh(2A(x - \lfloor x \rfloor - 0.5)) - \tanh(-A)}{\tanh(A) - \tanh(-A)}$$



Rounding the values

Apply the smoothing function to part of the prediction or the entire prediction.

Smoothing

$$\hat{x}_{ij} = \sum_{k=1}^{K_1} u_{ik} m_{kj} + \text{sr}\left(\sum_{k=K_1+1}^{K} u_{ik} m_{kj} \right)$$

Feature Maps

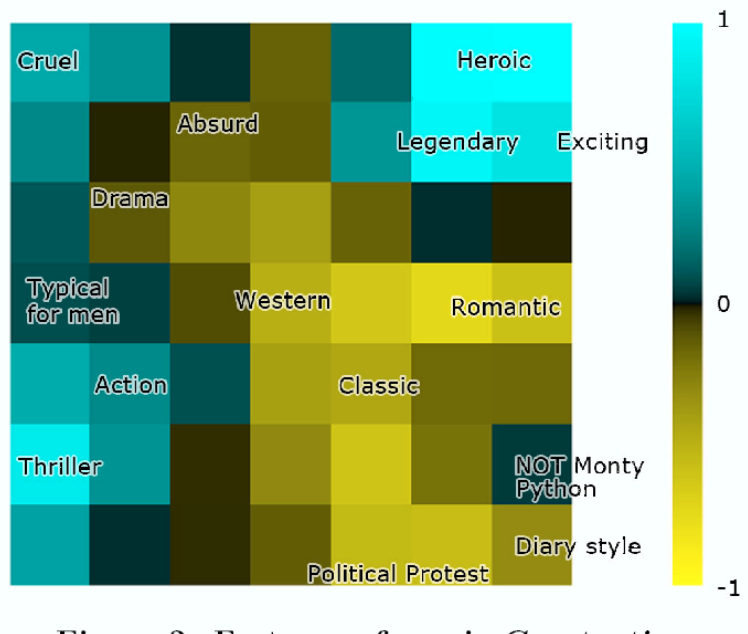


Figure 2: Features of movie *Constantine*

Idea: Arrange the features in a 2 dimensional feature map.

The labels are human assigned.

2D Modification of the matrix factorization algorithm

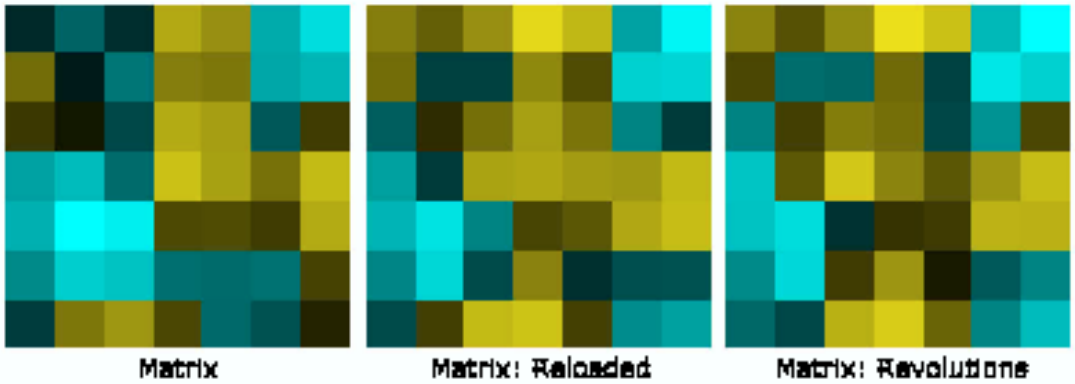


Figure 3: Features of *The Matrix* episodes

After each learning step modify feature values toward the direction of neighboring feature values.

Smoothing with adjacent user feature values

$$\Delta u'_{i,k_1,k_2} = \Delta u_{i,k_1,k_2} + \xi(\Delta u_{i,k_1+1,k_2} + \Delta u_{i,k_1-1,k2} + \Delta u_{i,k1,k2+1} + \Delta u_{i,k_1,k_2-1}) + \frac{\xi}{\sqrt{2}}(\Delta u_{i,k_1+1,k_2+1} + \Delta u_{i,k_1-1,k_2-1} + \Delta u_{i,k_1-1,k_2-1} + \Delta u_{i,k_1+1,k_2-1}))$$

Parameters of matrix factorization

1. the number of features K;
2. different learning rates and regularization factor
    1. for users and movies
    2. at different learning cycles
    3. values of η and λ on k
    4. for users or movies with different numbers of ratings
    5. for the corresponding variables of constant features
3. probability distribution to initialize U and M
4. offset to subtract from X before learning
5. nonlinear smoothing functions on parts of the output

All of these options lead to ....

