




Advanced Recommendations with Collaborative Filtering

Remember Recommendations?


Let's review the basics.

Recommendations


NETFLIX Watch Instantly ▾ Just for Kids ▾ Taste Profile ▾ DVDs ▾ DVD Queue

Movies, TV shows, actors, directors, genres  Everaldo ▾

Because you liked The Notebook



Because you liked Justin Bieber: Never Say Never



Recommendations are Everywhere

amazon.com[®]

 reddit

ebay

last.fm

You Tube

 *Grooveshark*

 delicious

Linked in

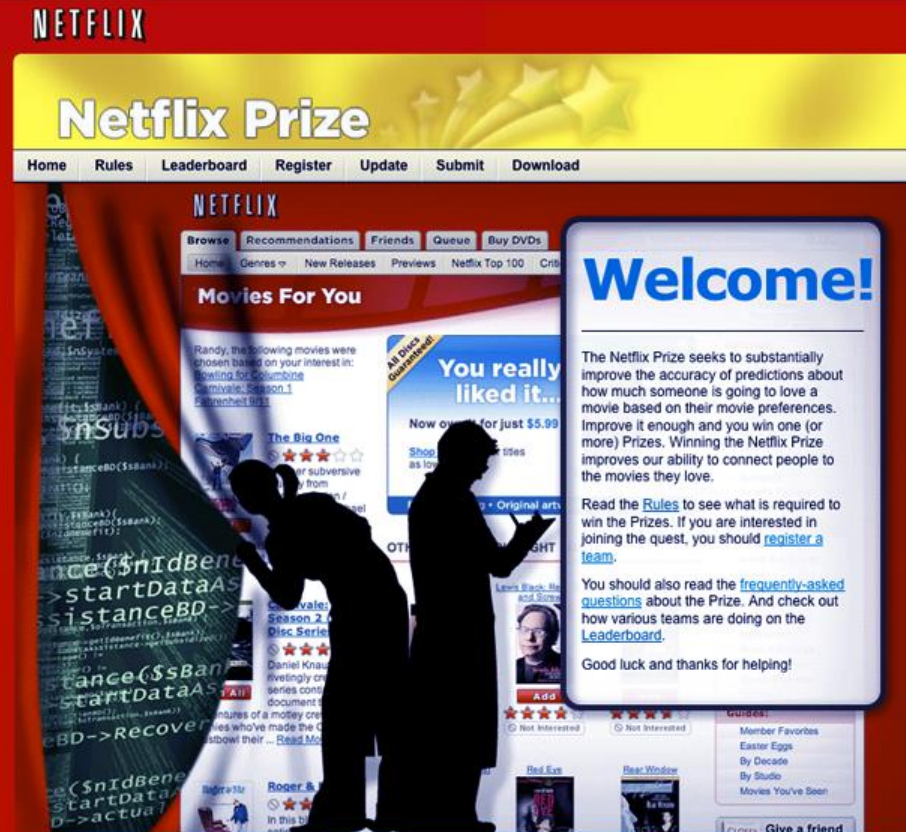
IMDb[®]

NETFLIX

digg



The Netflix Prize (2006-2009)



The Netflix Prize (2006-2009)



What was the Netflix Prize?

- In October, 2006 Netflix released a dataset containing 100 million anonymous movie ratings and challenged the data mining, machine learning, and computer science communities to develop systems that could beat the accuracy of its recommendation system, Cinematch.
- Thus began the Netflix Prize, **an open competition for the best collaborative filtering algorithm to predict user ratings for films**, solely based on previous ratings without any other information about the users or films.

The Netflix Prize Datasets

- Netflix provided a *training* dataset of 100,480,507 ratings that 480,189 users gave to 17,770 movies.
 - Each training rating (or instance) is of the form $\langle \text{user}, \text{movie}, \text{data of rating}, \text{rating} \rangle$.
 - The user and movie fields are integer IDs, while ratings are from 1 to 5 (integral) stars.

The Netflix Prize Datasets

- The *qualifying* dataset contained over 2,817,131 instances of the form ⟨user, movie, date of rating⟩, with ratings known only to the jury.
- A participating team's algorithm had to predict grades on the entire qualifying set, consisting of a *validation* and *test* set.
 - During the competition, teams were only informed of the score for a *validation* or *quiz* set of 1,408,342 ratings.
 - The jury used a *test* set of 1,408,789 ratings to determine potential prize winners.

The Netflix Prize Data

		<i>Movie Ratings</i>			
		1	2	..	m
<i>Users</i>	1	5	2	5	4
	2	2	5		3
	:	2	2	4	2
	n	5	1	5	?

The Netflix Prize Data

Movie Ratings

		<hr/>			
		1	2	..	m
Instances (samples, examples, observations)	1	5	2	5	4
	2	2	5		3
	:	2	2	4	2
	n	5	1	5	?
		<hr/>			

The Netflix Prize Data

Features (attributes, dimensions)

