Recommender Systems

Machine Learning – CSEP546

Carlos Guestrin
University of Washington

February 10, 2014

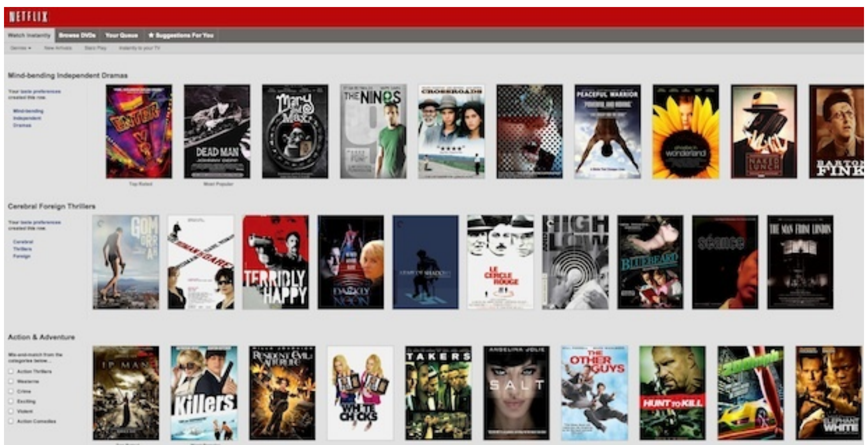
Personalization is transforming our experience of the world

- Information overload ➔ "Browsing" is history
  - Need fundamentally new ways to discover content
- Personalization: Connects *users & items*

