



10-601 Introduction to Machine Learning

Machine Learning Department School of Computer Science Carnegie Mellon University

Ensemble Methods



Recommender Systems

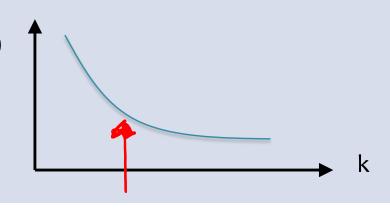
Matt Gormley Lecture 28 Apr. 27, 2020

Reminders

- Homework 9: Learning Paradigms
 - Out: Wed, Apr. 22
 - Due: Wed, Apr. 29 at 11:59pm
 - Can only be submitted up to 3 days late,
 so we can return grades before final exam
- Final Exam Practice Problems
 - Out: Wed, Apr. 29
- Final Exam
 - Mon, May 04 (1pm 4pm)
- Today's In-Class Poll
 - http://poll.mlcourse.org

Q&A

- In k-Means, since we don't have a validation set, how do we pick k?
- A: Look at the training objective function as a function of k
 and pick the value at the "elbo" of the curve.



- Q: What if our random initialization for k-Means gives us poor performance?
- A: Do random restarts: that is, run k-means from scratch, say, 10 times and pick the run that gives the lowest training objective function value.

The objective function is **nonconvex**, so we're just looking for the best local minimum.

EXAM LOGISTICS

Final Exam

Time / Location

- Time: Registrar-scheduled Exam
 Mon, May 4th at 1:00pm 4:00pm
- Online Exam: Same format as Midterm Exam 2
- Please watch Piazza carefully for announcements logistics

Logistics

- Distribution of Topics: Lectures 19 28 (95%), Lectures 1 18 (5%)
- Format of questions:
 - Multiple choice
 - True / False (with justification)
 - Derivations
 - Short answers
 - Interpreting figures
 - Implementing algorithms on paper
- You are encouraged to bring one 8½ x 11 sheet of notes (front and back)
- Open book according to my definition on Piazza: https://piazza.com/class/k4wzus8w2c11u6?cid=1673

Final Exam

How to Prepare

- Attend (or watch) this final exam review session
- Review Practice Problems: Exam 3
 - Disclaimer: the practice problems are somewhere between homework-style problems and exam-style problems
- Review this year's homework problems
- Review the poll questions from each lecture
- Consider whether you have achieved the learning objectives for each lecture / section

Final 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
 - Stochastic Gradient
 Descent
 - Regularization
 - Feature Engineering
- Feature Learning
 - Neural Networks
 - Basic NN Architectures
 - Backpropagation

- Learning Theory
 - PAC Learning
- Generative Models
 - Generative vs.
 Discriminative
 - MLE / MAP
 - Naïve Bayes

Topics for Final Exam

- Graphical Models
 - HMMs
 - Learning and Inference
 - Bayesian Networks
- Reinforcement Learning
 - Value Iteration
 - Policy Iteration
 - Q-Learning
 - Deep Q-Learning

- Other Learning Paradigms
 - K-Means
 - PCA
 - SVM (large-margin)
 - Kernels
 - Ensemble Methods
 - Recommender Systems

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 & (e.g. mixed graphical models)

cont.

Application Areas

Key challenges?

NLP, Speech, Compute
Vision, Robotics, Medic

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

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:

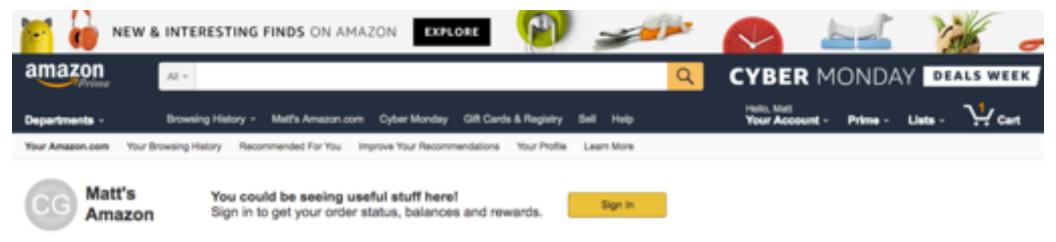
- Ensemble Methods: combine or learn multiple classifiers into one
 (i.e. a family of algorithms)
- 2. Recommender Systems: produce recommendations of what a user will like (i.e. the solution to a particular type of task)

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

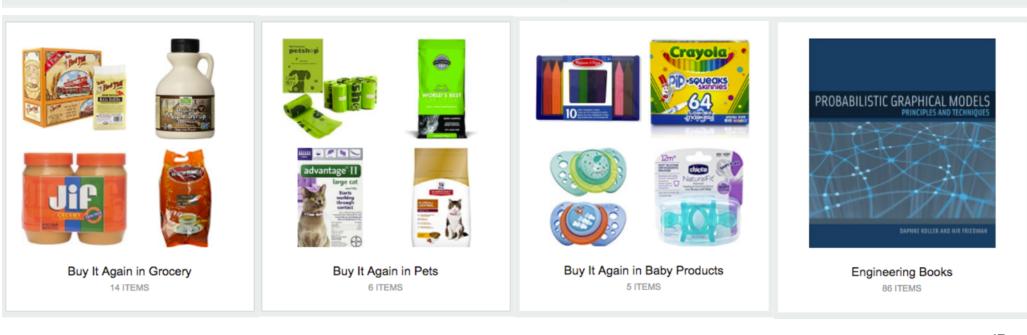
RECOMMENDER SYSTEMS

A Common Challenge:

