

Introduction to Data Science

WFAiS UJ, Informatyka Stosowana
I stopień studiów

This lecture is
based on course by E. Fox and C. Guestrin, Univ of Washington

Principal component analysis

- Input data may have thousands or millions of dimensions!
 - e.g., text data
- **Dimensionality reduction**: represent data with fewer dimensions
 - easier learning – fewer parameters
 - visualization – hard to visualize more than 3D or 4D
 - discover “intrinsic dimensionality” of data
 - high dimensional data that is truly lower dimensional

Lowering dimensionality projection

- Rather than picking a subset of the features, we can create new features that are combinations of existing features

$$\begin{aligned} \text{\textit{i}^{th} obs} \rightarrow z_i[1] &= 2.5 x_i[1] + 3 x_i[2] + 7 x_i[3] + \dots \\ z_i[2] &= \dots \\ \vdots \end{aligned}$$

First (new) feat

(other weights on old features)

- Let's see this in the unsupervised setting
 - just x, but no y

Lowering dimensionality projection

Linear projection...

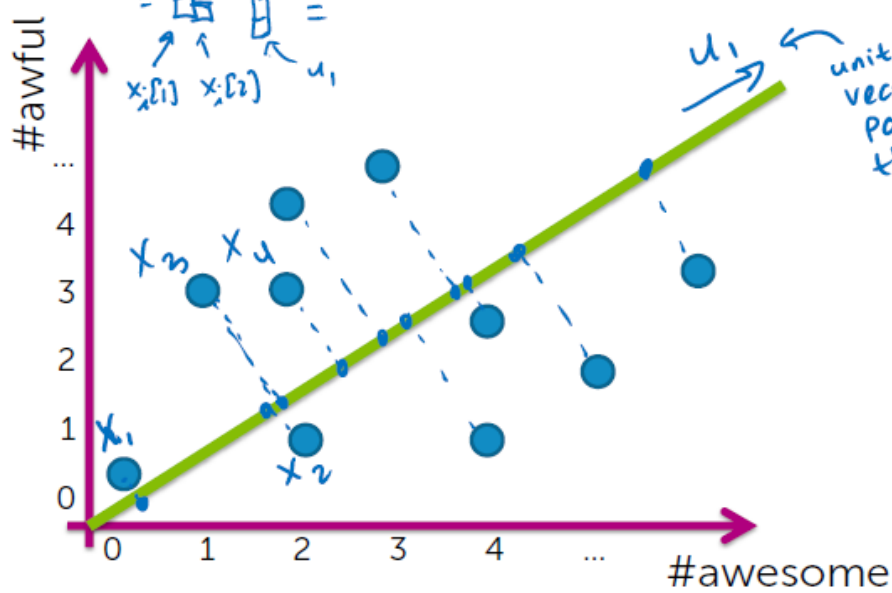
In eqn:

$$z_i = x_i \cdot u_1 = X_i^T u_1$$

$= \begin{bmatrix} x_1[1] & x_1[2] \end{bmatrix} \cdot \begin{bmatrix} u_1[1] \\ u_1[2] \end{bmatrix}$

$X^T u = \text{elementwise product + sum}$

Project onto 1-dimension



unit length vector pointing in this direction



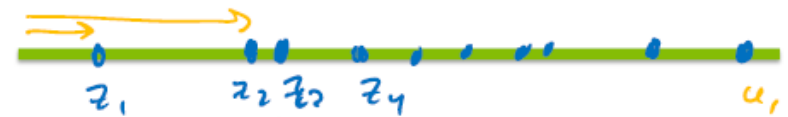
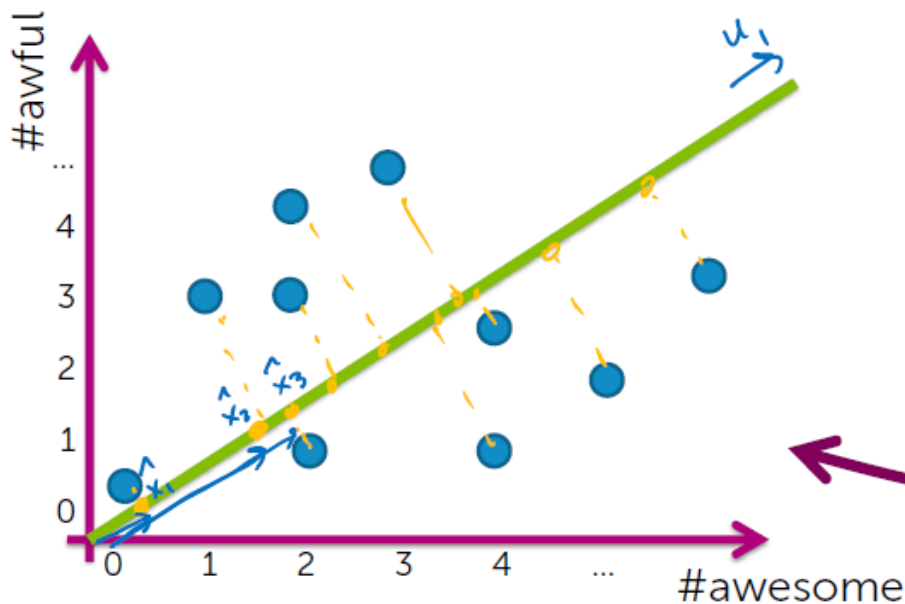
Lowering dimensionality projection

Linear projection and reconstruction

In eqn:

$$\hat{x}_i = z_i u_i$$

put back into $x[1], x[2]$
1 and



Reconstruction:
Only knowing z ,
what was $(x[1], x[2])$?

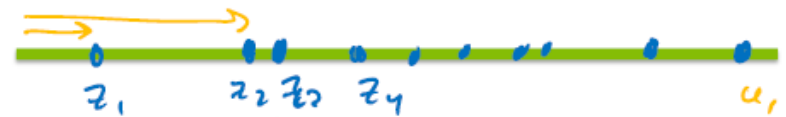
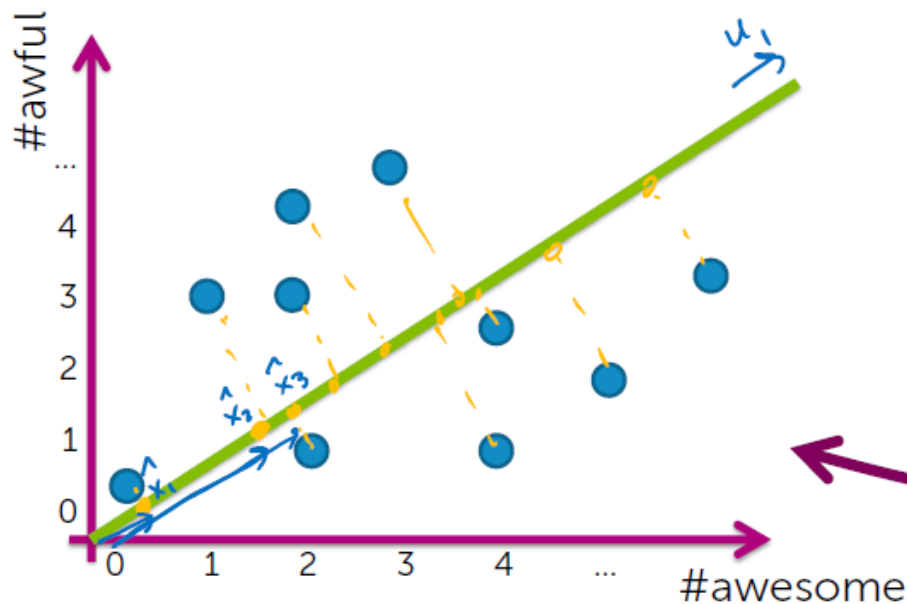
Lowering dimensionality projection

Linear projection and reconstruction

In eqn:

$$\hat{x}_i = z_i u_i$$

put back into $x[1], x[2]$
1 and



Reconstruction:
Only knowing z ,
what was $(x[1], x[2])$?

Lowering dimensionality projection

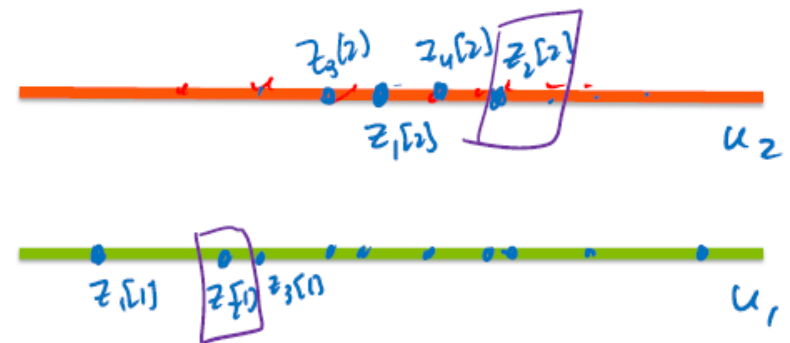
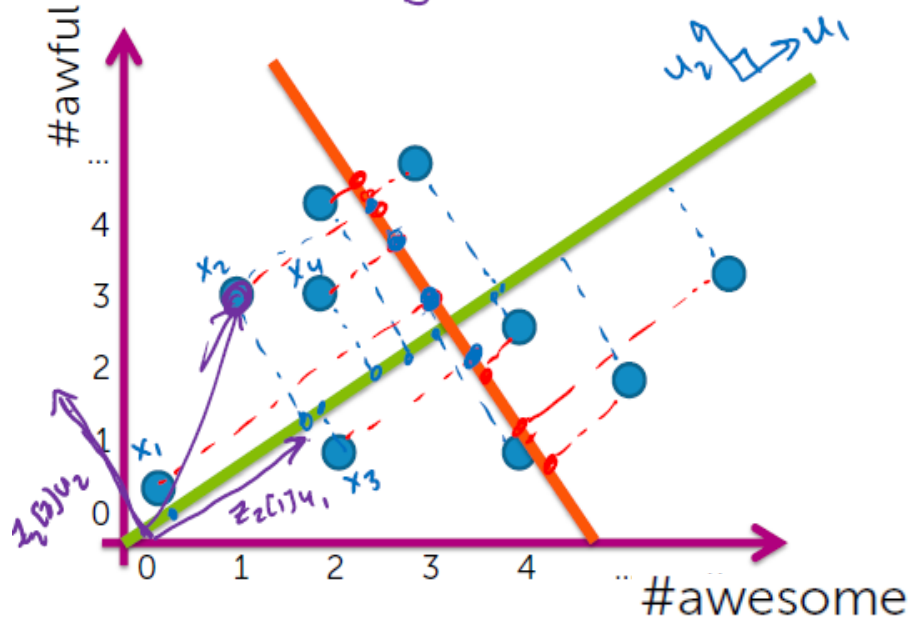
What if we project onto d orthogonal vectors?