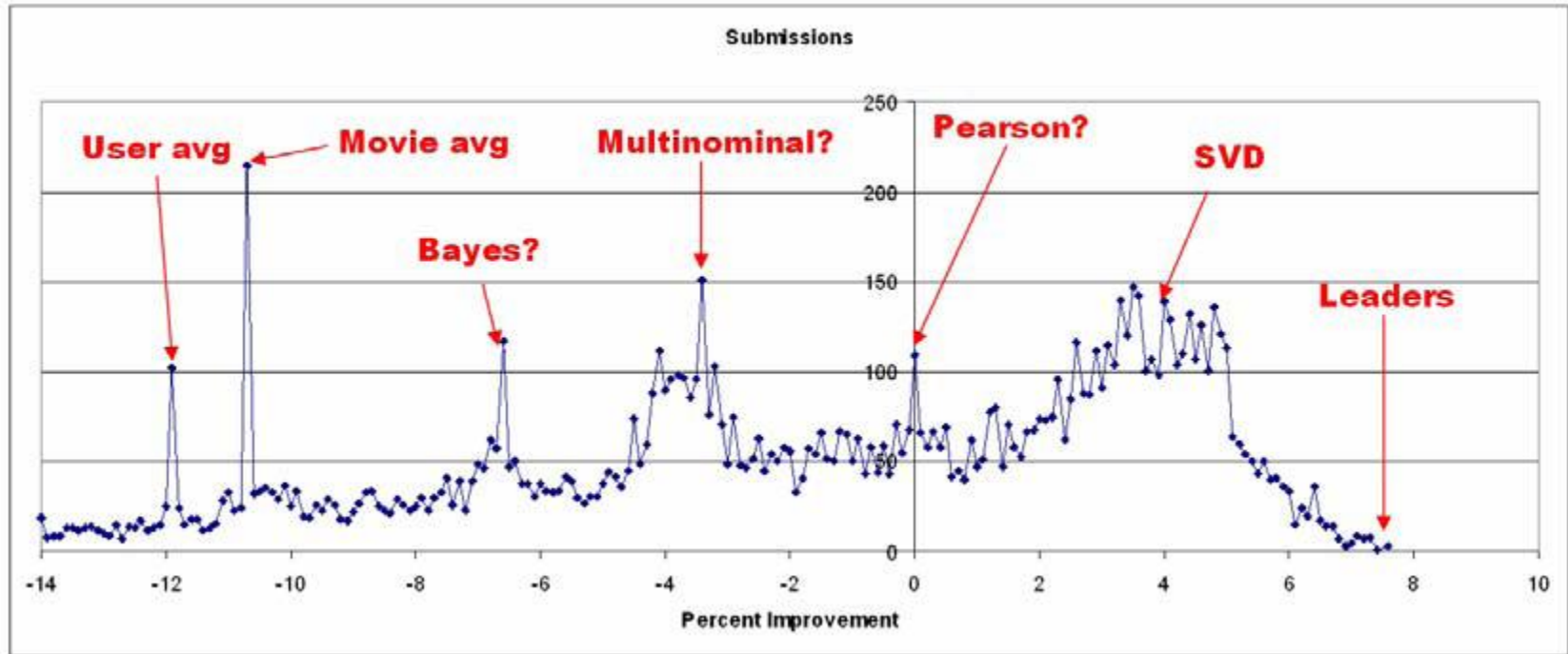
		1	2	..	m
<i>Users</i>	1	5	2	5	4
	2	2	5		3
	:	2	2	4	2
	n	5	1	5	?

The Netflix Prize Goal

		<i>Movie Ratings</i>			
		Star Wars	Hoop Dreams	Contact	Titanic
<i>Users</i>	Joe	5	2	5	4
	John	2	5		3
	Al	2	2	4	2
	Everaldo	5	1	5	?

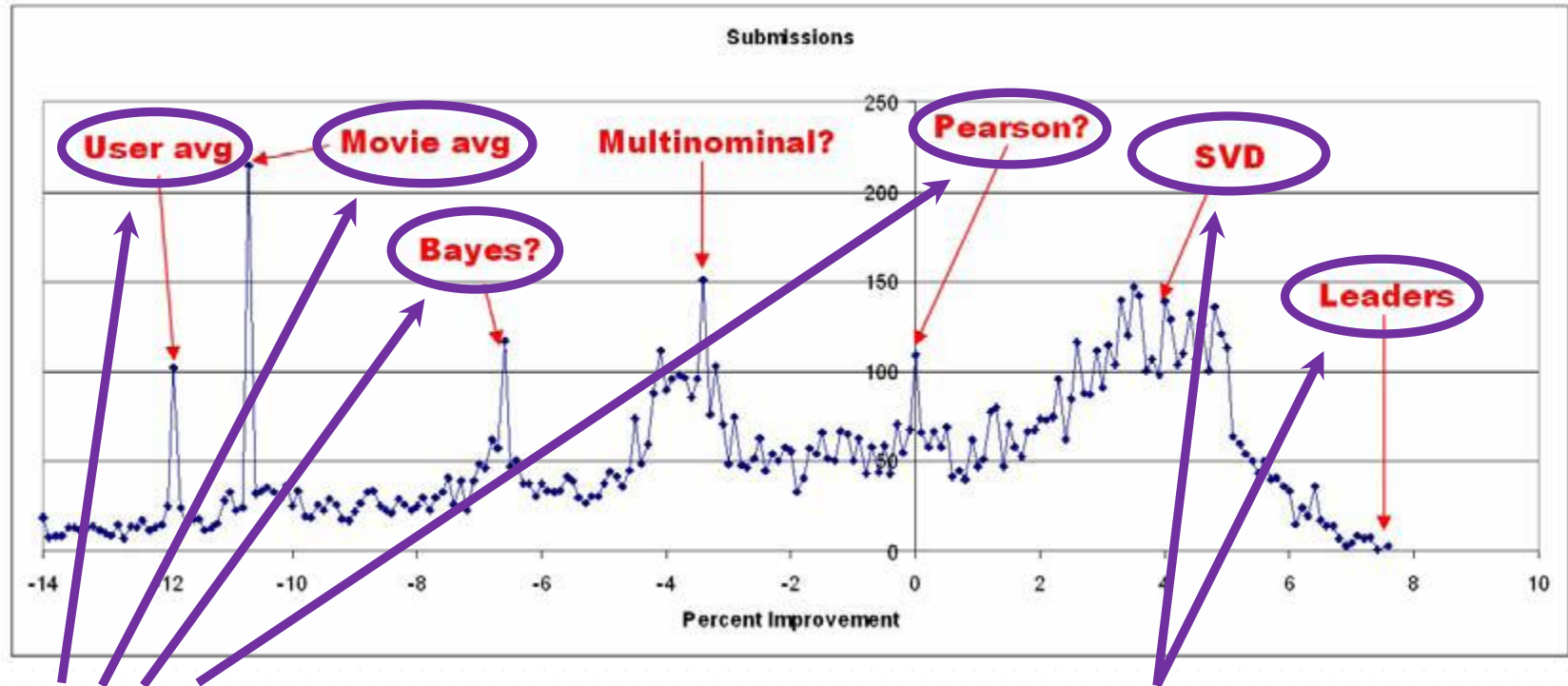
Goal: Predict ? (a movie rating) for a user

The Netflix Prize Methods



Bennett, James, and Stan Lanning. "The Netflix Prize." *Proceedings of KDD Cup and Workshop*. Vol. 2007. 2007.

