10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Midterm Review
+
Ensemble Methods
+
Recommender Systems

Matt Gormley
Lecture 21
Nov. 4, 2019

Reminders

- Homework 6: Information Theory / Generative Models
  - Out: Fri, Oct. 25
  - Due: Fri, Nov. 8 at 11:59pm
- Midterm Exam 2
  - Thu, Nov. 14, 6:30pm – 8:00pm
  - more details announced on Piazza
- Homework 7: HMMs
  - Out: Fri, Nov. 8
  - Due: Sun, Nov. 24 at 11:59pm

- Today's In-Class Poll
  - http://p21.mlcourse.org

MIDTERM EXAM LOGISTICS

Midterm Exam

- **Time / Location**
  - **Time:** Evening Exam
    Thu, Nov. 14 at 6:30pm – 8:00pm
  - **Room:** We will contact each student individually with **your room assignment.** The rooms are **not** based on section.
  - **Seats:** There will be **assigned seats.** Please arrive early.
  - Please watch Piazza carefully for announcements regarding room / seat assignments.
- **Logistics**
  - Covered material: Lecture 9 – Lecture 19 (95%), Lecture 1 – 8 (5%)
  - Format of questions:
    - Multiple choice
    - True / False (with justification)
    - Derivations
    - Short answers
    - Interpreting figures
    - Implementing algorithms on paper
  - No electronic devices
  - You are allowed to **bring** one 8½ x 11 sheet of notes (front and back)

Midterm Exam

- **How to Prepare**
  - Attend the midterm review lecture (right now!)
  - Review prior year’s exam and solutions (we’ll post them)
  - Review this year’s homework problems
  - Consider whether you have achieved the “learning objectives” for each lecture / section

Midterm Exam

- **Advice (for during the exam)**
  - Solve the easy problems first (e.g. multiple choice before derivations)
    - if a problem seems extremely complicated you're likely missing something
  - Don't leave any answer blank!
  - If you make an assumption, write it down
  - If you look at a question and don't know the answer:
    - we probably haven't told you the answer
    - but we've told you enough to work it out
    - imagine arguing for some answer and see if you like it

Topics for Midterm 1

- Foundations
  - Probability, Linear Algebra, Geometry, Calculus
  - Optimization
- Important Concepts
  - Overfitting
  - Experimental Design
- Classification
  - Decision Tree
  - KNN
  - Perceptron
- Regression
  - Linear Regression

Topics for Midterm 2

- Classification
  - Binary Logistic Regression
  - Multinomial Logistic Regression
- Important Concepts
  - Regularization
  - Feature Engineering
- Feature Learning
  - Neural Networks
  - Basic NN Architectures
  - Backpropagation
- Reinforcement Learning
  - Value Iteration
  - Policy Iteration
  - Q-Learning
  - Deep Q-Learning
- Learning Theory
  - Information Theory

SAMPLE QUESTIONS

Sample Questions

3.2 Logistic regression

Given a training set $\{(x_i, y_i), i = 1, \ldots, n\}$ where $x_i \in \mathbb{R}^d$ is a feature vector and $y_i \in \{0, 1\}$ is a binary label, we want to find the parameters $\hat{w}$ that maximize the likelihood for the training set, assuming a parametric model of the form

$$p(y = 1|x; w) = \frac{1}{1 + \exp(-w^T x)}.$$

The conditional log likelihood of the training set is

$$\ell(w) = \sum_{i=1}^{n} y_i \log p(y_i, |x_i; w) + (1 - y_i) \log(1 - p(y_i, |x_i; w)),$$

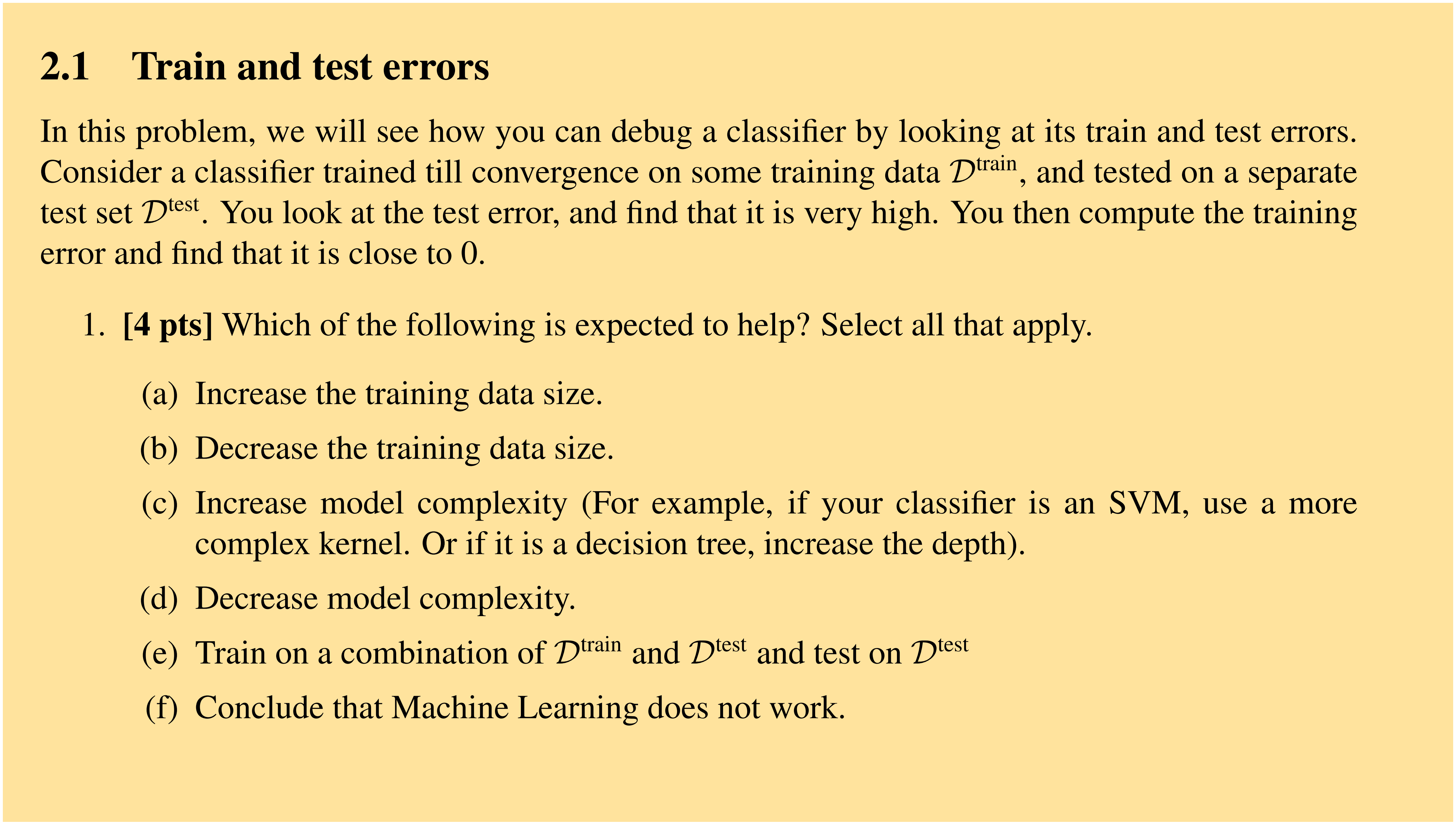
and the gradient is

$$\nabla \ell(w) = \sum_{i=1}^{n} (y_i - p(y_i|x_i; w)) x_i.$$

(b) [5 pts.] What is the form of the classifier output by logistic regression?

(c) [2 pts.] **Extra Credit:** Consider the case with binary features, i.e, $x \in \{0, 1\}^d \subset \mathbb{R}^d$, where feature x_1 is rare and happens to appear in the training set with only label 1. What is $\hat{w}_1$? Is the gradient ever zero for any finite w? Why is it important to include a regularization term to control the norm of $\hat{w}$?

Samples Questions



2.1 Train and test errors

In this problem, we will see how you can debug a classifier by looking at its train and test errors. Consider a classifier trained till convergence on some training data $\mathcal{D}^{\text{train}}$, and tested on a separate test set $\mathcal{D}^{\text{test}}$. You look at the test error, and find that it is very high. You then compute the training error and find that it is close to 0.

1. **[4 pts]** Which of the following is expected to help? Select all that apply.

   (a) Increase the training data size.

   (b) Decrease the training data size.

   (c) Increase model complexity (For example, if your classifier is an SVM, use a more complex kernel. Or if it is a decision tree, increase the depth).

   (d) Decrease model complexity.

   (e) Train on a combination of $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ and test on $\mathcal{D}^{\text{test}}$

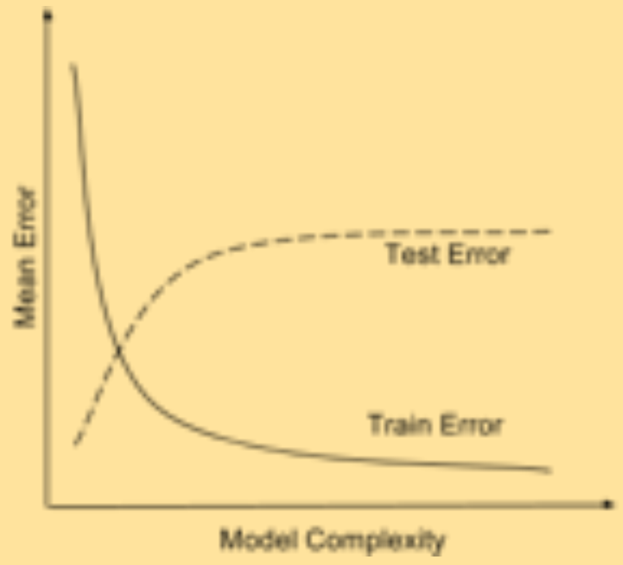
   (f) Conclude that Machine Learning does not work.

Samples Questions

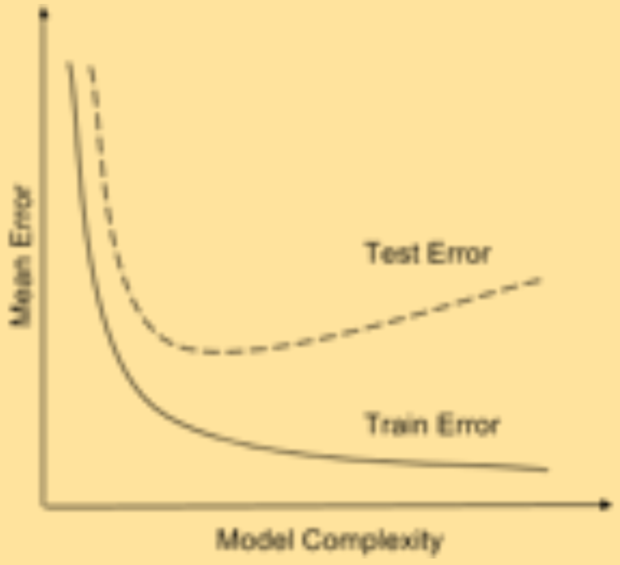
2.1 Train and test errors

In this problem, we will see how you can debug a classifier by looking at its train and test errors. Consider a classifier trained till convergence on some training data $\mathcal{D}^{\text{train}}$, and tested on a separate test set $\mathcal{D}^{\text{test}}$. You look at the test error, and find that it is very high. You then compute the training error and find that it is close to 0.

4. **[1 pts]** Say you plot the train and test errors as a function of the model complexity. Which of the following two plots is your plot expected to look like?



(a)

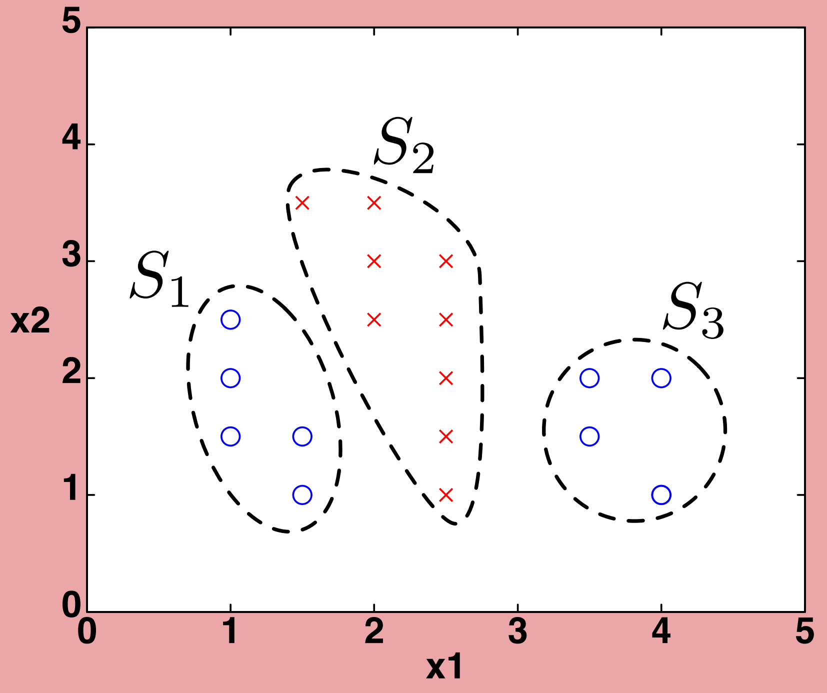


(b)

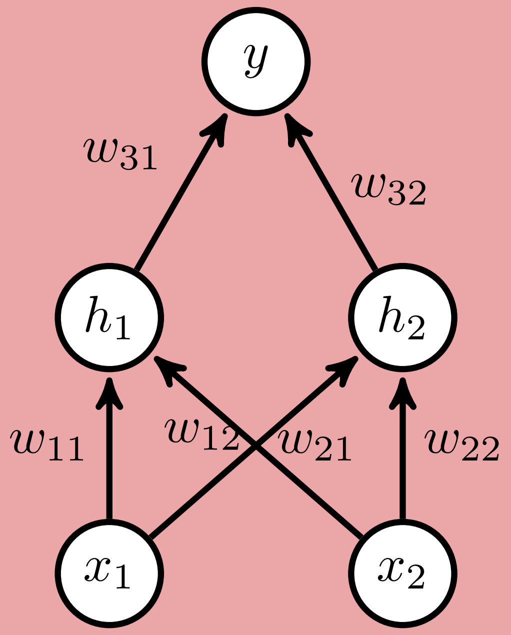
Sample Questions

Neural Networks

Can the neural network in Figure (b) correctly classify the dataset given in Figure (a)?



(a) The dataset with groups S_1, S_2, and S_3.

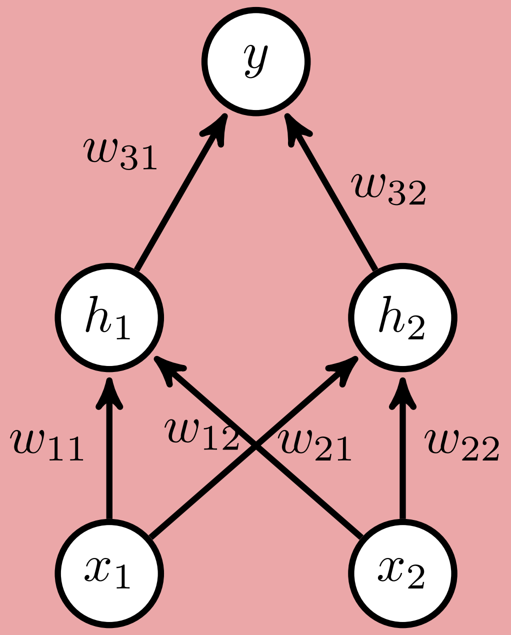


(b) The neural network architecture

Sample Questions

Neural Networks

Apply the backpropagation algorithm to obtain the partial derivative of the mean-squared error of y with the true value y* with respect to the weight w_{22} assuming a sigmoid nonlinear activation function for the hidden layer.



(b) The neural network architecture

Sample Questions

7.1 Reinforcement Learning

3. (1 point) **Please select one statement that is true for reinforcement learning and supervised learning.**

    ◯ Reinforcement learning is a kind of supervised learning problem because you can treat the reward and next state as the label and each state, action pair as the training data.

    ◯ Reinforcement learning differs from supervised learning because it has a temporal structure in the learning process, whereas, in supervised learning, the prediction of a data point does not affect the data you would see in the future.

4. (1 point) **True or False:** Value iteration is better at balancing exploration and exploitation compared with policy iteration.