In eqns:

$$\hat{x}_i[1:2] = z_i[1]u_1 + z_i[2]u_2$$

$$= \# \theta + \# \theta = \theta$$

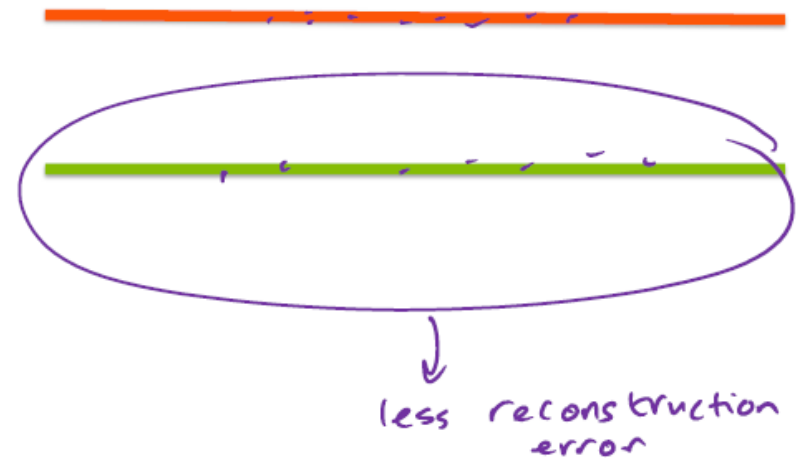
$z_i[1] \leftarrow$ projection onto u_1
 $z_i[2] \leftarrow$ projection onto u_2



Perfect
reconstruction!

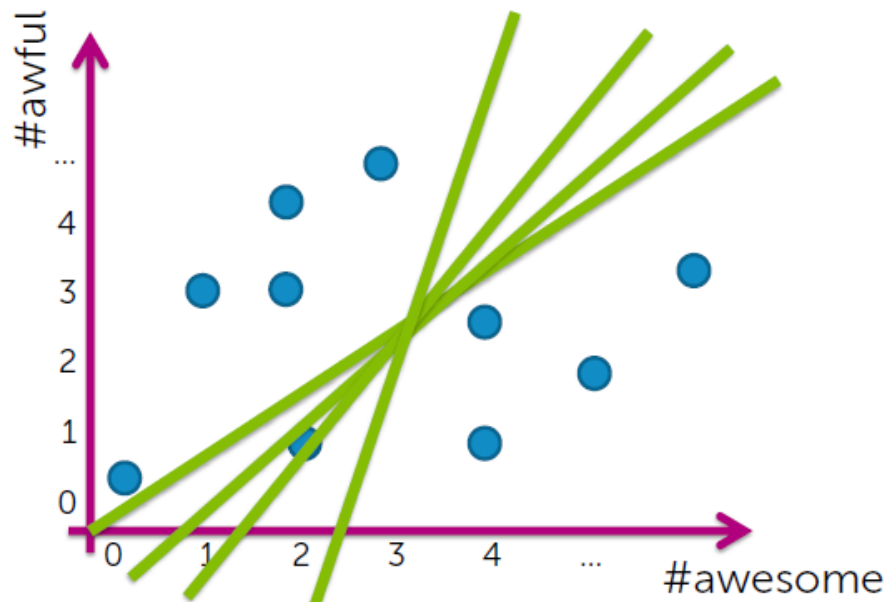
Lowering dimensionality projection

If I had to choose one of these vectors, which do I prefer?

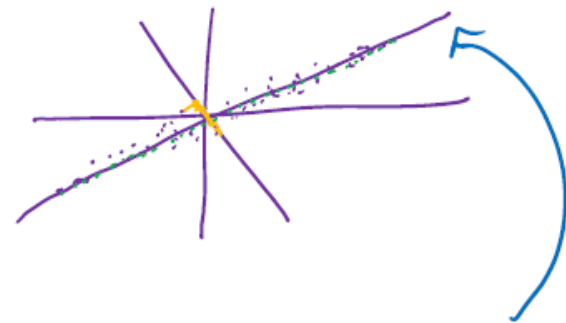


Lowering dimensionality projection

What about over all single vectors?



Consider extreme data example:



choose direction of
greatest variations

Principal component analysis (PCA)

Basic idea

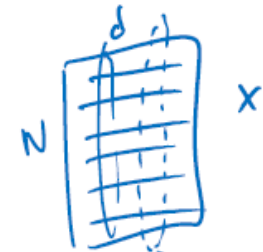
- Project d -dimensional data into k -dimensional space while preserving as much information as possible:
 - e.g., project space of 10000 words into 3-dimensions
 - e.g., project 3-d into 2-d
- Choose projection with minimum reconstruction error

Principal component analysis (PCA)

Basic PCA algorithm



- Form **data matrix** X
 - Each row is a different data point...like our typical data tables



- Recenter: **subtract the mean** from each row of $X \rightarrow X_c$
- Spread/orientation calculation: Compute the **covariance matrix** Σ :

$$\Sigma_{ts} = \frac{1}{N} \sum_{i=1}^N x_{c,i}[t]x_{c,i}[s]$$



- Find basis:
 - Compute **eigendecomposition** of Σ
 - Select $(u_1, \dots, u_k)$ to be **eigenvectors** with **largest eigenvalues**

discard all
others
 $u_{k+1}, \dots, u_d$

- Project data: **Project each data point** onto each vector
 $x_i[1:d] \rightarrow z_i[1:k] \quad k \leq d$

Principal component analysis (PCA)

Reconstruction

Using our principal components, reconstruct observation in original domain:

$$\hat{X}_i[1:d] = \bar{X}[1:d] + \sum_{j=1}^k z_i[j] u_j$$

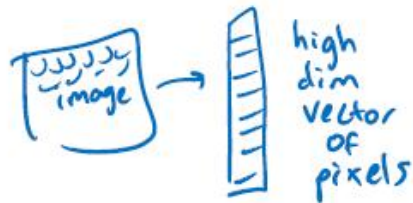
*recenter...
add back
subtracted
mean*

*amount
in this direction*

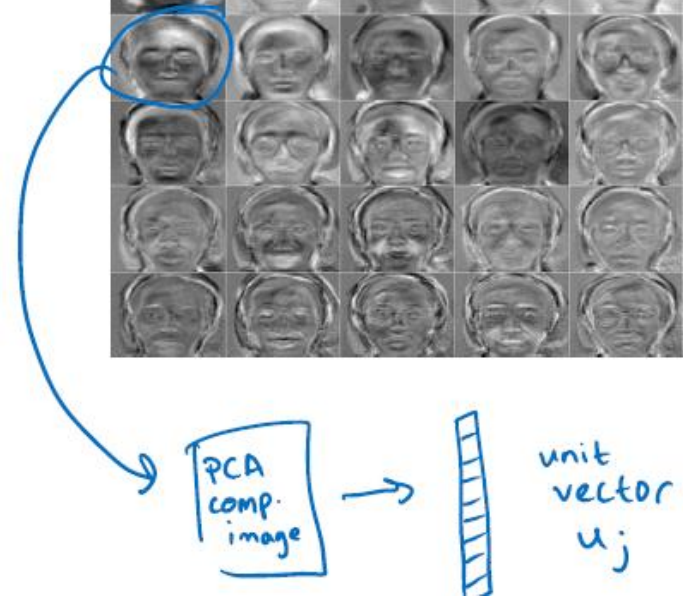
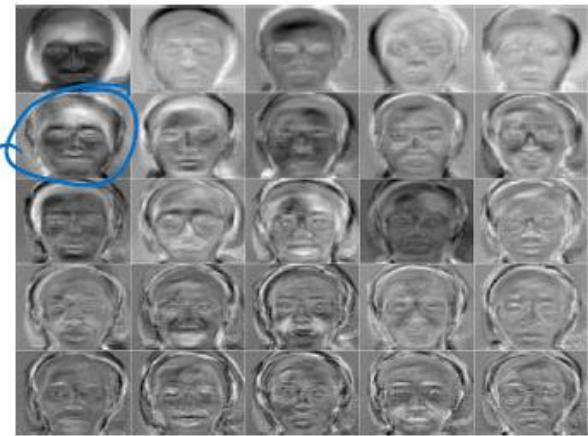
Principal component analysis (PCA)

Eigenfaces [Turk, Pentland '91]

Input images:



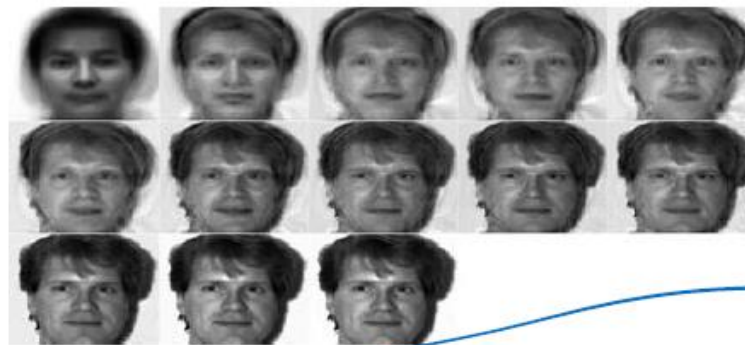
Principal components:



