Recommender Systems

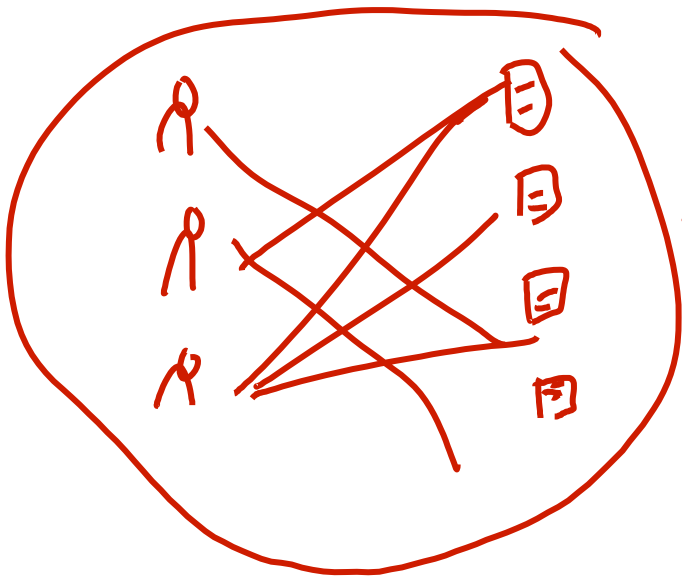
Machine Learning – CSEP546
Carlos Guestrin
University of Washington

February 10, 2014

Personalization is transforming our experience of the world

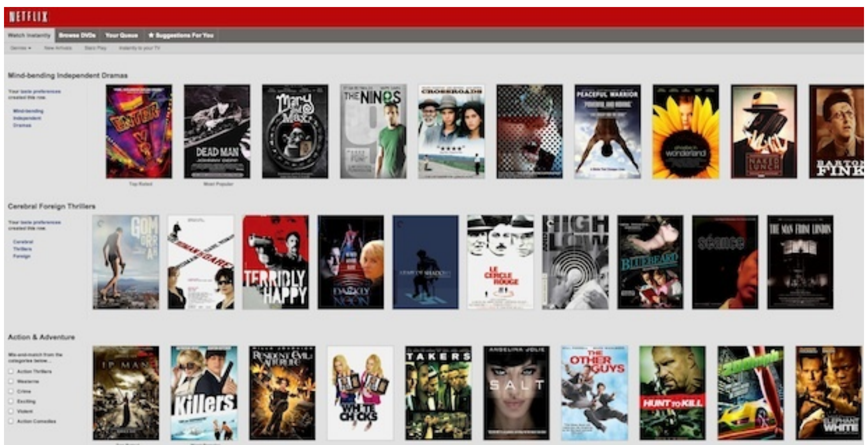
- Information overload ➔ "Browsing" is history
  - Need fundamentally new ways to discover content
- Personalization: Connects *users & items*

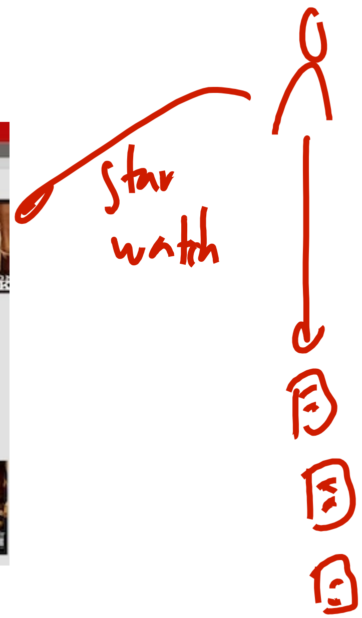
100 Hours a Minute
What do I care about?



