NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update

The Netflix Prize Rules

For a printable copy of these rules, go here.

Overview:

We're quite curious, really. To the tune of one million dollars.

Netflix is all about connecting people to the movies they love. To help customers find those movies, we've developed our world-class movie recommendation system: Cinematch[SM]. Its job is to predict whether someone will enjoy a movie based on how much they liked or disliked other movies. We use those predictions to make personal movie recommendations based on each customer's unique tastes. And while Cinematch is doing pretty well, it can always be made better.

Now there are a lot of interesting alternative approaches to how Cinematch works that we haven't tried. Some are described in the literature, some aren't. We're curious whether any of these can beat Cinematch by making better predictions. Because, frankly, if there is a much better approach it could make a big difference to our customers and our business.

So, we thought we'd make a contest out of finding the answer. It's "easy" really. We provide you with a lot of anonymous rating data, and a prediction accuracy bar that is 10% better than what Cinematch can do on the same training data set. (Accuracy is a measurement of how closely predicted ratings of movies match subsequent actual ratings.) If you develop a system that we judge most beats that bar on the qualifying test set we provide, you get serious money and the bragging rights. But (and you knew there would be a catch, right?) only if you share your method with us and describe to the world how you did it and why it works.

Recommender Systems

- Content filtering approach (profiles)
  - Create and use profiles of users and products to find good matches
- Collaborative filtering approach (no profiles)
  - Neighborhood methods
    - Relationship between users or between products
  - Latent factor models
    - Discover factors from the rating patterns

Ref: Koren, Bell, and Volinsky, Matrix Factorization Techniques for Recommender Systems

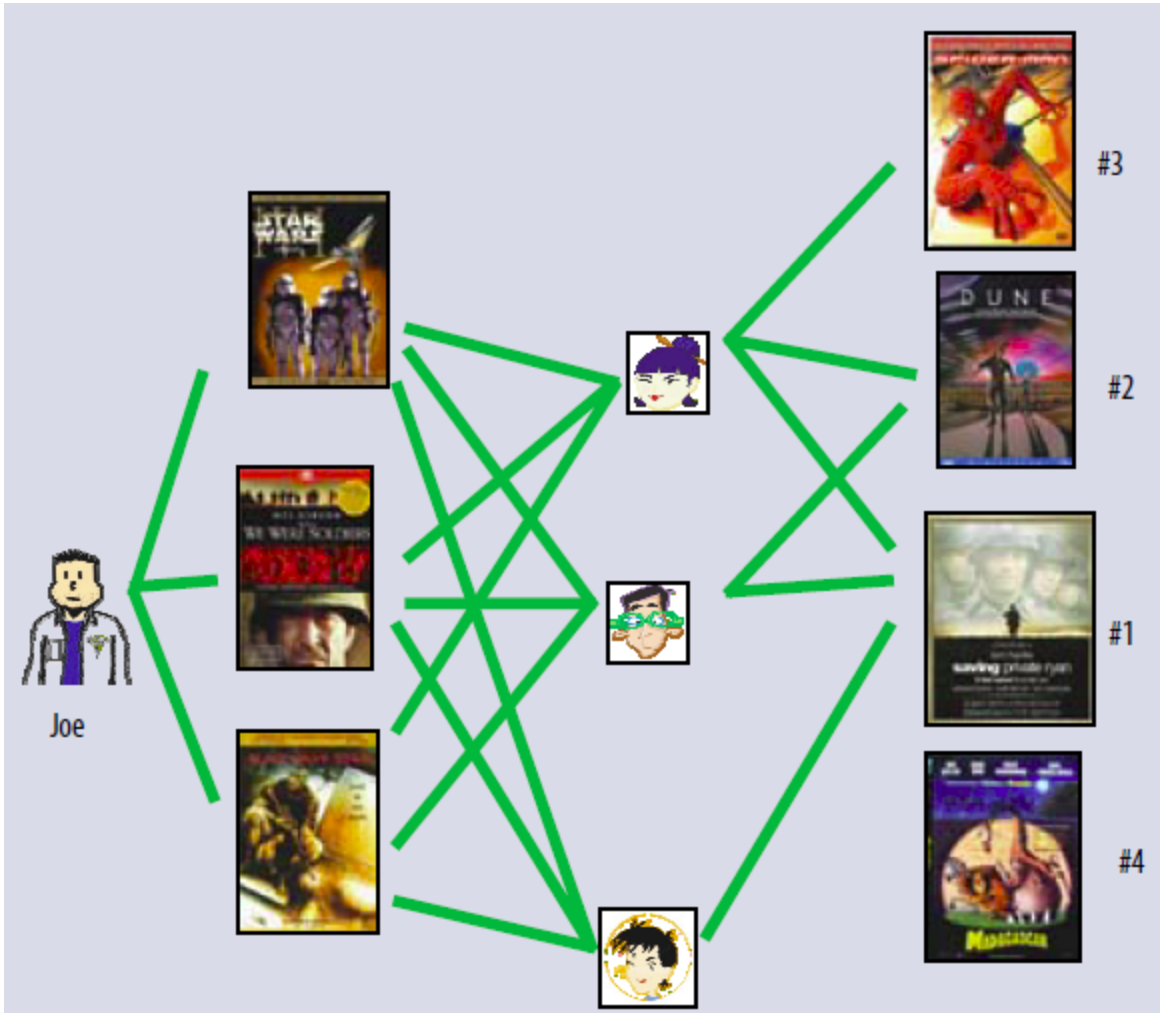

Figure 1. The user-oriented neighborhood method. Joe likes the three movies on the left. To make a prediction for him, the system finds similar users who also liked those movies, and then determines which other movies they liked. In this case, all three liked *Saving Private Ryan*, so that is the first recommendation. Two of them liked *Dune*, so that is next, and so on.