Principal component analysis (PCA)

Eigenfaces reconstruction

Each image corresponds to adding 8 principal components:



reconst. image $_i = \sum_{j=1}^k z_{i(j)} u_j + \bar{X}$

← PCA comp images

← image dataset mean image

proj. of x_i onto u_j

$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} = H$

Principal component analysis (PCA)

Scaling up

- Covariance matrix can be really big!
 - Σ is d by d
 - Say, only 10000 features
 - finding eigenvectors is very slow...
- Use singular value decomposition (SVD)
 - finds up to k eigenvectors
 - great implementations available

Recommender system: films

Machine learning:
recommender system

- Personalizacja

You Tube

100 Hours a Minute
What do I care about?

Information overload



Browsing is "history"
– Need new ways
to discover content

Personalization: Connects *users & items*

viewers

videos

Recomender system: films



Connect users with movies
they may want to watch

Recomender system: music



Recommendations form
coherent & diverse sequence

Recomender system: friends

Friend recommendations

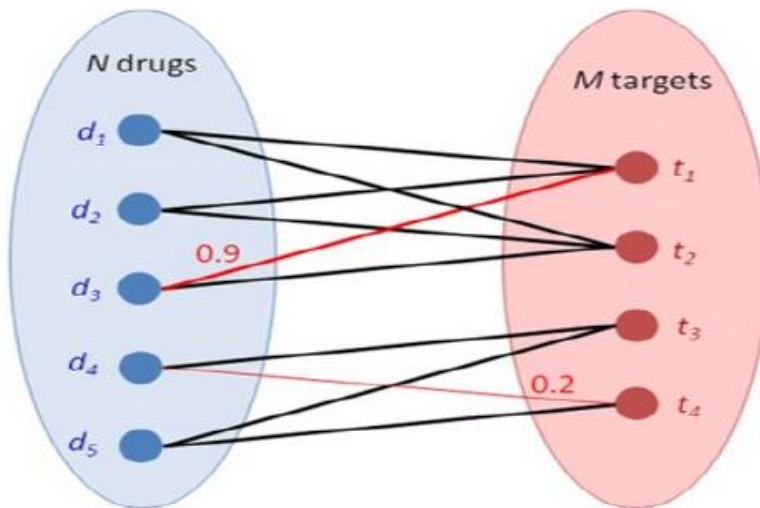


Users and "items"
are of the same "type"

Recomender system: medicine

Drug-target interactions

Cobanoglu et al. '13



What drug should we
"repurpose" for some disease?

Recomender system:

amazon.com[®] [Help](#) | [Close window](#)

Recommended for You

High Performance Web Sites: Essential Knowledge for Front-End Engineers
by Steve Souders (Author)
Our Price: \$19.79
Used & new from \$16.24

[Add to Cart](#) [Add to Wish List](#)

Because you purchased...

Programming Collective Intelligence: Building Smart Web 2.0 Applications (Paperback)
by Toby Segaran (Author)

Today's Recommendations For You

Here's a daily sample of items recommended for you. Click here to [see all recommendations](#)

Even Faster Web Sites: Performance... (Paperback) by Steve Souders
★★★★☆ (7) \$23.10
[Fix this recommendation](#)

Simply JavaScript (Paperback) by Kevin Yank
★★★★☆ (19) \$26.37
[Fix this recommendation](#)

The Art & Science of JavaScript (Paperback) by Peter DeWitt
★★★★☆ (5)
[Fix this recommendation](#)

[Any Category](#) Algorithms Boxed Sets Business & Culture Java
Networking Networks, Protocols & APIs New SQL

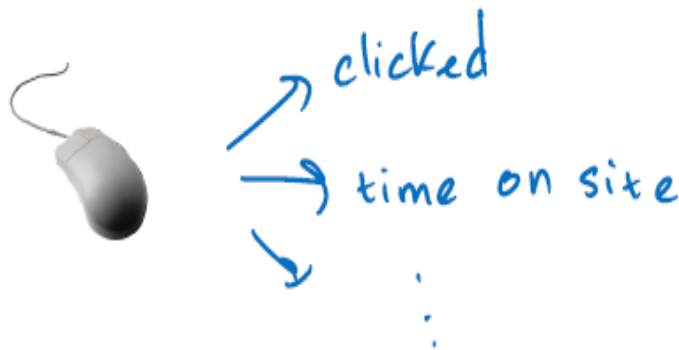
Recommendations combine
global & session interests

Challenges: Type of feedback

- Explicit – user tells us what she likes



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



Challenges: what is the goal ?

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-faceted & want to hedge our bets
 - Rocky 2, It's Always Sunny in Philadelphia, Gandhi

Challenges: Cold-start problem

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...

Challenges: evolving with time

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

Challenges: Scalability

For N users and M movies, some approaches take $O(N^3 + M^3)$

- Not so good for billions of users...

Big focus has been on:

- Efficient implementations
- Fast exact & approximate methods as needed

Building a recomender system

Solution 0: Popularity

Solution 1: Classification model

Solution 2: People who bought this
also bought...

Solution 3: Discovering hidden structure
by matrix factorization

Recommender system: popularity?

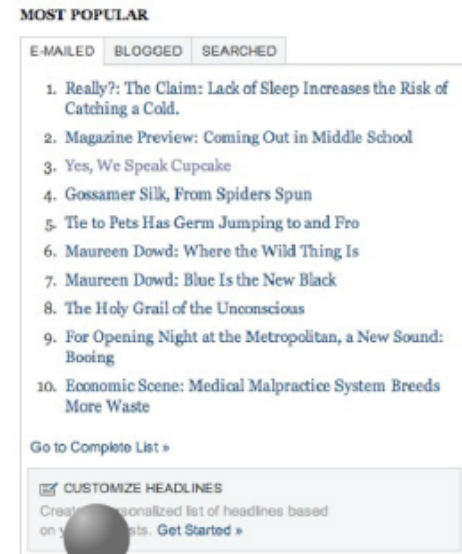
Simplest approach: Popularity

What are people viewing now?

- Rank by global popularity

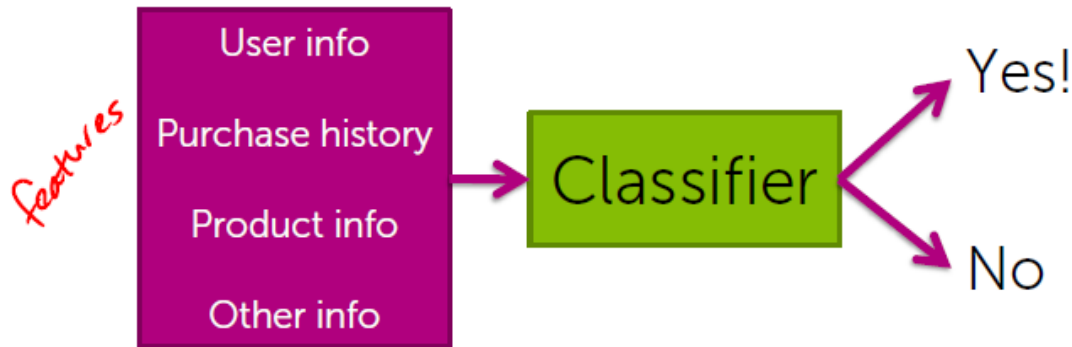
Limitation:

- No personalization



Recommender system: classification

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



Pros:

- ✱ - **Personalized:**
Considers user info & purchase history
- ✱ - **Features can capture context:**
Time of the day, what I just saw,...
- **Even handles limited user history:**
Age of user, ...

Cons:

- - Features may not be available
- Often doesn't perform as well as collaborative filtering methods (next)

Recommender system: co-occurrence

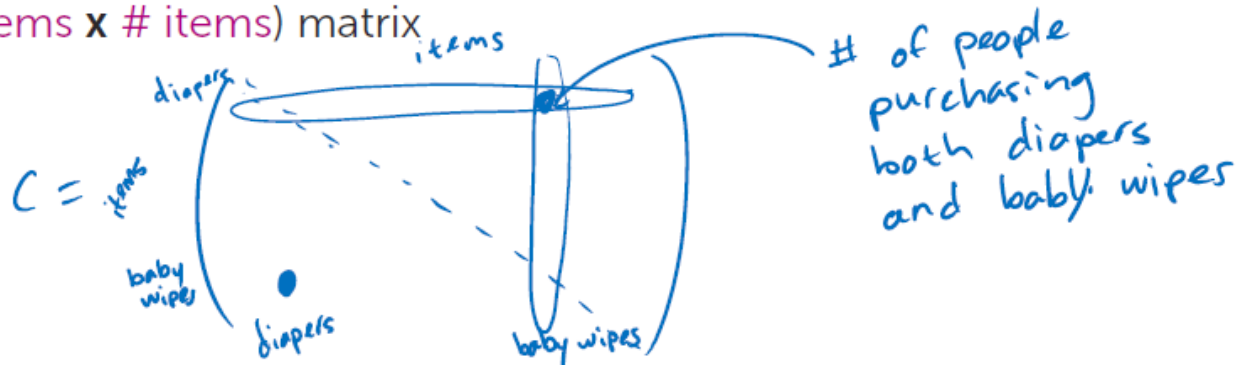
Co-occurrence matrix

- People who bought *diapers* also bought *baby wipes*

- **Matrix C:**

store # users who bought both items i & j

- (# items x # items) matrix



- **Symmetric:** # purchasing i & j same as # for j & i ($C_{ij} = C_{ji}$)

Recommender system: co-occurrence

Making recommendations using co-occurrences

User  purchased *diapers*

1. Look at *diapers* row of matrix

$$\begin{bmatrix} 0 & \dots & 4 & \dots & 100 & \dots \end{bmatrix}$$

DVD pacifiers baby wipes

2. Recommend other items with largest counts
- *baby wipes, milk, baby food,...*

Recommender system: correlations

Co-occurrence matrix must be normalized

What if there are very popular items?