Lots of opportunities for Numerology

Outline

Neighbor-based approaches

Use movies as neighbors, or users neighbors.

A rating can be estimated adding up the similarity times rating of its neighbors, then dividing by the sum of the similarities.

$$\hat{x}_{ij} = \frac{\sum_{k:(i,k \in R)} s_{jk} f_{jk}(x_{ik})}{\sum_{k:(i,k) \in R} s_{jk}}$$

Pearson correlation coefficient between two movies is:

$$r_{jk} = \frac{1}{|R_{jk}|} \sum_{i \in R_{jk}} \frac{(x_{ij} - \mu_j)(x_{ik} - \mu_k)}{\sigma_j \sigma_k}$$

Glossary

R: set of neighbors
s_{jk}: similarity metric between jth and kth movie
f_{jk}: function that predicts rating of the movie

Neighbor based approaches

Ratings with low support

If there are few ratings between neighbors then empirical correlation is a bad estimator.

Strategy

1. Transform, to a normal distribution
2. Regularize, shrink the weights

Neighbor based approaches

Fisher Z-Transform

Assume a mean of zero correlation, use the Fisher z-transform.
Fisher z-transform maps a correlation to a normal distribution.

$$z_{jk} = 0.5 \ln \frac{1+r_{jk}}{1-r_{jk}}$$

The distribution of z_{jk} is a normal with standard deviation $\frac{1}{\sqrt{|R_{jk}|-3}}$

Regularize

$$z'_{jk} = z_{jk} - \frac{\lambda}{\sqrt{|R_{jk}|-3}}$$

λ is the regularization factor.

Calculate the rating

Fisher Inverse Z- transform

$$r'_{jk} = \frac{\exp(2z'_{jk})-1}{\exp(2z'_{jk})+1}$$

Simularity metric, predictor function

$$s_{jk} = |r'_{jk}|^{\alpha}$$
$$f_{jk}(x) = \mu_j + r'_{jk}(x - \mu_k)$$

Glossary

s_{jk}: similarity metric
f_{jk}: movie-to-movie similarity
α: amplication factor, meant to increase similarity metric.
μ_j, μ_k: mean rating of movie j and k

Summary of Neighbor-based Approaches

Neighbor based approaches can be implemented without a training phase.(Memory Model)

Similar approach could be used for user-user relationships. Difficult to do with the netflix dataset.

Improvements could be consider only consider the K most similar neighbors.

The Gravity team did not improve their results with the neighbor-based method.

Outline

Clustering Overview

Apply clustering on users and movies simultaneously.

Create L user clusters and M movies clusters.

$|\text{Product Cluster}| = L \times M$

The rating x_{ij} can be predicted based on the cluster.

The naive approach use the average rating of the cluster.

Clustering Algorithm

1. Randomly create the initial clustering of users and movies.
2. For each user u : reassign u to a new user cluster, so that training RMSE is minimized.
3. For each movie m : reassign m to the movie cluster so that the training RMSE is minimized.
4. If the training RMSE decreased significantly, go to step 2.

Outline

Evaluation

Table 2: RMSE of the basic matrix factorization algorithm for various η and λ values ($K = 40$)

η \ λ	0.005	0.007	0.010	0.015	0.020
0.005	0.9280	0.9260	0.9237	0.9208	0.9190
0.007	0.9326	0.9301	0.9273	0.9243	0.9226
0.010	0.9397	0.9367	0.9337	0.9306	0.9287
0.015	0.9543	0.9518	0.9494	0.9473	0.9458
0.020	0.9781	0.9767	0.9753	0.9736	0.9719

Table 3: RMSE of various neighbor based methods

Parameters				RMSE
$K = \infty$,	$\alpha = 2$,	$\beta = 0.0$,	$\lambda = 0.0$	0.9483
$K = 16$,	$\alpha = 2$,	$\beta = 0.0$,	$\lambda = 0.0$	0.9399
$K = \infty$,	$\alpha = 2$,	$\beta = 0.2$,	$\lambda = 0.0$	0.9451
$K = \infty$,	$\alpha = 2$,	$\beta = 0.0$,	$\lambda = 2.3$	0.9445
$K = 16$,	$\alpha = 2$,	$\beta = 0.2$,	$\lambda = 2.3$	0.9313