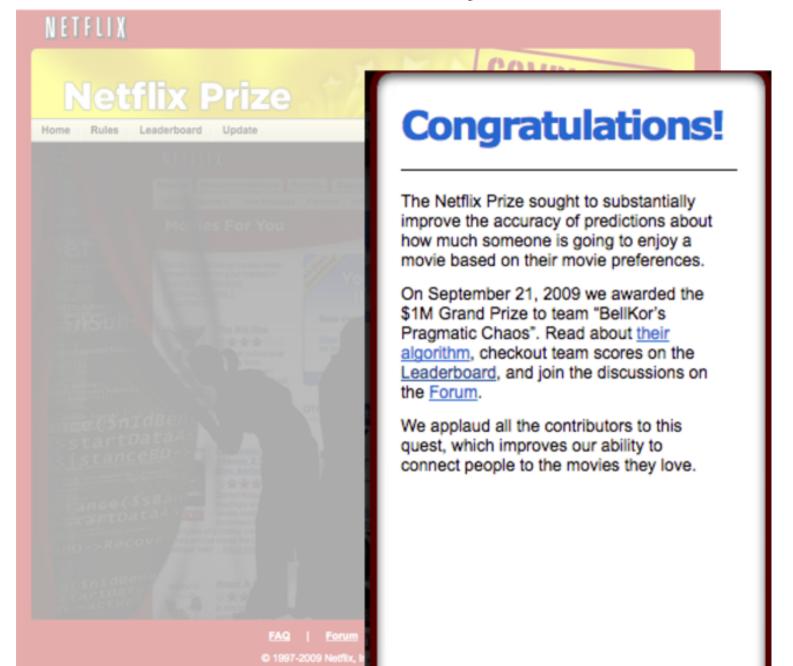
- Assume you're a company selling **items** of some sort: movies, songs, products, etc.
- Company collects millions of ratings from users of their items
- To maximize profit / user happiness, you want to recommend items that users are likely to want

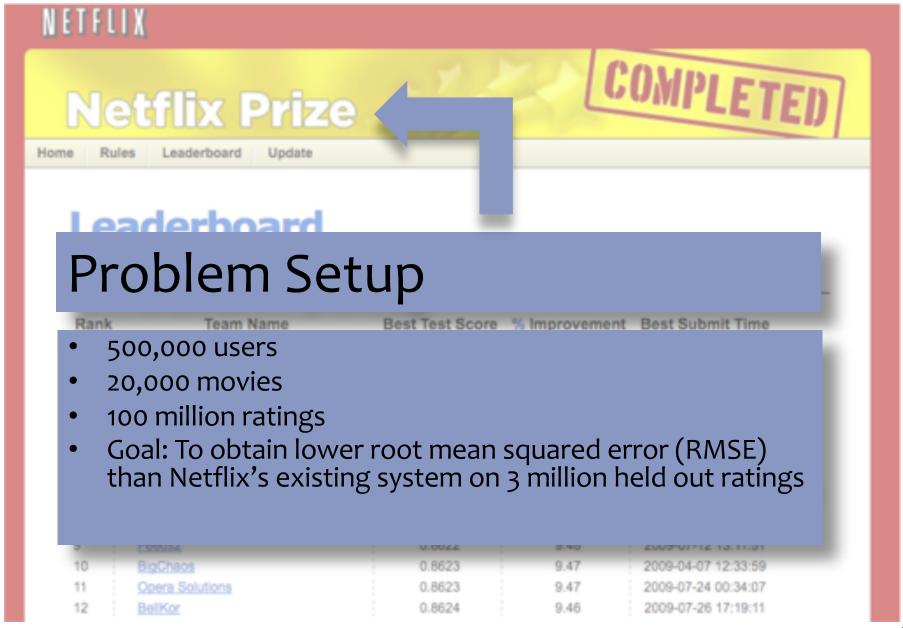


Recommended for you, Matt

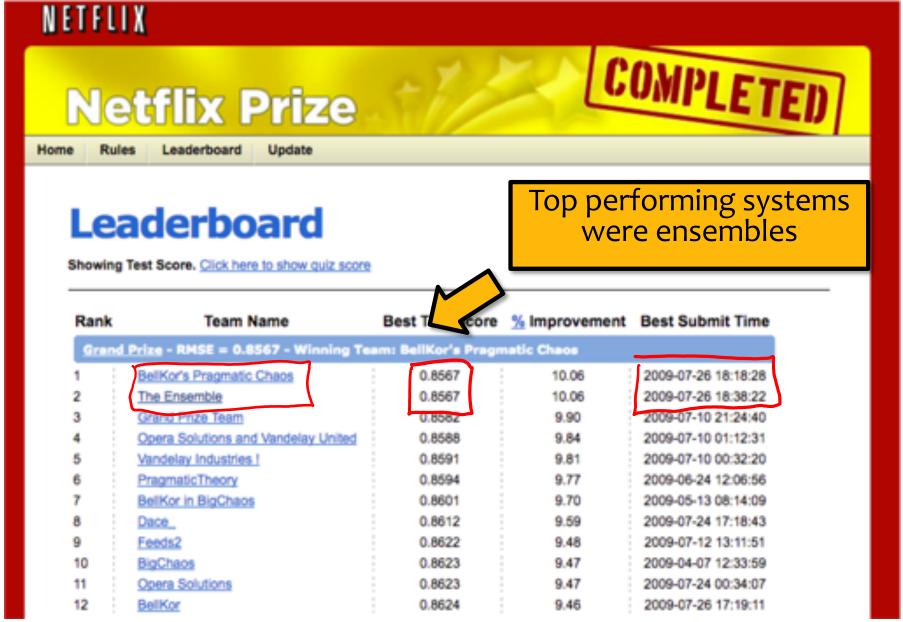








ENSEMBLE METHODS



Weighted Majority Algorithm

(Littlestone & Warmuth, 1994)

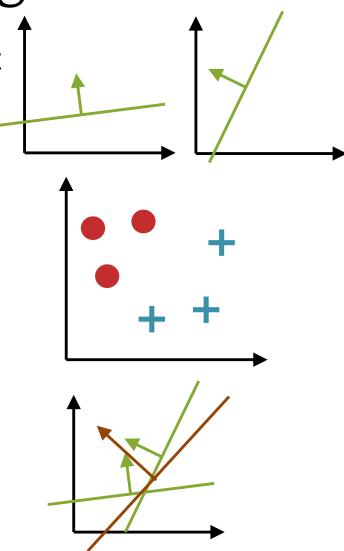
Given: pool A of binary classifiers (that you know nothing about)