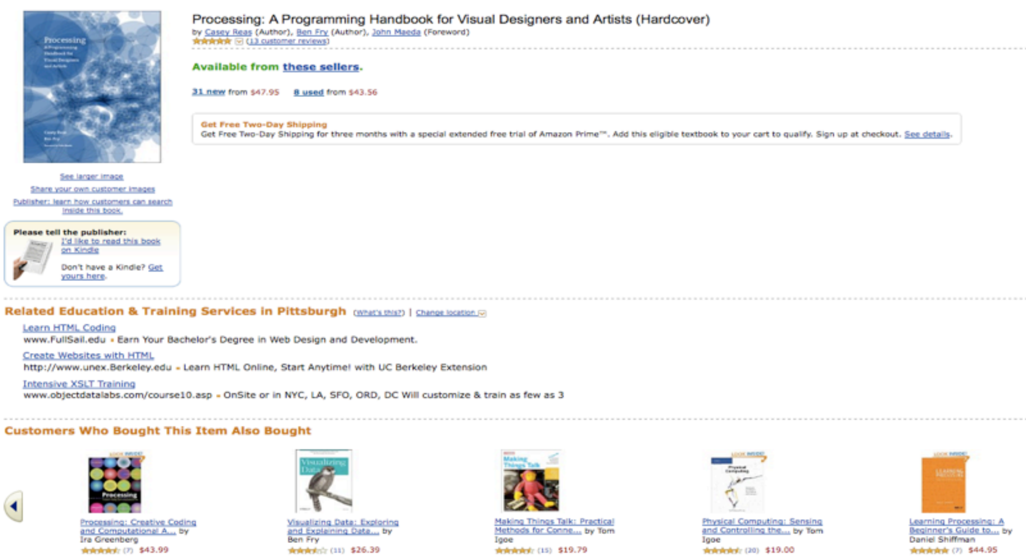
YouTube

100 Hours a Minute
What do I care about?

Movie recommendations

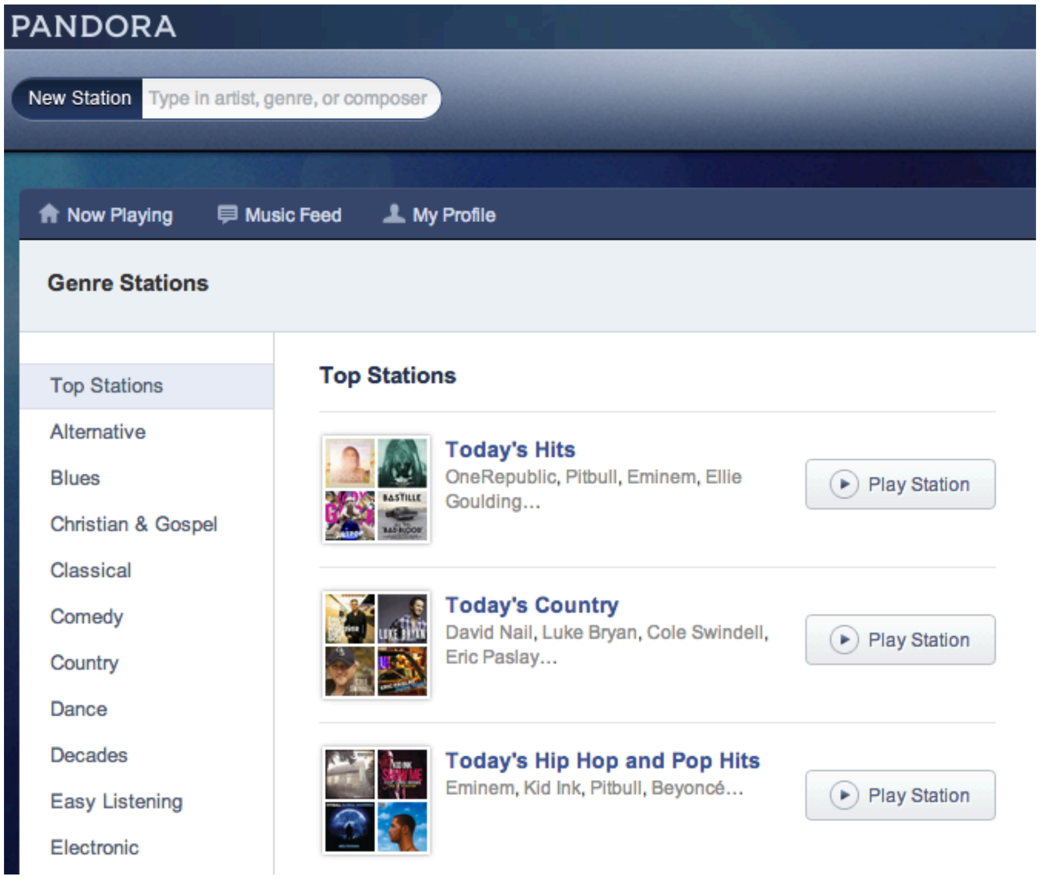


Product recommendations



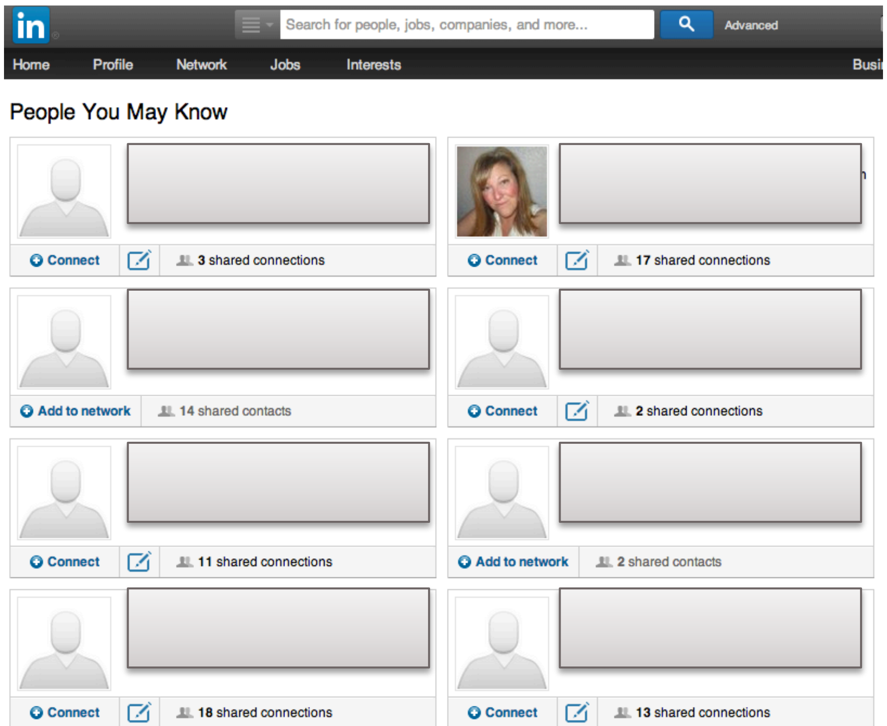
Recommendations combine global & session interests