    ◯ True

    ◯ False

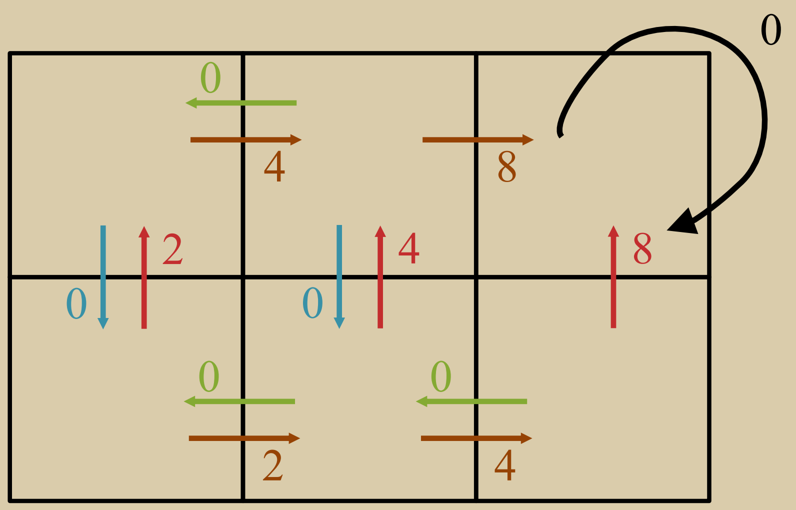
Sample Questions

7.1 Reinforcement Learning

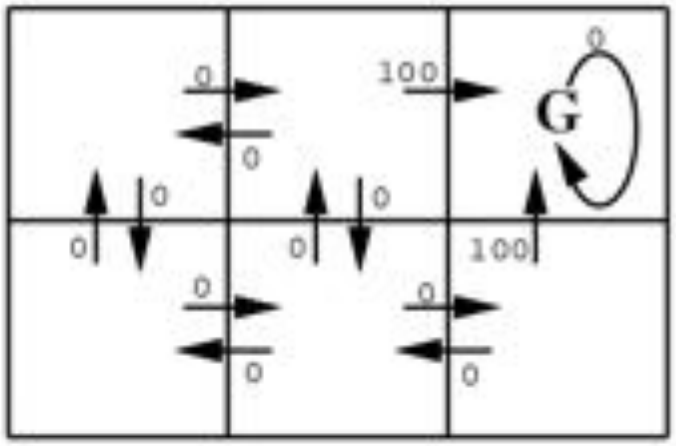
1. For the R(s,a) values shown on the arrows below, what is the corresponding optimal policy? Assume the discount factor is 0.1

2. For the R(s,a) values shown on the arrows below, which are the corresponding V*(s) values? Assume the discount factor is 0.1

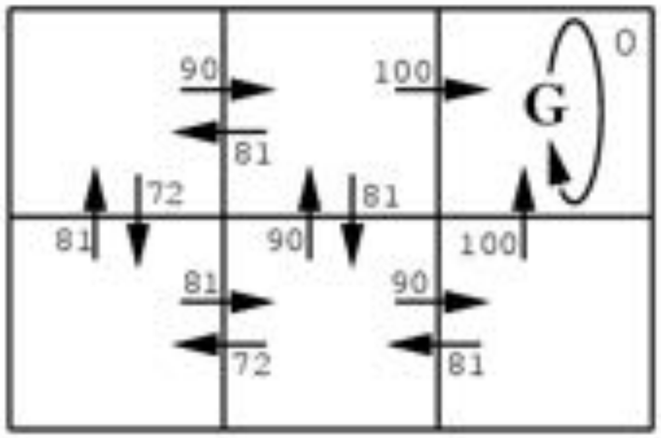
3. For the R(s,a) values shown on the arrows below, which are the corresponding Q*(s,a) values? Assume the discount factor is 0.1



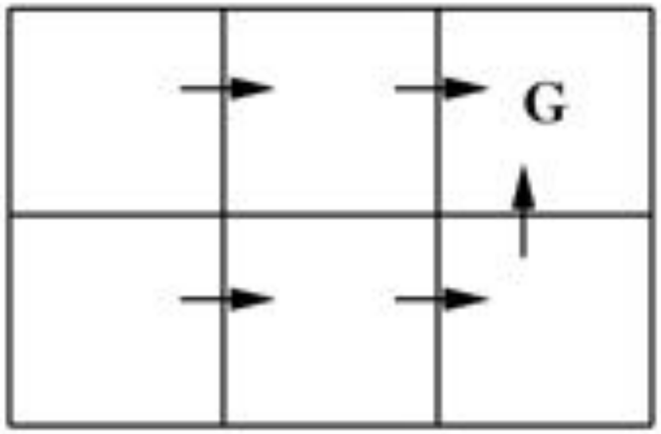
Example: Robot Localization



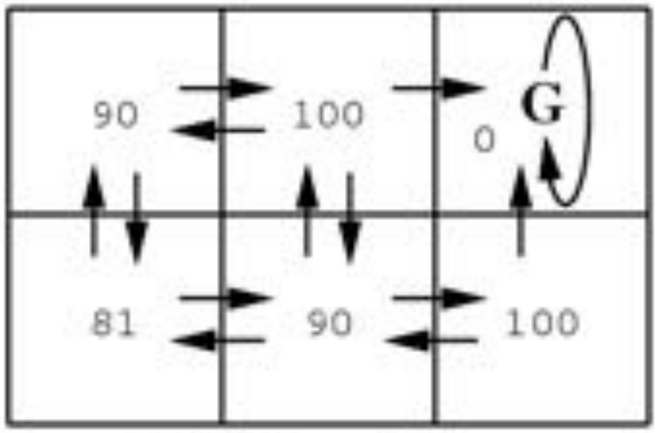
$r(s,a)$ (immediate reward) values



$Q(s,a)$ values



One optimal policy



$V^*(s)$ values

Figure from Tom Mitchell

ML Big Picture

Learning Paradigms:

What data is available and when? What form of prediction?

- supervised learning
- unsupervised learning
- semi-supervised learning
- reinforcement learning
- active learning
- imitation learning
- domain adaptation
- online learning
- density estimation
- recommender systems
- feature learning
- manifold learning
- dimensionality reduction
- ensemble learning
- distant supervision
- hyperparameter optimization

Theoretical Foundations:

What principles guide learning?

- ❑ probabilistic
- ❑ information theoretic
- ❑ evolutionary search
- ❑ ML as optimization

Problem Formulation:

What is the structure of our output prediction?

boolean	Binary Classification
categorical	Multiclass Classification
ordinal	Ordinal Classification
real	Regression
ordering	Ranking
multiple discrete	Structured Prediction
multiple continuous	(e.g. dynamical systems)
both discrete & cont.	(e.g. mixed graphical models)

Application Areas

Key challenges?

NLP, Speech, Computer Vision, Robotics, Medicine, Search

Facets of Building ML Systems:

How to build systems that are robust, efficient, adaptive, effective?

1. *Data prep*
2. *Model selection*
3. *Training (optimization / search)*
4. *Hyperparameter tuning on validation data*
5. *(Blind) Assessment on test data*

Big Ideas in ML:

Which are the ideas driving development of the field?

- *inductive bias*
- *generalization / overfitting*
- *bias-variance decomposition*
- *generative vs. discriminative*
- *deep nets, graphical models*
- *PAC learning*
- *distant rewards*

Outline for Today

We'll talk about two distinct topics:

1. **Ensemble Methods:** combine or learn multiple classifiers into one
   (i.e. a family of algorithms)
2. **Recommender Systems:** produce recommendations of what a user will like
   (i.e. the solution to a particular type of task)

We'll use a prominent example of a recommender systems (the Netflix Prize) to motivate both topics…

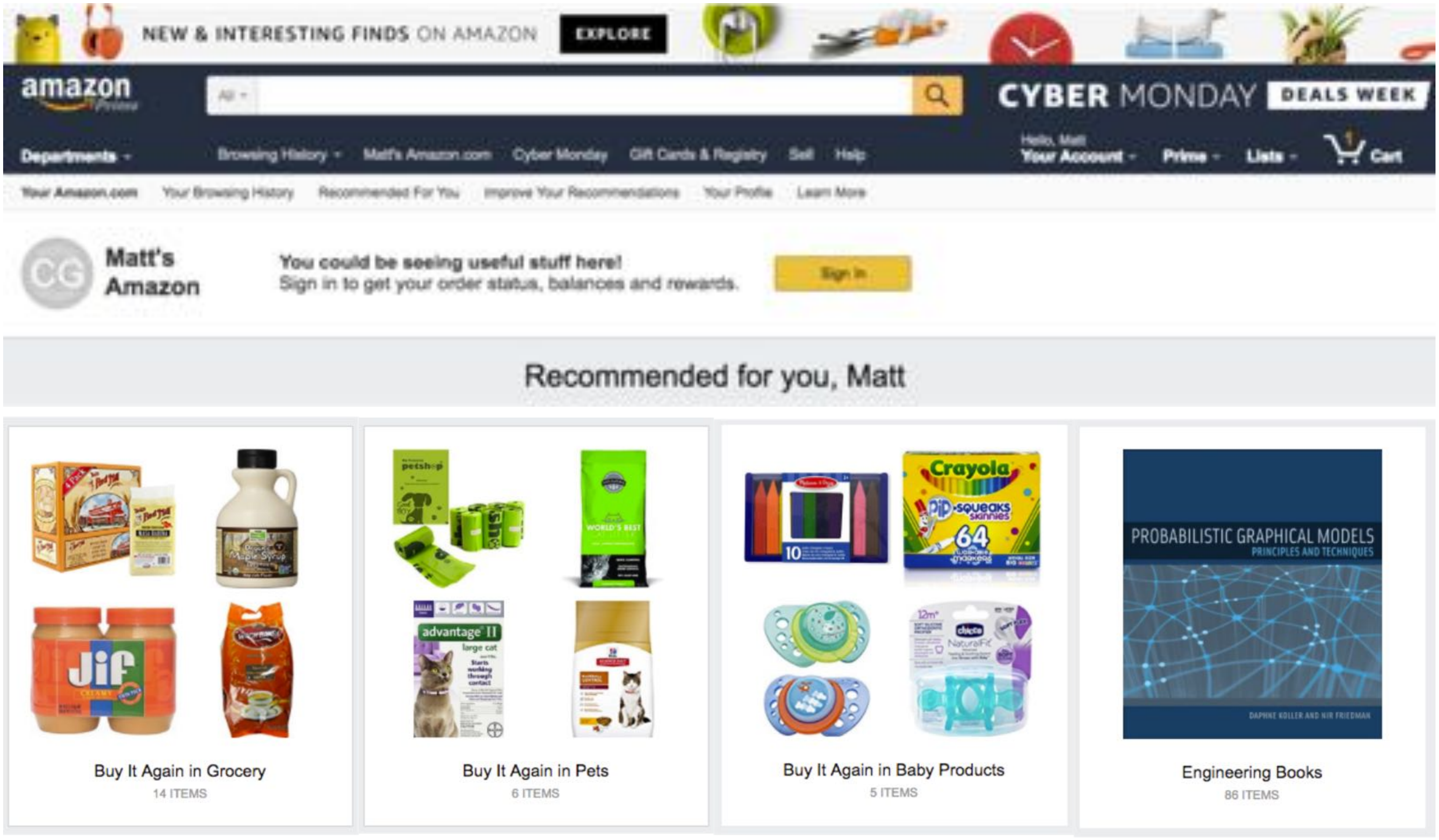
RECOMMENDER SYSTEMS

Recommender Systems

A Common Challenge:

- Assume you're a company selling **items** of some sort: movies, songs, products, etc.
- Company collects millions of **ratings** from **users** of their **items**
- To maximize profit / user happiness, you want to **recommend** items that users are likely to want

Recommender Systems



You could be seeing useful stuff here!
Sign in to get your order status, balances and rewards.

Recommended for you, Matt

Buy It Again in Grocery
14 ITEMS

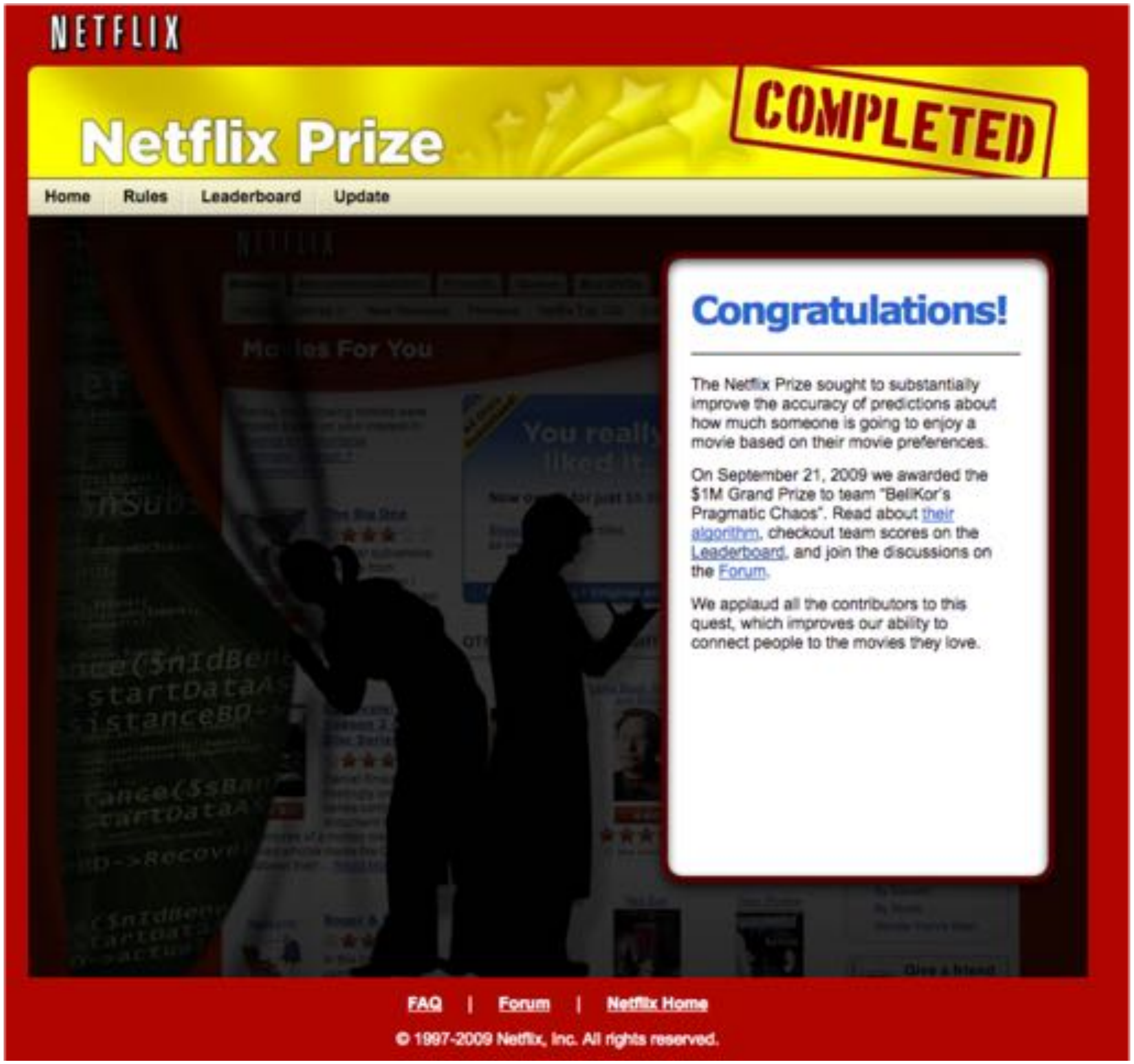
Buy It Again in Pets
6 ITEMS

Buy It Again in Baby Products
5 ITEMS

Engineering Books
86 ITEMS

PROBABILISTIC GRAPHICAL MODELS
PRINCIPLES AND TECHNIQUES
DAPHNE KOLLER AND NIR FRIEDMAN

Recommender Systems



Congratulations!

The Netflix Prize sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences.

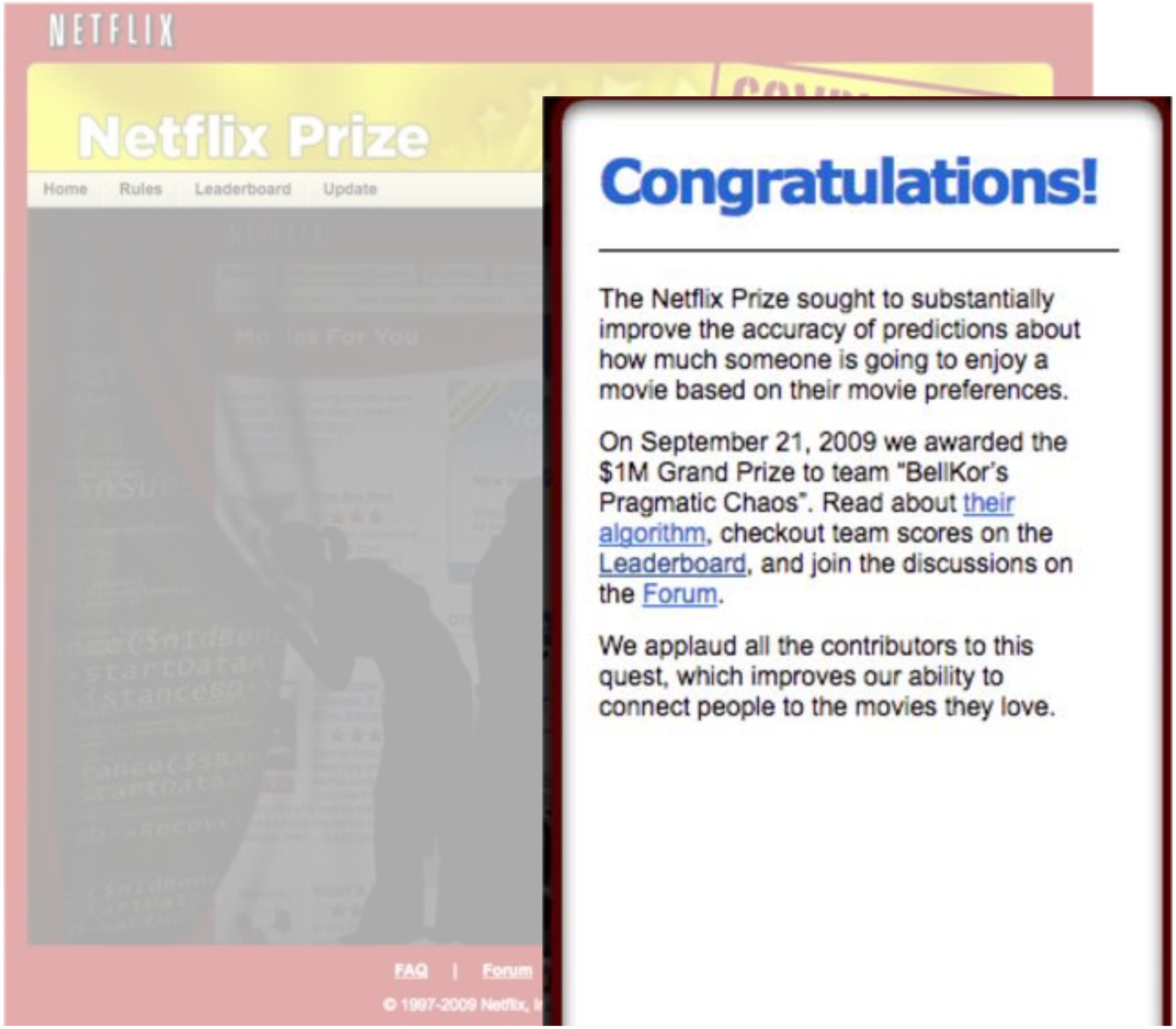
On September 21, 2009 we awarded the $1M Grand Prize to team "BellKor's Pragmatic Chaos". Read about their algorithm, checkout team scores on the Leaderboard, and join the discussions on the Forum.