- Popular baby item:

Pampers Swaddlers diapers



- For any baby item (e.g., *i=Sophie giraffe* )
large count C_{ij} for *j=Pampers Swaddlers*

Result: *Sophie* $\begin{bmatrix} 0 & \dots & 1 \text{ million} \\ \text{DVD} & & \text{diaper} & \dots & \text{baby wipes} \dots \end{bmatrix}$

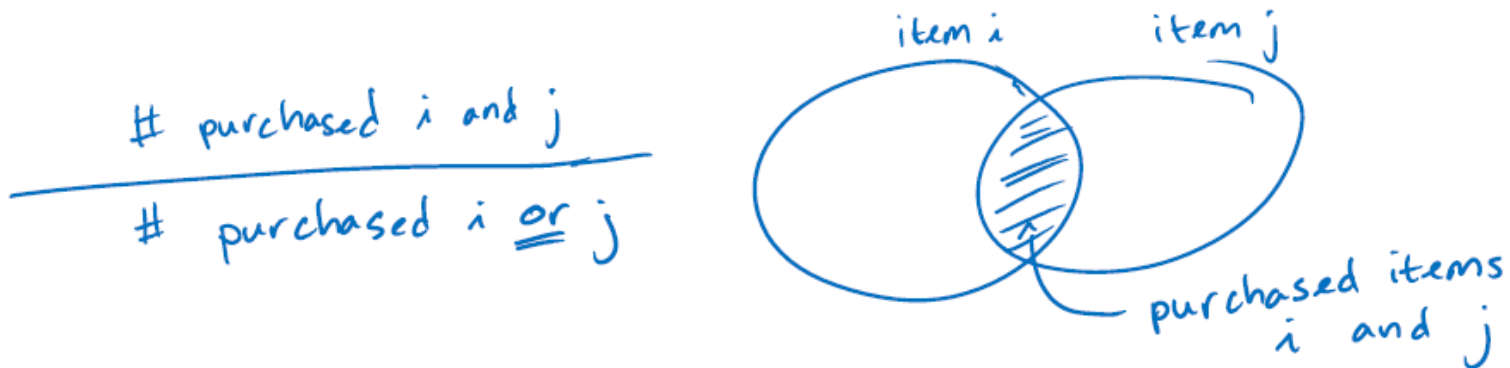
- Drowns out other effects
- Recommend based on popularity

Recommender system: co-occurrence

Normalize co-occurrences: Similarity matrix

Jaccard similarity: normalizes by popularity

- Who purchased *i* **and** *j* divided by who purchased *i* **or** *j*



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

Recommender system: co-occurrence

Limitations

- Only current page matters, **no history**
 - Recommend similar items to the one you bought
- What if you purchased many items?
 - Want recommendations based on purchase history

Recommender system: co-occurrence

(Weighted) Average of purchased items

User  bought items {*diapers*, *milk*}

- Compute user-specific score for each item j in inventory by combining similarities:

should we recommend this?

$$\text{Score}(\text{stick figure}, \text{baby wipes}) = \frac{1}{2} (S_{\text{baby wipes}, \text{diapers}} + S_{\text{baby wipes}, \text{milk}})$$

- Could also weight recent purchases more

Sort $\text{Score}(\text{stick figure}, j)$ and find item j with highest similarity

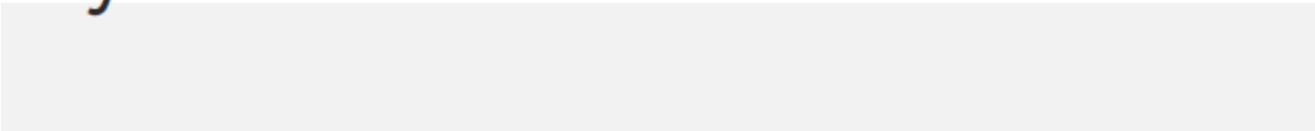
Recommender system: co-occurrence

Limitations

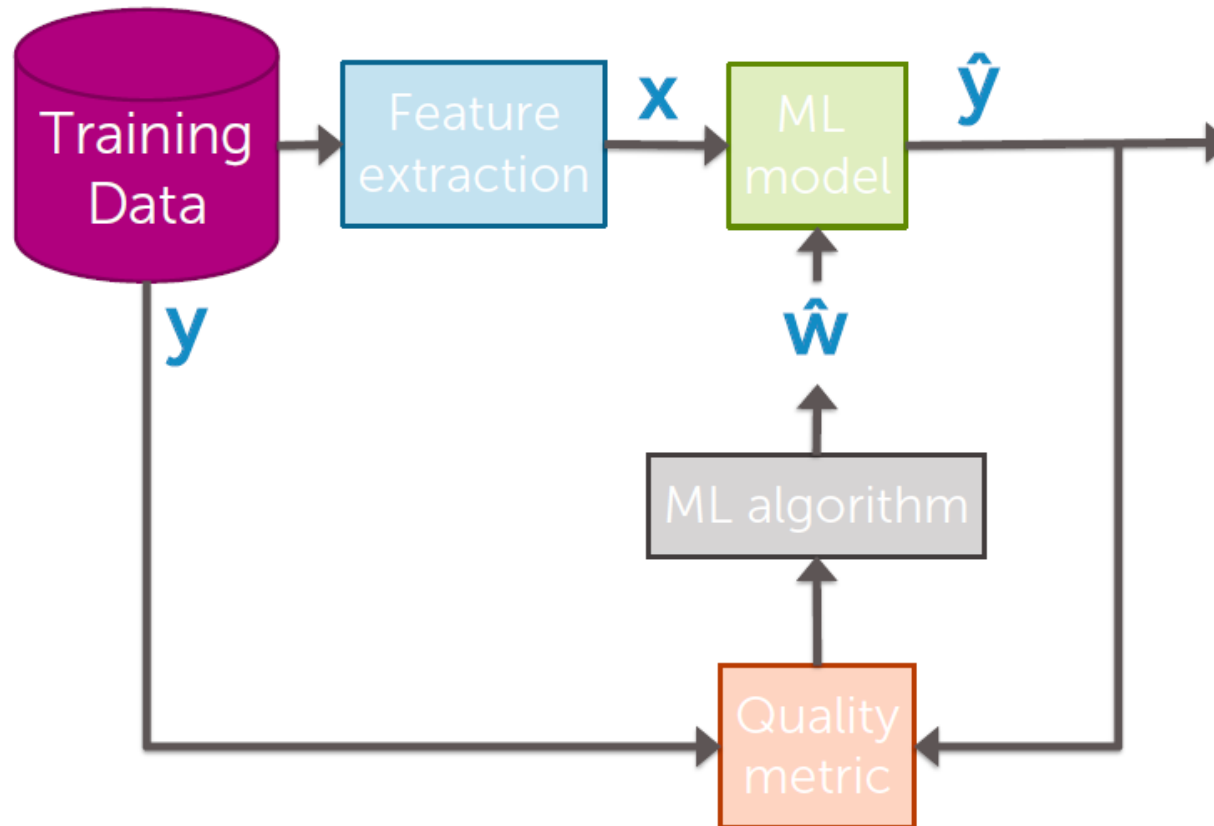
- Does **not** utilize:
 - context (e.g., time of day)
 - user features (e.g., age)
 - product features (e.g., baby vs. electronics)
- Scalability – similarity matrix M^2 size
- Cold start problem
 - What if a new user or product arrives?

Recommender system: matrix factorization

Discovering hidden structure
by matrix factorization



Flow chart



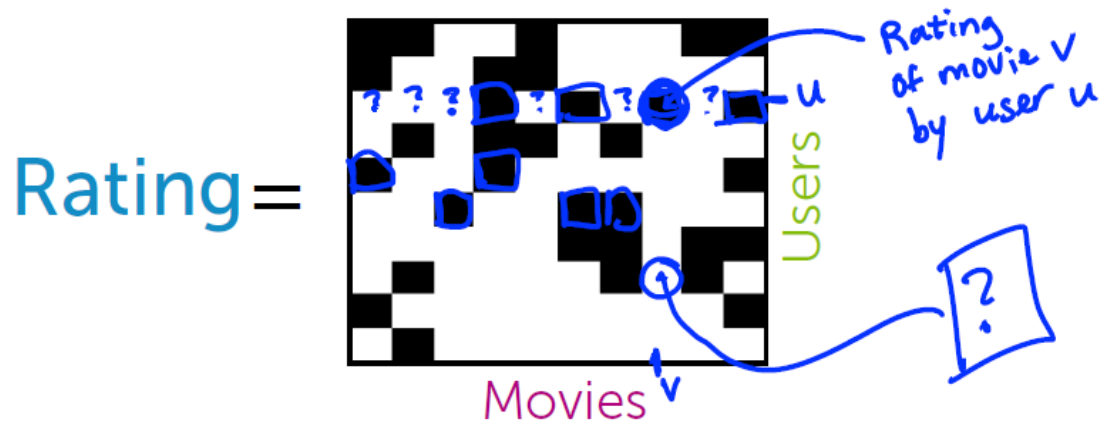
Training Data

- Users watch movies and rate them

User	Movie	Rating
		★★★★☆
		★★★★★
		★★★☆☆
		★★★☆☆
		★★★★☆
		★★☆☆☆
		★★★☆☆
		★★★★★
		★★★★☆