Playlist recommendations



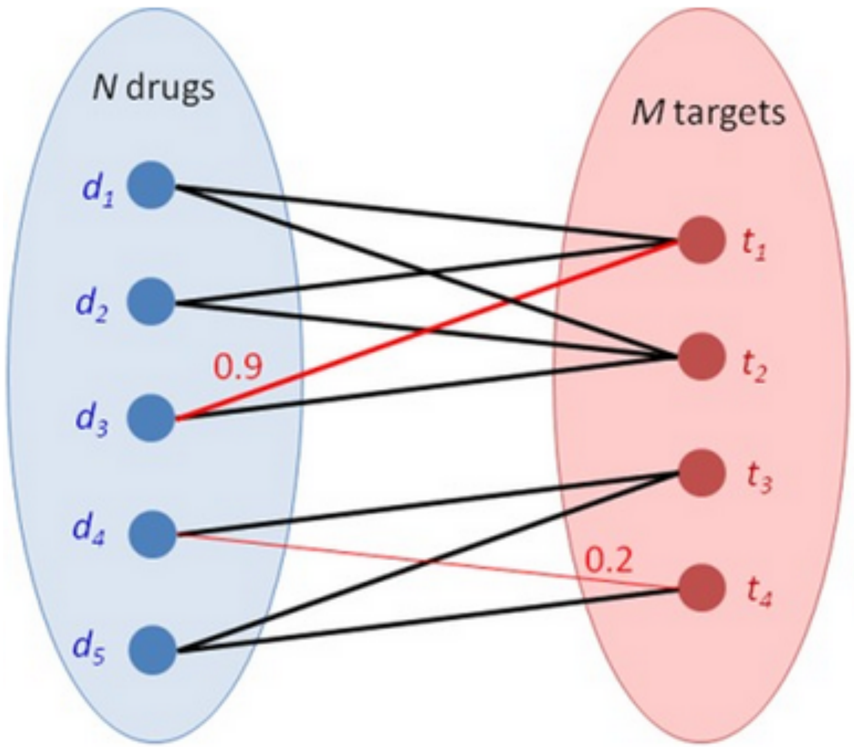
Recommendations form coherent & diverse sequence

Friend recommendations



Users and "items" are of the same type

Drug-target interactions



What drug should we "repurpose" for some disease?

Cobanoglu et al. '13

Challenges of developing recommender systems

Type of feedback

- Explicit – user tells us what she likes

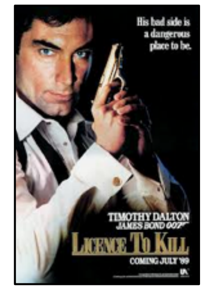
- Implicit – we try to infer what she likes from usage data

Top K versus diverse outputs

- Top K recommendations may be very redundant
  - *People who liked Rocky 1 also enjoyed Rocky 2, Rocky 3, Rocky 4, Rocky 5,…*
- Diverse recommendations
  - Users are multi-facetted & want to hedge our bets
  - Rocky 2, It never rains in Philadelphia, Gandhi

A new movies walks into a bar…

IN THEATERS



- **Cold-start problem**: recommendations for new users or new movies
  - Need side information about user/movie
    - A.K.A. features!
  - Could also play 20-questions game…

That's so last year…

- Interests change over time…
  - Is it 1967?
  - Or 1977?
  - Or 1988?
  - Or 1998?
  - Or 2011?
- Models need flexibility to adapt to users
  - Macro scale
  - Micro scale
- And keep checking that system still accurate

macys.com

Scalability

- For N users and M movies, some approaches take $O(N^3+M^3)$
  - Not so good for billions of users…

- GraphLab can help…
  - Efficient implementations
  - Fast exact & approximate methods as needed

Building a recommender system

Solution 0: Popularity

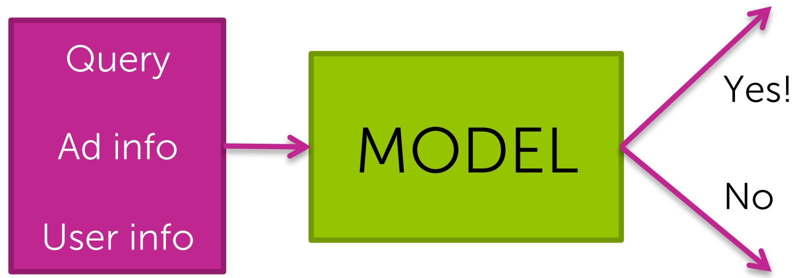
Simplest approach: popularity

- What people are viewing now
  - Super popular

- Limitations:
  - No context (what's my intention now)
  - No personalization

Solution 1: Click prediction

What's the probability I'll buy this product?