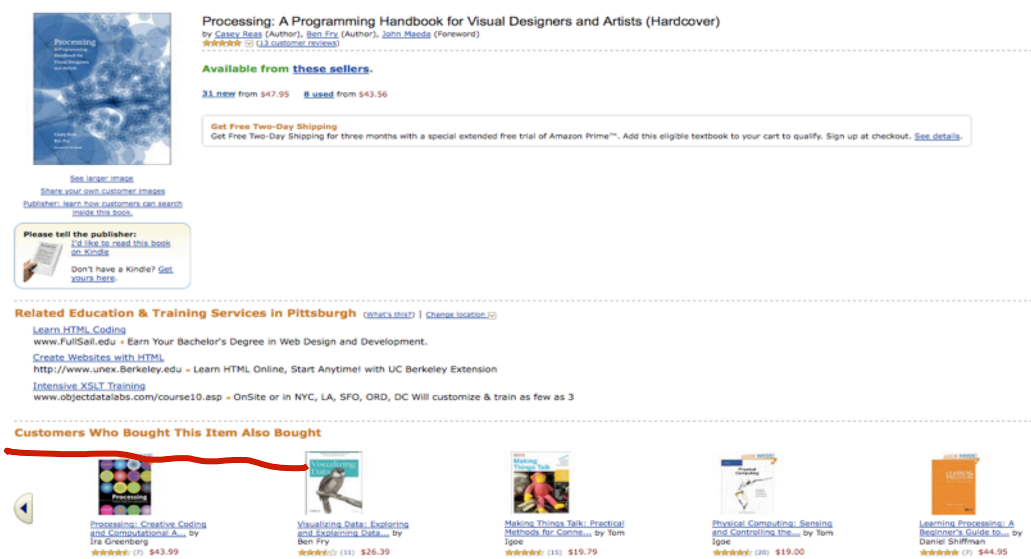
recommendations

Movie recommendations





Product recommendations

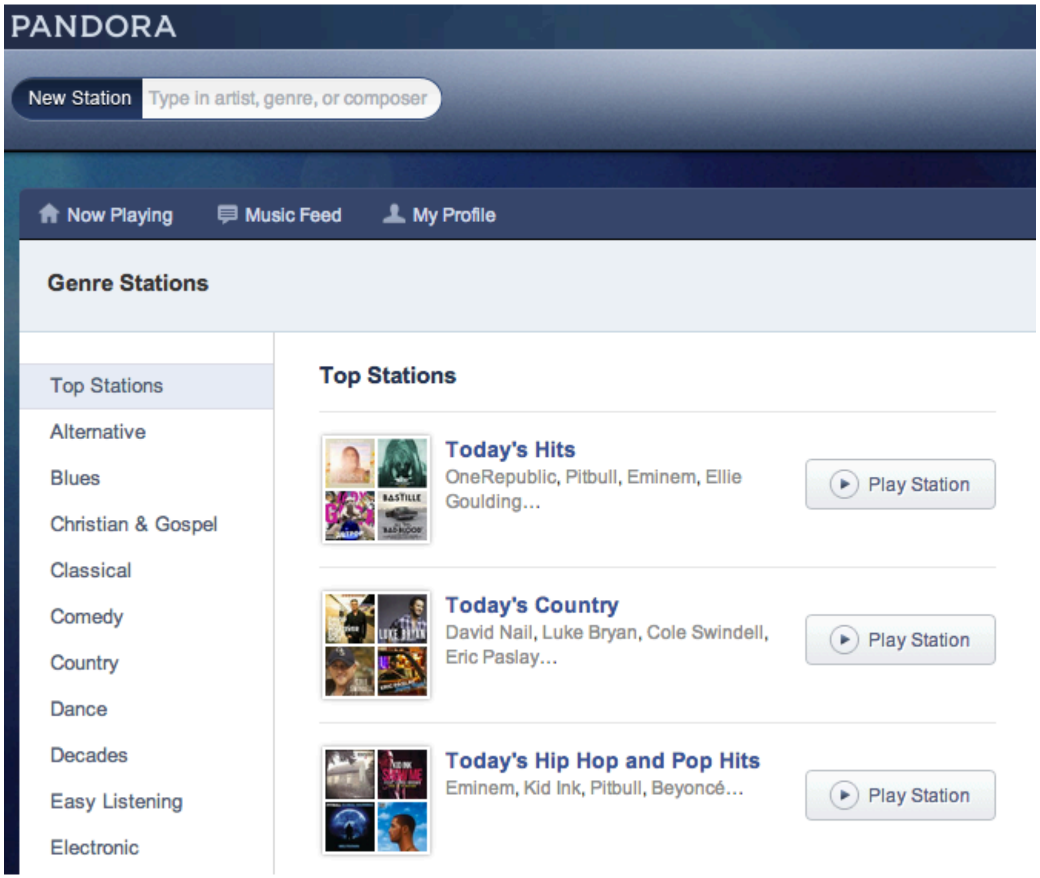


Recommendations combine global & session interests

what I bought in the past

intention

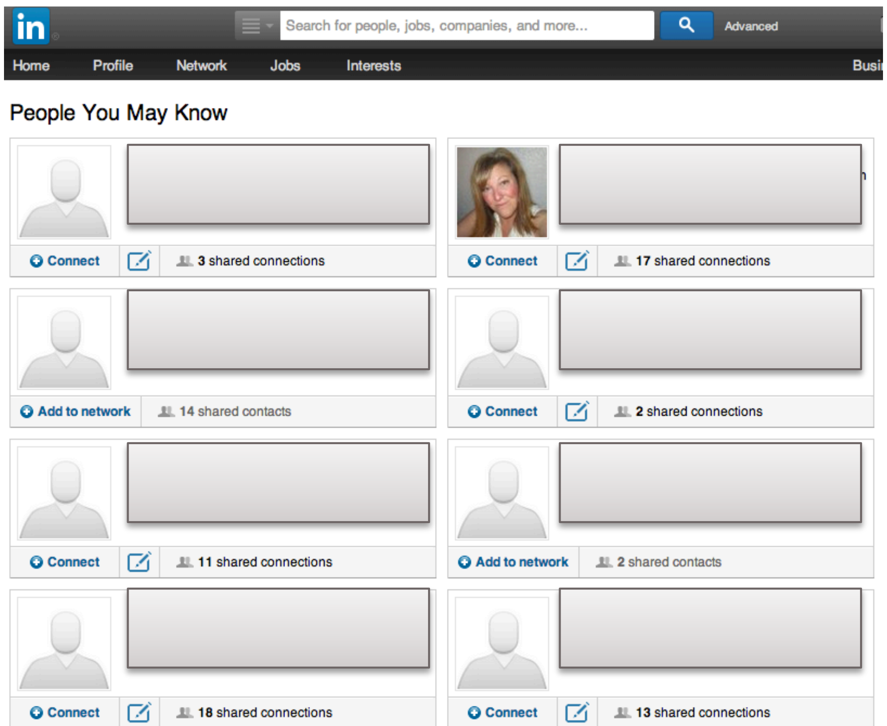
Playlist recommendations



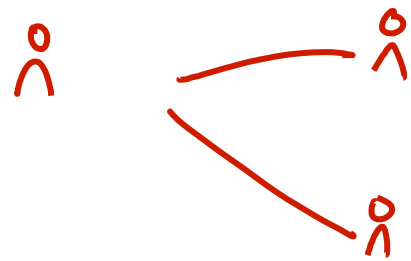
Recommendations form coherent & diverse sequence