Each user only watches a few of the available movies

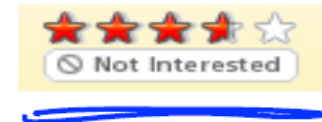
Training Data: matrix completion



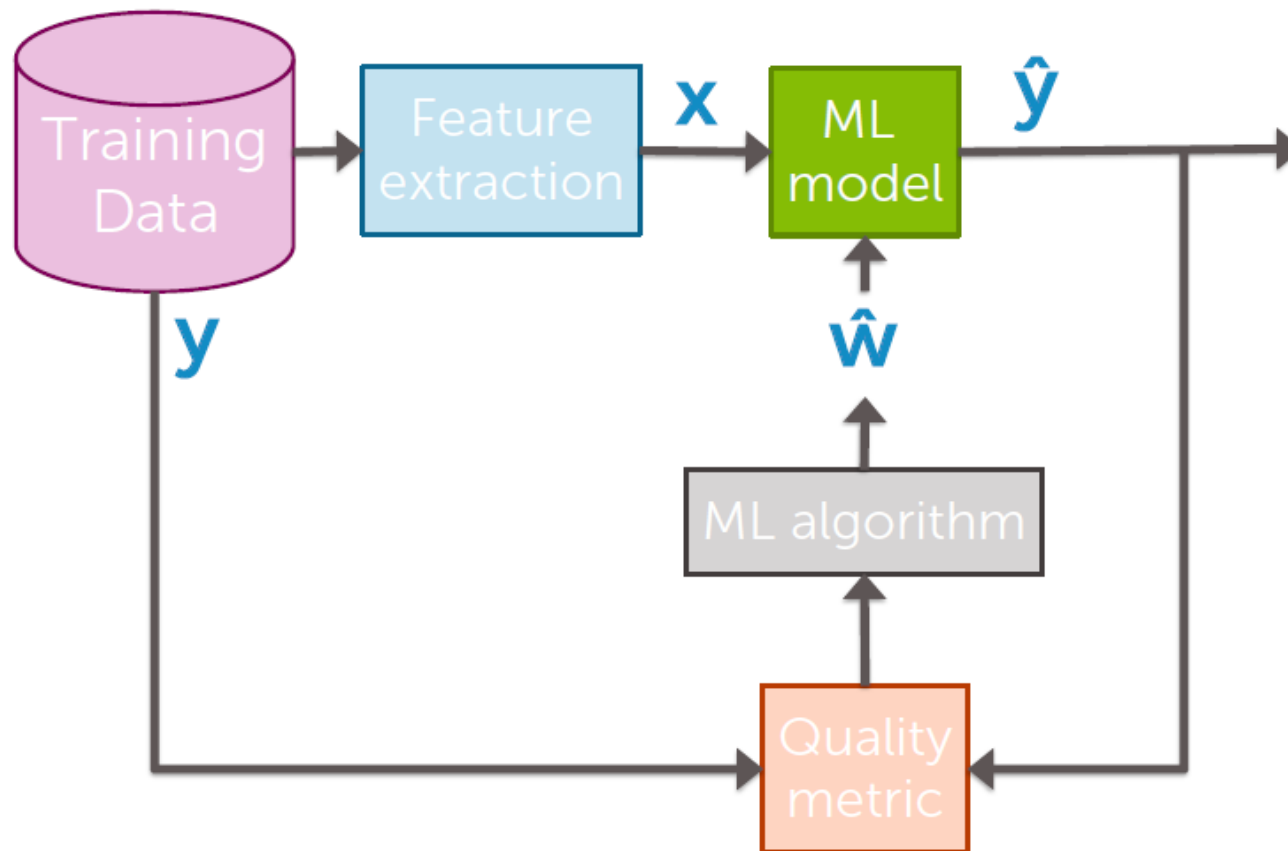
- **Data:** Users score some movies

$Rating(u, v)$ known for black cells
 $Rating(u, v)$ unknown for white cells

- **Goal:** Filling missing data?



Flow chart



ML model

Suppose we had d topics for each user & movie

- Describe movie v  with topics R_v
 - How much is it action, romance, drama,...

$$R_v = [0.3 \quad 0.01 \quad 1.5 \quad \dots]$$

- Describe user u  with topics L_u
 - How much she likes action, romance, drama,...

$$L_u = [2.5 \quad 0 \quad 0.8 \quad \dots]$$

- $\widehat{Rating}(u,v)$ is the product of the two vectors

$$\begin{array}{l} R_v = [0.3 \quad 0.01 \quad 1.5 \quad \dots] \\ L_u = [2.5 \quad 0 \quad 0.8 \quad \dots] \end{array} \rightarrow 0.3 * 2.5 + 0 + 1.5 * 0.8 + \dots = 7.2$$

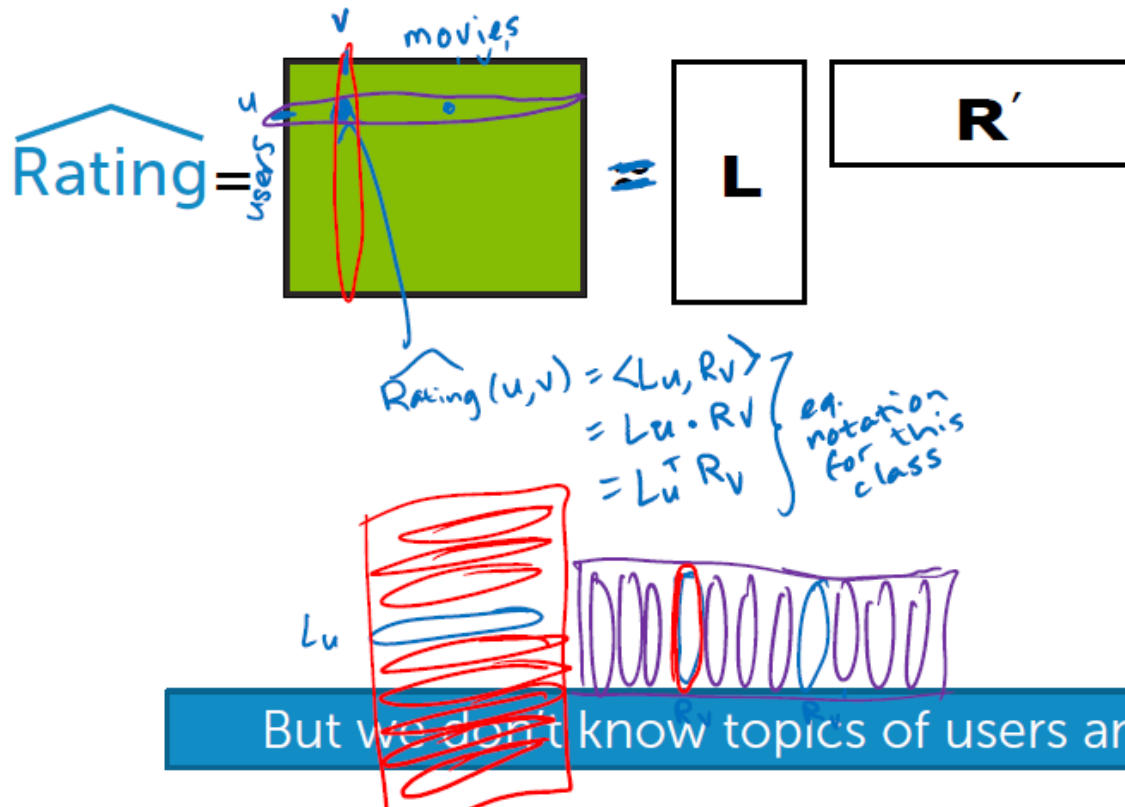
not 0, ..., 5
↓
star

$$L_{u'} = [0 \quad 3.5 \quad 0.01 \quad \dots] \rightarrow 0 + 0.01 * 3.5 + 1.5 * 0.01 + \dots = 0.8$$

- Recommendations:** sort movies user hasn't watched by $\widehat{Rating}(u,v)$

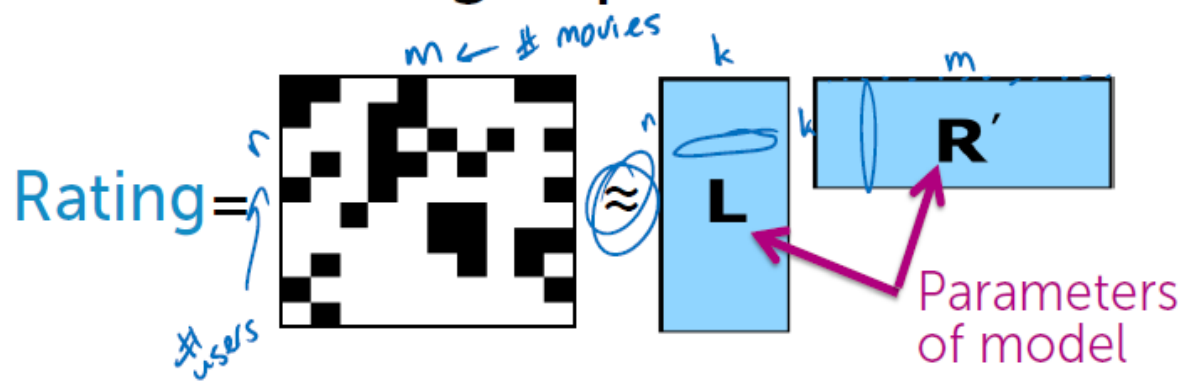
ML model

Predictions in matrix form



Matrix factorisation model

Matrix factorization model: Discovering topics from data



params for model w/ k topics:

$nk + k \cdot m \ll n \cdot m$
learn this using only black squares predict this
↑
full matrix

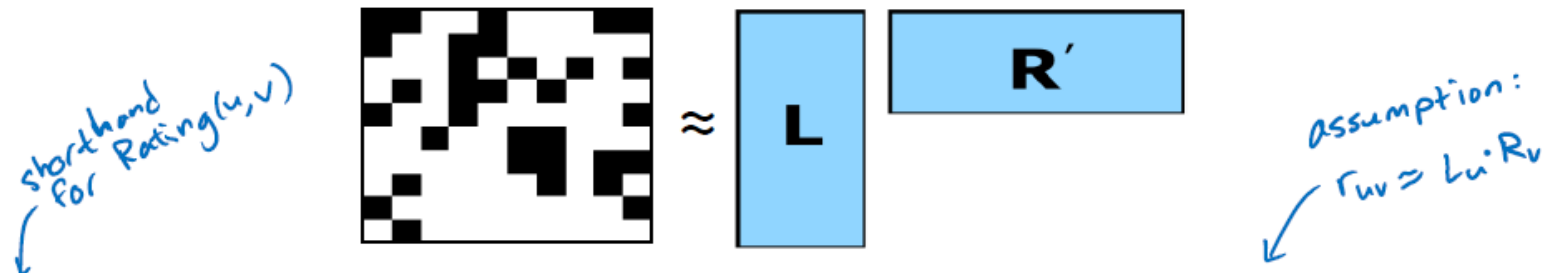
- Only use observed values to estimate "topic" vectors $\hat{L}_u$ and $\hat{R}_v$
- Use estimated $\hat{L}_u$ and $\hat{R}_v$ for recommendations

Many efficient algorithms for factorization

Matrix factorisation model

Is the problem well posed?