 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 $\boldsymbol{\beta}$



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 $\alpha_t = 1, \ \forall t \in \{1, \dots, T\}$
- 3: **for** each training example (x, y) **do**
- 4: Predict majority vote class (splitting ties randomly)

$$\hat{h}(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \underline{\alpha_t h_t(x)}\right)$$

- if a mistake is made $\hat{h}(x) \neq y$ then for each classifier $t \in \{1, \dots, T\}$ do
- 7: If $h_t(x) \neq y$, then $\alpha_t \leftarrow \beta \alpha_t$

Weighted Majority Algorithm

Theorems (Littlestone & Warmuth, 1994)

For the general case where WM is applied to a pool \mathcal{A} of algorithms we show the following upper bounds on the number of mistakes made in a given sequence of trials:

- 1. $O(\log |\mathcal{A}| + m)$, if one algorithm of \mathcal{A} makes at most m mistakes.
- 2. $O(\log \frac{|A|}{k} + m)$, if each of a subpool of k algorithms of A makes at most m mistakes.
- 3. $O(\log \frac{|A|}{k} + \frac{m}{k})$, if the total number of mistakes of a subpool of k algorithms of A is at most m.

These are

"mistake
bounds" of the
variety we saw
for the
Perceptron
algorithm

ADABOOST

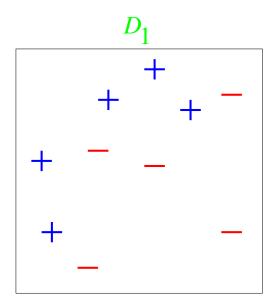
Comparison

Weighted Majority Algorithm

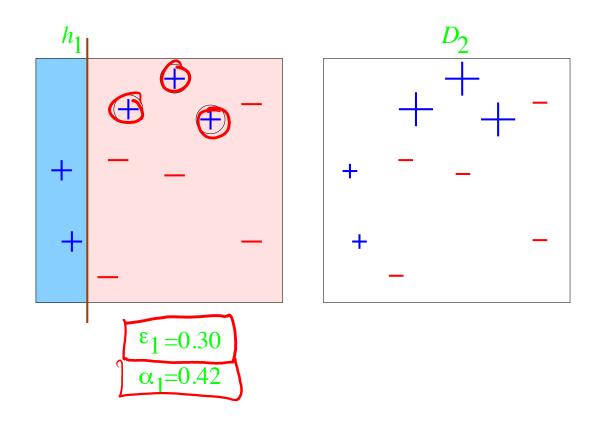
- an example of an ensemble method
- assumes the classifiers are learned ahead of time
- only learns (majority vote) weight for each classifiers

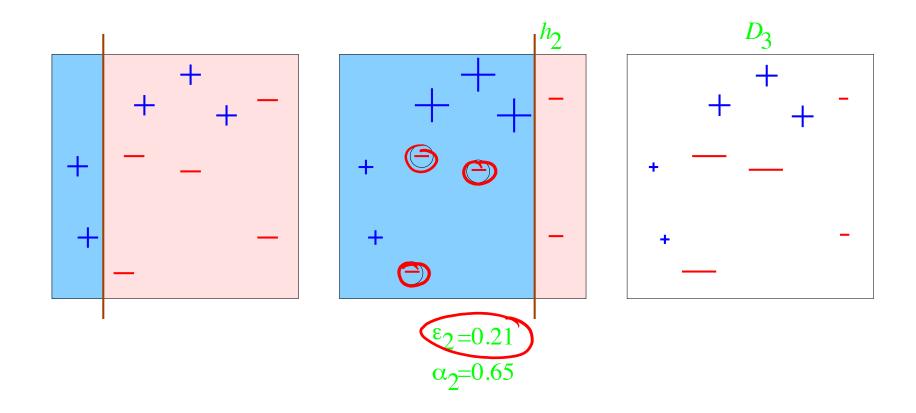
AdaBoost

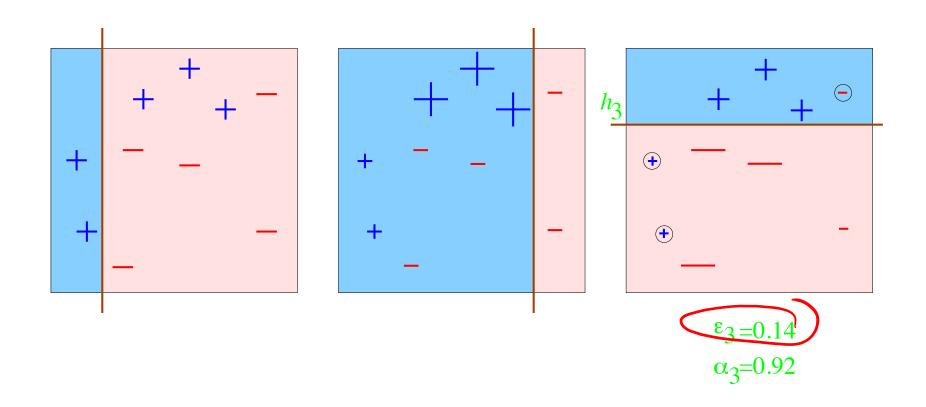
- an example of a boosting method
- simultaneously learns:
 - the classifiers themselves
 - (majority vote) weight for each classifiers