Diverse

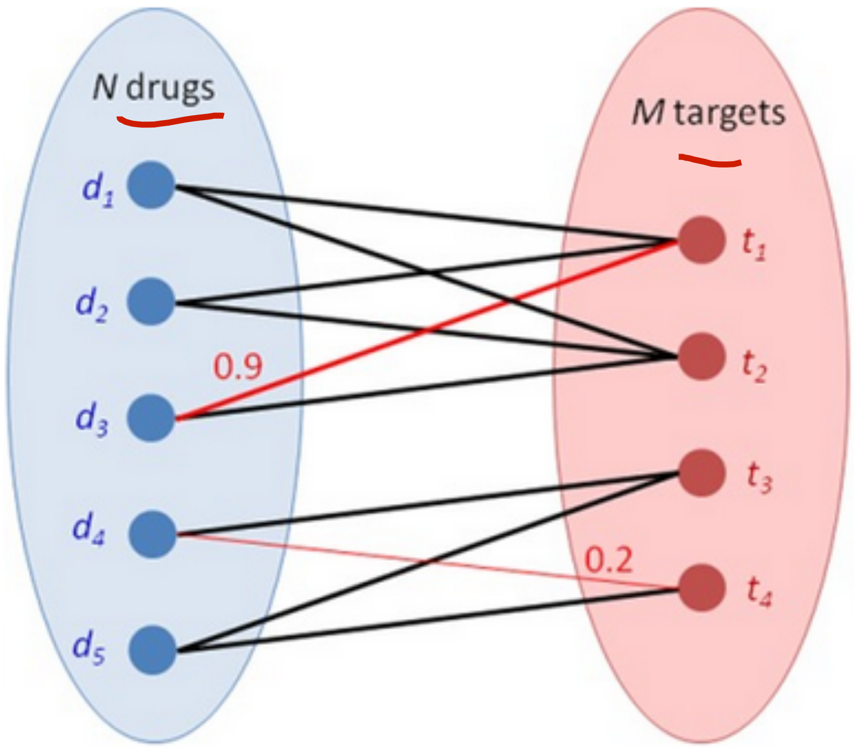
Friend recommendations



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

clicked

time on site

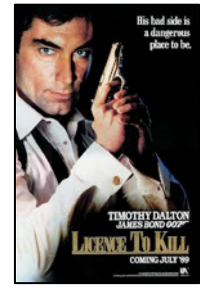
⋮

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!

action, actors, sequel, …

  - 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 intention now
- 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 (easily with GraphLab)

Solution 0: Popularity

Simplest approach: popularity

- What people are viewing now
  - Super popular

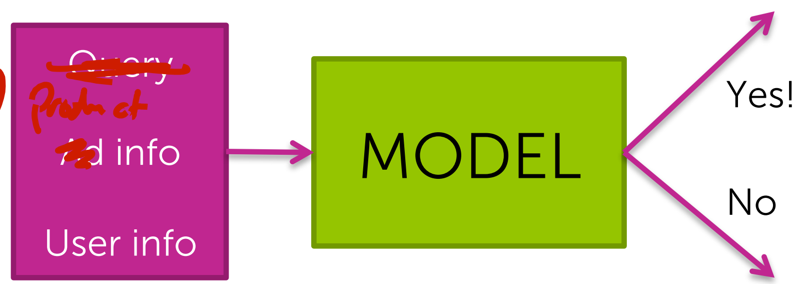
- Limitations:
  - No context (what's my intention now)
  - No personalization

Solution 1: Click prediction

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

Product: (price, model, ..)

R: (age, location...)



prob. of purchase

- 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

x → Beer, diapers ...

item
diapers

Beer +1

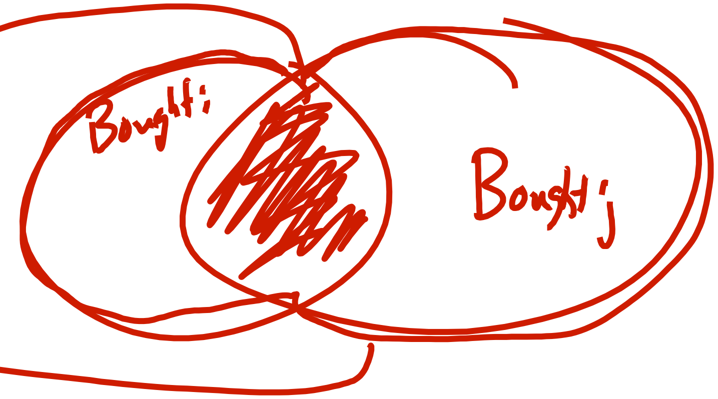
C = item

- $C_{ij} = C_{ji}$ 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*

$$S_{ij} = \frac{\text{bought } i \text{ and } j}{\text{bought } i \text{ or } j}$$



- 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

  Bud Light Org Vanilla Milk Gerber Pear & Potato mix

  - Beer, milk, baby food,…

- Limitation:
  - Only current page matters, no history

Solution 3: Average item-item similarity

(Weighted) average over items user bought

{Diapers, milk}

- User *u* bought items B_u
  - Define user specific similarity (score) for each item *j*
    - Average similarity for items in B_u

$$Score(Ru, Beer) = \frac{1}{2}\left(S_{Beer,Milk} + S_{Beer,Diapers}\right)$$

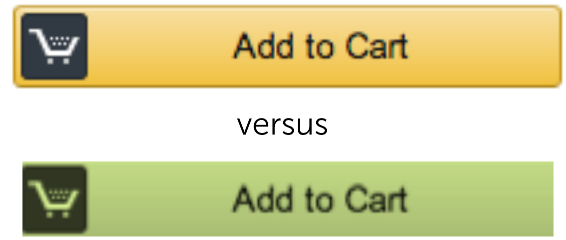
    - Could also weight recent purchases more