The Netflix Prize Methods



We discussed these methods.

We will discuss these methods now.

All of these methods are based upon collaborative filtering.

What was that again?

Key to Collaborative Filtering

Common insight: personal tastes are *correlated*

If Alice and Bob both like X and Alice likes Y , then Bob is more likely to like Y , especially (perhaps) if Bob knows Alice.

Types of Collaborative Filtering

- 1 Neighborhood- or Memory-based
- 2 Model-based
- 3 Hybrid

Types of Collaborative Filtering

1 Neighborhood- or Memory-based

We'll talk about this type now.

2 Model-based

3 Hybrid

Neighborhood-based CF

A subset of users are chosen based on their similarity to the active users, and a weighted combination of their ratings is used to produce predictions for this user.

Neighborhood-based CF

It has three steps:

- 1 Assign a weight to all users with respect to similarity with the active user
- 2 Select k users that have the highest similarity with the active user—commonly called the *neighborhood*.
- 3 Compute a prediction from a weighted combination of the selected neighbors' ratings.

Neighborhood-based CF

Step 1

In step 1, the weight $w_{a,u}$ is a measure of similarity between the user u and the active user a . The most commonly used measure of similarity is the Pearson correlation coefficient between the ratings of the two users:

$$w_{a,u} = \frac{\sum_{i \in I} (r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I} (r_{a,i} - \bar{r}_a)^2 \sum_{i \in I} (r_{u,i} - \bar{r}_u)^2}}$$

where I is the set of items rated by both users, $r_{u,i}$ is the rating given to item i by user u , and \bar{r}_u is the mean rating given by user u .

Neighborhood-based CF

Step
2

In step 2, some sort of threshold is used on the similarity score to determine the “neighborhood.”

Neighborhood-based CF

Step
3

In step 3, predictions are generally computed as the weighted average of deviations from the neighbor's mean, as in:

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u \in K} (r_{u,i} - \bar{r}_u) \times w_{a,u}}{\sum_{u \in K} w_{a,u}}$$

where $p_{a,i}$ is the prediction for the active user a for item i , $w_{a,u}$ is the similarity between users a and u , and K is the neighborhood or set of most similar users.

But how do we compute the similarity $w_{a,u}$?

Item-to-Item Matching

- An extension to neighborhood-based CF.
- Addresses the problem of high computational complexity of searching for similar users.
- The idea:

Rather than matching similar users, match a user's rated items to similar items.

Item-to-Item Matching

In this approach, similarities between pairs of items i and j are computed off-line using Pearson correlation, given by:

$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2 \sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}}$$

where U is the set of all users who have rated both items i and j , $r_{u,i}$ is the rating of user u on item i , and \bar{r}_i is the average rating of the i th item across users.