- Features capture context
  - Time of the day, what I just saw, user info, what I bought in the past
- Helps mitigate cold-start problem
  - Rate new movie from features of other movies user liked

- Limitation:
  - May not have context available
  - Often doesn't perform as well as collaborative filtering methods (next)

Solution 2: People who bought this also bought…

Co-occurrence matrix

- Matrix C: item by item

- $C_{ij} = C_{ji}$ number of users who bought both items i & j

Normalizing co-occurrences: similarity matrix

- C_{ij} very large if either *i* or *j* are very popular movies ➔ drowns out other effects
  - just recommends by popularity
- Jaccard similarity: normalizes by popularity
  - Who watched *i* **and** *j divided by* who watched *i* **or** *j*

- Many other similarity metrics possible, e.g., cosine similarity

Using similarity matrix to recommend

- People who bought diapers also bought beer
- For *i=diapers*, sort S_{ij} and find *j* with highest similarity
  - Beer, milk, baby food,…

- Limitation:
  - Only current page matters, no history

Solution 3: Average item-item similarity

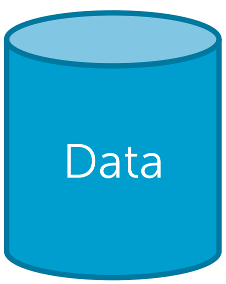
(Weighted) average over items user bought

- User *u* bought items B_u
  - Define user specific similarity (score) for each item *j*
    - Average similarity for items in B_u
    - Could also weight recent purchases more
- For B_u=*{diapers,beer}* sort *Score(u,j)* and find *j* with highest similarity
- Limitation:
  - Scalability – similarity matrix M^2 size

Training versus testing data

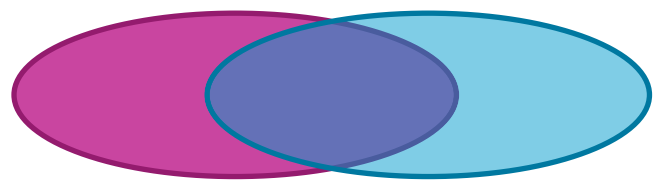
- A/B testing standard in industry:
  - Randomly split users into groups A & B
  - Show different websites
  - Compare outcomes

- Same idea fundamental in ML
  - Randomly split data into train and test sets
  - Train on training data, evaluate on test data



Example Performance metric for recommenders

- User *u* liked *m* movies, we showed her *k* movies



- *Recall:* what fraction of the liked movies we found
- *Precision:* what fraction of the movies we showed she liked
- Precision-Recall curve:

Limitations of item-based similarity

- Scalability – similarity matrix M^2 size
- Cold-start problem

Solution 4: Discovering hidden structure by matrix factorization

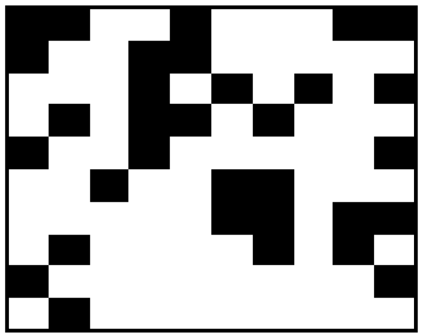
Suppose we had *d* features of movies and users

- Describe movie *v* with features R_v
  - How much is it action, romance, drama,…
- Describe user *u* with features L_u
  - How much she likes action, romance, drama,…
- *Score(u,v)* is the product of the two vectors