Can we uniquely identify the latent factors?



If r_{uv} is described by L_u, R_v what happens if we redefine the "topics" as

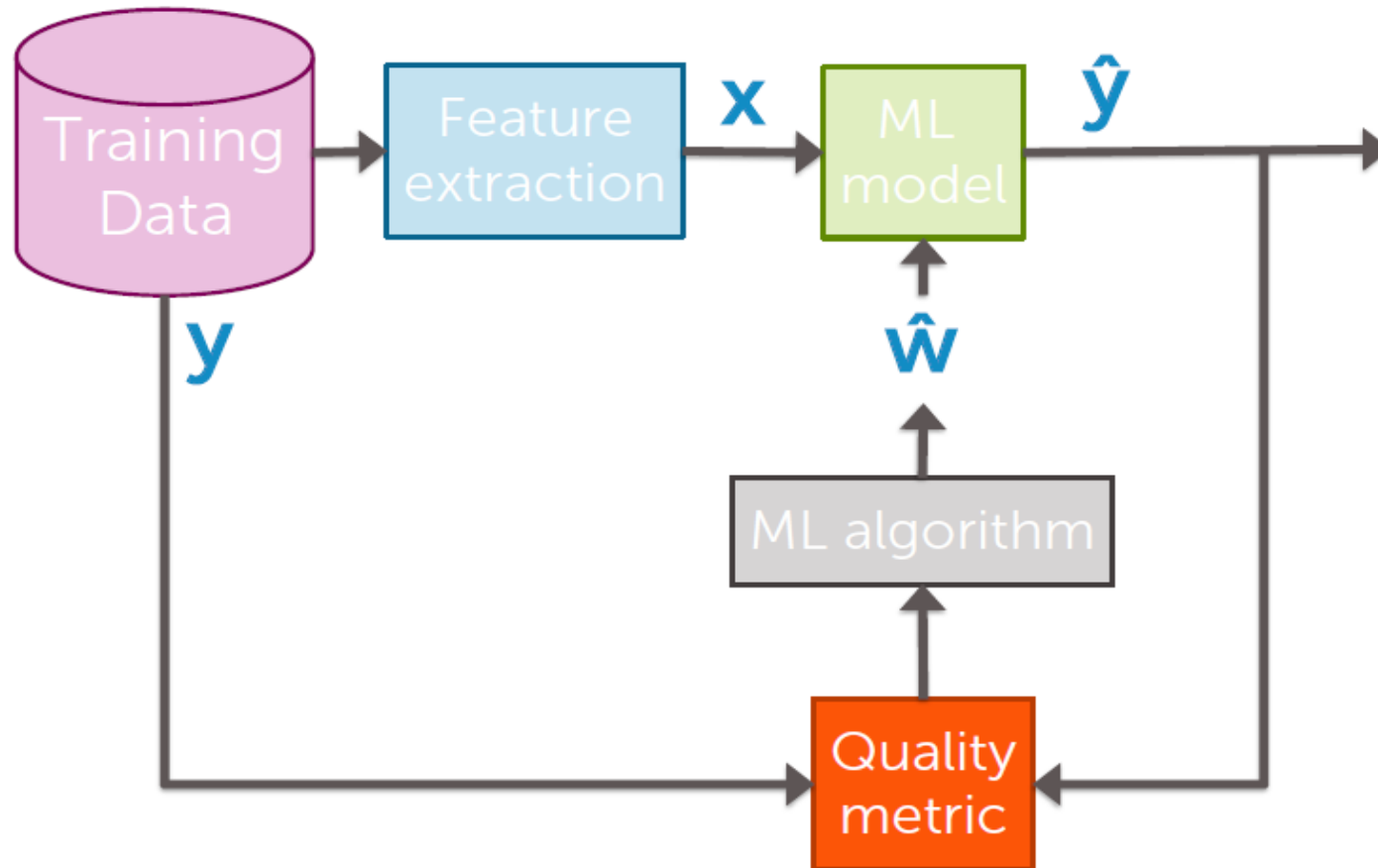
$$\tilde{L}_u = c L_u \quad \tilde{R}_v = \frac{1}{c} R_v$$

Then,

$$\tilde{L}_u \cdot \tilde{R}_v = c L_u \cdot \frac{1}{c} R_v = c \frac{1}{c} (L_u \cdot R_v) = L_u \cdot R_v \approx r_{uv}$$

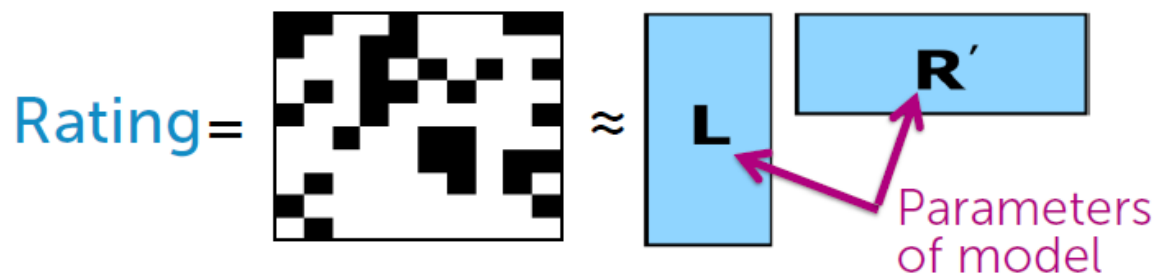
Other (orthonormal) transformations can have the same effect.
(other trans. have same effect... orthonormal trans)
can't uniquely identify $L_u, R_v \Rightarrow$ don't interpret individually, only product

Flow chart



Matrix factorisation model

Matrix factorization objective



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

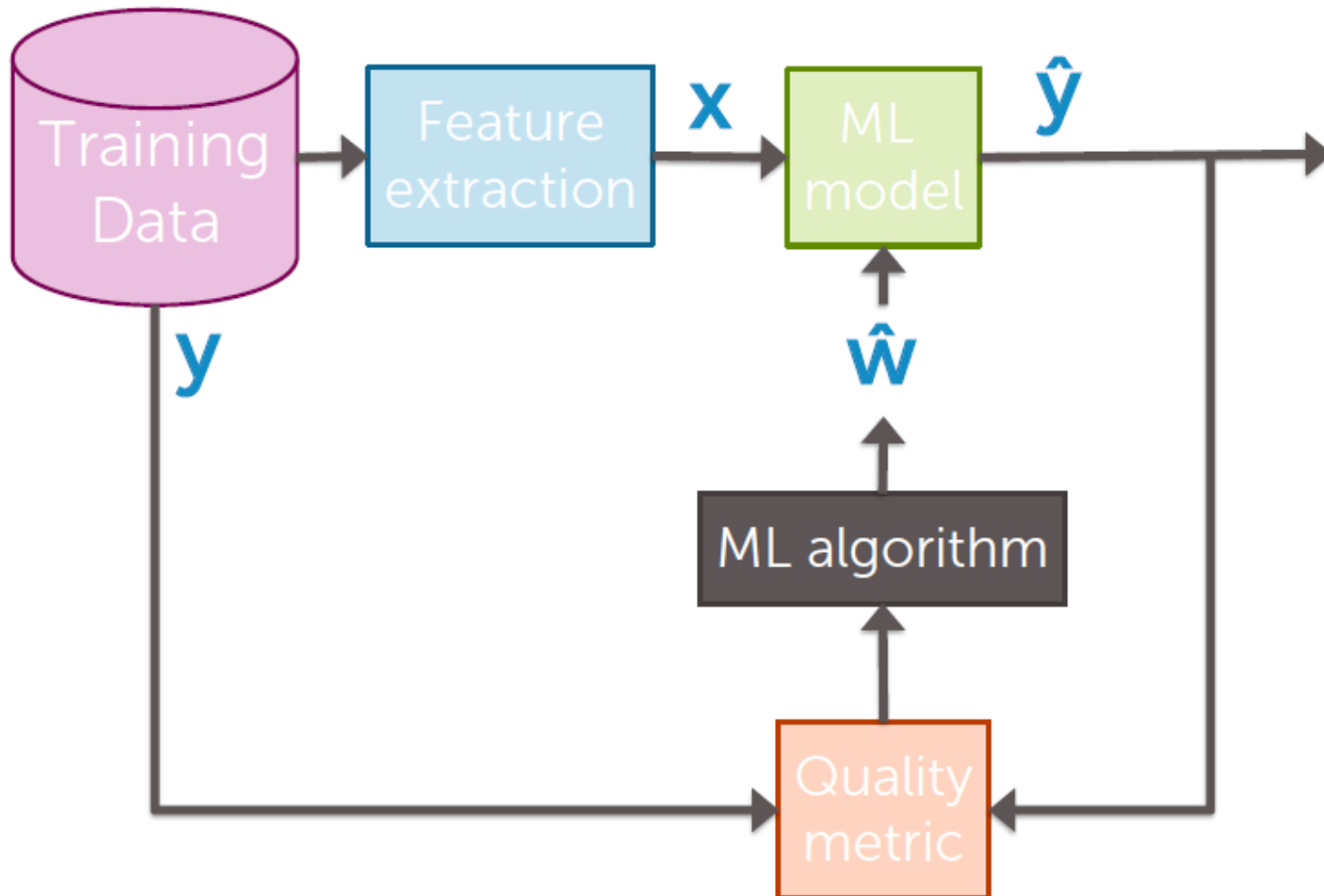
$$\min_{L, R} \sum_{u,v: r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2$$

predicted $\leftarrow$ actual rating

$\uparrow$ only over obs. values

- Non-convex objective
subject to convergence to local mode

Flow chart



ML algorithm

Coordinate descent

Goal: Minimize some function g $\min_w g(w)$

$$g(w) = g(w_0, w_1, \dots, w_D)$$

Often, hard to find minimum for all coordinates, but **easy for each coordinate**