- For B_u=*{diapers,beer}* sort *Score(u,j)* and find *j* with highest similarity

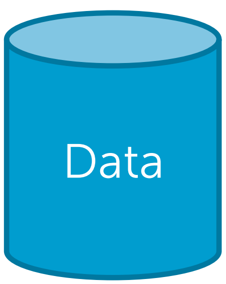
Demo: Item-based collaborative filtering with GraphLab

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



Compare on test data

Test

Train

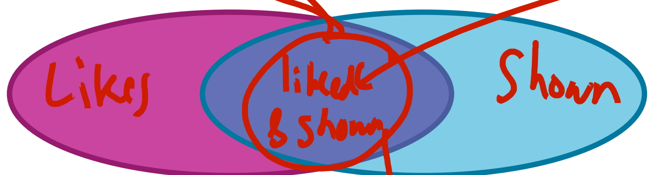
Popularity

item-item sim

Example Performance metric for recommenders

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

want to maximize



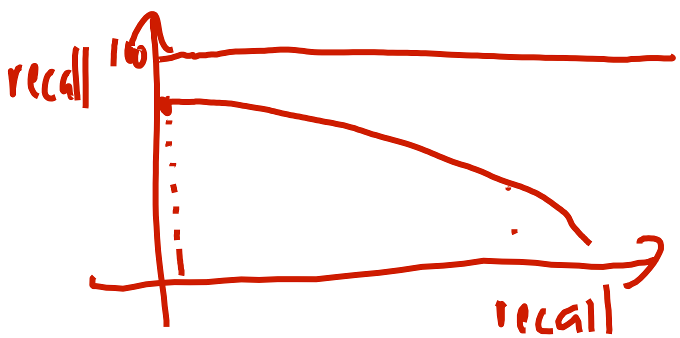
- *Recall:* what fraction of the liked movies we found

$$\frac{\text{liked \& shown}}{\text{liked}}$$

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

$$\frac{\text{liked \& shown}}{\text{shown}}$$

- Precision-Recall curve:



Demo for reals: Item-based collaborative filtering with GraphLab

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,…

$$R = (\underset{0.9}{\text{action}},\ \underset{0.2}{\text{romance}},\ \underset{0.5}{\text{drama}})$$

- Describe user *u* with features L_u
  - How much she likes action, romance, drama,…

$$L = (0.01,\ 0,\ 0.9)$$

- *Score(u,v)* is the product of the two vectors

$$\text{Score}(\ ,\ ) = 0.9 \times 0.01 + 0.2 \times 0 + 0.5 \times 0.9$$

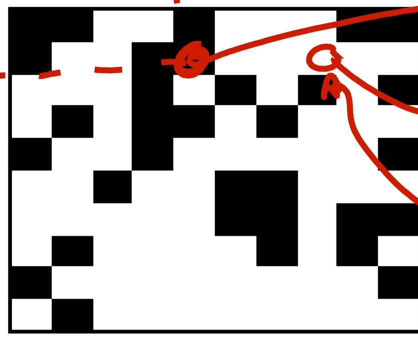
But we don't know features of users and movies…

Matrix Completion Problem

movies
PoC
Tit.

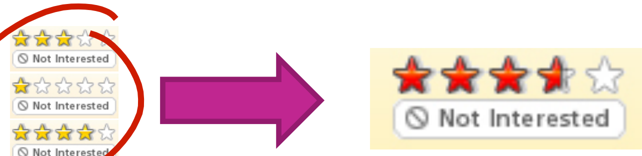
Score = A

users



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

- Users score some movies

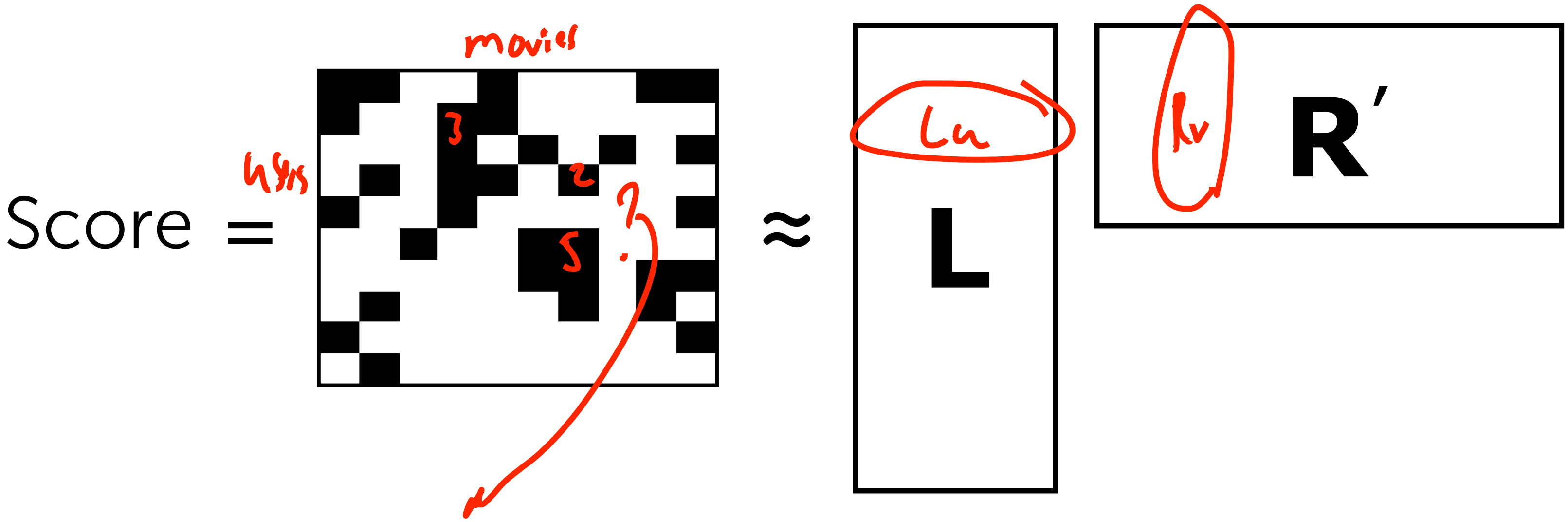


- Filling missing data?

