# The Netflix Prize

## and SVD

Stephen Gower

April 24, 2014

# Table of Contents

# Table of Contents

# Basics



- Q: What is the Goal? What are we trying to do?

# Basics

- Q: What is the Goal? What are we trying to do?
- A: Recommender Systems seek to match users with items that they will enjoy

# Basics



- Q: What is the Goal? What are we trying to do?
- A: Recommender Systems seek to match users with items that they will enjoy

Pretty basic stuff. But how does the system know what the users will like?

# Types of Systems



Content Filtering: Creates a profile on items and users based on information gathered outside of interaction with the user.

Collaborative Filtering: Analyzing relations between users and items based on interdependencies and use these relations to form new connections.

Content has better accuracy with little user action, but Collaborative is better in the long term as long as a user is active.

# Types of Systems



We focus on the Collaborative model and from there we get two branches:

- Neighborhood Method
- Latent Factor Model

# Types of Systems



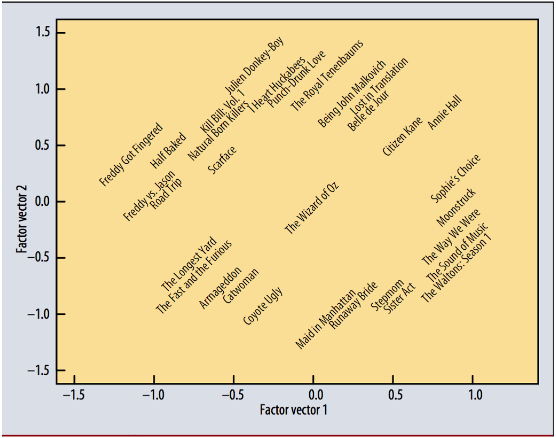


Figure : An example of a Latent model

# Types of Systems

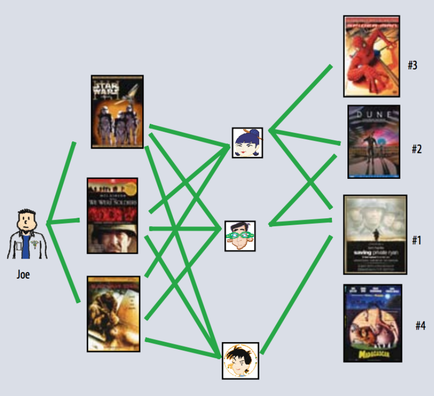


Figure : An example of a Neighborhood method

# Explicit Vs. Implicit



- Explicit data is anything obtained from the user by action of the user, such as profile details, ratings on items, etc.

# Explicit Vs. Implicit



- Explicit data is anything obtained from the user by action of the user, such as profile details, ratings on items, etc.
- Implicit data includes information such as keystrokes, mouse movement, search history, etc.

# Explicit Vs. Implicit

- Explicit data is anything obtained from the user by action of the user, such as profile details, ratings on items, etc.
- Implicit data includes information such as keystrokes, mouse movement, search history, etc.

Explicit data is going to be the basis of the SVD methods, while implicit data will impact parameters in the model.

# Table of Contents

# The Model



## Basic Approximation

We define two vectors, $\mathbf{q}_i \in \mathbb{R}^f$ represents our item vector while $\mathbf{p}_u \in \mathbb{R}^f$ is the user vector. An inner product of these vectors allows for an approximate rating as:

$$r_{ui} \approx \mathbf{q}_i^* \mathbf{p}_u$$

# Refining the Model



What we have so far is great, but it can be much better. We now add *biases* to the model. The bias $b_{ui}$ is defined as:

$$b_{ui} = \mu + b_i + b_u$$

# Refining the Model

What we have so far is great, but it can be much better. We now add *biases* to the model. The bias $b_{ui}$ is defined as:

$$b_{ui} = \mu + b_i + b_u$$

**Refined Model**

$$r_{ui} = \mu + b_i + b_u + \mathbf{q}_i^* \mathbf{p}_u$$

# Refining the Model



Next we need to make use of these other data inputs, so far we include biases and ratings but we don't include profile data and/or that implicit data mentioned prior.

# Refining the Model



Next we need to make use of these other data inputs, so far we include biases and ratings but we don't include profile data and/or that implicit data mentioned prior.