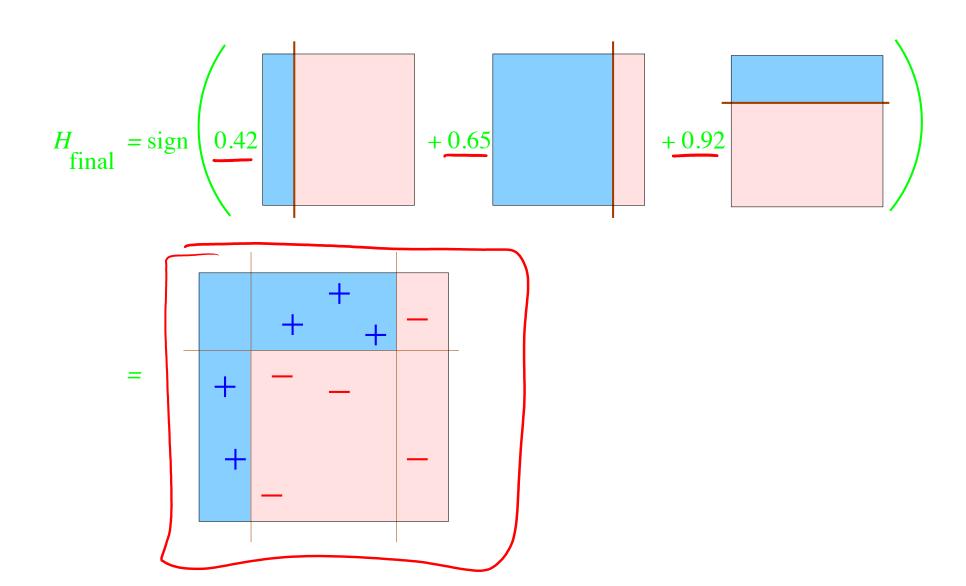


weak classifiers = vertical or horizontal half-planes









AdaBoost

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

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

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

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

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

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

Output the final hypothesis:

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

AdaBoost

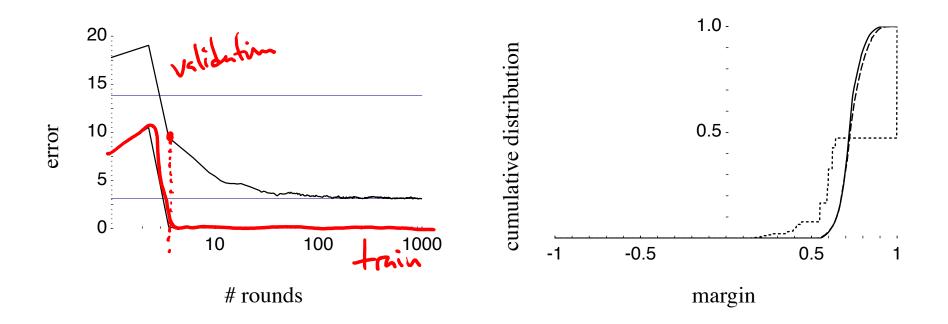


Figure 2: Error curves and the margin distribution graph for boosting C4.5 on the letter dataset as reported by Schapire et al. [41]. *Left*: the training and test error curves (lower and upper curves, respectively) of the combined classifier as a function of the number of rounds of boosting. The horizontal 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.

Learning Objectives

Ensemble Methods / Boosting

You should be able to...

- 1. Implement the Weighted Majority Algorithm
- 2. Implement AdaBoost
- 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

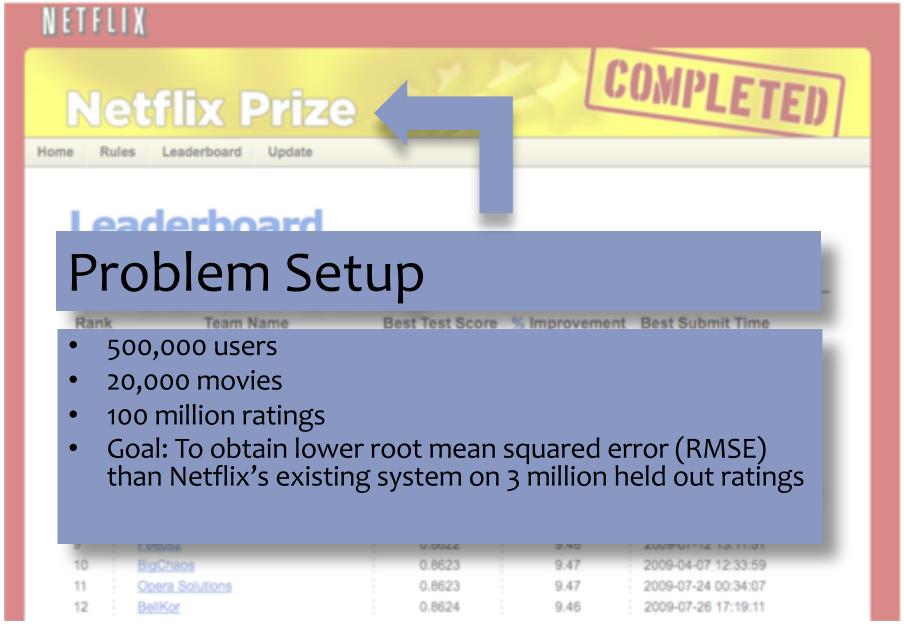
Recommender Systems

- 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



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?"

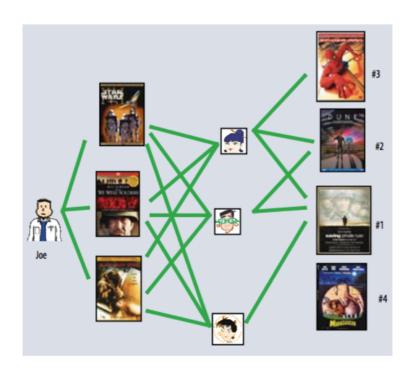
– ...

Common insight: personal tastes are correlated

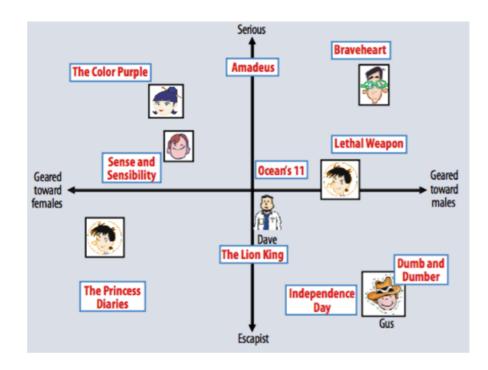
- If Alice and Bob both like X and Alice likes Y then Bob is more likely to like Y
- especially (perhaps) if Bob knows Alice