But we don't know features of users and movies…

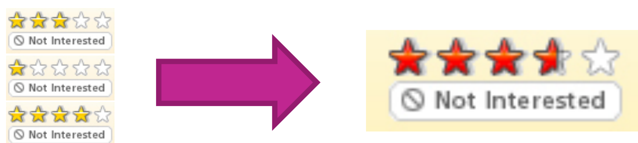
Matrix Completion Problem

Score =



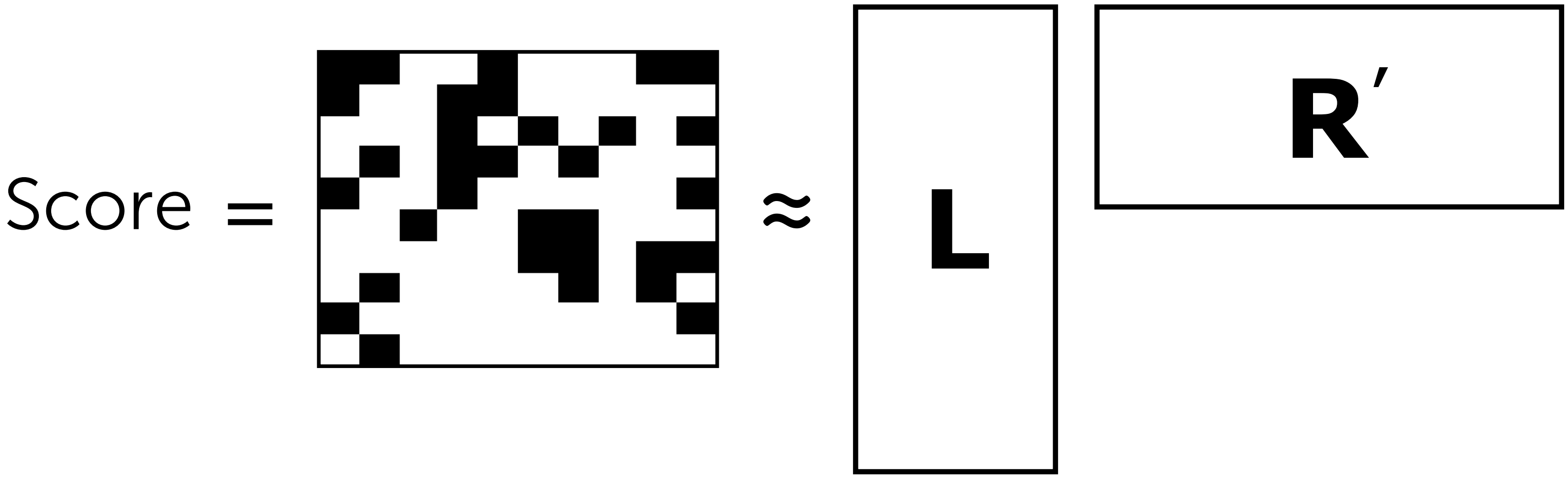
Score(u,v) known for black cells
Score(u,v) unknown for white cells
Rows index users
Columns index movies

- Users score some movies



- Filling missing data?

Matrix Factorization: discovering features of users and movies



Many efficient algorithms for
matrix factorization implemented in GraphLab

Using the results of matrix factorization

- Discover “features” R_v for each movie *v*

- Discover “features” L_u for each user *u*

- *Score(u,v)* is the product of the two vectors ➔ predict how much a user will like a movie

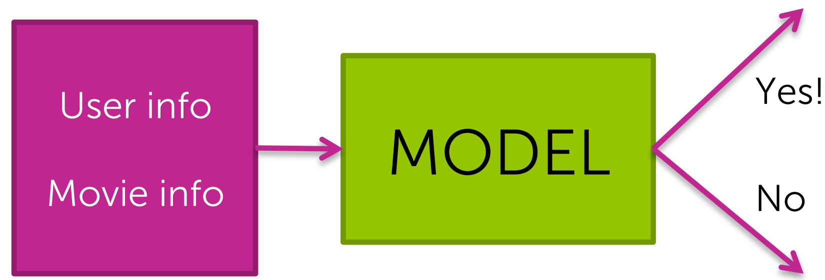
- Recommendations: sort movies user hasn’t watched by *Score(u,v)*

Limitations of matrix factorization

- Cold-start problem

Bringing it all together: Featurized matrix factorization

Combining real and discovered features



- Real features capture context
  - Time of the day, what I just saw, user info, what I bought in the past
- Discovered features from matrix factorization capture groups of users who behave similarly
  - Hipster wannabes from Seattle who teach and have a startup

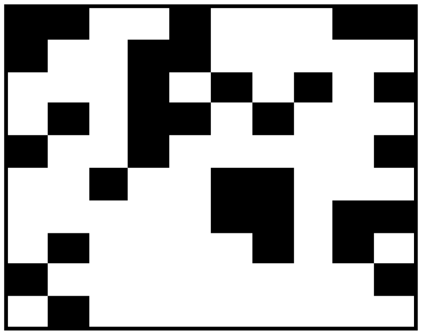
- Mitigates cold-start problem
  - Ratings for a new user from real features only
  - As more information about user is discovered, matrix factorization “features” become more relevant