We applaud all the contributors to this quest, which improves our ability to connect people to the movies they love.

FAQ | Forum | Netflix Home

© 1997-2009 Netflix, Inc. All rights reserved.

Recommender Systems



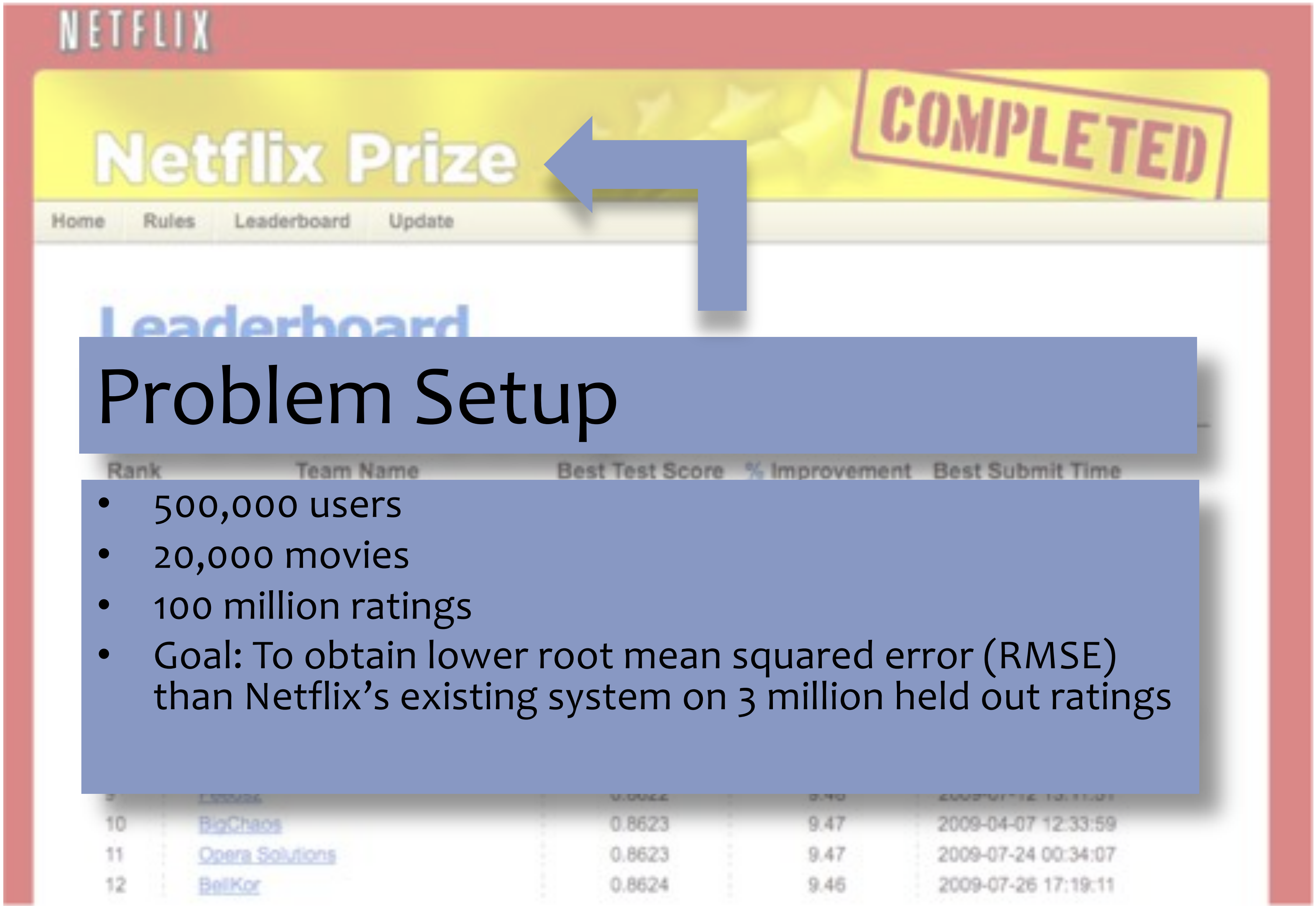
Congratulations!

The Netflix Prize sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences.

On September 21, 2009 we awarded the $1M Grand Prize to team "BellKor's Pragmatic Chaos". Read about their algorithm, checkout team scores on the Leaderboard, and join the discussions on the Forum.

We applaud all the contributors to this quest, which improves our ability to connect people to the movies they love.

Recommender Systems



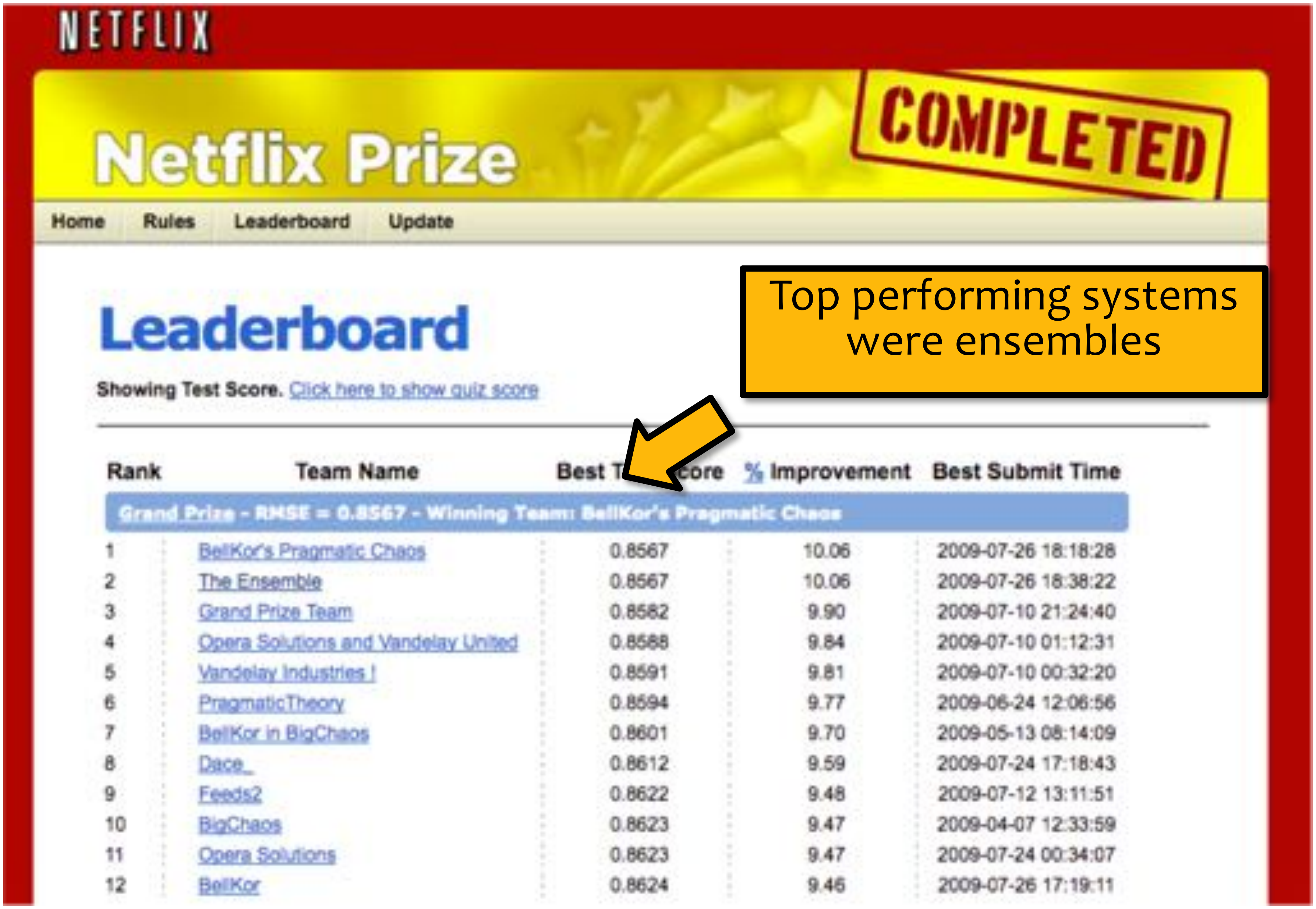
Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11

Problem Setup

- 500,000 users
- 20,000 movies
- 100 million ratings
- Goal: To obtain lower root mean squared error (RMSE) than Netflix's existing system on 3 million held out ratings

ENSEMBLE METHODS

Recommender Systems



Top performing systems were ensembles

Leaderboard

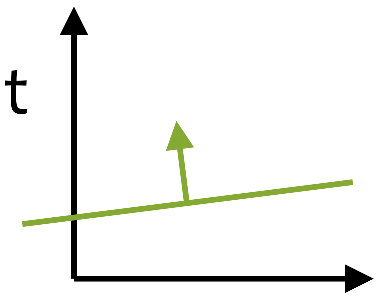
Showing Test Score. Click here to show quiz score

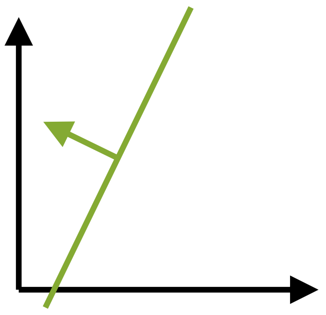
Rank	Team Name	Best T core	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11

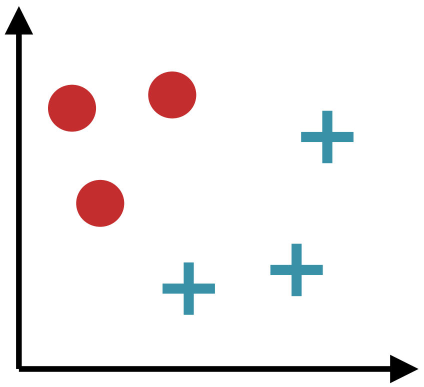
Weighted Majority Algorithm

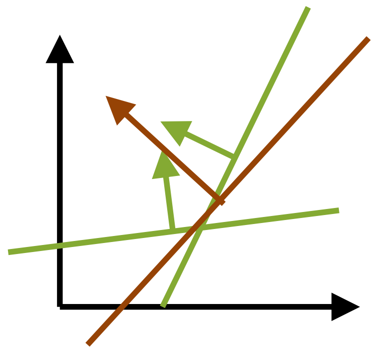
(Littlestone & Warmuth, 1994)

- **Given:** pool *A* of binary classifiers (that you know nothing about)
- **Data:** stream of examples (i.e. online learning setting)
- **Goal:** design a new learner that uses the predictions of the pool to make new predictions
- **Algorithm:**
  - Initially weight all classifiers equally
  - Receive a training example and predict the (weighted) majority vote of the classifiers in the pool
  - Down-weight classifiers that contribute to a mistake by a factor of β









Weighted Majority Algorithm

(Littlestone & Warmuth, 1994)

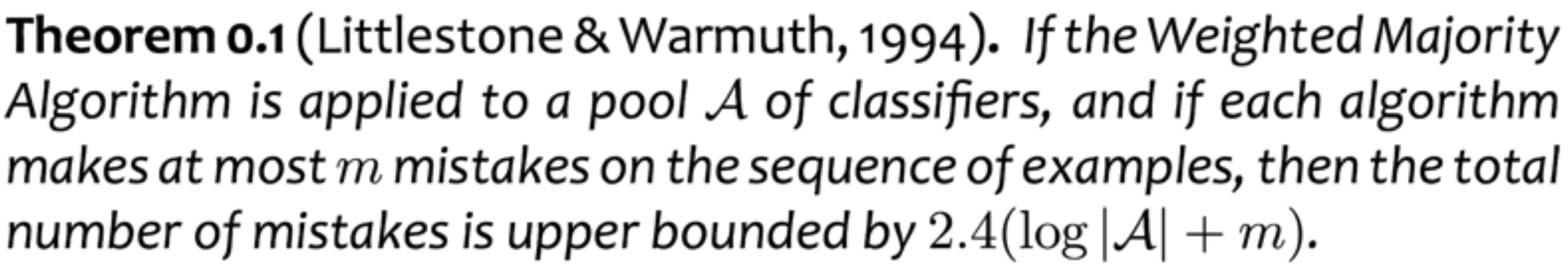
Suppose we have a pool of T binary classifiers $\mathcal{A} = \{h_1, \ldots, h_T\}$ where $h_t : \mathbb{R}^M \to \{+1, -1\}$. Let α_t be the weight for classifier h_t.

Algorithm 1 Weighted Majority Algorithm

```
1: procedure WEIGHTEDMAJORITY($\mathcal{A}$, $\beta$)
2:     Initialize classifier weights $\alpha_t = 1, \ \forall t \in \{1, \ldots, T\}$
3:     for each training example $(\mathbf{x}, y)$ do
4:         Predict majority vote class (splitting ties randomly)
               $\hat{h}(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
5:         if a mistake is made $\hat{h}(x) \neq y$ then
6:             for each classifier $t \in \{1, \ldots, T\}$ do
7:                 If $h_t(x) \neq y$, then $\alpha_t \leftarrow \beta\alpha_t$
```

Weighted Majority Algorithm

(Littlestone & Warmuth, 1994)



Theorem 0.1 (Littlestone & Warmuth, 1994). *If the Weighted Majority Algorithm is applied to a pool $\mathcal{A}$ of classifiers, and if each algorithm makes at most m mistakes on the sequence of examples, then the total number of mistakes is upper bounded by $2.4(\log|\mathcal{A}| + m)$.*

This is a "mistake bound" of the variety we saw for the Perceptron algorithm

ADABOOST

Comparison

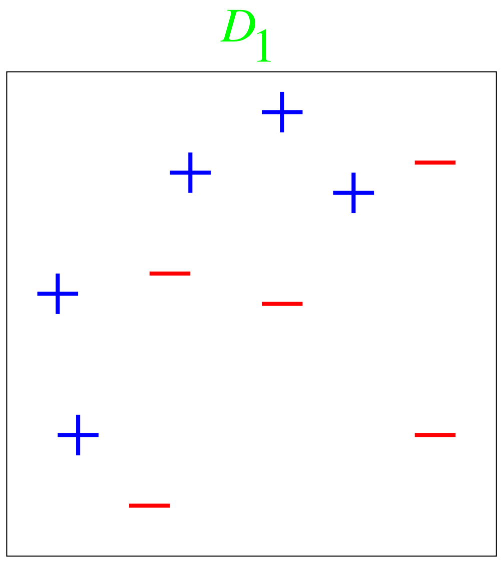
Weighted Majority Algorithm

- an example of an ensemble method
- assumes the classifiers are learned ahead of time
- only learns (majority vote) weight for each classifiers

AdaBoost

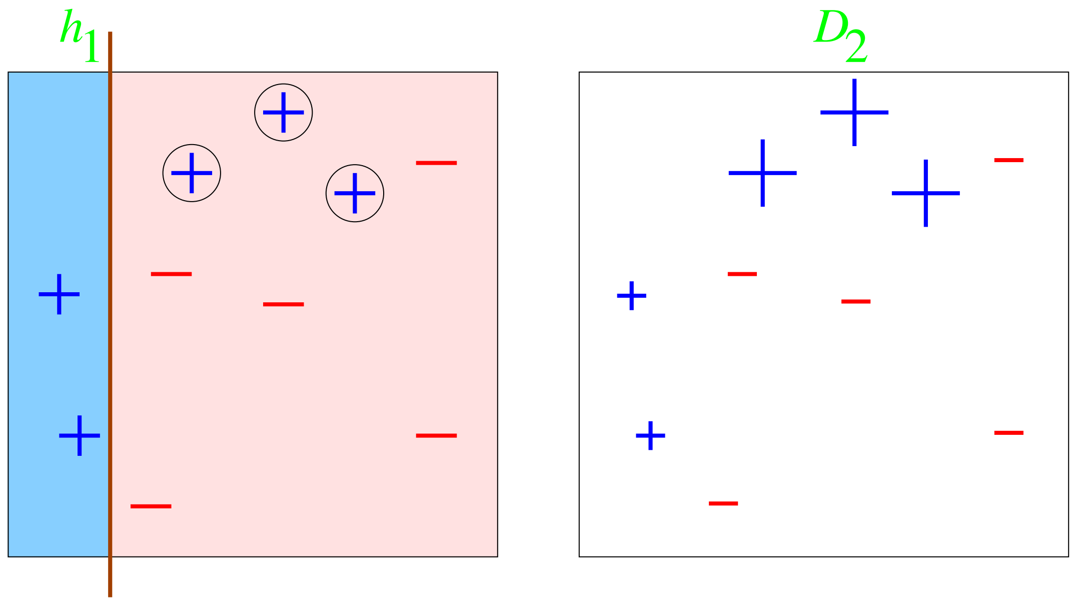
- an example of a boosting method
- simultaneously learns:
  - the classifiers themselves
  - (majority vote) weight for each classifiers