If I had user & movie features

Score(A, Titanic) = A × $\boxed{\text{Titanic}}$

Matrix Factorization: discovering topics for users and movies

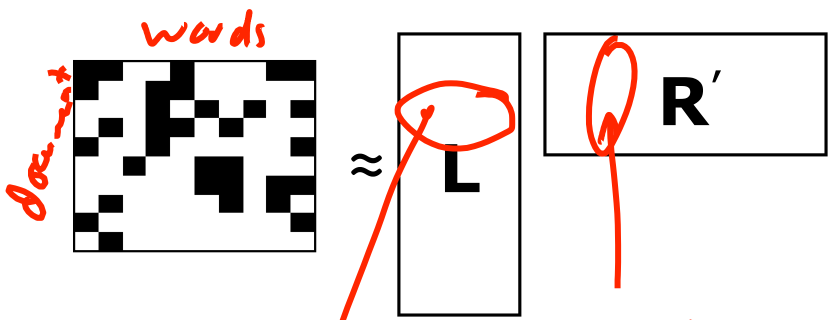


$$\text{Score}(17, \text{PoC}) = L_{17} \cdot R_{\text{PoC}}$$

Many efficient algorithms for
matrix factorization implemented in GraphLab

Example topics discovered from Wikipedia

Application to text data:



K topics
each topic
is associated
with words

Lu:
for each doc
how much
is it about
each of these discovered topics

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

$$Score(u,v) = L_u \cdot R_v$$

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

? Tianyi

$Score(Tianyi, v) \quad \forall v$

sort

Limitations of matrix factorization

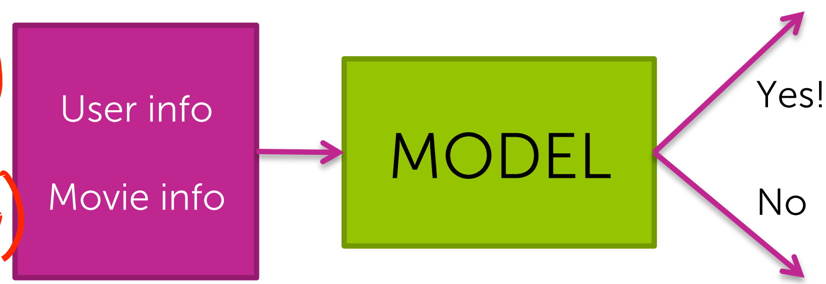
- Cold-start problem

Bringing it all together: Featurized matrix factorization

Combining real and discovered features

h(?, 1, 2, ...)
h(age, gender, location, L_u)

h(genres, actors, years, R_v)



- 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

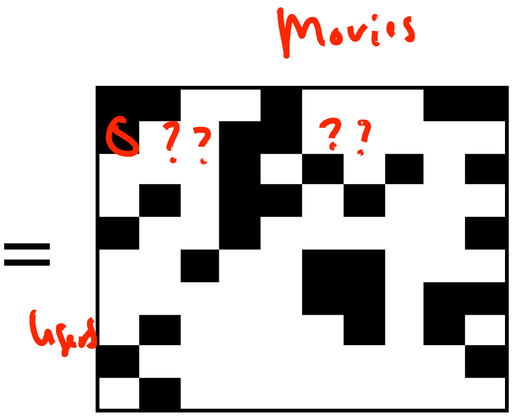
- Mitigates cold-start problem
  - Ratings for a new user from real features only
  - As more information about user is discovered, matrix factorization “features” become more relevant

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?