Coordinate descent:

Initialize $\hat{w} = 0$ (or smartly...)

while not converged

pick a coordinate j

$$\hat{w}_j \leftarrow \min_w g(\hat{w}_0, \dots, \hat{w}_{j-1}, w, \hat{w}_{j+1}, \dots, \hat{w}_D)$$

fixed from prev. iteration
just min over i th coord.



ML algorithm

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 R_v , optimize for user factors L_u
- First key insight:

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

$V_u \triangleq$ set of movies
user u has rated

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

← ind. opt. problem
for each user

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

ML algorithm

Comments on coordinate descent

How do we pick next coordinate?

- At random ("random" or "stochastic" coordinate descent), round robin, ...

No stepsize to choose!

Super useful approach for *many* problems

- Converges to optimum in some cases (e.g., "strongly convex")

ML algorithm

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 R_v , optimize for user factors L_u
- First key insight:

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

$V_u \triangleq$ set of movies
user u has rated

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

← ind. opt. problem
for each user

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

ML algorithm

Minimize objective separately for each user

- For each user u : $\min_{L_u} \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2$
param vector $\uparrow$ $\overline{\text{fixed}}$ $\uparrow$ *obs.*

- Second key insight: Looks like linear regression!

$$\min_w \sum_{i=1}^N (\underbrace{w \cdot h(x_i)}_{w^T h} - y_i)^2$$

opt. w/ grad. desc.

ML algorithm

Overall coordinate descent algorithm

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

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

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

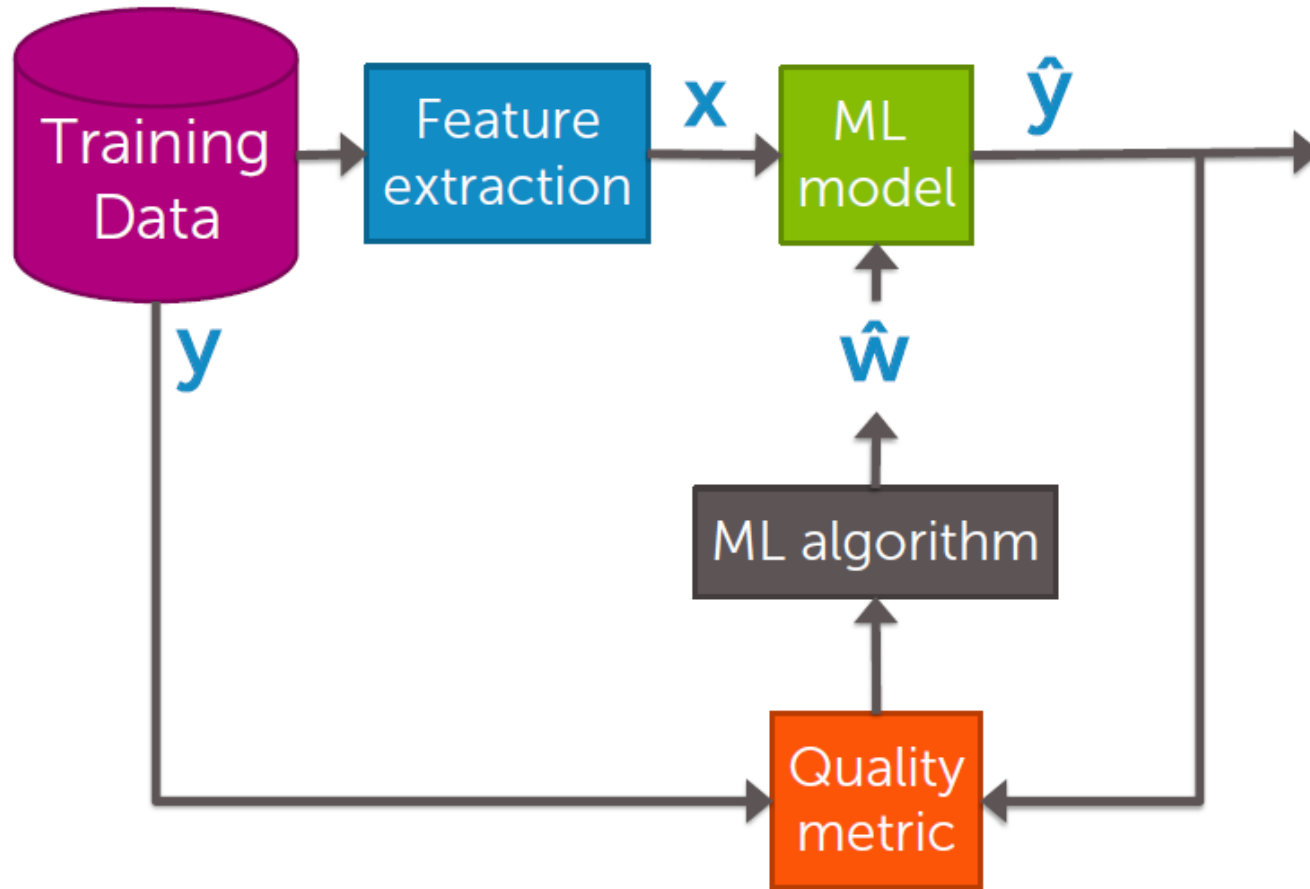
- Fix user factors optimize for movie factors
– Independent least-squares over movies

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

- System may be underdetermined: use regularization
- Converges to local optima
- Choices of regularizers and impact on algorithm:

$$\begin{aligned} L_2: \sum_u \|L_u\|_2^2 &\triangleq \|L\|_F^2 \rightarrow \text{ridge} \\ L_1: \sum_i \|L_i\|_1 &\rightarrow \text{lasso} \end{aligned}$$

Flow chart



Recommendation

Using the results of matrix factorization

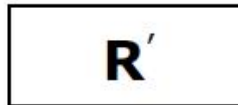
- Discover "topics" R_v for each movie $v \rightarrow \hat{R}_v$
- Discover "topics" L_u for each user $u \rightarrow \hat{L}_u$
- $Score(u,v)$ is the product of the two vectors $\rightarrow \hat{L}_u \cdot \hat{R}_v$
Predict how much a user will like a movie
- Recommendations: sort movies user hasn't watched by $Score(u,v)$
recommend movies v w/ highest score (u,v)

Example topics discovered from Wikipedia

Application to text data:



$\approx$



party law
government
election court
president elected
council general minister
political national members
committee united office federal
member house parliament vote
public elections democratic held
son died
married family
king daughter john
death william father
born wife royal ireland
irish henry house lord
charles sir prince brother
children england queen duke
thomas years marriage george
earl edward english

school students
university high college schools
education year program student
career community group time training
senior members senior national youth
public academic association science arts
educational institute class institute
department teachers colleges classes
office activities activities school
engineering meeting founded family gis
sports student looks professional sound
teaching students secondary established

york county
american united
city washington john
texas served virginia
pennsylvania war moved ohio
chicago william carolina north
florida illinois george james died
massachusetts president
named jersey born boston
years seed company gangs shift began
chicago fort years philadelphia white
season team
game league games
played coach football
record teams baseball field year
second career play basketball
hockey three yards won
politics war series player head conference
championship seasons players shift high
time named national led nfl miss major
traveled stadium champion best playing time
history role foundation signed

album band
song released
music songs single records
recorded rock bands release
live tour video record albums
label group recording
album version tracks number featured time
chart 10 us top performed studio played
singles second time peak artist song of studio
singer artists members released early
second time

century king
roman empire greek
bc ancient emperor ii
kingdom period battle city
time great war ad early reign
kings ii son rule power greece
army centuries dynasty
modern history imperial medieval death
attorney peace led expansion dedicated
ruled year throne attacks capital battle
military left a massive control
species family
birds small long large animals
bird plants genus
habitat rose last tropical white trunk
leaves brown american forests trees as the
flowers eggs worldwide field insect
adapted well length time including
habitable range food female fruit about
maize endemic forest group including
include insect threatened led

radio station
news television
channel broadcast
stations network media tv
broadcasting time format local
program bbc programming live
in morning total larger studio live air public
sat hosted coverage music per to inter
daily channels digital also added managed
against launched collaborations
programs day live broadcasts moved the
photo security task right

engine car
design model cars
production built engines
vehicle class models
speed vehicles designed
produced power front system rear
version type series motor rear
standard gun company
introduced range ford sold fuel
view wheel turn third factory machine
classified based regional vehicle time
powered small high weight aircraft body
recorded early
art museum work
works artists collection design
arts painting artist gallery
paintings exhibition style
including painted architecture york london
painted the early created sculpture artists
fading contemporary industrialism great
museum visited images time photograph
figures sculpture environment modern
period photography began studio showing
include portrait produced landscape
painted equal

age 18 population
income average years
median living 65 males
females households 100 family
people families older town size
city household miles density
american
household 2000 square 40 20 84
population 24 44 white female and including
white living 2000 individuals included
poverty united village

war army military
forces battle force british
command general navy ship
division ships troops corps
service naval regiment
commander infantry attack men
officer
squad soldiers units officers
operations unit joint august brigade july the
training march battalion left care for
action december time military united
action was royal garment machine major

white red
black blue called
color will head green gold side
small hand long arms top flag
horses wear silver common light
dog wood body type large
yellow
term wear upper cut circular left
generally traditional full front female shape
four feet square line coat three female
modern face shape

music musical opera
festival orchestra dance
performed jazz piano theatre
performance works
symphony composer plays performances
instrumental musician classical including
work composed major writing songs folk
instrumental social composition
performing concerto playing stage
opera inside produce chair ensemble
squad style line with feet piece chamber
newspapers doing