AdaBoost: Toy Example



weak classifiers = vertical or horizontal half-planes

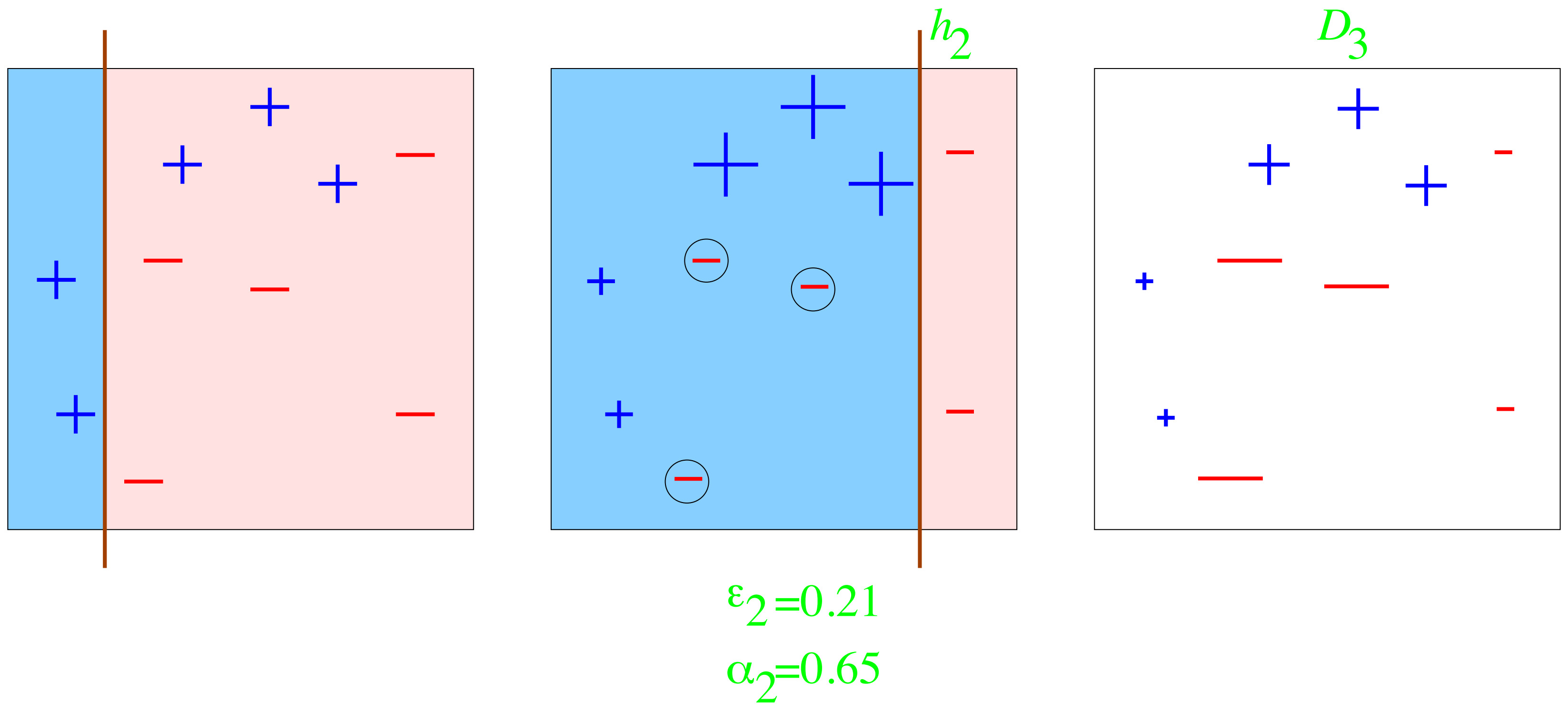
AdaBoost: Toy Example



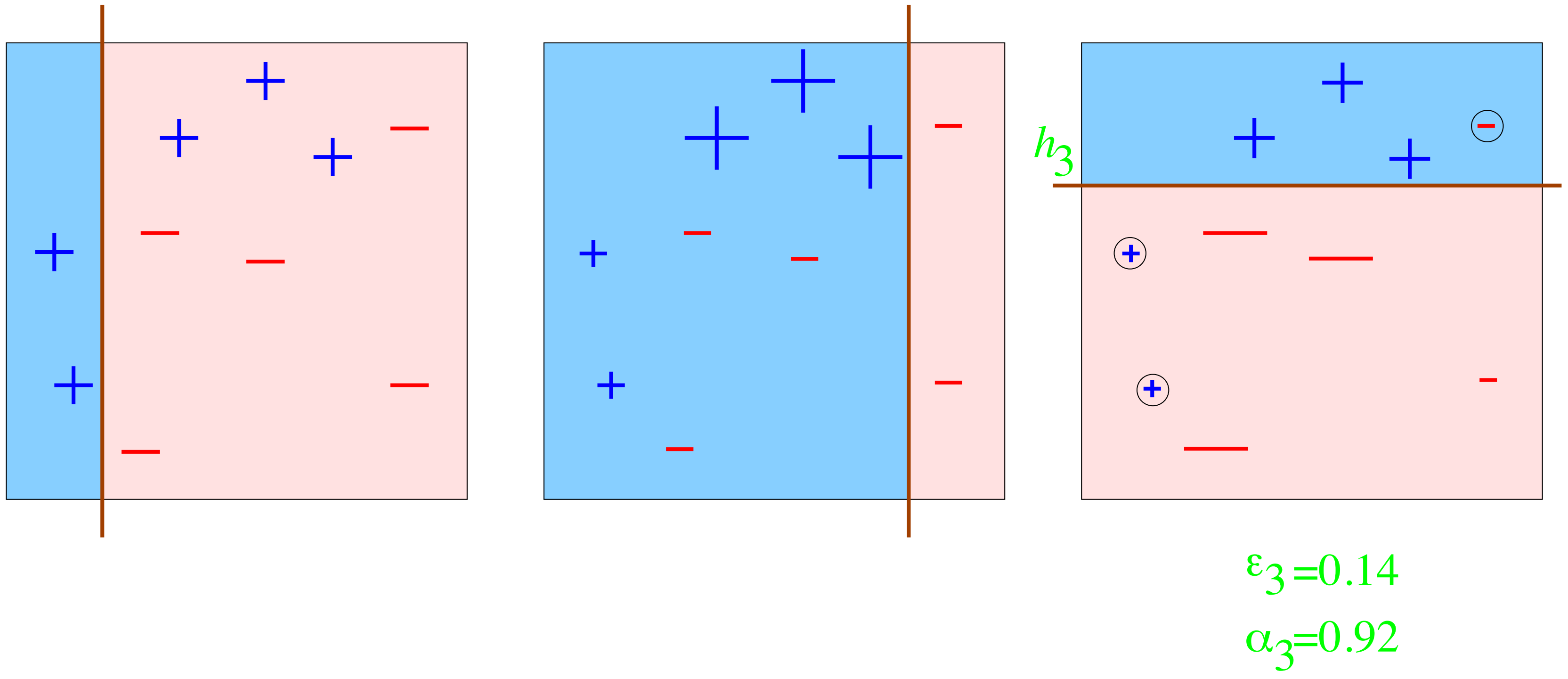
$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$

AdaBoost: Toy Example

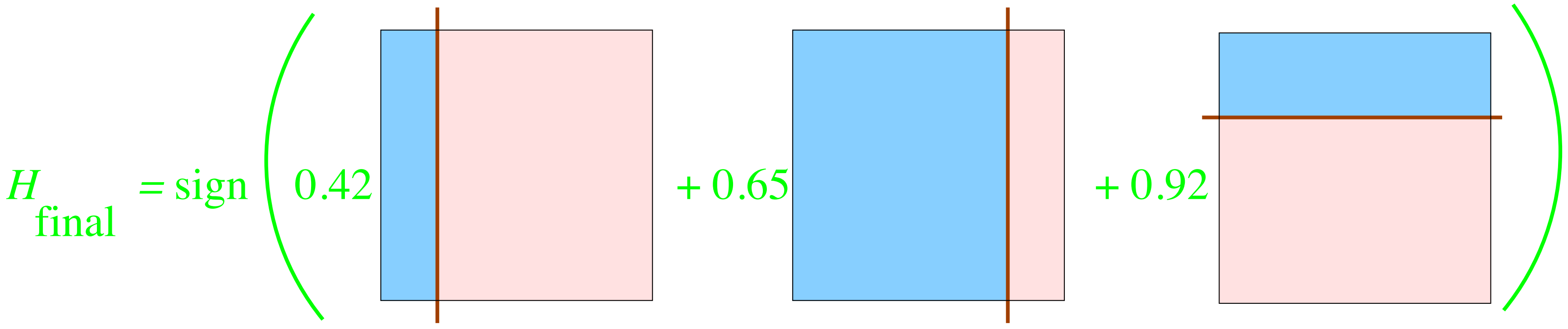


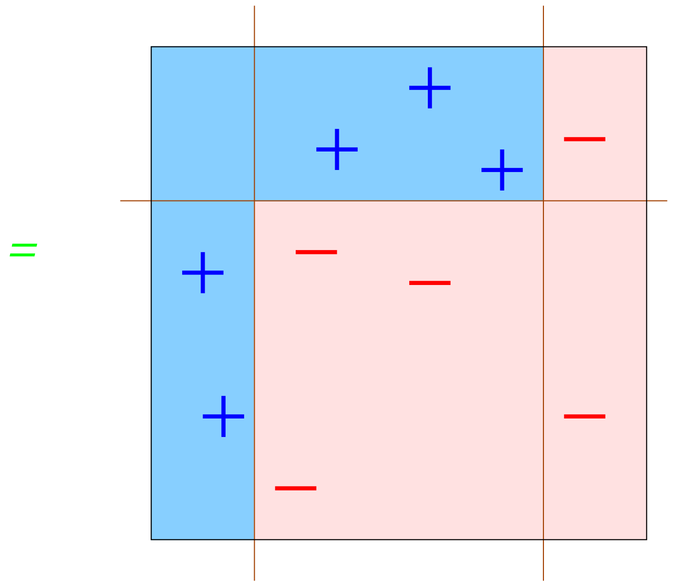
AdaBoost: Toy Example



Slide from Schapire NIPS Tutorial

AdaBoost: Toy Example





Slide from Schapire NIPS Tutorial

AdaBoost

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$.
- Update:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Algorithm from (Freund & Schapire, 1999)

AdaBoost

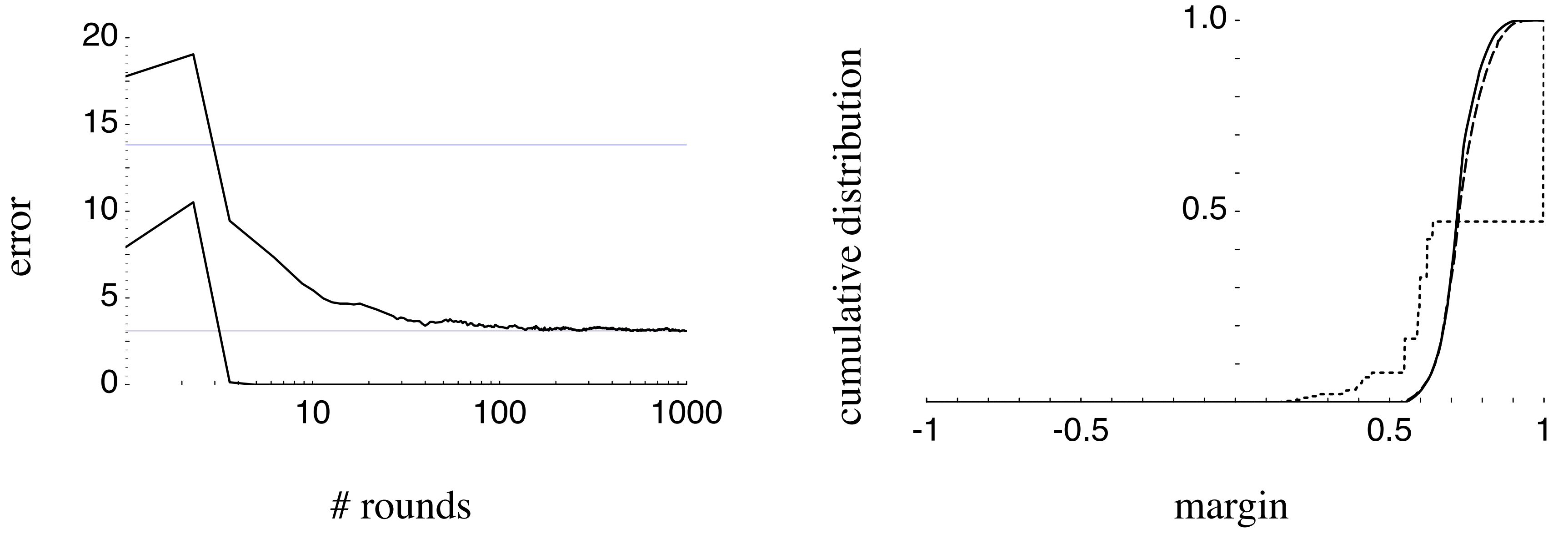


Figure 2: Error curves and the margin distribution graph for boosting C4.5 on the letter dataset as reported by Schapire et al. [41]. *Left*: the training and test error curves (lower and upper curves, respectively) of the combined classifier as a function of the number of rounds of boosting. The horizontal lines indicate the test error rate of the base classifier as well as the test error of the final combined classifier. *Right*: The cumulative distribution of margins of the training examples after 5, 100 and 1000 iterations, indicated by short-dashed, long-dashed (mostly hidden) and solid curves, respectively.

Figure from (Freund & Schapire, 1999)

Learning Objectives

Ensemble Methods / Boosting

You should be able to…

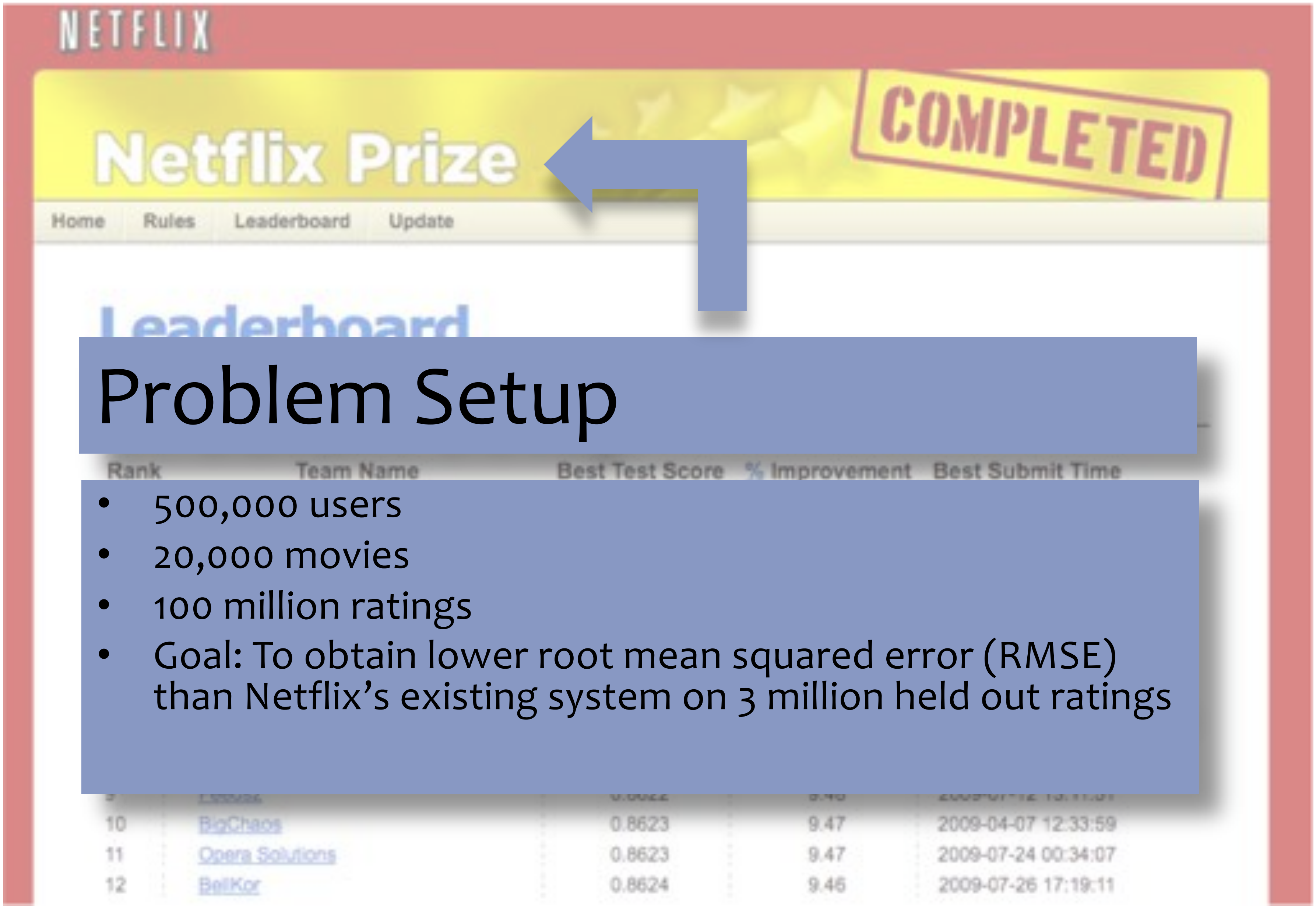
1. Implement the Weighted Majority Algorithm
2. Implement AdaBoost
3. Distinguish what is learned in the Weighted Majority Algorithm vs. Adaboost
4. Contrast the theoretical result for the Weighted Majority Algorithm to that of Perceptron
5. Explain a surprisingly common empirical result regarding Adaboost train/test curves