Two Types of Collaborative Filtering

1. Neighborhood Methods

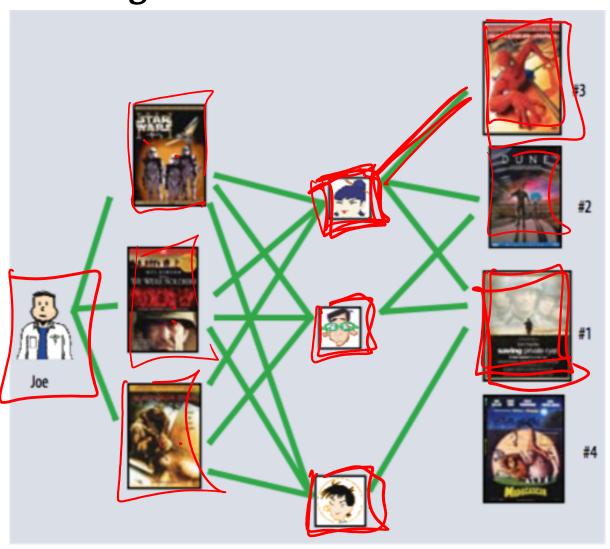


2. Latent Factor Methods



Two Types of Collaborative Filtering

1. Neighborhood Methods



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

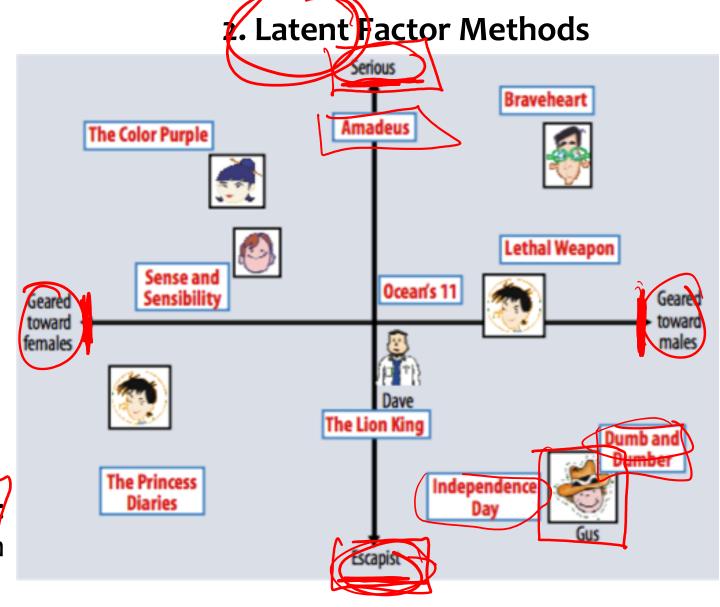
Algorithm:

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

Two Types of Collaborative Filtering

- Assume that both movies and users live in some lowdimensional space describing their properties
- Recommend a
 movie based on its
 proximity to the
 user in the latent
 space

Example AlgorithmMatrix Factorization



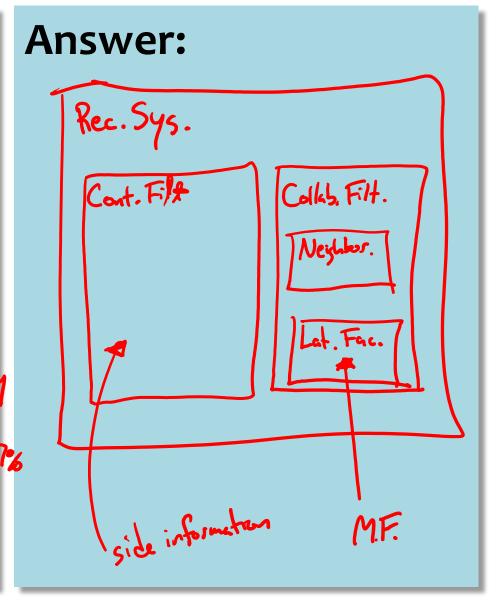
Recommending Movies

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 20%
- B. latent factor methods 44%.
- C ensemble methods calamit
- √D. content filtering **%**%
 - E. neighborhood methods
 - F. recommender systems 1/2%



MATRIX FACTORIZATION

Matrix Factorization

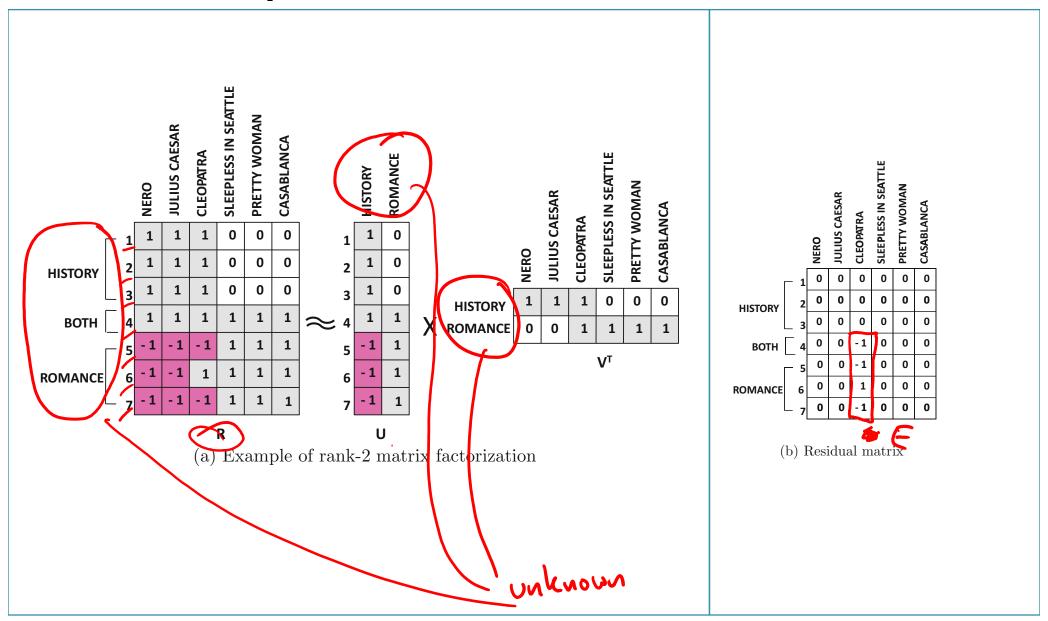
- Many different ways of factorizing a matrix
- We'll consider three:
 - 1. Unconstrained Matrix Factorization 🖔
 - 2. Singular Value Decomposition
 - 3. Non-negative Matrix Factorization
- MF is just another example of a common recipe:
 - define a model
 - 2. define an objective function
 - 3. optimize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