Matrix Factorization
Alternating Least Squares

Machine Learning – CSEP546
Carlos Guestrin
University of Washington

February 10, 2014

Matrix Completion Problem

$\mathbf{X} =$ 

X_{ij} known for black cells

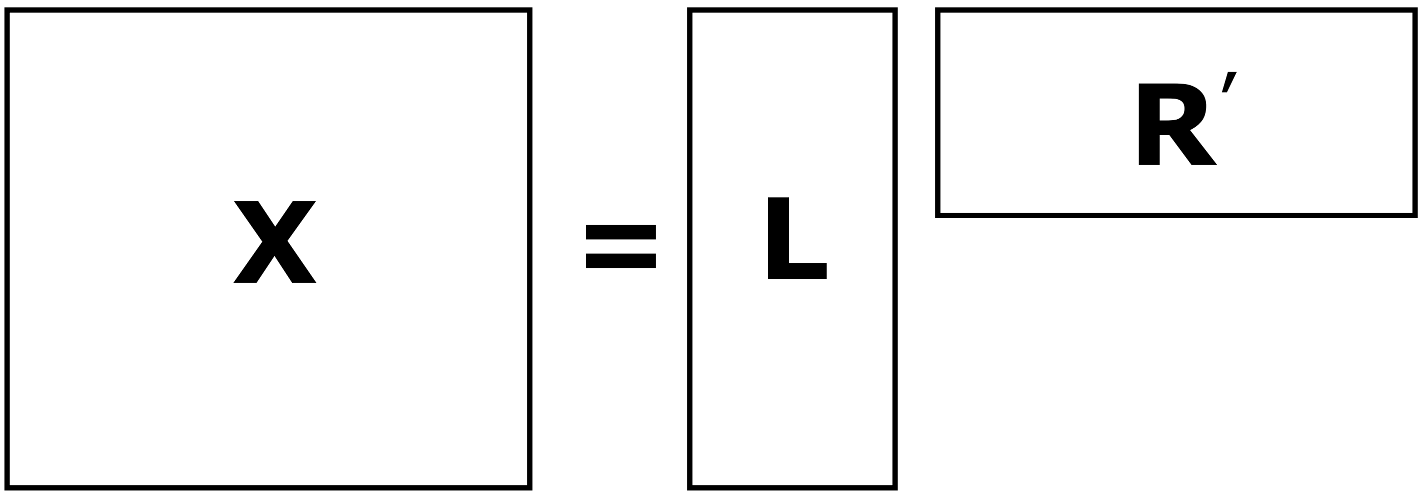
X_{ij} unknown for white cells

Rows index users

Columns index movies

- Filling missing data?

Interpreting Low-Rank Matrix Completion (aka Matrix Factorization)



Matrix Completion via Rank Minimization

- Given observed values:
- Find matrix
- Such that:
- But…
- Introduce bias:
- Two issues:

Approximate Matrix Completion

- Minimize squared error:
  - (Other loss functions are possible)

- Choose rank k:

- Optimization problem:

Coordinate Descent for Matrix Factorization

$$\min_{L,R} \sum_{(u,v):r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2$$

- Fix movie factors, optimize for user factors
- First Observation:

Minimizing Over User Factors

- For each user *u*: $\min_{L_u} \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2$

- In matrix form:

- Second observation: Solve by

Coordinate Descent for Matrix Factorization: Alternating Least-Squares

$$\min_{L,R} \sum_{(u,v):r_{uv}\neq ?} (L_u \cdot R_v - r_{uv})^2$$

- Fix movie factors, optimize for user factors
  - Independent least-squares over users

$$\min_{L_u} \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2$$

- Fix user factors, optimize for movie factors
  - Independent least-squares over movies