Outline

- **Recommender Systems**
  - Content Filtering
  - Collaborative Filtering (CF)
  - CF: Neighborhood Methods
  - CF: Latent Factor Methods
- **Matrix Factorization**
  - Background: Low-rank Factorizations
  - Residual matrix
  - Unconstrained Matrix Factorization
    - Optimization problem
    - Gradient Descent, SGD, Alternating Least Squares
    - User/item bias terms (matrix trick)
  - Singular Value Decomposition (SVD)
  - Non-negative Matrix Factorization

RECOMMENDER SYSTEMS

Recommender Systems

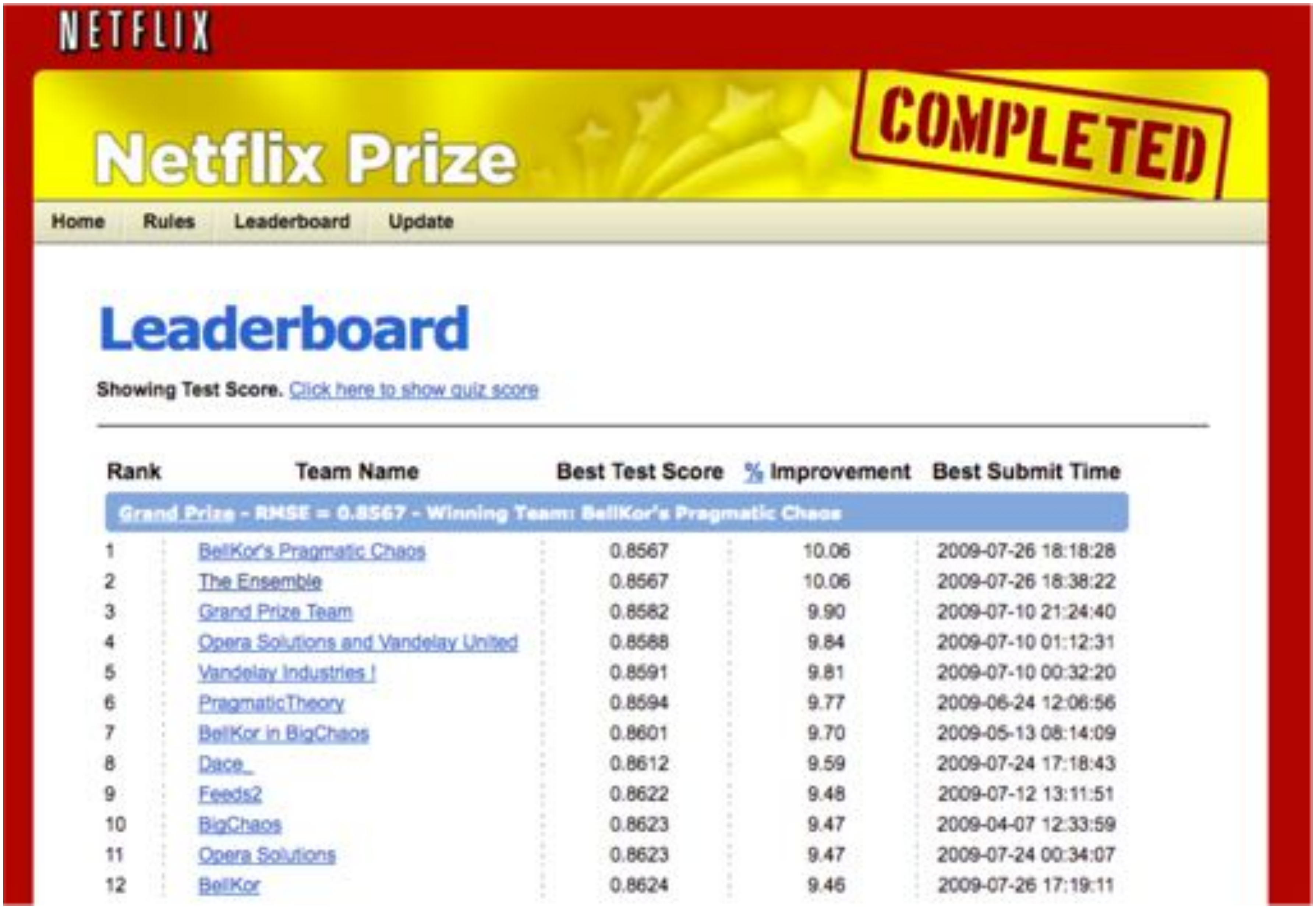


Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11

Problem Setup

- 500,000 users
- 20,000 movies
- 100 million ratings
- Goal: To obtain lower root mean squared error (RMSE) than Netflix's existing system on 3 million held out ratings

Recommender Systems



Leaderboard

Showing Test Score. Click here to show quiz score

Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11

Recommender Systems

- **Setup:**
  - Items:
    movies, songs, products, etc.
    (often many thousands)
  - Users:
    watchers, listeners, purchasers, etc.
    (often many millions)
  - Feedback:
    5-star ratings, not-clicking 'next', purchases, etc.
- **Key Assumptions:**
  - Can represent ratings numerically as a user/item matrix
  - Users only rate a small number of items (the matrix is sparse)

	Doctor Strange	Star Trek: Beyond	Zootopia
Alice	1		5
Bob	3	4	
Charlie	3	5	2

Two Types of Recommender Systems

Content Filtering

- *Example*: Pandora.com music recommendations (Music Genome Project)
- **Con:** Assumes access to side information about items (e.g. properties of a song)
- **Pro:** Got a new item to add? No problem, just be sure to include the side information

Collaborative Filtering

- *Example*: Netflix movie recommendations
- **Pro:** Does not assume access to side information about items (e.g. does not need to know about movie genres)
- **Con:** Does not work on new items that have no ratings

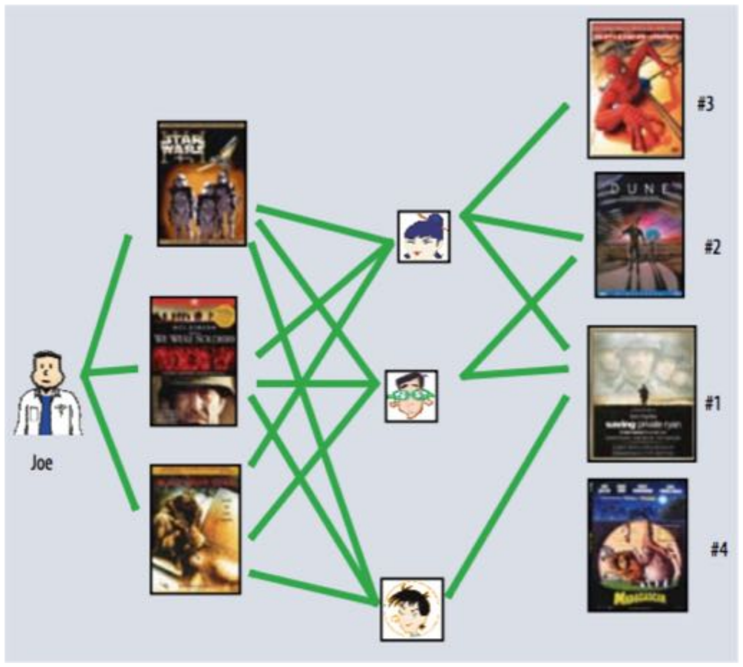
COLLABORATIVE FILTERING

Collaborative Filtering

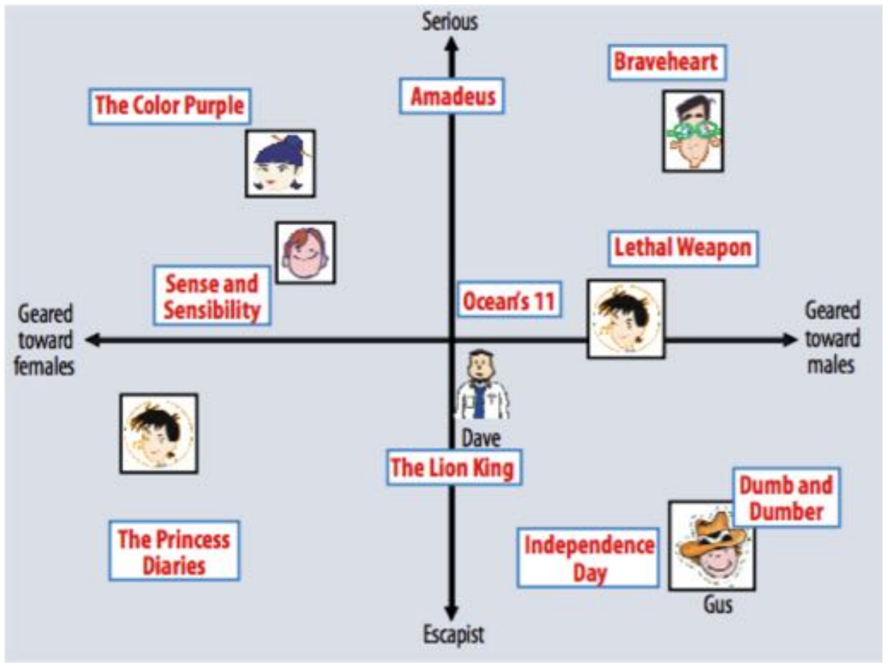
- **Everyday Examples of Collaborative Filtering...**
  - Bestseller lists
  - Top 40 music lists
  - The "recent returns" shelf at the library
  - Unmarked but well-used paths thru the woods
  - The printer room at work
  - "Read any good books lately?"
  - …
- **Common insight:** personal tastes are 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

Two Types of Collaborative Filtering

1. Neighborhood Methods



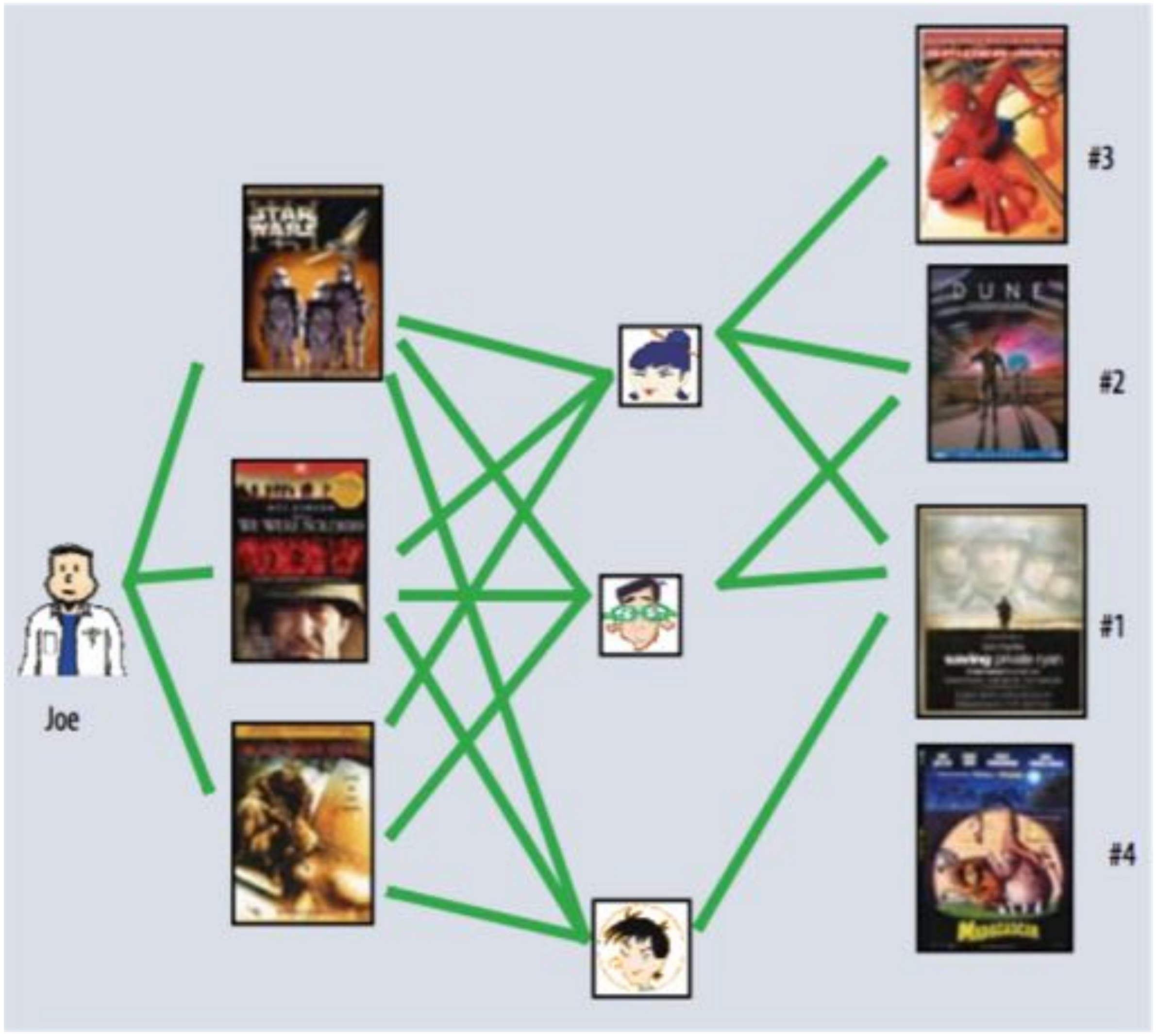
2. Latent Factor Methods



Figures from Koren et al. (2009)

Two Types of Collaborative Filtering

1. Neighborhood Methods



In the figure, assume that a green line indicates the movie was **watched**

Algorithm:

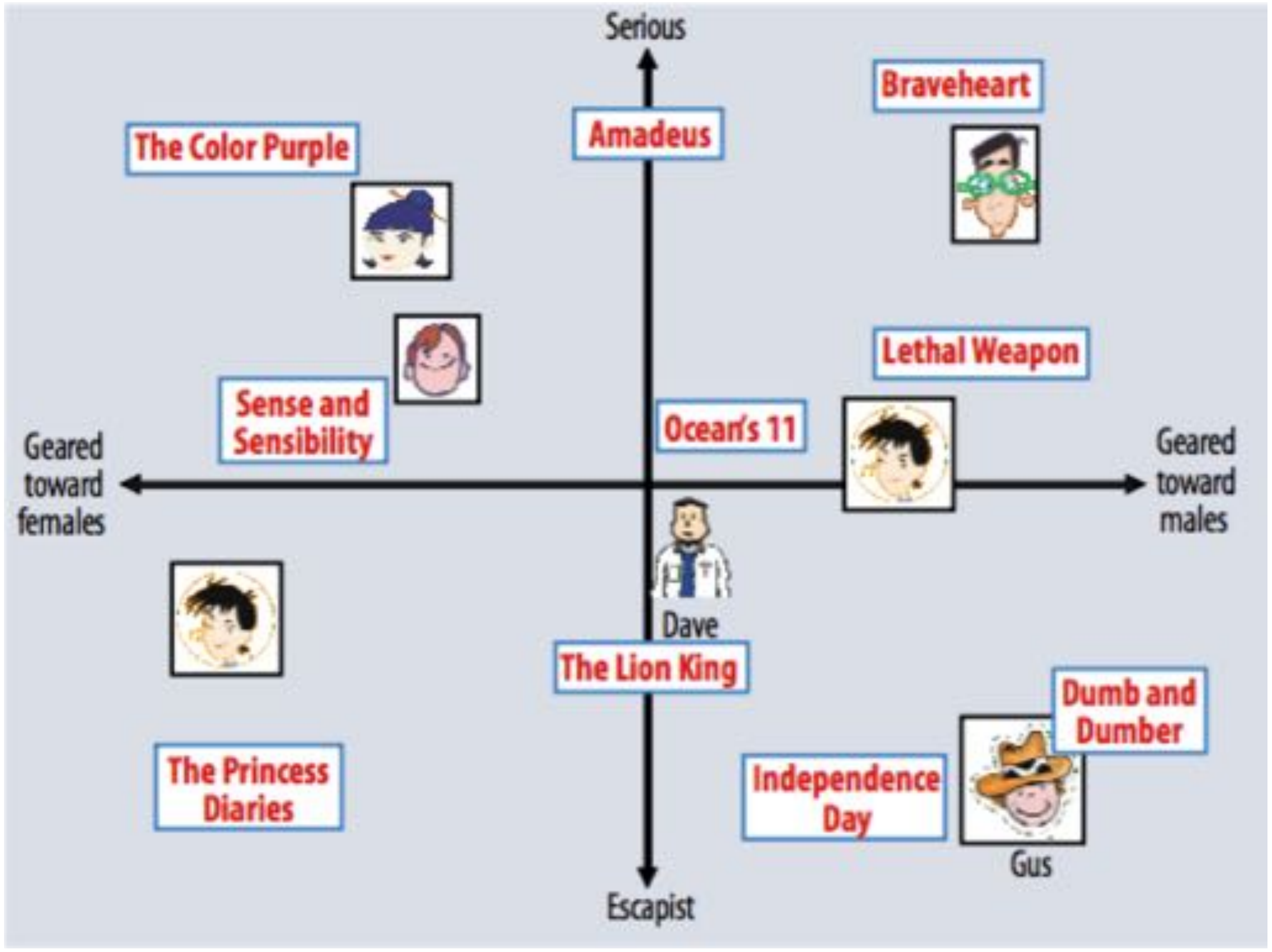
1. **Find neighbors** based on similarity of movie preferences
2. **Recommend** movies that those neighbors watched

Figures from Koren et al. (2009)

Two Types of Collaborative Filtering

2. Latent Factor Methods

- Assume that both movies and users live in some **low-dimensional space** describing their properties
- **Recommend** a movie based on its **proximity** to the user in the latent space
- **Example Algorithm:** Matrix Factorization



Figures from Koren et al. (2009)

MATRIX FACTORIZATION

Recommending Movies

Question:

Applied to the Netflix Prize problem, which of the following methods *always* requires side information about the users and movies?