## Final Model

The general SVD++ model:

$$r_{ui} = \mu + b_i + b_u + \mathbf{q}_i^* \left[ \mathbf{p}_u + |N(u)|^{-0.5} \sum_{i \in N(u)} \mathbf{x}_i + \sum_{a \in A(u)} \mathbf{y}_a \right]$$

# Learning Algorithms

To refine these estimated ratings, the system also attempts to minimize its regularized squared error which goes as

$$\min_{q*,p*} \sum_{(u,i)\in\kappa} (r_{ui} - \mathbf{q}_i^* \mathbf{p}_u)^2 + \lambda(||\mathbf{q}_i||^2 + ||\mathbf{p}_u||^2)$$

Where $\lambda$ controls the extent of this regularization. To minimize this error the system first computes the prediction error and then it modifies the vectors using

$$\mathbf{q}_i \leftarrow \mathbf{q}_i + \gamma(e_{ui}\mathbf{p}_u - \lambda\mathbf{q}_i)$$

and

$$\mathbf{p}_u \leftarrow \mathbf{p}_u + \gamma(e_{ui}\mathbf{q}_i - \lambda\mathbf{p}_u).$$

# Temporal Dynamics

Time can also be factored into the model through the following factors:

- $b_u$ - User bias can vary with time
- $\mathbf{p}_u$ - User taste can vary with time
- $b_i$ the relative perceived quality of an item can change with time

# How Does SVD Play a Role?

So far we haven't seen the application of SVD in the model directly. SVD is used to form the user and item vectors from the massive rating matrix comprised of all users and their ratings on all items. These are defined as:

$$\mathbf{p}_u = U\sqrt{S}^T$$

and

$$\mathbf{q}_i = \sqrt{S}V^T$$

Where $A = USV^*$

# How Does SVD Play a Role?



This method of forming the user and item vectors focuses on relationships formed through latent factors.

Next we look at Principal Component Analysis which should shed light on the Neighborhood end of things and at the end we blend the two methods in our final vectors.

# PCA

We begin by denoting the matrix $C = A^*A$. We can solve for matrices such that

$$C = E^T \Lambda E$$

Thus

$$ECE^T = \Lambda$$

Denote a transformation $B$ such that $B = AE^T$ so

$$C_B = B^T B = ECE^T = \Lambda$$

This allows us to establish variance of two vectors and to estimate ratings based upon approximations comprised of the most "similar" vectors. This method lends well to the Neighborhood method.

# Table of Contents

# Regularized SVD

Another high performing method is Regularized SVD. In this method, ratings are approximated in the same way as we begun in the SVD++ mode, with an inner product of parameter vectors. These vectors are "trained" using the following algorithm:

$$r_{ij} = y_{ij} - \bar{y}_{ij}$$

$$u_{ik} += lrate * (r_{ij} v_{jk} - \lambda u_{ik})$$

$$v_{jk} += lrate * (r_{ij} u_{jk} - \lambda v_{ik})$$

# Regularized SVD



The final prediction is given by

$$\bar{y}_{ij} = c_i + d_j + \mathbf{u}_i^* \mathbf{v}_j$$

where the parameters are

$$c_i += lrate * (r_{ij} - \lambda_2(c_i + d_j - global\_mean)$$

and

$$d_j += lrate * (r_{ij} - \lambda_2(c_i + d_j - global\_mean)$$

This method achieves a root mean squared error (RSME) of 0.9094 (Cinematch scores an RSME of 0.9514)

# Iterative SVD



Iterative SVD is defined as

$$R_{iu}^{(t)} = \begin{cases} R_{iu}, & \text{if } iu \in D \\ [\sum_k U_k S_k V_k^*]_{iu}^{(t-1)}, & \text{otherwise} \end{cases}$$

Where $D$ is the original rating matrix. Then the system attempts to minimize the equation

$$\sum_{iu} (R_{iu}^{(t)} - \mu_{iu})^2$$

given that $\mu_{iu} = [\sum_k U_k S_k V_k^*]_{iu}$.

# Table of Contents

# References



- Y. Koren, R. Bell, C. Volinsky, *Matrix Factorization Techniques for Recommender Systems*. IEE Computer Society, 2009.
- Y. Koren, *The BellKor Solution to the Netflix Grand Prize*. 2009.
- M. Piotte, M. Chabbert, *The Pragmatic Theory Solution to the Netflix Grand Prize*. Pragmatic Theory inc, Canada, 2009.
- A. Toscher, M. Jahrer, *The BigChaos Solution to the Netflix Grand Prize*. AT&T Labs, New Jersey, 2009.
- R. Salakutdinov, A. Mnih, G. Hinton, *Restricted Boltzmann Machines for Collaborative Filtering*. U. of Toronto, Canada, 2007.

# References


- M. Ali Ghanzanfar, A. Prugel-Bennett, *The Advantage of Careful Imputation Sources in Sparse Data-Environment of Recommender Systems: Generating Improved SVD-based Recommendations*. School of Electronics and Computer Science, UK, 2013.
- A. Paterek, *Improving regularized singular value decomposition for collaborative filtering*. Institute of Informatics, Poland, 2007.
- Y. Koren, *Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model*. AT&T Labs, New Jersey, 2008.
- K. Goldberg, T. Roeder, D. Gupta, C. Perkins, *Eigentaste: A Constant Time Collaborative Filtering Algorithm*. IOER and EECS, California, 2000.

# References



- M. Vozalis, K. Margaritis, *Applying SVD on Generalized Item-based Filtering*. International Journal of Computer Science & Applications, 2006