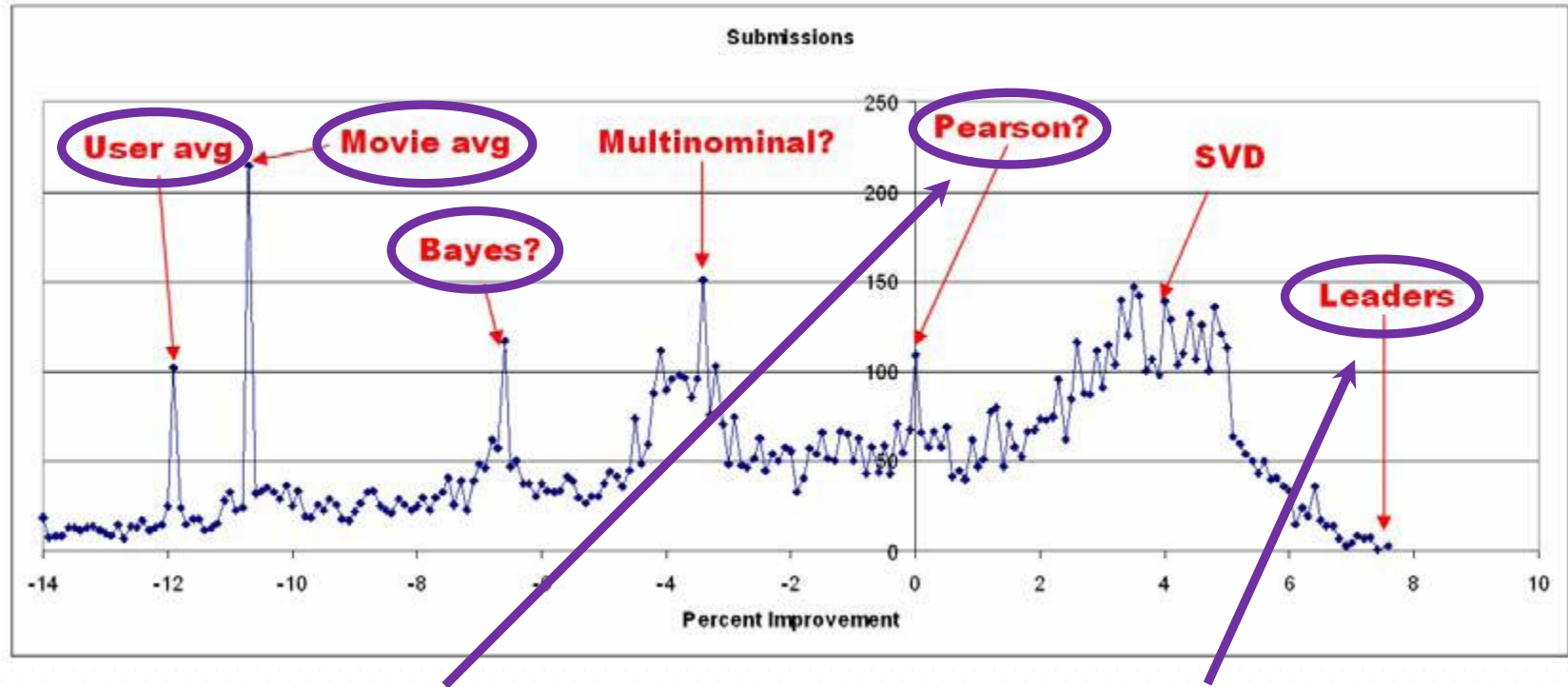
Item-to-Item Matching

Now, the rating for item i for user a can be predicted using a simple weighted average, as in:

$$p_{a,i} = \frac{\sum_{j \in K} r_{u,i} w_{i,j}}{\sum_{j \in K} |w_{i,j}|}$$

where K is the neighborhood set of the k items rated by a that are most similar to i .

The Netflix Prize Methods



Item-oriented collaborative filtering using
Pearson correlation gets us right about here.

So how do we get here?

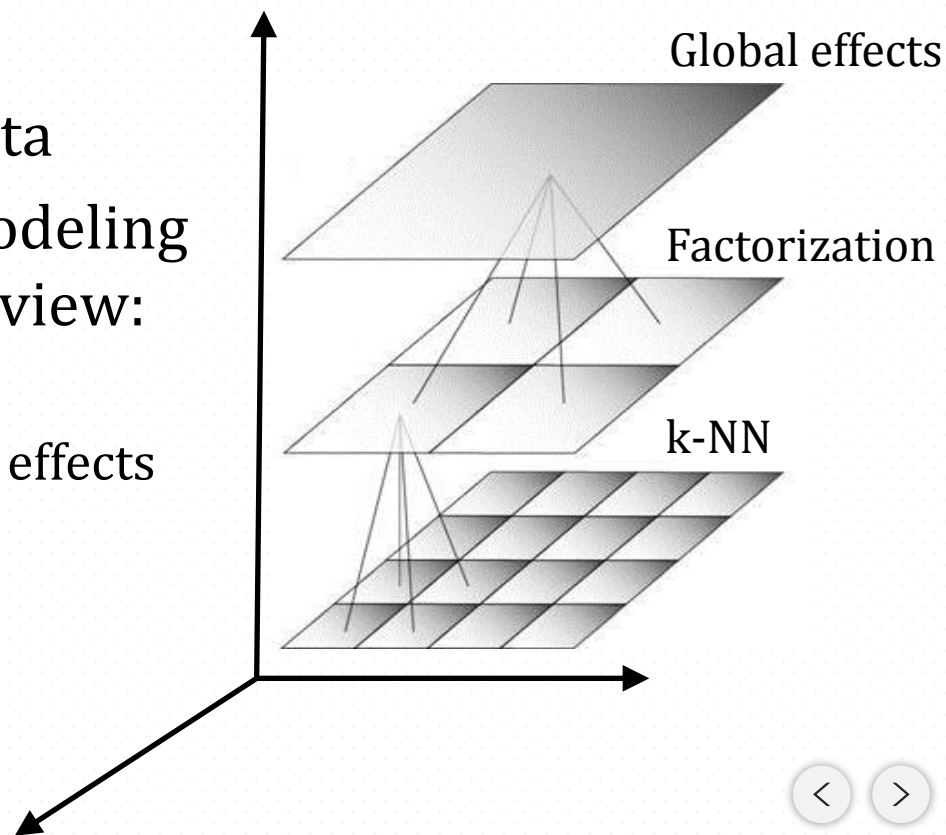
Generalizing the Recommender System

- Use an ensemble of complementing predictors.
- Many seemingly different models expose similar characteristics of the data, and will not mix well.
- Concentrate efforts along three axes.
 - Scale
 - Quality
 - Implicit/explicit

The First Axis: Scale

The first axis:

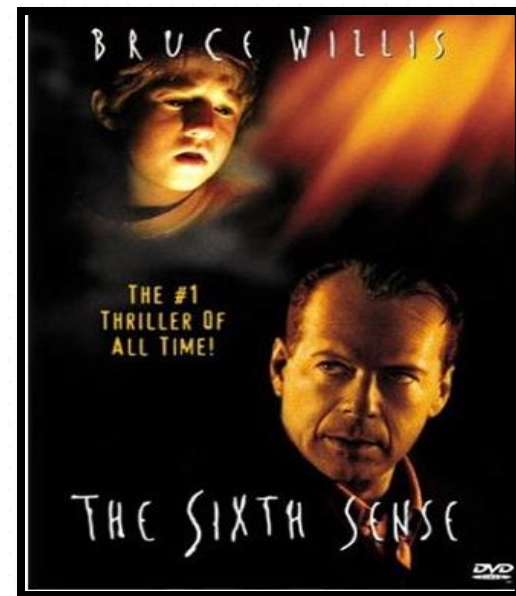
- Multi-scale modeling of the data
- Combine top level, regional modeling of the data, with refined, local view:
 - *kNN*: Extracts local patterns
 - Factorization: Addresses regional effects



Multi-Scale Modeling: 1st Tier

Global effects:

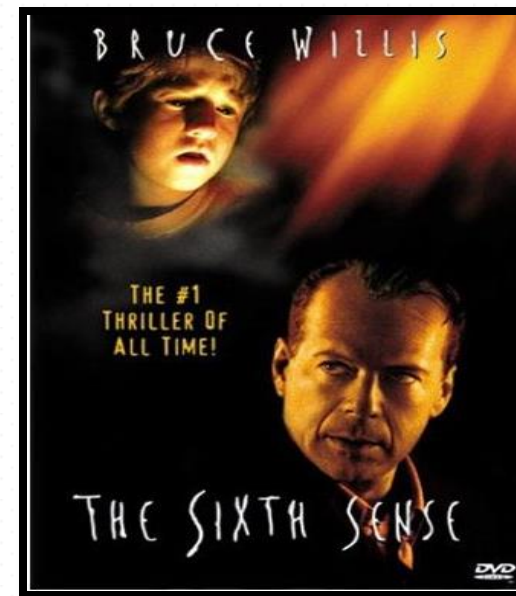
- Mean movie rating: 3.7 stars
- *The Sixth Sense* is 0.5 stars above average
- Joe rates 0.2 stars below average
 - Baseline estimation:
Joe will rate *The Sixth Sense* 4 stars



Multi-Scale Modeling: 2nd Tier

Factors model:

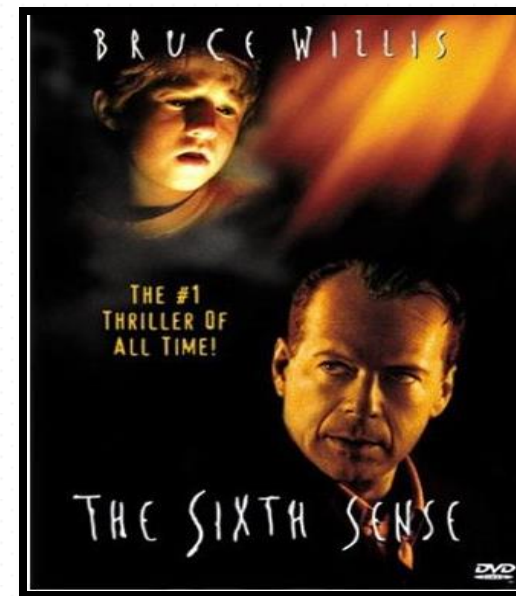
- Both *The Sixth Sense* and *Joe* are placed high on the “Supernatural Thrillers” scale
- Adjusted estimate:
Joe will rate *The Sixth Sense* 4.5 stars



Multi-Scale Modeling: 3rd Tier

Neighborhood Model

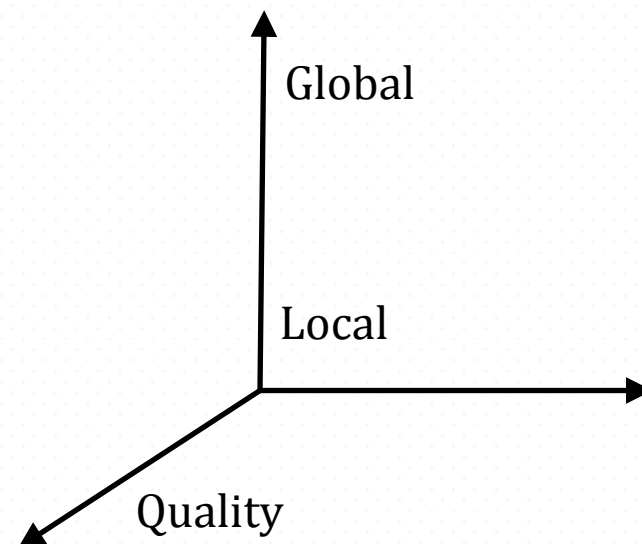
- *Joe* didn't like related movie *Signs*
- Final estimate:
Joe will rate *The Sixth Sense* 4.2 stars



The Second Axis: Model Quality

The second axis:

- Quality of modeling
- Make the best out of a model
- Strive for:
 - Fundamental derivation
 - Simplicity
 - Avoid overfitting
 - Robustness against number of iterations, parameter settings, etc.
- Optimizing is good, but don't overdo it!



Local Modeling via k NN

- Earliest and most popular collaborative filtering method.
- Derive unknown ratings from those of “similar” items (**movie-movie** variant).
- A parallel **user-user** flavor.
 - Rely on ratings of like-minded users

Collaborative Filtering with k NN

		Users											
		1	2	3	4	5	6	7	8	9	10	11	12
Movies	1	1		3			5			5		4	
	2			5	4			4			2	1	3
	3	2	4		1	2		3		4	3	5	
	4		2	4		5			4			2	
	5			4	3	4	2					2	5
	6	1		3		3			2			4	



— Unknown rating



— Rating from 1 to 5



Collaborative Filtering with k NN

		Users											
		1	2	3	4	5	6	7	8	9	10	11	12
Movies	1	1		3		?	5			5		4	
	2			5	4			4			2	1	3
	3	2	4		1	2		3		4	3	5	
	4		2	4		5			4			2	
	5			4	3	4	2					2	5
	6	1		3		3			2			4	



— Estimate rating of movie 1 by user 5

Collaborative Filtering with k NN

		Users											
		1	2	3	4	5	6	7	8	9	10	11	12
Movies	1	1		3		?	5			5		4	
	2			5	4			4			2	1	3
	3	2	4		1	2		3		4	3	5	
	4		2	4		5			4			2	
	5			4	3	4	2					2	5
	6	1		3		3			2			4	

Neighbor selection: Identify movies similar to 1, rated by user 5

Collaborative Filtering with k NN

		Users											
		1	2	3	4	5	6	7	8	9	10	11	12
Movies	1	1		3		?	5			5		4	
	2			5	4			4			2	1	3
	3	2	4		1	2		3		4	3	5	
	4		2	4		5			4			2	
	5			4	3	4	2					2	5
	6	1		3		3			2			4	