Matrix Factorization

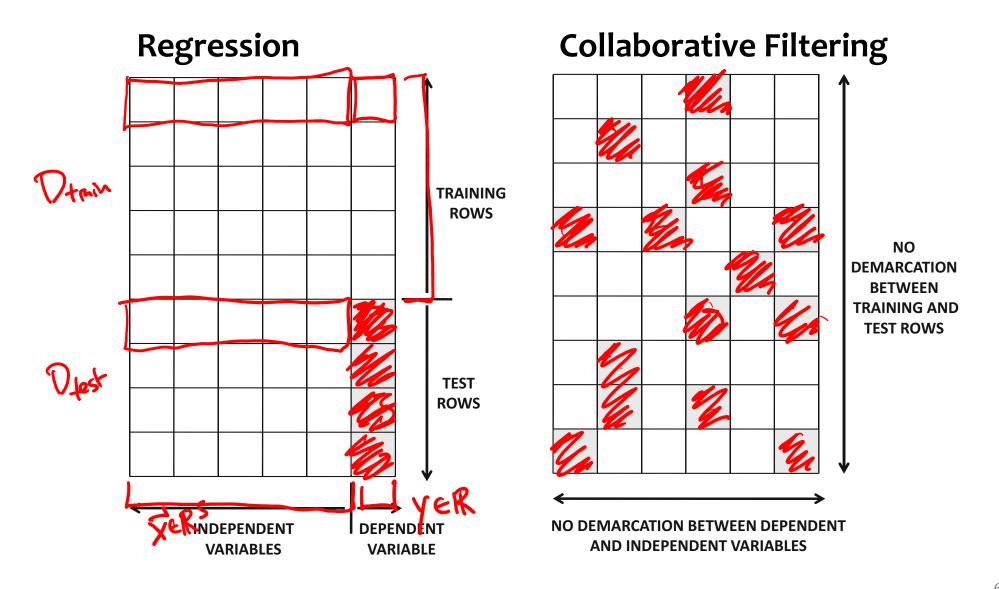
Whiteboard

- Background: Low-rank Factorizations
- Residual matrix

Example: MF for Netflix Problem



Regression vs. Collaborative Filtering



UNCONSTRAINED MATRIX FACTORIZATION

Whiteboard

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

SGD for UMF:

While not conveyed:

(1) Suple (i,j) from Z uniformly at random

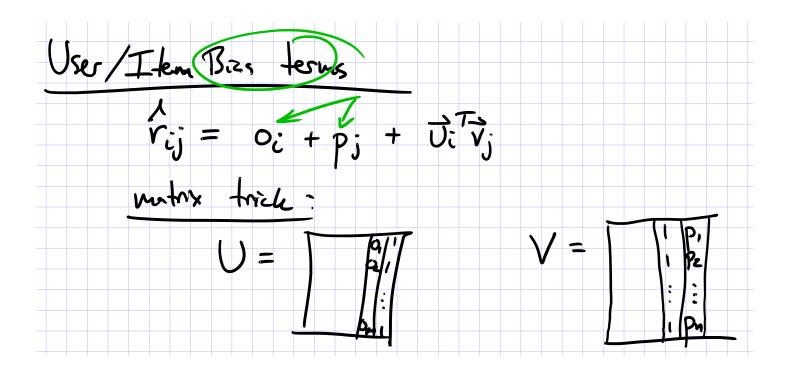
(2) Couple
$$e_{ij} = r_{ij} - \vec{U}_i \vec{\nabla}_j$$

(3) Update

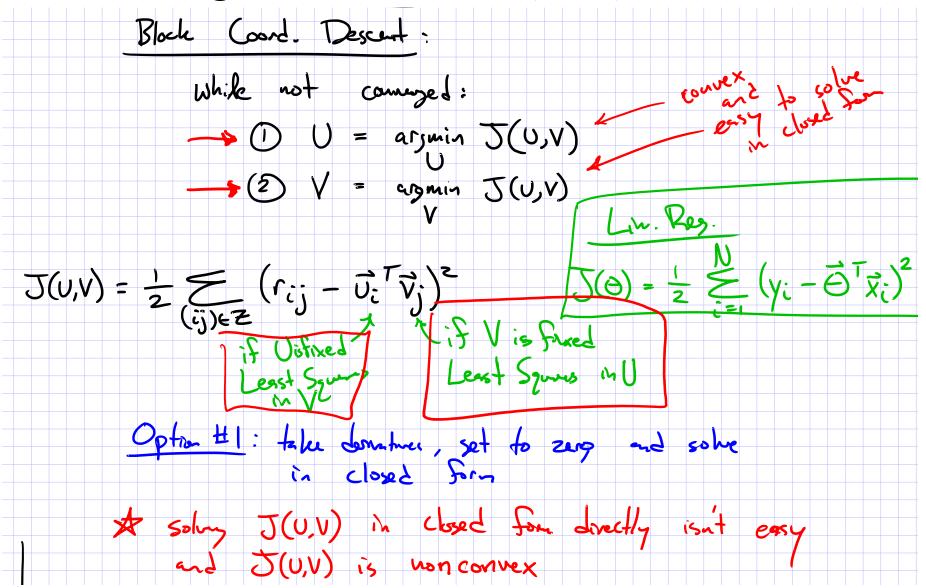
 $\vec{U}_i \leftarrow \vec{U}_i - \gamma \vec{V}_{ij} \vec{J}_{ij}(U,V)$
 $\vec{V}_j \leftarrow \vec{V}_j - \gamma \vec{V}_{ij} \vec{J}_{ij}(U,V)$
 $\vec{V}_{ij} \leftarrow \vec{V}_{ij} - \gamma \vec{V}_{ij} \vec{J}_{ij}(U,V)$
 $\vec{V}_{ij} = \frac{1}{2} (r_{ij} - \vec{U}_i \vec{V}_j)^2 + \lambda (||U_i||_2^2 + ||V_j||_2^2)$
 $\vec{V}_{ij} \vec{J}_{ij}(U,V) = -e_{ij} \vec{U}_i + \lambda \vec{V}_j$