Select all that apply

A. collaborative filtering
B. latent factor methods
C. ensemble methods
D. content filtering
E. neighborhood methods
F. recommender systems

Answer:

Matrix Factorization

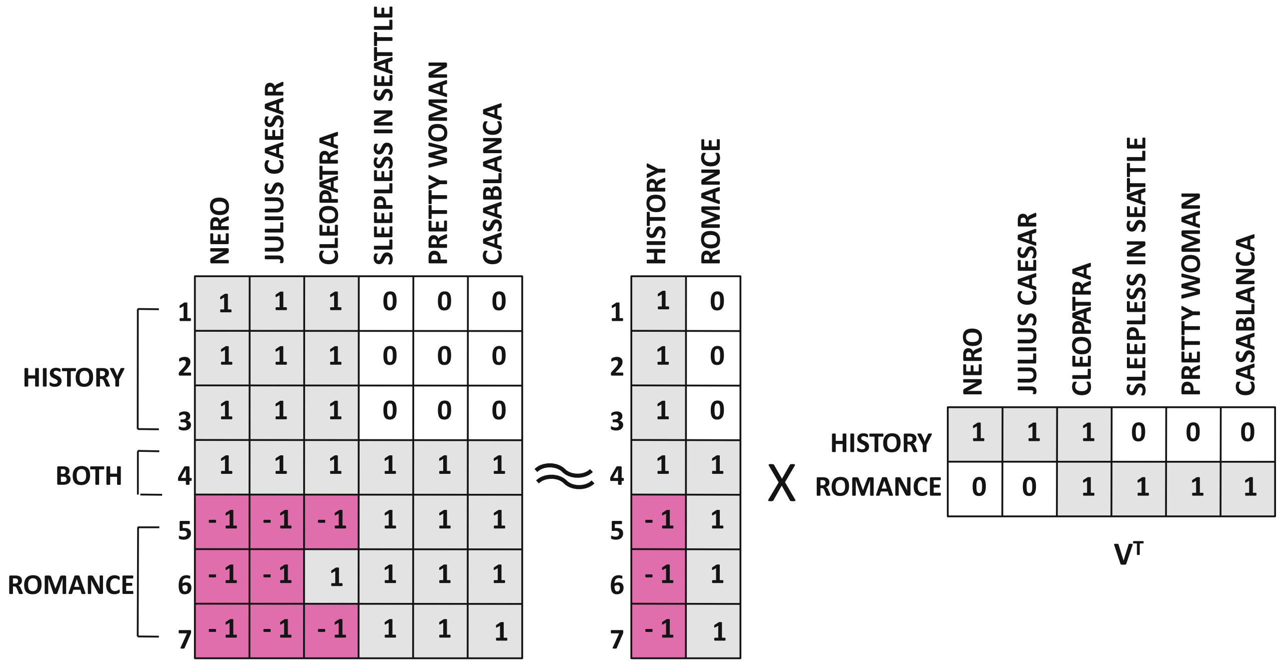
- Many different ways of factorizing a matrix
- We'll consider three:
  1. Unconstrained Matrix Factorization
  2. Singular Value Decomposition
  3. Non-negative Matrix Factorization
- MF is just another example of a **common recipe**:
  1. define a model
  2. define an objective function
  3. optimize with SGD

Matrix Factorization

Whiteboard

- Background: Low-rank Factorizations
- Residual matrix

Example: MF for Netflix Problem



R

		NERO	JULIUS CAESAR	CLEOPATRA	SLEEPLESS IN SEATTLE	PRETTY WOMAN	CASABLANCA
HISTORY	1	1	1	1	0	0	0
	2	1	1	1	0	0	0
	3	1	1	1	0	0	0
BOTH	4	1	1	1	1	1	1
ROMANCE	5	- 1	- 1	- 1	1	1	1
	6	- 1	- 1	1	1	1	1
	7	- 1	- 1	- 1	1	1	1

≈

U

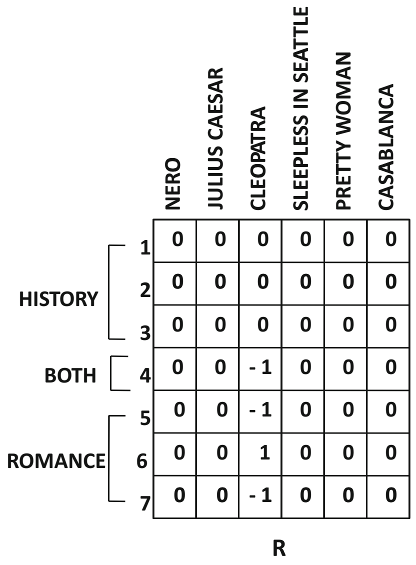
	HISTORY	ROMANCE
1	1	0
2	1	0
3	1	0
4	1	1
5	- 1	1
6	- 1	1
7	- 1	1

X

V^T

	NERO	JULIUS CAESAR	CLEOPATRA	SLEEPLESS IN SEATTLE	PRETTY WOMAN	CASABLANCA
HISTORY	1	1	1	0	0	0
ROMANCE	0	0	1	1	1	1

(a) Example of rank-2 matrix factorization



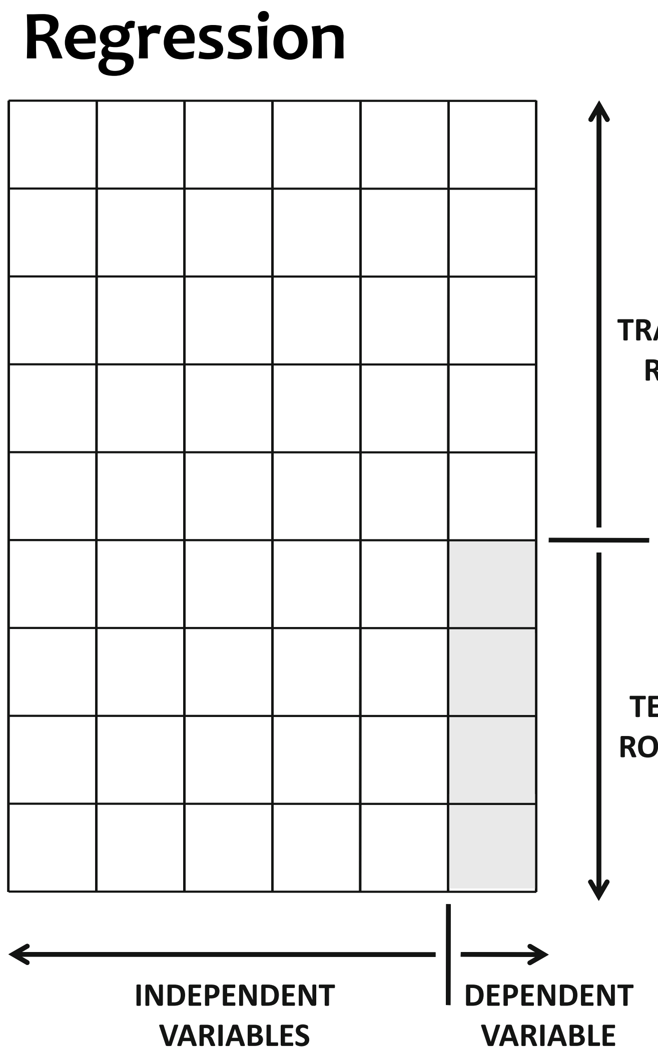
R

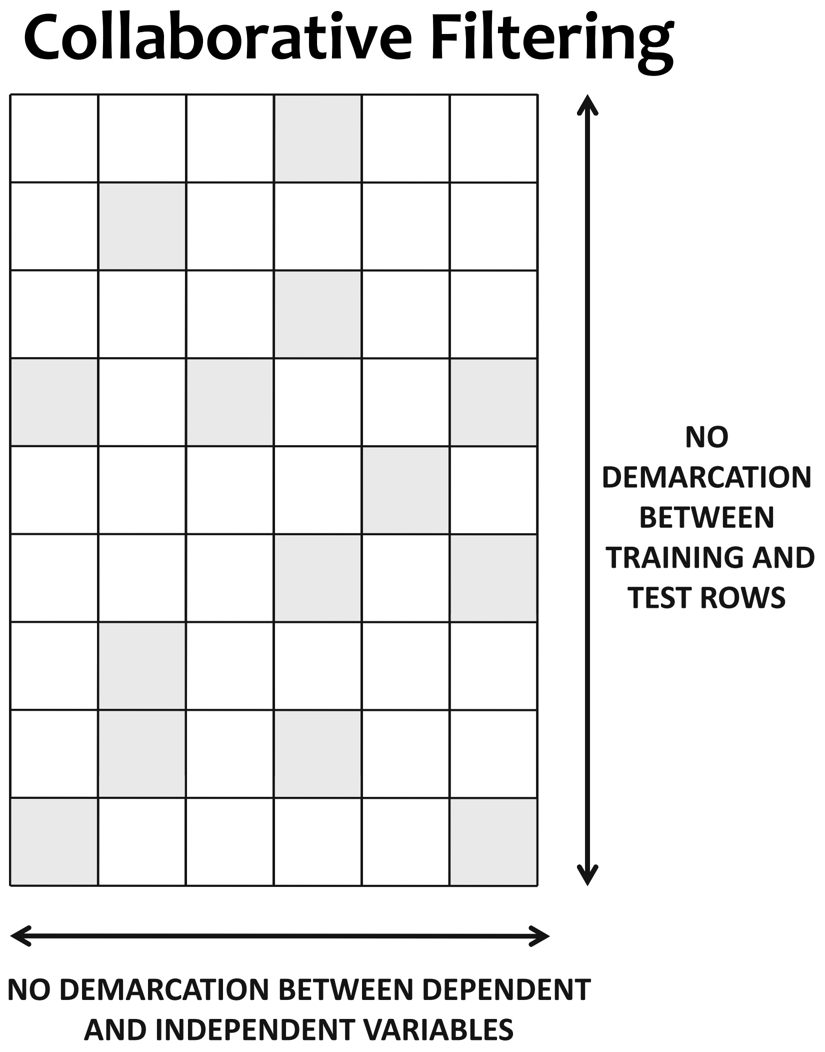
		NERO	JULIUS CAESAR	CLEOPATRA	SLEEPLESS IN SEATTLE	PRETTY WOMAN	CASABLANCA
HISTORY	1	0	0	0	0	0	0
	2	0	0	0	0	0	0
	3	0	0	0	0	0	0
BOTH	4	0	0	- 1	0	0	0
ROMANCE	5	0	0	- 1	0	0	0
	6	0	0	1	0	0	0
	7	0	0	- 1	0	0	0

(b) Residual matrix

Figures from Aggarwal (2016)

Regression vs. Collaborative Filtering





Collaborative Filtering
NO DEMARCATION BETWEEN TRAINING AND TEST ROWS
NO DEMARCATION BETWEEN DEPENDENT AND INDEPENDENT VARIABLES

Figures from Aggarwal (2016)

UNCONSTRAINED MATRIX FACTORIZATION

Unconstrained Matrix Factorization

Whiteboard

- Optimization problem
- SGD
- SGD with Regularization
- Alternating Least Squares
- User/item bias terms (matrix trick)

Unconstrained Matrix Factorization

SGD for UMF:

```
While not converged:
    ① Sample $(i,j)$ from $\mathcal{Z}$ uniformly at random
    ② Compute $e_{ij} = r_{ij} - \vec{u}_i^T \vec{v}_j$
    ③ Update
        $\vec{u}_i \leftarrow \vec{u}_i - \gamma \nabla_{\vec{u}_i} J_{ij}(U,V)$
        $\vec{v}_j \leftarrow \vec{v}_j - \gamma \nabla_{\vec{v}_j} J_{ij}(U,V)$
```

W/ Regularization

$$J_{ij}(U,V) = \frac{1}{2}\left(r_{ij} - \vec{u}_i^T \vec{v}_j\right)^2 + \lambda\left(\|u_i\|_2^2 + \|v_j\|_2^2\right)$$

$$\nabla_{\vec{u}_i} J_{ij}(U,V) = -e_{ij}\vec{v}_j + \lambda \vec{u}_i$$

$$\nabla_{\vec{v}_j} J_{ij}(U,V) = -e_{ij}\vec{u}_i + \lambda \vec{v}_j$$

where $e_{ij} = r_{ij} - \vec{u}_i^T \vec{v}_j$

Unconstrained Matrix Factorization

SGD for UMF:

User/Item Bias terms

$$\hat{r}_{ij} = o_i + p_j + \vec{u}_i^T \vec{v}_j$$

matrix trick:

$$U = \begin{bmatrix} & o_1 & 1 \\ & o_2 & 1 \\ & \vdots & \vdots \\ & o_M & 1 \end{bmatrix} \qquad V = \begin{bmatrix} & 1 & p_1 \\ & 1 & p_2 \\ & \vdots & \vdots \\ & 1 & p_N \end{bmatrix}$$

Unconstrained Matrix Factorization

Alternating Least Squares (ALS) for UMF:

Block Coord. Descent:

```
while not converged:
    ① $U = \underset{U}{\operatorname{argmin}} J(U,V)$
    ② $V = \underset{V}{\operatorname{argmin}} J(U,V)$
```

convex and easy to solve in closed form

Lin. Reg.

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \vec{\theta}^T\vec{x}_i)^2$$

$$J(U,V) = \frac{1}{2}\sum_{(i,j)\in Z}(r_{ij} - \vec{u}_i^T\vec{v}_j)^2$$

if U is fixed Least Squares in V

if V is fixed Least Squares in U

Option #1: take derivatives, set to zero and solve in closed form

★ solving $J(U,V)$ in closed form directly isn't easy and $J(U,V)$ is nonconvex

Unconstrained Matrix Factorization

In-Class Exercise

Derive a block coordinate descent algorithm for the Unconstrained Matrix Factorization problem.