Compute similarity weights: $s_{13} = 0.2, s_{16} = 0.3$

Collaborative Filtering with k NN

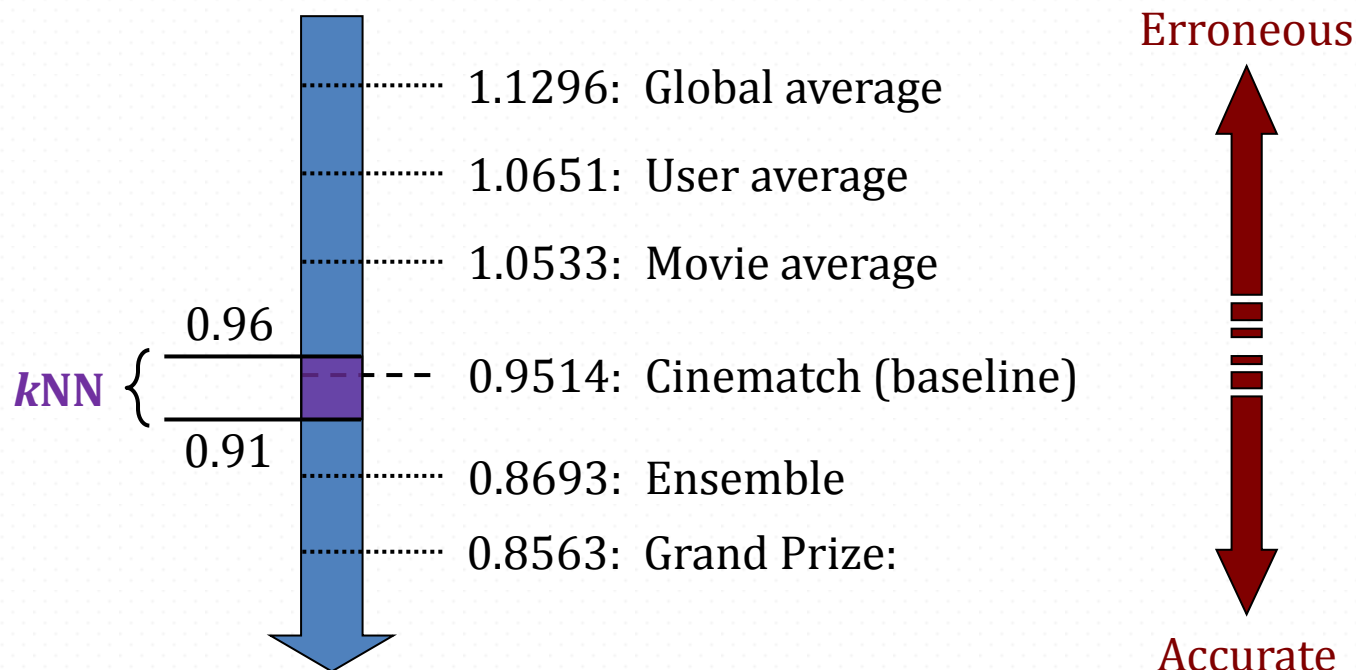
		Users											
		1	2	3	4	5	6	7	8	9	10	11	12
Movies	1	1		3		2.6	5			5		4	
	2			5	4			4			2	1	3
	3	2	4		1	2		3		4	3	5	
	4		2	4		5			4			2	
	5			4	3	4	2					2	5
	6	1		3		3			2			4	

Predict by taking weighted average: $(0.2 \times 2 + 0.3 \times 3)/(0.2 + 0.3) = 2.6$

Properties of k NN

- Intuitive.
- No substantial preprocessing is required.
- Easy to explain reasoning behind a recommendation.
- Accurate?

k NN on the Error (RMSE) Scale



Item-Oriented k NN CF

- Problems:

- Suppose that a particular item is predicted perfectly by a subset of the neighbors, where the predictive subset should receive all the weight. Pearson correlation cannot do this.
- Suppose the neighbors set contains three movies that are highly correlated with each other. Basic neighborhood methods do not account for interactions among neighbors.
- Suppose that an item has no useful neighbors rated by a particular user. The standard formula uses a weighted average of rates for the uninformative neighbors.

Interpolation Weights

To address the problem of arbitrary similarity measures, we can use a weighted sum rather than a weighted average:

$$p_{a,i} = \bar{r}_a + \sum_{u \in K} (r_{u,i} - \bar{r}_u) \times w_{a,u}$$

Now, we can allow $\sum_{u \in K} w_{a,u} \neq 1$.

Interpolation Weights

To address the other problems, we can model relationships between item i and its neighbors. This can be learned through a least squares problem from all other users that rated i :

$$\min_w \sum_{v \neq K} \left((r_{vi} - b_{vi}) - \sum_{u \in K} w_{a,u} (r_{vu} - b_{vu}) \right)^2$$

Interpolation Weights

The Result:

- Interpolation weights derived based on their role; no use of an arbitrary similarity measure.
- Explicitly account for interrelationships among the neighbors.

Challenges:

- Dealing with missing values.
- Avoiding overfitting.
- Efficient implementation.

From Local to Latent Trends

*Inherently, nearest neighbors is a local technique.
What about capturing non-local, or latent, trends?*