Where $e_{ij} = r_{ij} - \vec{U}_i \vec{V}_j$

SGD for UMF:



Alternating Least Squares (ALS) for UMF:



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:

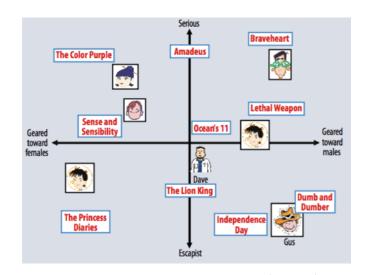
$$(W_{u*})^T \in \mathbb{R}^r$$

Item vectors:

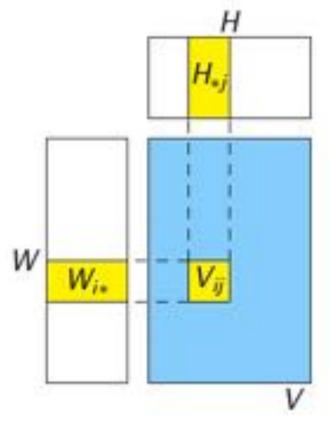
$$H_{*i} \in \mathbb{R}^r$$

Rating prediction:

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



Figures from Koren et al. (2009)



Figures from Gemulla et al. $(2011)_{71}$

User vectors:

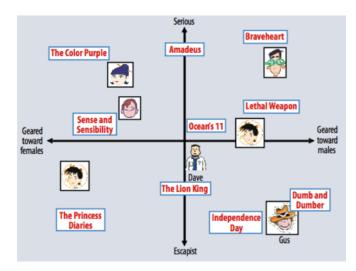
$$\mathbf{w}_u \in \mathbb{R}^r$$

Item vectors:

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

Rating prediction:

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



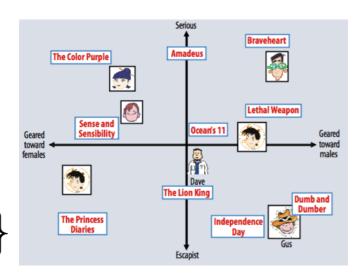
Figures from Koren et al. (2009)

Set of non-missing entries:

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

Objective:

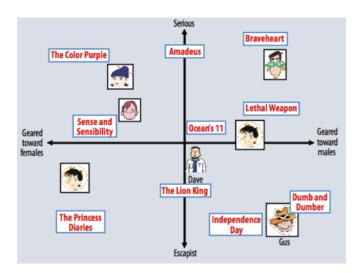
$$\underset{\mathbf{w},\mathbf{h}}{\operatorname{argmin}} \sum_{(u,i)\in\mathcal{Z}} (v_{ui} - \mathbf{w}_u^T \mathbf{h}_i)^2$$



Figures from Koren et al. (2009)

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)

Regularized Objective:

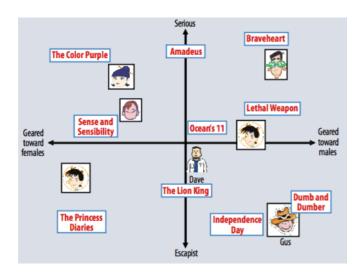
$$\underset{\mathbf{w},\mathbf{h}}{\operatorname{argmin}} \sum_{(u,i)\in\mathcal{Z}} (v_{ui} - \mathbf{w}_{u}^{T} \mathbf{h}_{i})^{2} + \lambda (\sum_{i} ||\mathbf{w}_{i}||^{2} + \sum_{u} ||\mathbf{h}_{u}||^{2})$$

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:

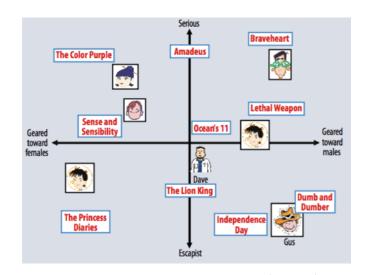
$$(W_{u*})^T \in \mathbb{R}^r$$

Item vectors:

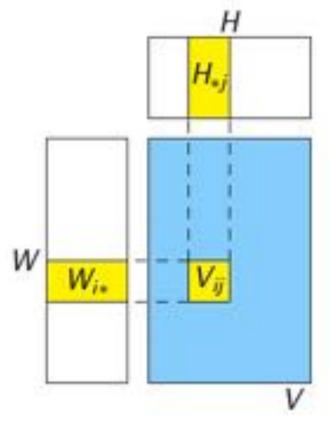
$$H_{*i} \in \mathbb{R}^r$$

Rating prediction:

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



Figures from Koren et al. (2009)



Figures from Gemulla et al. $(2011)_{76}$

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.

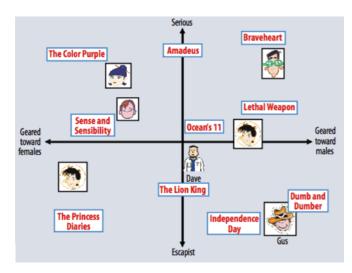
$$W'_{i*} \leftarrow W_{i*} - \epsilon_n N \frac{\partial}{\partial W_{i*}} l(V_{ij}, W_{i*}, H_{*j})$$

$$\boldsymbol{H}_{\star j} \leftarrow \boldsymbol{H}_{\star j} - \epsilon_n N \frac{\partial}{\partial \boldsymbol{H}_{\star j}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i\star}, \boldsymbol{H}_{\star j})$$

$$W_{i*} \leftarrow W'_{i*}$$
 end while

step size

Figure from Gemulla et al. (2011)



Figures from Koren et al. (2009)