- User vectors:

$$\mathbf{w}_u \in \mathbb{R}^r$$

- Item vectors:

$$\mathbf{h}_i \in \mathbb{R}^r$$

- Rating prediction:

$$v_{ui} = \mathbf{w}_u^T \mathbf{h}_i$$

- Set of non-missing entries

$$\mathcal{Z} = \{(u,i) : v_{ui} \text{ is observed}\}$$

- Objective:

$$\underset{\mathbf{w},\mathbf{h}}{\operatorname{argmin}} \sum_{(u,i)\in\mathcal{Z}} (v_{ui} - \mathbf{w}_u^T \mathbf{h}_i)^2$$

Matrix Factorization
(with matrices)

- User vectors:

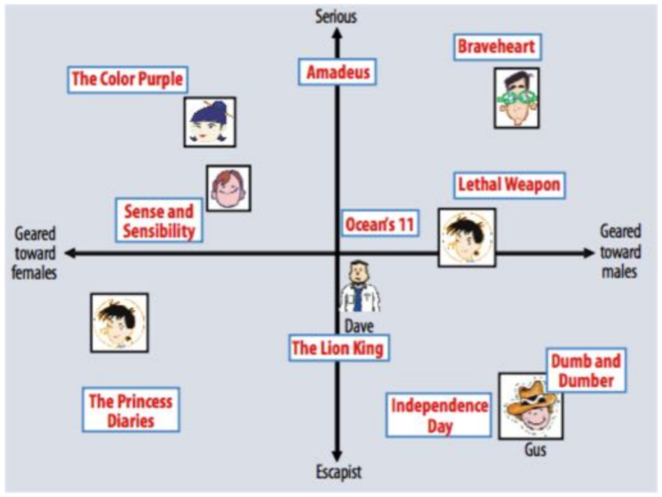
$$(W_{u*})^T \in \mathbb{R}^r$$

- Item vectors:

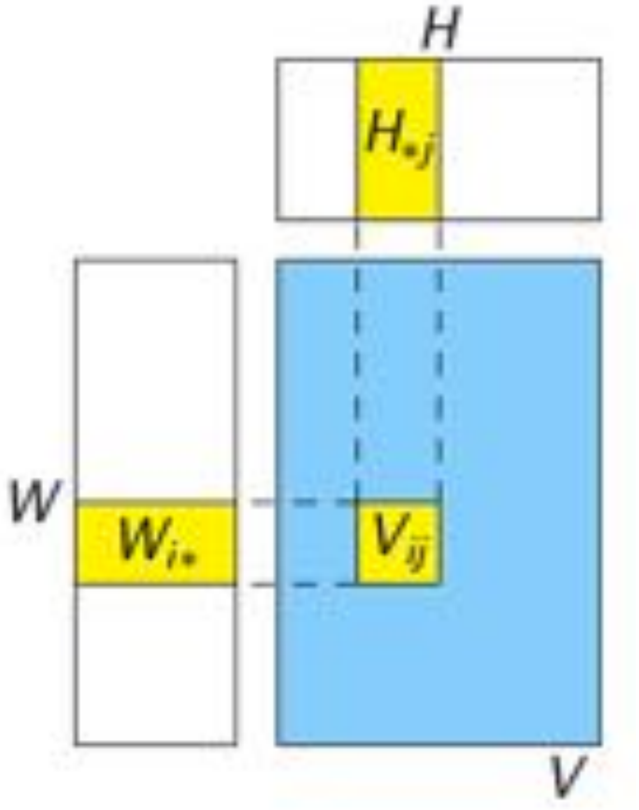
$$H_{*i} \in \mathbb{R}^r$$

- Rating prediction:

$$V_{ui} = W_{u*}H_{*i}$$
$$= [WH]_{ui}$$



Figures from Koren et al. (2009)



Figures from Gemulla et al. (2011)

Matrix Factorization
(with vectors)

- User vectors:

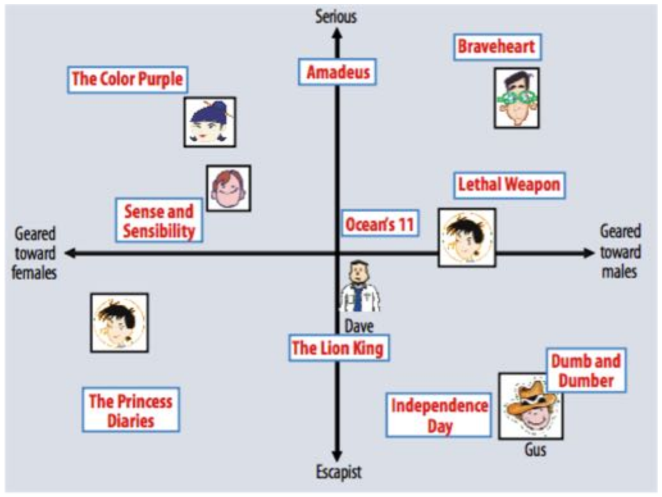
$$\mathbf{w}_u \in \mathbb{R}^r$$

- Item vectors:

$$\mathbf{h}_i \in \mathbb{R}^r$$

- Rating prediction:

$$v_{ui} = \mathbf{w}_u^T \mathbf{h}_i$$



Figures from Koren et al. (2009)

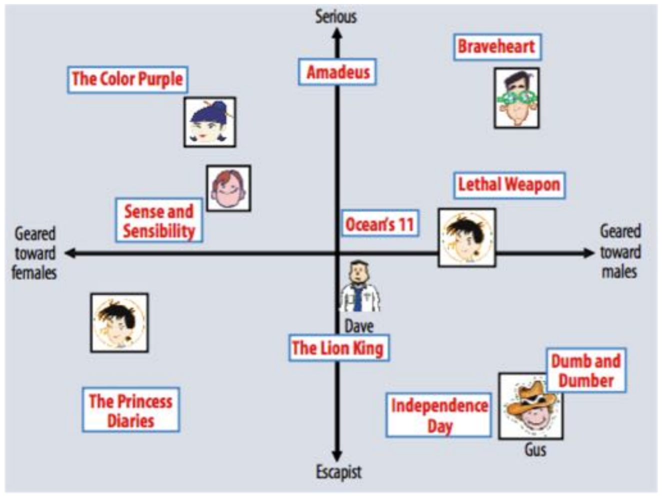
Matrix Factorization (with vectors)

- Set of non-missing entries:

$$\mathcal{Z} = \{(u,i) : v_{ui} \text{ is observed}\}$$

- Objective:

$$\underset{\mathbf{w},\mathbf{h}}{\text{argmin}} \sum_{(u,i)\in\mathcal{Z}} (v_{ui} - \mathbf{w}_u^T \mathbf{h}_i)^2$$



Figures from Koren et al. (2009)

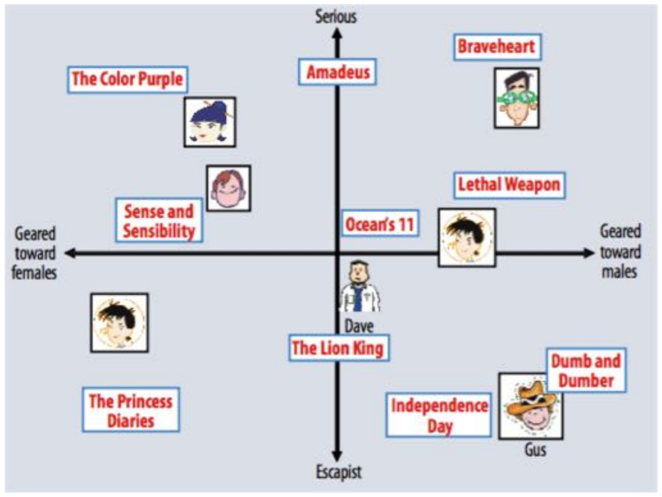
Matrix Factorization

(with vectors)

- Regularized Objective:

$$\underset{\mathbf{w},\mathbf{h}}{\operatorname{argmin}} \sum_{(u,i)\in\mathcal{Z}} (v_{ui} - \mathbf{w}_u^T \mathbf{h}_i)^2$$

$$+ \lambda(\sum_i ||\mathbf{w}_i||^2 + \sum_u ||\mathbf{h}_u||^2)$$



Figures from Koren et al. (2009)

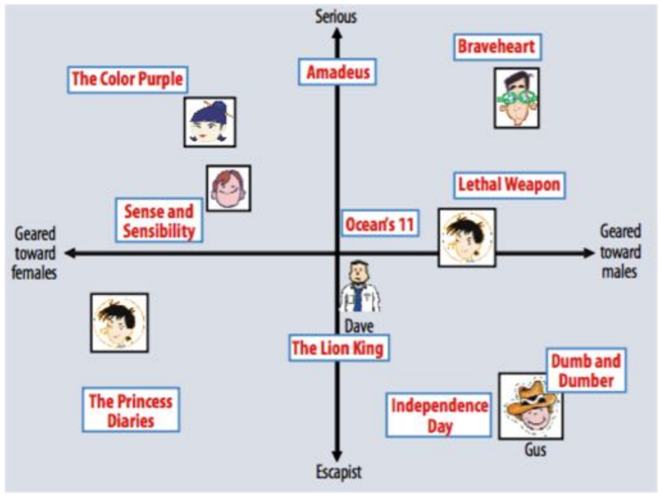
Matrix Factorization (with vectors)

- Regularized Objective:

$$\underset{\mathbf{w},\mathbf{h}}{\text{argmin}} \sum_{(u,i)\in\mathcal{Z}} (v_{ui} - \mathbf{w}_u^T \mathbf{h}_i)^2$$
$$+ \lambda(\sum_i ||\mathbf{w}_i||^2 + \sum_u ||\mathbf{h}_u||^2)$$

- SGD update for random (u,i):

$$e_{ui} \leftarrow v_{ui} - \mathbf{w}_u^T \mathbf{h}_i$$
$$\mathbf{w}_u \leftarrow \mathbf{w}_u + \gamma(e_{ui}\mathbf{h}_i - \lambda \mathbf{w}_u)$$
$$\mathbf{h}_i \leftarrow \mathbf{h}_i + \gamma(e_{ui}\mathbf{w}_u - \lambda \mathbf{h}_i)$$



Figures from Koren et al. (2009)

Matrix Factorization
(with matrices)

- User vectors:

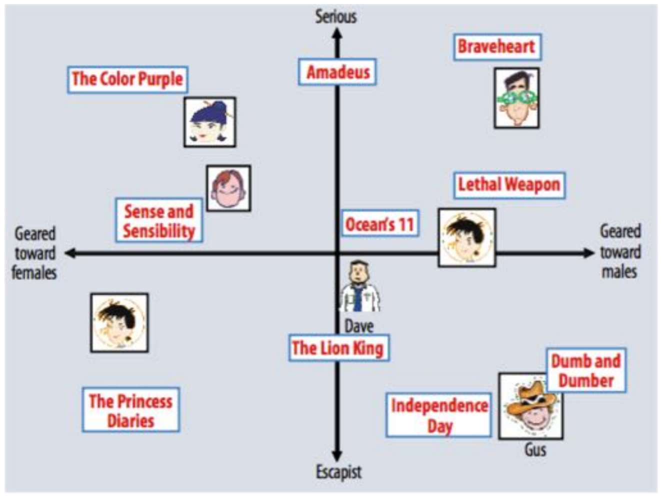
$$(W_{u*})^T \in \mathbb{R}^r$$

- Item vectors:

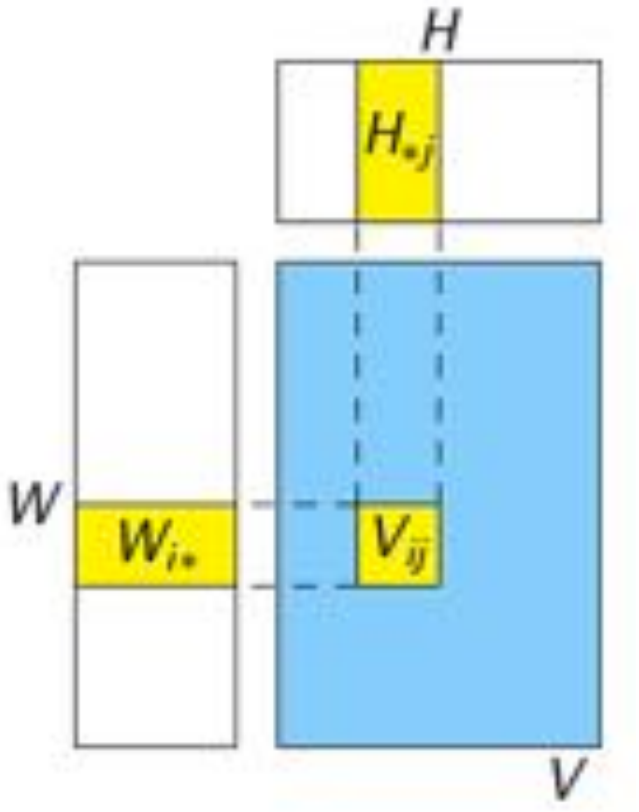
$$H_{*i} \in \mathbb{R}^r$$

- Rating prediction:

$$V_{ui} = W_{u*} H_{*i}$$
$$= [WH]_{ui}$$



Figures from Koren et al. (2009)



Figures from Gemulla et al. (2011)

Matrix Factorization
(with matrices)

- SGD

require that the loss can be written as

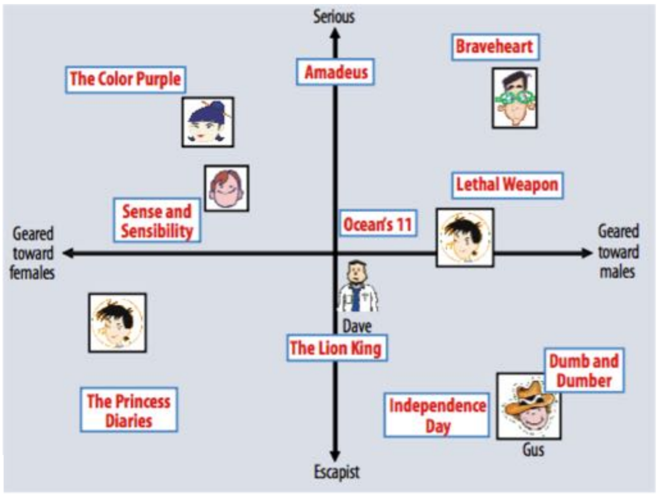
$$L = \sum_{(i,j)\in Z} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$$

Algorithm 1 SGD for Matrix Factorization

```
Require: A training set $Z$, initial values $W_0$ and $H_0$
  while not converged do {step}
    Select a training point $(i,j) \in Z$ uniformly at random.
    $W'_{i*} \leftarrow W_{i*} - \epsilon_n N \frac{\partial}{\partial W_{i*}} l(V_{ij}, W_{i*}, H_{*j})$
    $H_{*j} \leftarrow H_{*j} - \epsilon_n N \frac{\partial}{\partial H_{*j}} l(V_{ij}, W_{i*}, H_{*j})$
    $W_{i*} \leftarrow W'_{i*}$
  end while
```

step size

Figure from Gemulla et al. (2011)



Figures from Koren et al. (2009)

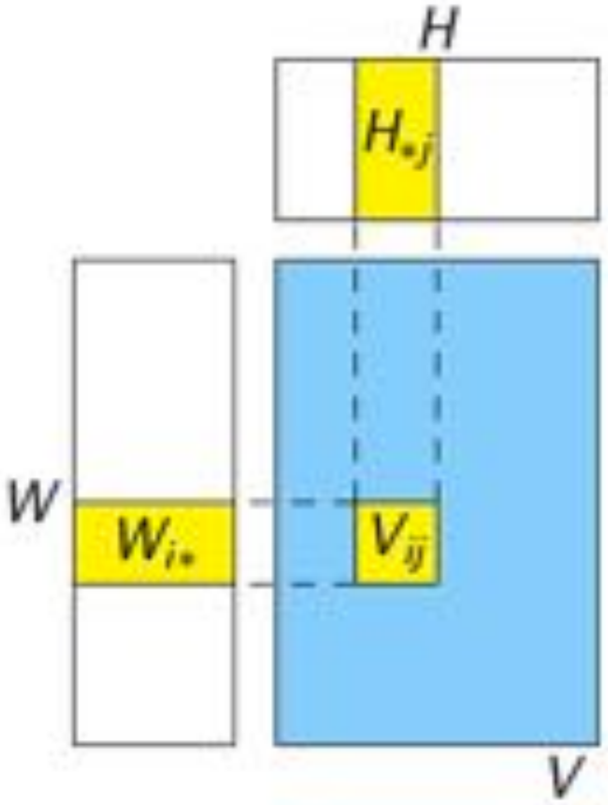


Figure from Gemulla et al. (2011)