Limitations of matrix factorisation

- Cold-start problem
 - This model still cannot handle a new user or movie



Cold-start problem more formally

Consider a new user u' and predicting that user's ratings $\rightarrow L_{u'}$

- No previous observations

$$r_{u'v} = ? \quad \forall v$$

- Objective considered so far:

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

does not depend on $L_{u'}$ (no obs. ratings)

only term that appears

- Optimal user factor:

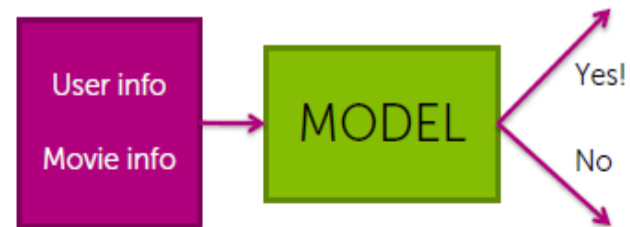
$$L_{u'} = 0 \quad \text{only penalty term present}$$

- Predicted user ratings:

always predict: $r_{u'v} = 0 \quad \forall v \quad \dots \text{problem!}$

Combining features and topics

- Features capture **context**
 - *Time of day, what I just saw, user info, past purchases,...*
- Discovered topics from matrix factorization capture **groups of users** who behave similarly
 - *Women from Seattle who teach and have a baby*
- **Combine** to mitigate cold-start problem
 - Ratings for a new user from **features** only
 - As more information about user is discovered, matrix factorization **topics** become more relevant



Colaborative filtering

- Create feature vector for each movie (often have this even for new movies):

$$\phi(v) = (\text{genre, year, director, ...})$$

genre, year, director, ...
('action', 1994, Tarantino, ...)

- Define weights on these features for how much all users like each feature

w = vector of same length

- Fit linear model:

For all users, $r_{uv} \approx w \cdot \underline{\phi(v)}$ standard regression model

- Minimize:

$$\min_w \sum_{r_{uv} \neq ?} (w \cdot \phi(v) - r_{uv})^2 + \lambda_w \|w\| \quad \leftarrow \text{LS, lasso, ridge}$$

Building in personalization

- Of course, users do *not* have identical preferences
- Include a **user-specific deviation** from the global set of user weights:
- If we don't have any observations about a user, **use wisdom of the crowd**
- As we gain more information about the user, **forget the crowd**
- Can add in **user-specific features**, and cross-features, too

Featurized matrix factorization: combined approach

Feature-based approach:

- Feature representation of user and movies fixed
- Can address cold-start problem

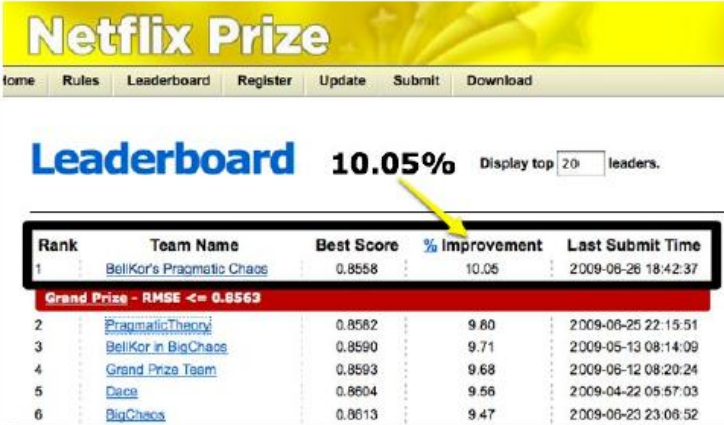
Matrix factorization approach:

- Suffers from cold-start problem
- User & movie features are learned from data

A unified model: $r_{uv} \approx L_u \cdot R_v + (w + w_u) \cdot \phi(u, v)$
Solve via coord. desc., grad. desc., etc.

Blending models

- Squeezing last bit of accuracy by blending models
- Netflix Prize 2006-2009
 - 100M ratings
 - 17,770 movies
 - 480,189 users
 - Predict 3 million ratings to highest accuracy
 - **Winning team blended over 100 models**



The screenshot shows the Netflix Prize Leaderboard interface. At the top, there's a yellow banner with "Netflix Prize" and navigation links: Home, Rules, Leaderboard, Register, Update, Submit, Download. Below this, the "Leaderboard" section is highlighted in blue, followed by "10.05%" and "Display top 20 leaders." A yellow arrow points from the "10.05%" text to the "% Improvement" column header in the table below. The table has five columns: Rank, Team Name, Best Score, % Improvement, and Last Submit Time. The first row is highlighted in red and indicates the Grand Prize threshold: "Grand Prize - RMSE <= 0.8563".

Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	BellKor's Pragmatic Chaos	0.8558	10.05	2009-06-26 18:42:37
Grand Prize - RMSE <= 0.8563				
2	PragmaticTheory	0.8582	9.80	2009-06-25 22:15:51
3	BellKor in BigChaos	0.8590	9.71	2009-05-13 08:14:09
4	Grand Prize Team	0.8593	9.68	2009-06-12 08:20:24
5	Dase	0.8604	9.56	2009-04-22 05:57:03
6	BigChaos	0.8613	9.47	2009-06-23 23:08:52

Recommender system: how effective?

The world of all baby products



Recommending system: how effective?

User likes subset of items



Recommender system: how effective?

Why not use classification accuracy?

- Classification accuracy = fraction of items correctly classified (*liked* vs. *not liked*)
- Here, **not** interested in what a person *does not like*
- Rather, how quickly can we discover the relatively few *liked* items?
 - (Partially) an imbalanced class problem

How many liked items were recommended?



Recommending system: how effective?

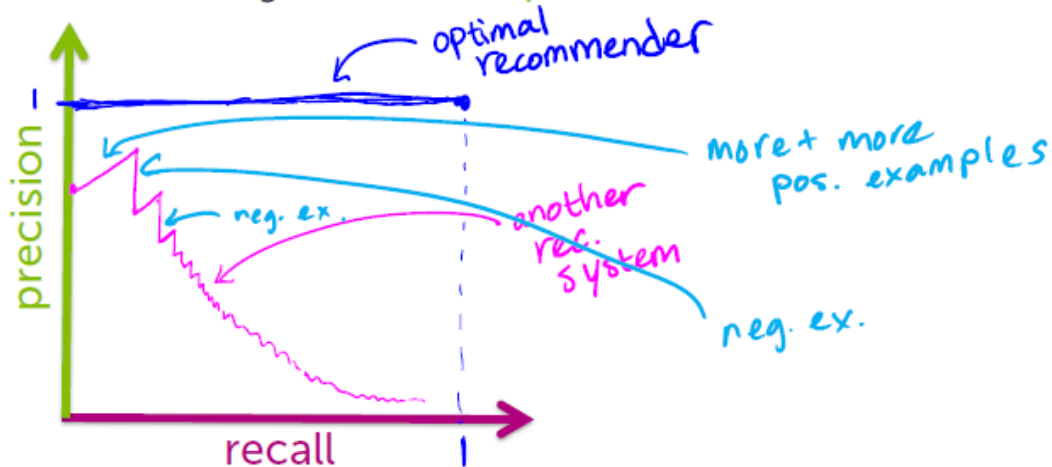
How many recommended items were liked?



Recommending system: how effective?

Precision-recall curve

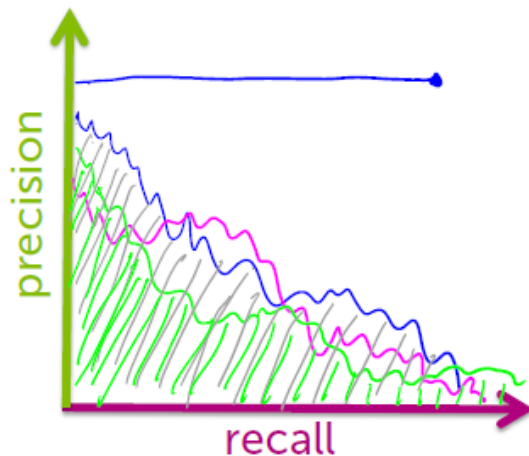
- **Input:** A specific recommender system
- **Output:** Algorithm-specific precision-recall curve
- To draw curve, vary threshold on # items recommended
 - For each setting, calculate the precision and recall



Recommending system: how effective?

Which Algorithm is Best?

- For a given **precision**, want **recall** as large as possible (or vice versa)
- One metric: largest area under the curve (AUC)
- Another: set desired recall and maximize precision (precision at k)



Recommender system

Models

- Collaborative filtering
- Matrix factorization
- PCA

Algorithms

- Coordinate descent
- Eigen decomposition
- SVD

Concepts

- Matrix completion, eigenvalues, random projections, cold-start problem, diversity, scaling up