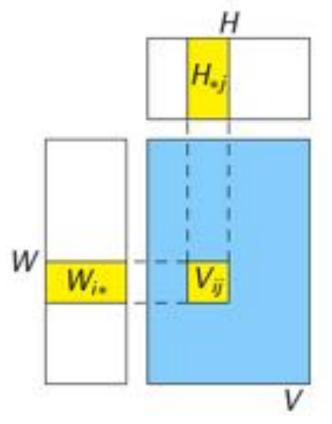


Figure from Gemulla et al. $(2011)_{77}$

Matrix Factorization

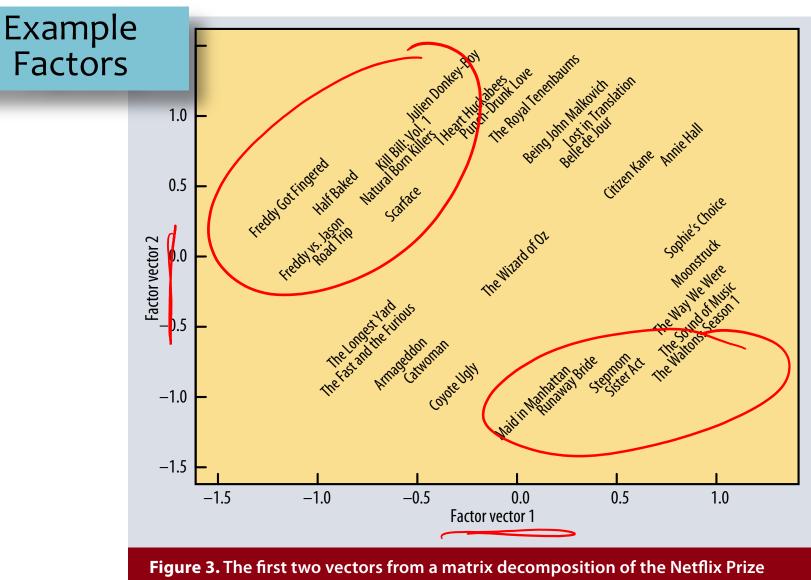
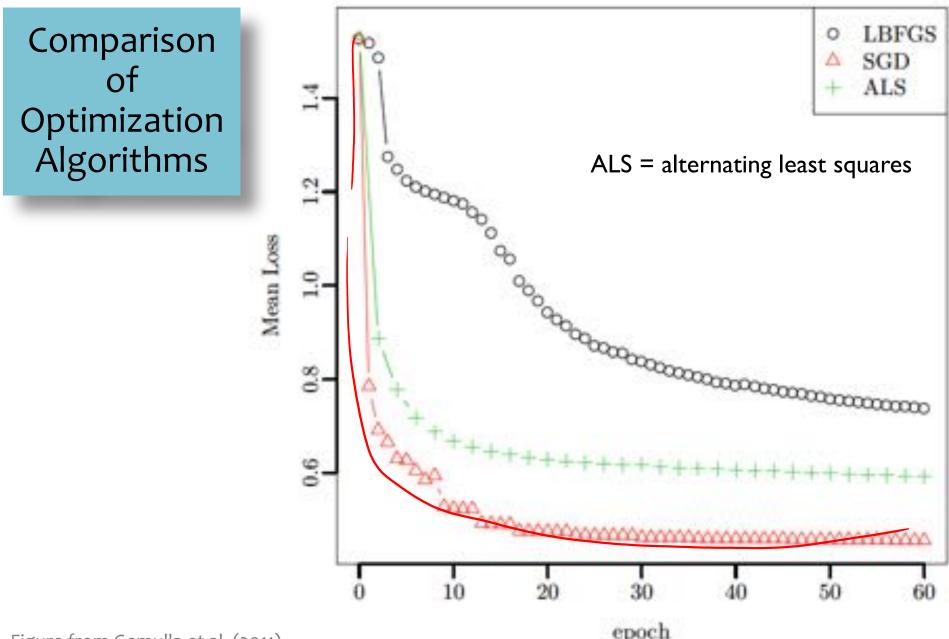


Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky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

Matrix Factorization



SVD FOR COLLABORATIVE FILTERING

Singular Value Decomposition for Collaborative Filtering

For any arbitrary matrix A, SVD gives a decomposition:

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

where Λ is a diagonal matrix, and ${f U}$ and ${f V}$ 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

Non-negative Matrix Factorization

Constrained Optimization Problem:

$$U, V = \operatorname*{argmin} \frac{1}{2}||R - UV^T||_2^2$$
 s.t. $U_{ij} \geq 0$ 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 $S_j = \sum_{l \in \Omega_j} u_l u_l^T + \hat{U}\hat{U}^T + \lambda I_\ell$. 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 $g_j^T U^T S_j^{-1}$ appears in all the partial derivatives that correspond to elements in column j of \hat{X} and can be precomputed in each iteration of the algorithm and reused for computing partial derivatives with respect to different antidote

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

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 X, and we define the polarization metric as the normalized sum of pairwise euclidean distances between estimated user ratings, i.e., between rows of X. In particular:

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

The normalization term 🚉 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_i^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

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

Then we define the individual unfairness as the variance of the user

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

To improve individual fairness, we seek to minimize R_{inds} :

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

$$L_{i} = \frac{||P_{\Omega_{G_{i}}}(\hat{\mathbf{X}} - \mathbf{X})||_{2}^{2}}{|\Omega_{G_{i}}|}$$
(14)

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

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

⁴We can derive it by rewriting (10) as $R_{pool}(\hat{\mathbf{X}}) = \frac{1}{d} \int_{j-1}^{d} \frac{1}{n^j} \sum_{k-1,l>k} (\hat{\mathbf{x}}_{k,j} - \hat{\mathbf{x}}_{l,j})^j$.

⁵Note that for a set of equally likely values $\mathbf{x}_1, \dots, \mathbf{x}_n$ the variance can be expressed without referring to the mean as: $\frac{1}{n^k} \sum_{l=1}^{d} [\mathbf{x}_l - \mathbf{x}_l)^l$.

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:
 - define a model
 - 2. define an objective function
 - optimize with your favorite black box optimizer (e.g. SGD, Gradient Descent, Block Coordinate Descent aka. Alternating Least Squares)