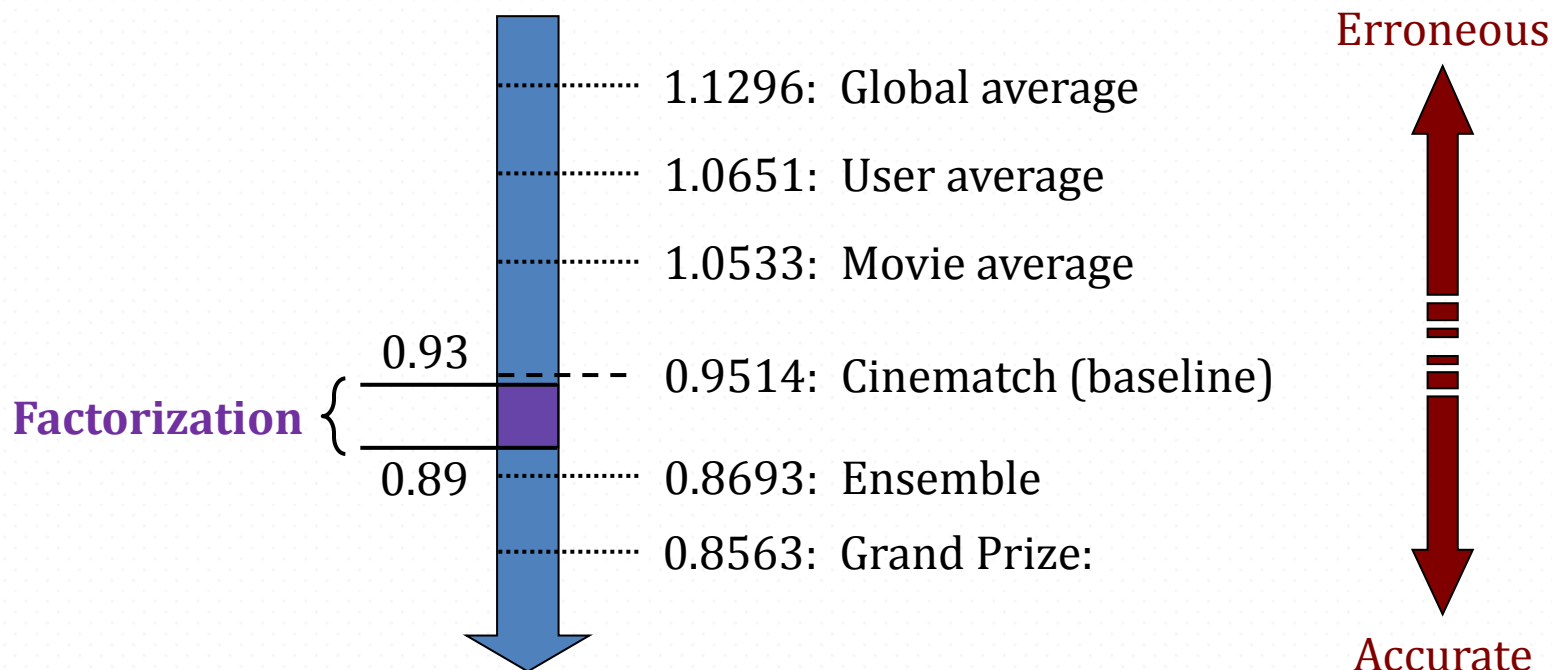
Latent Factor Models

- Decompose user ratings on movies into separate item and movie matrices to capture latent factors. Frequently performed using singular value decomposition (SVD).
- Estimate unknown ratings as inner-products of factors.

Ratings												Movies			Users											
1		3			5			5		4		.1	-.4	.2	1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
		5	4			4			2	1	3	-.5	.6	.5	-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2	4		1	2		3		4	3	5		-.2	.3	.5	2.1											
	2	4		5			4			2		1.1	2.1	.3	-.7	2.1	-2			.9	-.3	.4	.8	.7	-.6	.1
		4	3	4	2					2	5	-1	.7	.3												
1		3		3			2			4																

- Very powerful model, but can easily overfit.

Factorization on the Error (RMSE) Scale



Ensemble Creation

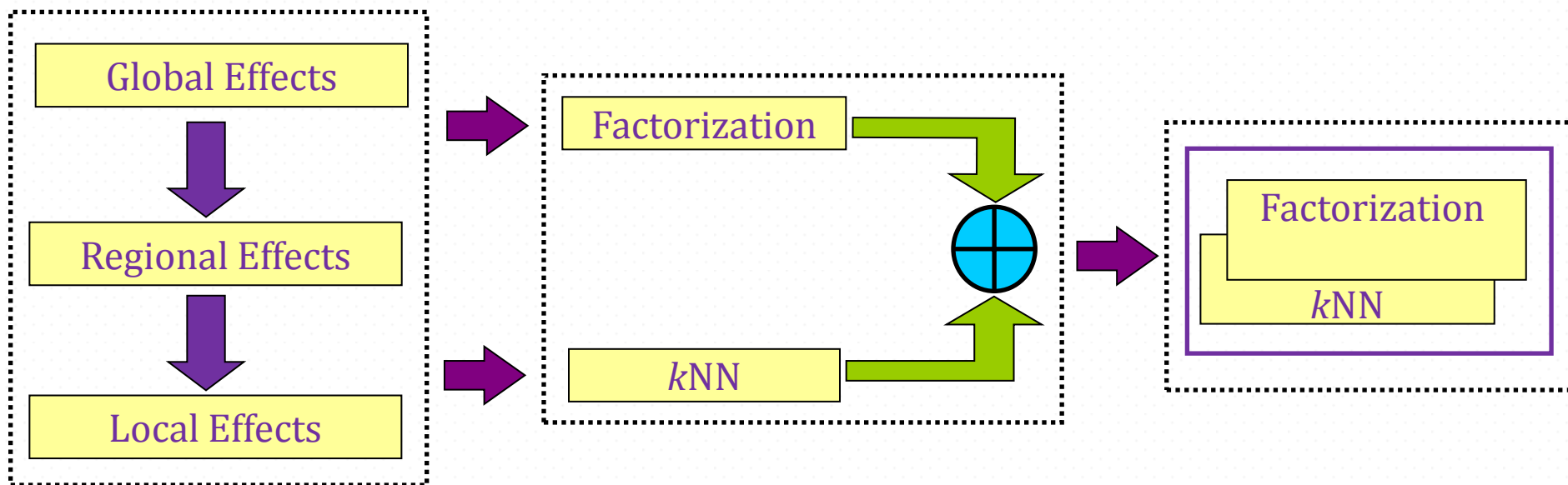
- Factorization and k NN models are used at various scales.
- These models can be combined to form an ensemble.
- Stacked generalization or blending is used.
 - A linear regression model can be trained over the base model predictions.
 - Models can be weighted differently at different scales.