if $k = 100$

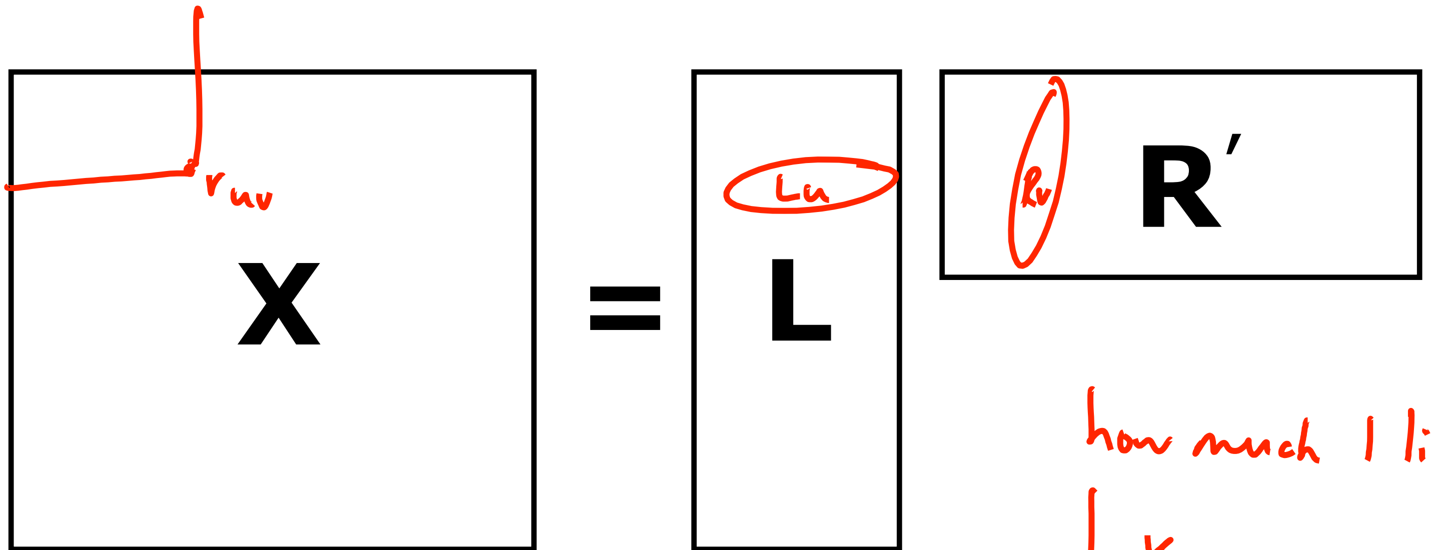
want to estimate $n \cdot m$ numbers

$m = 17k$ $n = 500k$ $n \cdot m \approx 8B$

$n \cdot k + m \cdot k$

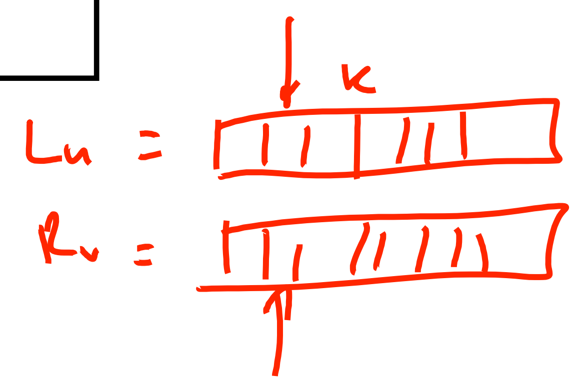
$50M + 1.7M \approx 51.7M$ params

Interpreting Low-Rank Matrix Completion (aka Matrix Factorization)



$r_{uv} \approx L_u \cdot R_v$

how much I like "sports"



how much movie is about "sports"

Matrix Completion via Rank Minimization

- Given observed values: $(u, v, r_{uv}) \in X$ $\quad r_{uv} \neq ?$
- Find matrix $\underset{n}{\overset{m}{\Theta}}$
- Such that: $\Theta_{uv} = r_{uv} \quad \forall\, r_{uv} \neq ?$
- But... predictions for $r_{uv} = ?$
- Introduce bias: $\text{rank}(\Theta) = k$
- Two issues:
  - Problem is NP-hard
  - you c

Approximate Matrix Completion

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

$$\min_{\Theta} \sum_{u,v:\, r_{uv} \neq ?} (\Theta_{uv} - r_{uv})^2$$

- Choose rank *k:*

$$ {}_{n}\overset{m}{\Theta} = {}_{n}\overset{k}{L}\;\left({}_{m}\overset{k}{R}\right)^{T} $$

- Optimization problem: $\min\limits_{R,L} \sum\limits_{u,v:\, r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2$

Non-Convex problem $\rightarrow$ local opt only.

Coordinate Descent for Matrix Factorization

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

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

$$\min_{L_1,\dots,L_n} \sum_{u,v:\, r_{uv}\neq ?} (L_u \cdot R_v - r_{uv})^2$$

$V_u \equiv$ set of movies user u rated

$$\equiv \min_{L_1,\dots,L_n} \sum_{u=1}^{n} \sum_{v\in V_u} (L_u \cdot R_v - r_{uv})^2$$

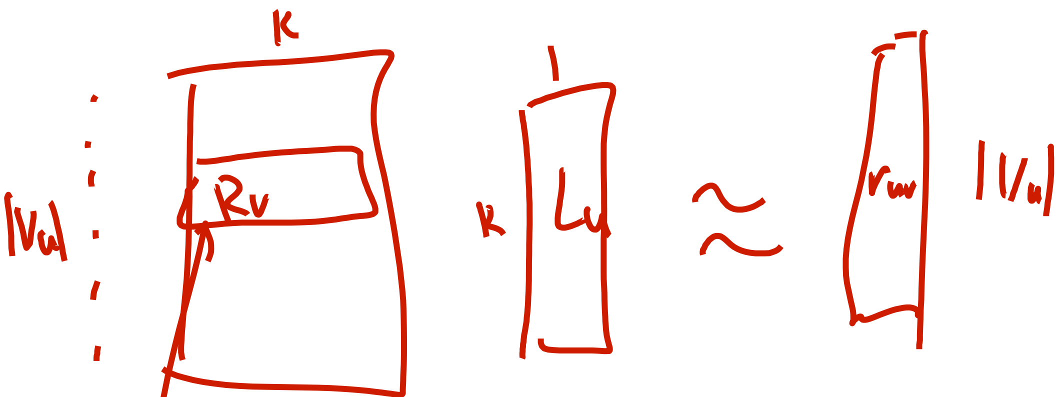
$\leftarrow$ indep opt probs per user

$$\equiv \sum_{u=1}^{n} \underbrace{\min_{L_u} \sum_{v\in V_u} (L_u \cdot R_v - r_{uv})^2}_{\text{Local regression, only depends on movies user } u \text{ rated}}$$

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:



become basis functions/features in regression

- Second observation: Solve by

matrix inversion

Coordinate Descent for Matrix Factorization: Alternating Least-Squares

prediction ↙ observed rating

$$\min_{L,R} \sum_{(u,v):r_{uv}\neq ?} (\underbrace{L_u \cdot R_v}_{\text{prediction}} - r_{uv})^2 + \lambda_u \|L\| + \lambda_v \|R\|$$