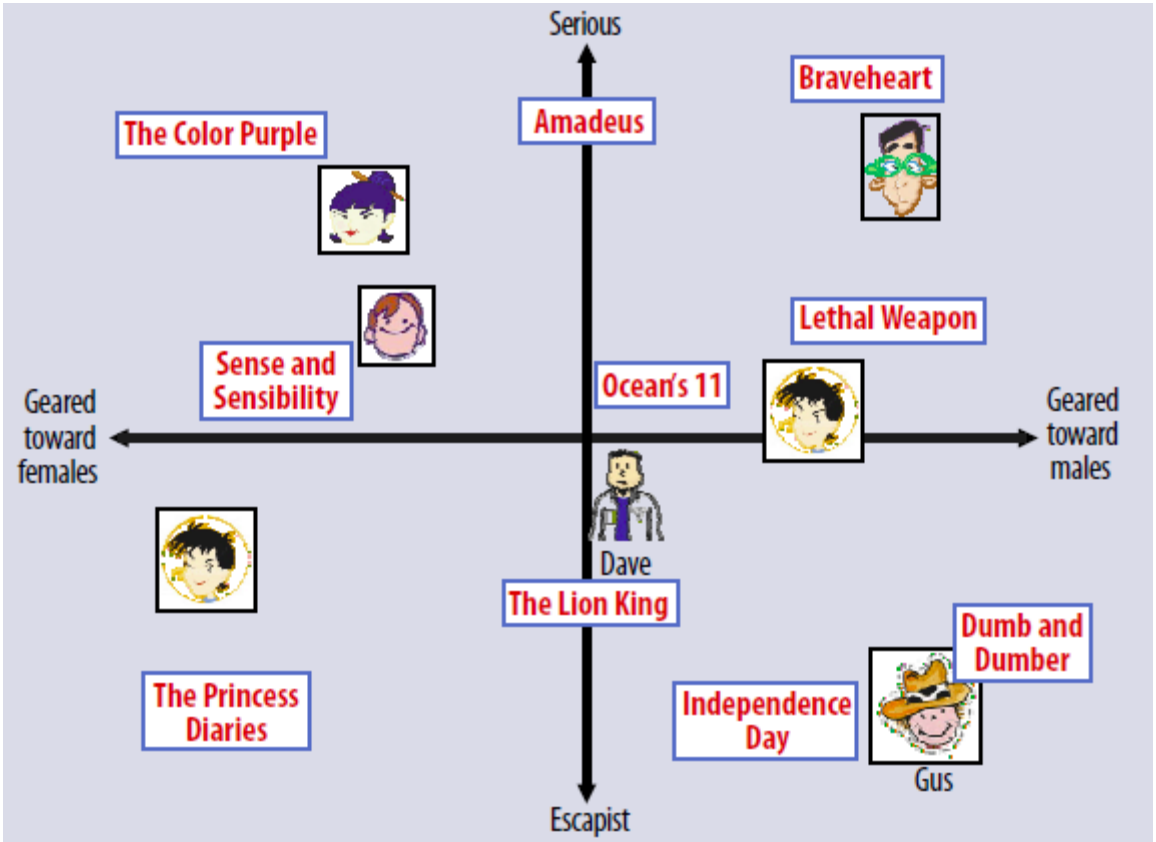

Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

Matrix Factorization Methods

- Example of a latent factor model
- f factors
- Factors for item i is q_i, vector of length f
- Factors for user u is p_u, vector of length f
- Estimated rating by user u for item i is $q_i^T p_u$

Minimization problem

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2$$

- r_{ui} with $(u,i) \in \kappa$ is the set of all known ratings

Regularized minimization problem

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

- To account for noise, solve the regularized minimization problem, where λ is a parameter

Stochastic gradient descent algorithm

Loop until convergence

Loop over all known ratings (training set)

$$q_i = q_i + \gamma(e_{ui}p_u - \lambda q_i)$$

$$p_u = p_u + \gamma(e_{ui}q_i - \lambda p_u)$$

Endloop

Endloop

Where $e_{ui} = r_{ui} - q_i^T p_u$

And γ is a scalar learning rate parameter

How to test for convergence?

- Compute the current value of the “minimum” and stop when it changes very slowly

How to compute predicted ratings?

- Given q_i and p_u for all items and users, compute $q_i^T p_u$

Parallel SGD algorithm

Loop until convergence

Parallel Loop over all known ratings (training set)

$$q_i = q_i + \gamma(e_{ui}p_u - \lambda q_i)$$

$$p_u = p_u + \gamma(e_{ui}q_i - \lambda p_u)$$

Endloop

Endloop

Not the same as the sequential algorithm, as updates may use “old” values

Ref: Niu, Recht, Re, and Wright, “Hogwild!: A lock-free approach to parallelizing stochastic gradient descent”