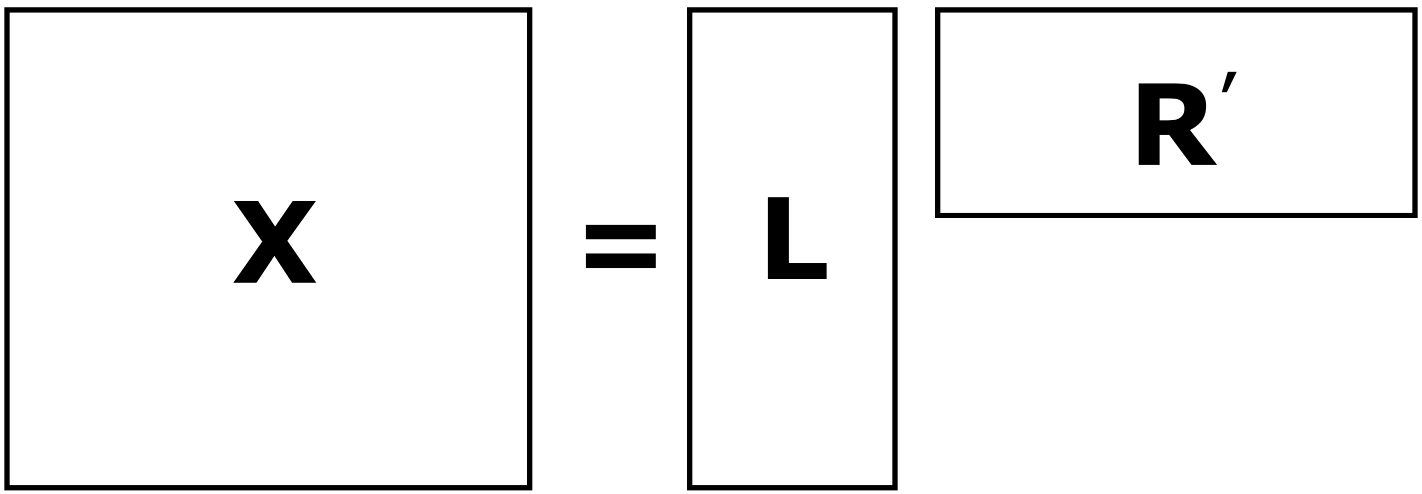
$$\min_{R_v} \sum_{u \in U_v} (L_u \cdot R_v - r_{uv})^2$$

- System may be underdetermined:

- Converges to

Effect of Regularization

$$\min_{L,R} \sum_{(u,v):r_{uv}\neq ?} (L_u \cdot R_v - r_{uv})^2$$



Stochastic Gradient Descent

$$\min_{L,R} \sum_{r_{uv}} (L_u \cdot R_v - r_{uv})^2 + \lambda_u ||L||_2^2 + \lambda_v ||R||_2^2$$

- Observe one rating at a time r_{uv}
- Gradient:
- Updates:

What you need to know…

- Matrix completion problem for collaborative filtering
- Over-determined -> low-rank approximation
- Rank minimization is NP-hard
- Minimize least-squares prediction for known values for given rank of matrix
  - Must use regularization
- Coordinate descent algorithm = "Alternating Least Squares"

Non-Negative Matrix Factorization

Machine Learning – CSEP546
Carlos Guestrin
University of Washington

February 10, 2014

Matrix factorization solutions can be unintuitive…

- Many, many, many applications of matrix factorization

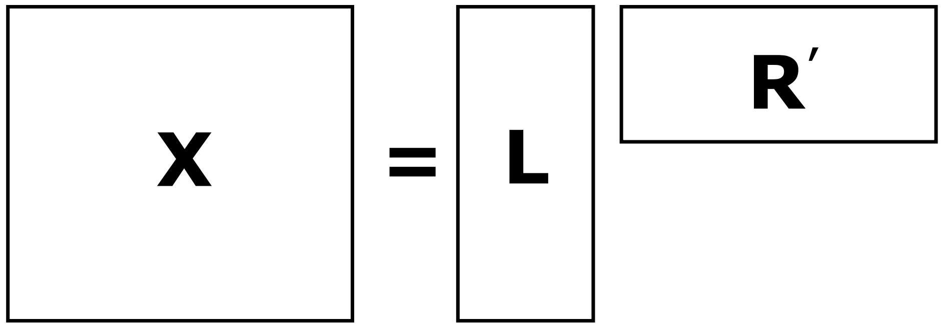
- E.g., in text data, can do topic modeling:

$$\mathbf{X} = \mathbf{L}\,\mathbf{R}'$$

- Would like:

- But…

Nonnegative Matrix Factorization



- Just like before, but

$$\min_{L \geq 0, R \geq 0} \sum_{r_{uv}} (L_u \cdot R_v - r_{uv})^2 + \lambda_u ||L||_F^2 + \lambda_v ||R||_F^2$$

- Constrained optimization problem
  - Many, many, many, many solution methods… we'll check out a simple one