Evaluation

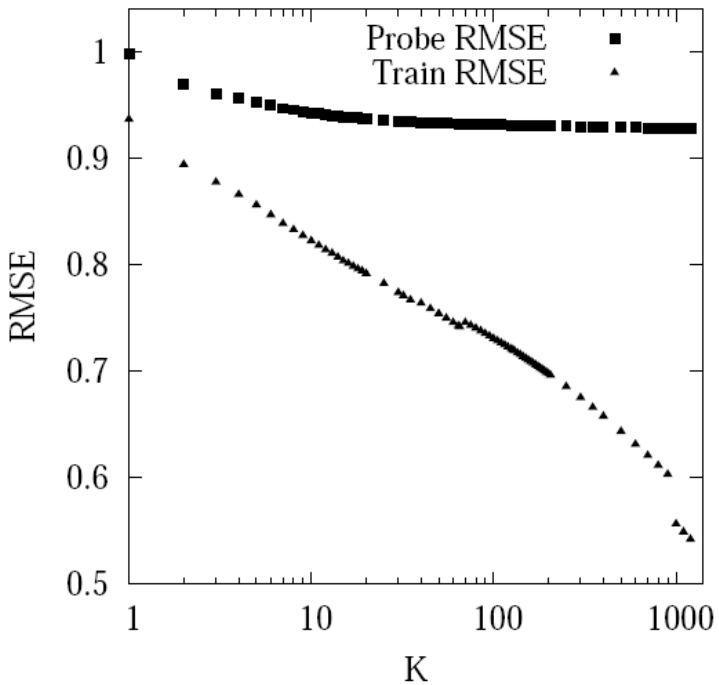


Figure 4: MFs with different values of K ($\eta = 0.01, \lambda = 0.01$)

Evaluation

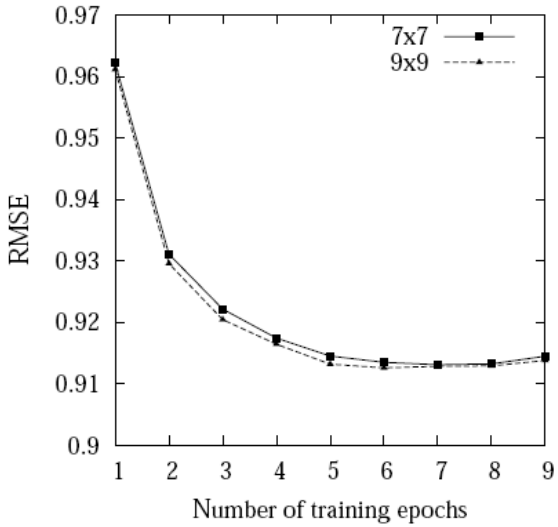


Figure 6: The learning curves of two 2D MF algorithm for various numbers of features

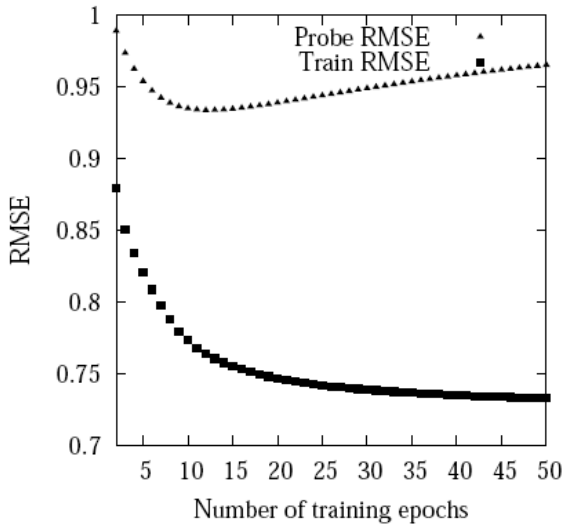


Figure 5: Learning curves of an MF run ($\eta = 0.01, \lambda = 0.01, K = 40$)

Evaluation

Table 4: RMSE of various clustering based methods

Parameters		RMSE
$L = 480189$,	$M = 5$	0.9659
$L = 5$,	$M = 17770$	0.9640
$L = 40$,	$M = 40$	0.9606

Table 5: Best results of single approaches and their combinations

Method/Combination	RMSE
MF	0.9190
NB	0.9313
CL	0.9606
NB + CL	0.9275
MF + CL	0.9137
MF + NB	0.9089
MF + NB + CL	0.9089

Conclusion

Combination of Matrix Factorization and Neighbor-based approaches worked the best.

Many different techniques for Matrix Factorization

1. Incorporating date information
2. Incorporating content
3. Rounding the values
4. Using 2D Smoothing

Some of the approaches worked well, others did not.

Neighbor-based approaches worked well with fixed number of neighbors and a regularization constant 2.3.