Matrix Factorization

Example Factors

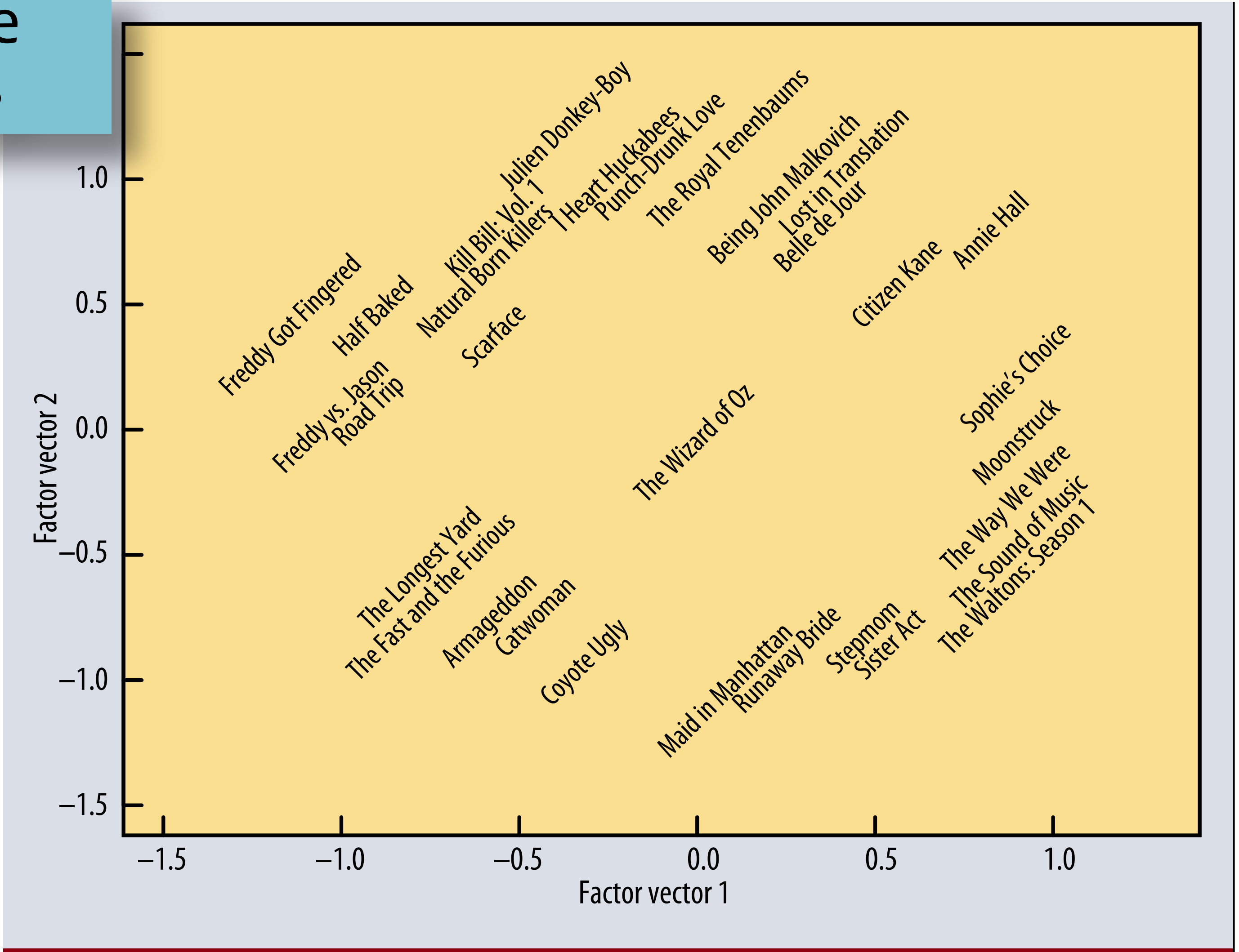
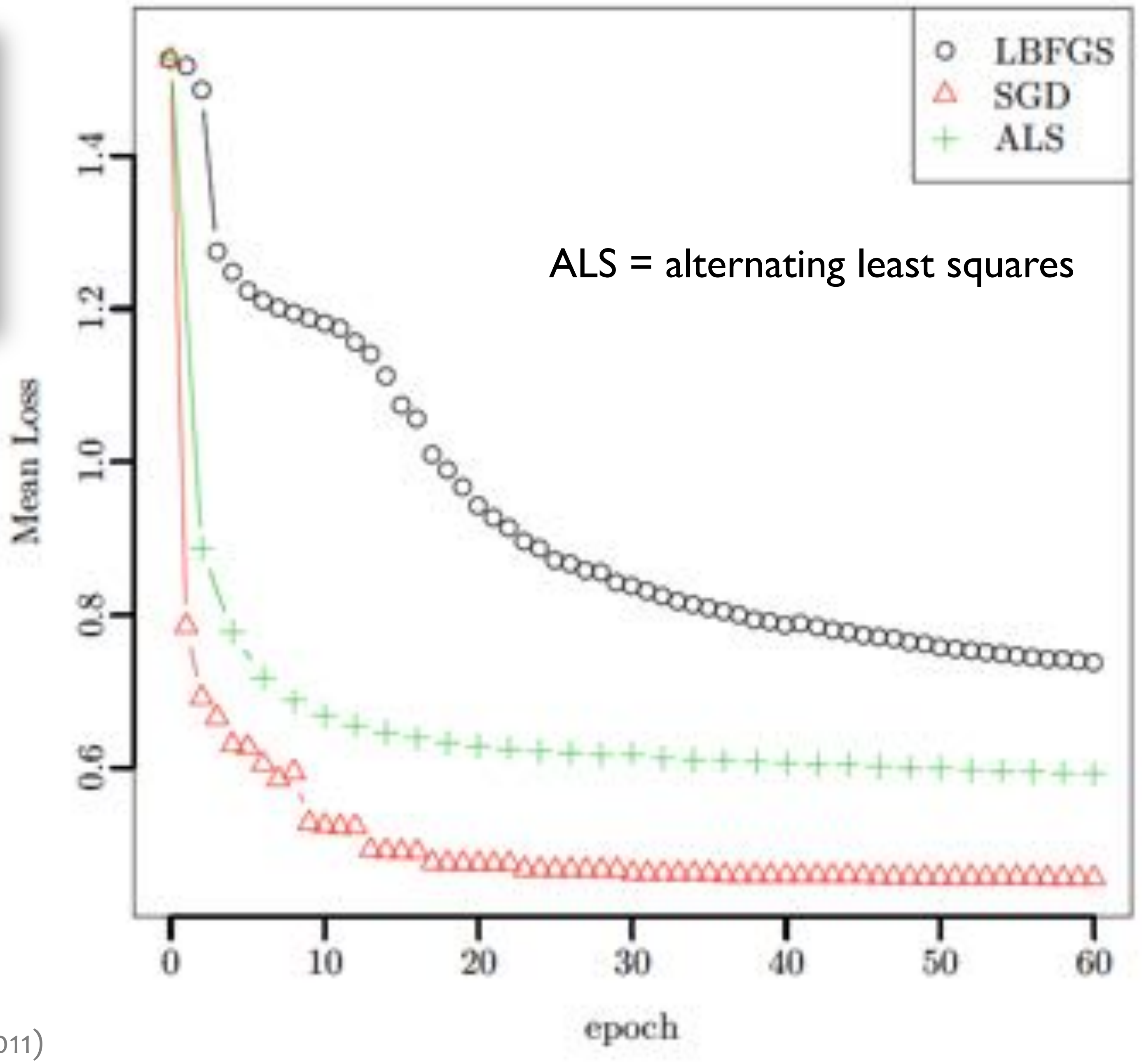


Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films.

Figure from Koren et al. (2009)

Matrix Factorization

Comparison of Optimization Algorithms



ALS = alternating least squares

Figure from Gemulla et al. (2011)

SVD FOR COLLABORATIVE FILTERING

Singular Value Decomposition for Collaborative Filtering

For any arbitrary matrix $\mathbf{A}$, SVD gives a decomposition:

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$$

where $\mathbf{\Lambda}$ is a diagonal matrix, and $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices.

Suppose we have the SVD of our ratings matrix

$$R = Q\Sigma P^T,$$

but then we truncate each of Q, Σ, and P s.t. Q and P have only k columns and Σ is $k \times k$:

$$R \approx Q_k \Sigma_k P_k^T$$

For collaborative filtering, let:

$$U \triangleq Q_k \Sigma_k$$
$$V \triangleq P_k$$
$$\Rightarrow U, V = \underset{U,V}{\operatorname{argmin}} \frac{1}{2} ||R - UV^T||_2^2$$

s.t. columns of U are mutually orthogonal

s.t. columns of V are mutually orthogonal

Theorem: If R fully observed and no regularization, the optimal UV^T from SVD equals the optimal UV^T from Unconstrained MF

NON-NEGATIVE MATRIX FACTORIZATION

Implicit Feedback Datasets

- What information does a five-star rating contain?

- Implicit Feedback Datasets:
  - In many settings, users don't have a way of expressing *dislike* for an item (e.g. can't provide negative ratings)
  - The only mechanism for feedback is to "like" something
- Examples:
  - Facebook has a "Like" button, but no "Dislike" button
  - Google's "+1" button
  - Pinterest pins
  - Purchasing an item on Amazon indicates a preference for it, but there are many reasons you might *not* purchase an item (besides dislike)
  - Search engines collect click data but don't have a clear mechanism for observing dislike of a webpage

Examples from Aggarwal (2016)

Non-negative Matrix Factorization

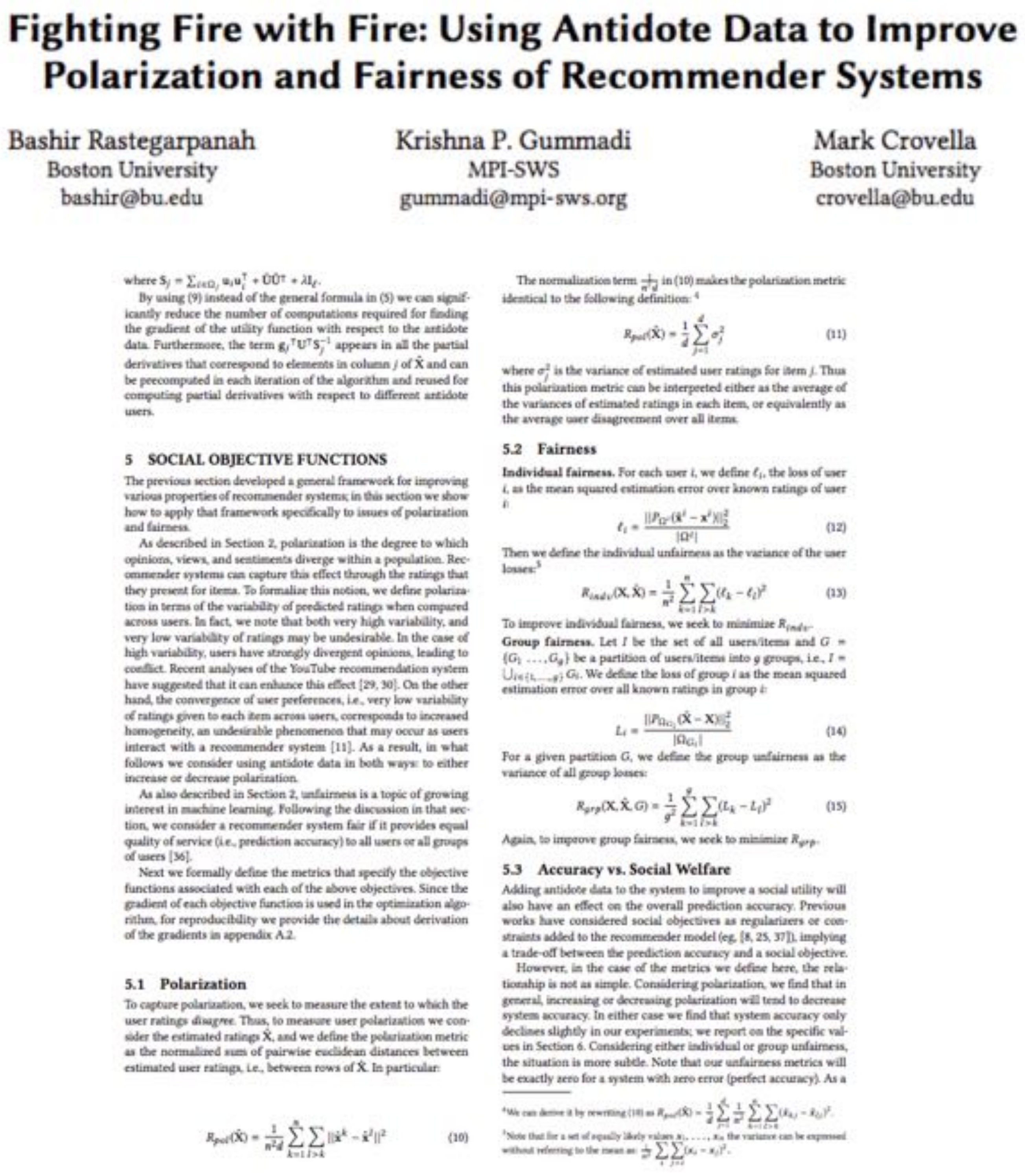
Constrained Optimization Problem:

$$U, V = \operatorname*{argmin}_{U,V} \frac{1}{2}||R - UV^T||_2^2$$

$$\text{s.t. } U_{ij} \geq 0$$

$$\text{s.t. } V_{ij} \geq 0$$

Multiplicative Updates: simple iterative algorithm for solving just involves multiplying a few entries together



Fighting Fire with Fire: Using Antidote Data to Improve Polarization and Fairness of Recommender Systems

Bashir Rastegarpanah
Boston University
bashir@bu.edu

Krishna P. Gummadi
MPI-SWS
gummadi@mpi-sws.org

Mark Crovella
Boston University
crovella@bu.edu

where $\mathbf{S}_j = \sum_{i \in \Omega_j} \mathbf{u}_i \mathbf{u}_i^\top + \tilde{\mathbf{U}}\tilde{\mathbf{U}}^\top + \lambda \mathbf{I}_{\ell}$.

By using (9) instead of the general formula in (5) we can significantly reduce the number of computations required for finding the gradient of the utility function with respect to the antidote data. Furthermore, the term $\mathbf{g}_j{}^\top \mathbf{U}^\top \mathbf{S}_j^{-1}$ appears in all the partial derivatives that correspond to elements in column j of $\hat{\mathbf{X}}$ and can be precomputed in each iteration of the algorithm and reused for computing partial derivatives with respect to different antidote users.

5 SOCIAL OBJECTIVE FUNCTIONS

The previous section developed a general framework for improving various properties of recommender systems; in this section we show how to apply that framework specifically to issues of polarization and fairness.

As described in Section 2, polarization is the degree to which opinions, views, and sentiments diverge within a population. Recommender systems can capture this effect through the ratings that they present for items. To formalize this notion, we define polarization in terms of the variability of predicted ratings when compared across users. In fact, we note that both very high variability, and very low variability of ratings may be undesirable. In the case of high variability, users have strongly divergent opinions, leading to conflict. Recent analyses of the YouTube recommendation system have suggested that it can enhance this effect [29, 30]. On the other hand, the convergence of user preferences, i.e., very low variability of ratings given to each item across users, corresponds to increased homogeneity, an undesirable phenomenon that may occur as users interact with a recommender system [11]. As a result, in what follows we consider using antidote data in both ways: to either increase or decrease polarization.

As also described in Section 2, unfairness is a topic of growing interest in machine learning. Following the discussion in that section, we consider a recommender system fair if it provides equal quality of service (i.e., prediction accuracy) to all users or all groups of users [36].

Next we formally define the metrics that specify the objective functions associated with each of the above objectives. Since the gradient of each objective function is used in the optimization algorithm, for reproducibility we provide the details about derivation of the gradients in appendix A.2.

5.1 Polarization

To capture polarization, we seek to measure the extent to which the user ratings *disagree*. Thus, to measure user polarization we consider the estimated ratings $\hat{\mathbf{X}}$, and we define the polarization metric as the normalized sum of pairwise euclidean distances between estimated user ratings, i.e., between rows of $\hat{\mathbf{X}}$. In particular:

$$R_{pol}(\hat{\mathbf{X}}) = \frac{1}{n^2 d} \sum_{k=1}^{n} \sum_{l>k} \|\hat{\mathbf{x}}^k - \hat{\mathbf{x}}^l\|^2 \qquad (10)$$

The normalization term $\frac{1}{n^2 d}$ in (10) makes the polarization metric identical to the following definition: [4]

$$R_{pol}(\hat{\mathbf{X}}) = \frac{1}{d} \sum_{j=1}^{d} \sigma_j^2 \qquad (11)$$

where σ_j^2 is the variance of estimated user ratings for item j. Thus this polarization metric can be interpreted either as the average of the variances of estimated ratings in each item, or equivalently as the average user disagreement over all items.

5.2 Fairness

Individual fairness. For each user i, we define ℓ_i, the loss of user i, as the mean squared estimation error over known ratings of user i:

$$\ell_i = \frac{\|P_{\Omega^i}(\hat{\mathbf{x}}^i - \mathbf{x}^i)\|_2^2}{|\Omega^i|} \qquad (12)$$

Then we define the individual unfairness as the variance of the user losses: [5]

$$R_{indv}(\mathbf{X}, \hat{\mathbf{X}}) = \frac{1}{n^2} \sum_{k=1}^{n} \sum_{l>k} (\ell_k - \ell_l)^2 \qquad (13)$$

To improve individual fairness, we seek to minimize R_{indv}.

Group fairness. Let I be the set of all users/items and $G = \{G_1 \dots, G_g\}$ be a partition of users/items into g groups, i.e., $I = \bigcup_{i \in \{1,\dots,g\}} G_i$. We define the loss of group i as the mean squared estimation error over all known ratings in group i:

$$L_i = \frac{\|P_{\Omega_{G_i}}(\hat{\mathbf{X}} - \mathbf{X})\|_2^2}{|\Omega_{G_i}|} \qquad (14)$$

For a given partition G, we define the group unfairness as the variance of all group losses:

$$R_{grp}(\mathbf{X}, \hat{\mathbf{X}}, G) = \frac{1}{g^2} \sum_{k=1}^{g} \sum_{l>k} (L_k - L_l)^2 \qquad (15)$$

Again, to improve group fairness, we seek to minimize R_{grp}.

5.3 Accuracy vs. Social Welfare

Adding antidote data to the system to improve a social utility will also have an effect on the overall prediction accuracy. Previous works have considered social objectives as regularizers or constraints added to the recommender model (eg, [8, 25, 37]), implying a trade-off between the prediction accuracy and a social objective.

However, in the case of the metrics we define here, the relationship is not as simple. Considering polarization, we find that in general, increasing or decreasing polarization will tend to decrease system accuracy. In either case we find that system accuracy only declines slightly in our experiments; we report on the specific values in Section 6. Considering either individual or group unfairness, the situation is more subtle. Note that our unfairness metrics will be exactly zero for a system with zero error (perfect accuracy). As a

[4] We can derive it by rewriting (10) as $R_{pol}(\hat{\mathbf{X}}) = \frac{1}{d} \sum_{j=1}^{d} \frac{1}{n^2} \sum_{k=1}^{n} \sum_{l>k} (\hat{x}_{kj} - \hat{x}_{lj})^2$.

[5] Note that for a set of equally likely values $x_1, \dots, x_n$ the variance can be expressed without referring to the mean as: $\frac{1}{n^2} \sum_i \sum_{j>i} (x_i - x_j)^2$.

Summary

- Recommender systems solve many **real-world** (*large-scale) **problems**
- Collaborative filtering by Matrix Factorization (MF) is an **efficient** and **effective** approach
- MF is just another example of a **common recipe:**
    1. define a model
    2. define an objective function
    3. optimize with SGD