Don't initialize R_v to 0 !! usually init to random

- Fix movie factors, optimize for user factors
  - Independent least-squares over users $L_u^{(t+1)}$: $\arg\min_{L_u} \sum_{v \in V_u} (L_u \cdot R_v^{(t)} - r_{uv})^2 + \lambda_u \|L_u\|$
- Fix user factors, optimize for movie factors
  - Independent least-squares over movies $R_v^{(t+1)}$: $\arg\min_{R_v} \sum_{u \in U_v} (L_u^{(t+1)} \cdot R_v - r_{uv})^2 + \lambda_v \|R_v\|$

$U_v \in$ users who rated movie v

- System may be underdetermined:

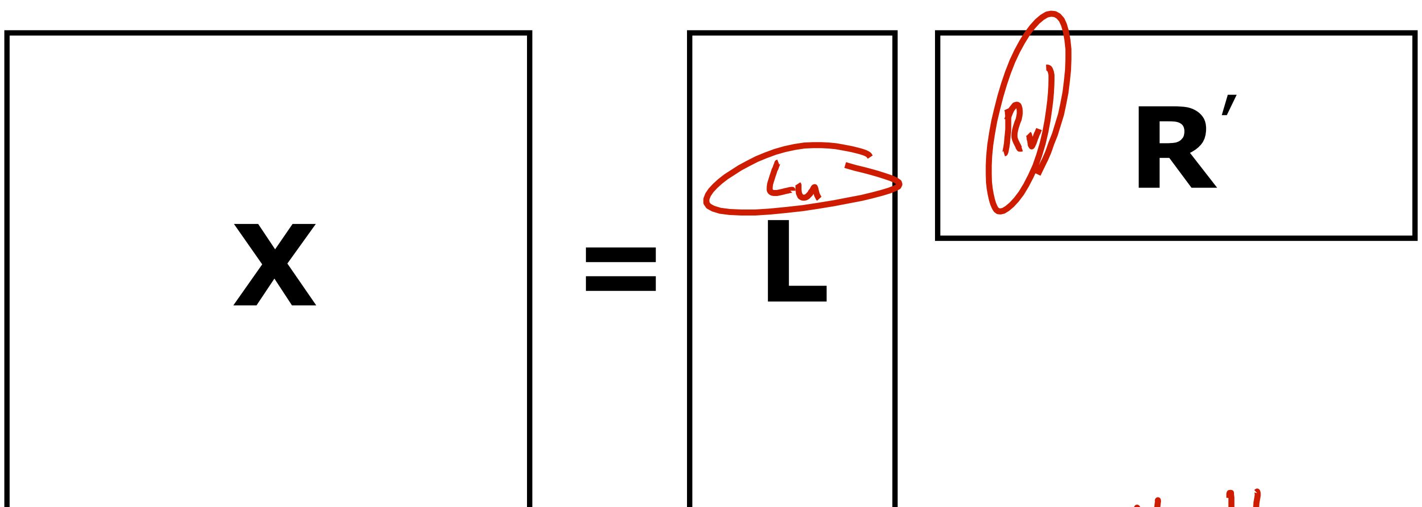
  use regularization

- Converges to local opt

Effect of Regularization

Frobinews $\|L\|_F = \sqrt{\sum_{uv} L_{uv}^2}$

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



if $\|\cdot\| = \|\cdot\|_F$

each subproblem uses L_2 regularization

→ ridge regression

$\|\cdot\| = \|\cdot\|_1$

→ each subproblem LASSO

↑ sparse latent factors

Stochastic Gradient Descent

$\|L\|_F^2$

$F =$

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

- Observe one rating at a time r_{uv}

$\epsilon_t = L_u^{(t)} \cdot R_v^{(t)} - r_{uv}$

- Gradient:

$$\frac{\partial F}{\partial L_u} = \epsilon_t R_v + \lambda_u L_u$$

$$\frac{\partial F}{\partial R_v} = \epsilon_t L_u + \lambda_v R_v$$

SGD observe rating r_{uv}

$$\begin{bmatrix} L^{(t+1)} \\ R^{(t+1)} \end{bmatrix} \leftarrow \begin{bmatrix} L^{(t)} \\ R^{(t)} \end{bmatrix} - \eta_t \nabla F_t$$

- Updates:

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

very simple to implement

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:

words

movies

$L_u \cdot R_v \approx r_{uv}$

document | users

$$X = L \; R'$$

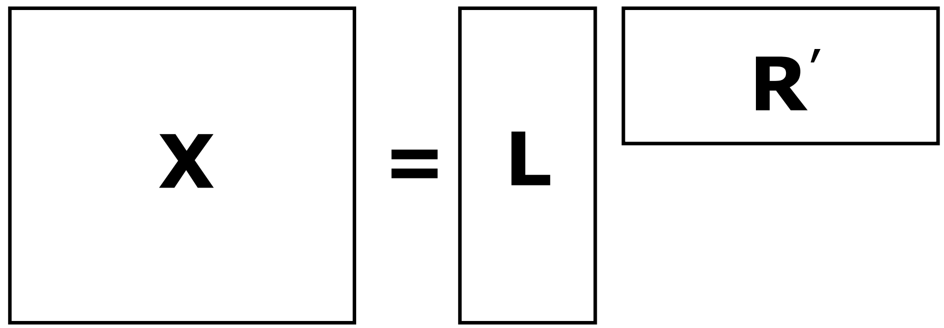
L_u $\quad$ R_v

- Would like: $L_u \geq 0$

$R_v \geq 0$ $\quad$ possible

- 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:

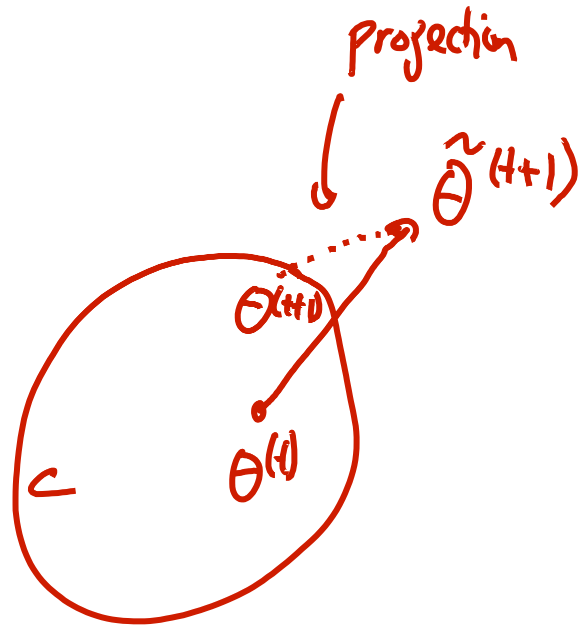
Projected Gradient

- Standard optimization:
  - Want to minimize: $\min_{\Theta} f(\Theta)$
  - Use e.g. 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):

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

  - Projection into feasible set:

$$\Pi_C(\theta) = \arg\min_{\beta \in C} \|\theta - \beta\|_2^2$$

often easy to compute (always convex)

$$\Theta^{(t+1)} = \Pi_C(\tilde{\Theta}^{(t+1)})$$

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: $L_u \geq 0$; $R_v \geq 0$ $\forall u,v$
- Projection step:

$\Pi_C(\theta) = \arg\min_{\beta \in C} \|\theta - \beta\|_2^2$ ← totally indep prob per dimension

Single dim

$= \arg\min_{\beta \geq 0} (\theta - \beta)^2$

$= \begin{cases} \theta, & \text{if } \theta \geq 0 \\ 0, & \text{if } \theta < 0 \end{cases} = (\theta)_+$

$$\begin{bmatrix} L^{(t+1)} \\ R^{(t+1)} \end{bmatrix} \leftarrow \begin{bmatrix} \tilde{L}^{(t+1)} \\ \tilde{R}^{(t+1)} \end{bmatrix}_+$$

set all neg coords to zero, universe on our side!! :)

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

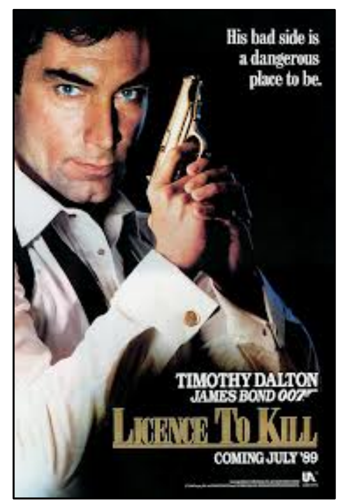
Machine Learning – CSEP546
Carlos Guestrin
University of Washington

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: u' no ratings : $\forall v \; r_{u'v} = ?$

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

doesn't depend on $L_{u'}$

$\Rightarrow L_{u'} = 0$, always predict $\forall v \; r_{u'v} = 0$ (constant)

- A simpler model for collaborative filtering:
  - Observe ratings: $r_{uv} \in X$
  - Given features of a movie:

$\phi(v) = ($ action (genre), 1994 (year), Tarantino (director), ... $)$

  - Fit linear model:

For all users u, $r_{uv} \sim N(w \cdot \phi(v), \sigma_r^2)$

  - Minimize: for model

$$\min_w \sum_{r_{uv}} (w \cdot \phi(v) - r_{uv})^2 + \lambda_w ||w|| \leftarrow$$ least squares, Lasso ...

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:

$$\phi(u) = (\underset{\text{age}}{25}, \underset{\text{gender}}{F}, \underset{\text{education}}{MSc}, \underset{\text{..grade in Big Data class}}{A+} \ldots)$$

- Combine with features of movie:

$$\phi(u,v) = (\ldots \phi(u) \ldots, \ldots \phi(v) \ldots, \ldots \text{cross features} \ldots)$$

- Unified linear model:

$$r_{uv} \sim N\left((w + w_u) \cdot \phi(u,v), \sigma_r^2\right)$$

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

hierarchical model

$\|w_u - w\|_2^2$

$$F \rightarrow \min_{L,R,w,\{w_u\}_u} \frac{1}{2}\sum_{r_{uv}}(L_u \cdot R_v + (w + w_u)\cdot \phi(u,v) - r_{uv})^2$$

$$+ \frac{\lambda_u}{2}||L||_F^2 + \frac{\lambda_v}{2}||R||_F^2 + \frac{\lambda_w}{2}||w||_2^2 + \frac{\lambda_{wu}}{2}\sum_u ||w_u||_2^2$$

- Gradient step observing $r_{uv}^{(t)}$ $\quad \epsilon_t = L_u^{(t)} \cdot R_v^{(t)} + (w_u^{(t)} + w^{(t)}) \cdot \phi(u,v) - r_{uv}^{(t)}$
  - 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: $\nabla_w F^{(t)} = \epsilon_t \phi(u,v) + \lambda_w w^{(t)}$

$$\begin{bmatrix} w^{(t)} \\ w_u^{(t)} \end{bmatrix} \leftarrow \begin{bmatrix} (1-\eta_t\lambda_w)w^{(t)} - \eta_t\epsilon_t\phi(u,v) \\ (1-\eta_t\lambda_{w_u})w_u^{(t)} - \eta_t\epsilon_t\phi(u,v) \end{bmatrix}$$

only update w_u for user u in $r_{uv}^{(t)}$

What you need to know…

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