Projected Gradient

- Standard optimization:
  - Want to minimize: $\min_{\Theta} f(\Theta)$
  - Use gradient updates:

$$\Theta^{(t+1)} \leftarrow \Theta^{(t)} - \eta_t \nabla f(\Theta^{(t)})$$

- Constrained optimization:
  - Given convex set *C* of feasible solutions
  - Want to find minima within *C*: $\min_{\Theta} f(\Theta)$

$$\Theta \in \mathcal{C}$$

- Projected gradient:
  - Take a gradient step (ignoring constraints):
  - Projection into feasible set:

Projected Stochastic Gradient Descent for Nonnegative Matrix Factorization

$$\min_{L \geq 0, R \geq 0} \frac{1}{2} \sum_{r_{uv}} (L_u \cdot R_v - r_{uv})^2 + \frac{\lambda_u}{2} ||L||_F^2 + \frac{\lambda_v}{2} ||R||_F^2$$

- Gradient step observing r_{uv} ignoring constraints:

$$\begin{bmatrix} \tilde{L}_u^{(t+1)} \\ \tilde{R}_v^{(t+1)} \end{bmatrix} \leftarrow \begin{bmatrix} (1 - \eta_t \lambda_u) L_u^{(t)} - \eta_t \epsilon_t R_v^{(t)} \\ (1 - \eta_t \lambda_v) R_v^{(t)} - \eta_t \epsilon_t L_u^{(t)} \end{bmatrix}$$

- Convex set:
- Projection step:

What you need to know…

- In many applications, want factors to be nonnegative
- Corresponds to constrained optimization problem
- Many possible approaches to solve, e.g., projected gradient

The Cold-Start Problem

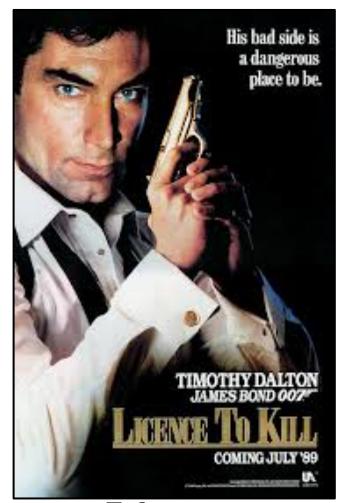
Machine Learning – CSEP546
Carlos Guestrin
University of Washington

February 10, 2014

Cold-Start Problem

- **Challenge:** Cold-start problem (new movie or user)
- **Methods:** use features of movie/user

IN THEATERS



Cold-Start More Formally

- No observations about a particular user:

$$\min_{L,R} \frac{1}{2} \sum_{r_{uv}} (L_u \cdot R_v - r_{uv})^2 + \frac{\lambda_u}{2} ||L||_F^2 + \frac{\lambda_v}{2} ||R||_F^2$$

- A simpler model for collaborative filtering:
  - Observe ratings:
  - Given features of a movie:
  - Fit linear model:
  - Minimize:

Personalization

- If we don't have any observations about a user, use wisdom of the crowd
  - Address cold-start problem

- But, as we gain more information about the user, forget the crowd:

User Features…

- In addition to movie features, may have information user:

- Combine with features of movie:

- Unified linear model:

Feature-based Approach versus Matrix Factorization

- Feature-based approach:
  - Feature representation of user and movies fixed
  - Can address cold-start

- Matrix factorization approach:
  - Suffers from cold-start problem
  - User & movie features are learned from data

- Unified model:

MAP for Unified Collaborative Filtering via SGD

$$\min_{L,R,w,\{w_u\}_u} \frac{1}{2}\sum_{r_{uv}}(L_u \cdot R_v + (w + w_u)\cdot\phi(u,v) - r_{uv})^2$$
$$+ \frac{\lambda_u}{2}||L||_F^2 + \frac{\lambda_v}{2}||R||_F^2 + \frac{\lambda_w}{2}||w||_2^2 + \frac{\lambda_{wu}}{2}\sum_u ||w_u||_2^2$$

- Gradient step observing r_{uv}
  - For L,R $\begin{bmatrix} L_u^{(t+1)} \\ R_v^{(t+1)} \end{bmatrix} \leftarrow \begin{bmatrix} (1-\eta_t\lambda_u)L_u^{(t)} - \eta_t\epsilon_t R_v^{(t)} \\ (1-\eta_t\lambda_v)R_v^{(t)} - \eta_t\epsilon_t L_u^{(t)} \end{bmatrix}$
  - For w and w_u:

What you need to know…

- Cold-start problem
- Feature-based methods for collaborative filtering
  - Help address cold-start problem
- Unified approach