Combining Multi-Scale Views

Residual Fitting

Weighted Average

A Unified Model



Seek Alternative Perspectives

*The previous models all address the movies.
The problem, however, is about users!*

The Third Axis: Implicit Information

- Improve accuracy by exploiting **implicit feedback**.
- Implicit behavior is abundant and easy to collect:
 - Rental history, search patterns, browsing history, etc.
- Allows predicting personalized ratings for users that never rated.

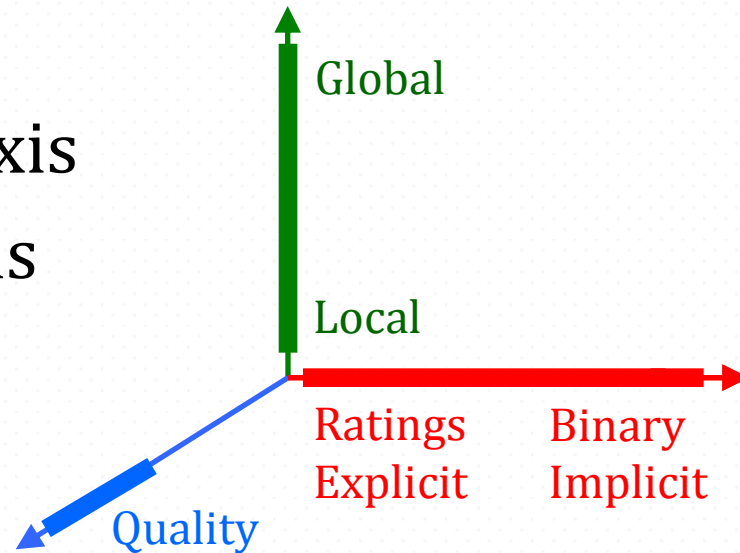
The Idea:

Characterize users by *which* movies
they rated, rather than *how* they rated.

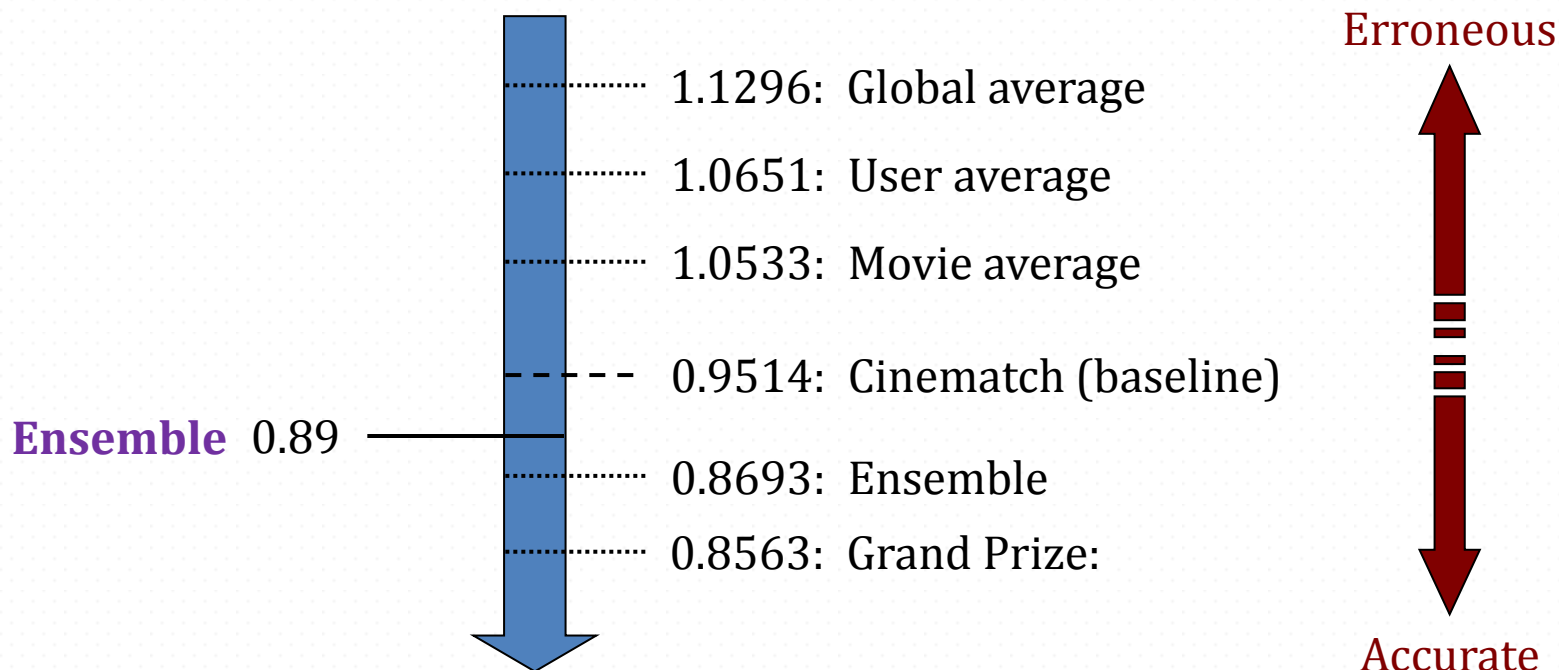
The Big Picture

Where do you want to be?

- All over the global-local axis
- Relatively high on the quality axis
- All over the explicit-implicit axis



Ensemble on the Error (RMSE) Scale



The Take-Away Messages

Solving challenging data mining and data science problems require you to:

1. Think deeply

- Design better, more innovative algorithms.

2. Think broadly

- Use ensembles of multiple predictors.

3. Think differently

- Model the data from different perspectives and in different ways.

