INTRODUCTION TO DATA SCIENCE

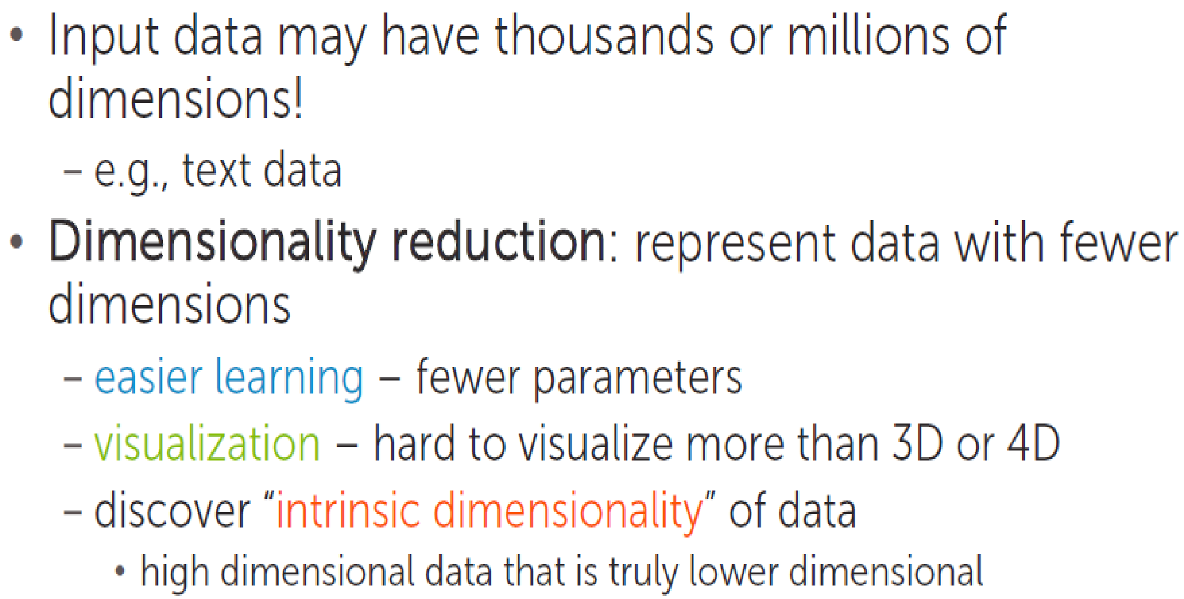
This lecture is
based on course by E. Fox and C. Guestrin, Univ of Washington

4/12/2024

WFAiS UJ, Informatyka Stosowana
I stopień studiów

Principal component analysis

Dimensionality reduction



- 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

first (new) feat

i^{th} obs $\rightarrow$

$$z_i[1] = 2.5\, x_i[1] + 3\, x_i[2] + 7\, x_i[3] + \dots$$

$$z_i[2] = \dots$$

$$\vdots$$

(other weights on old features)

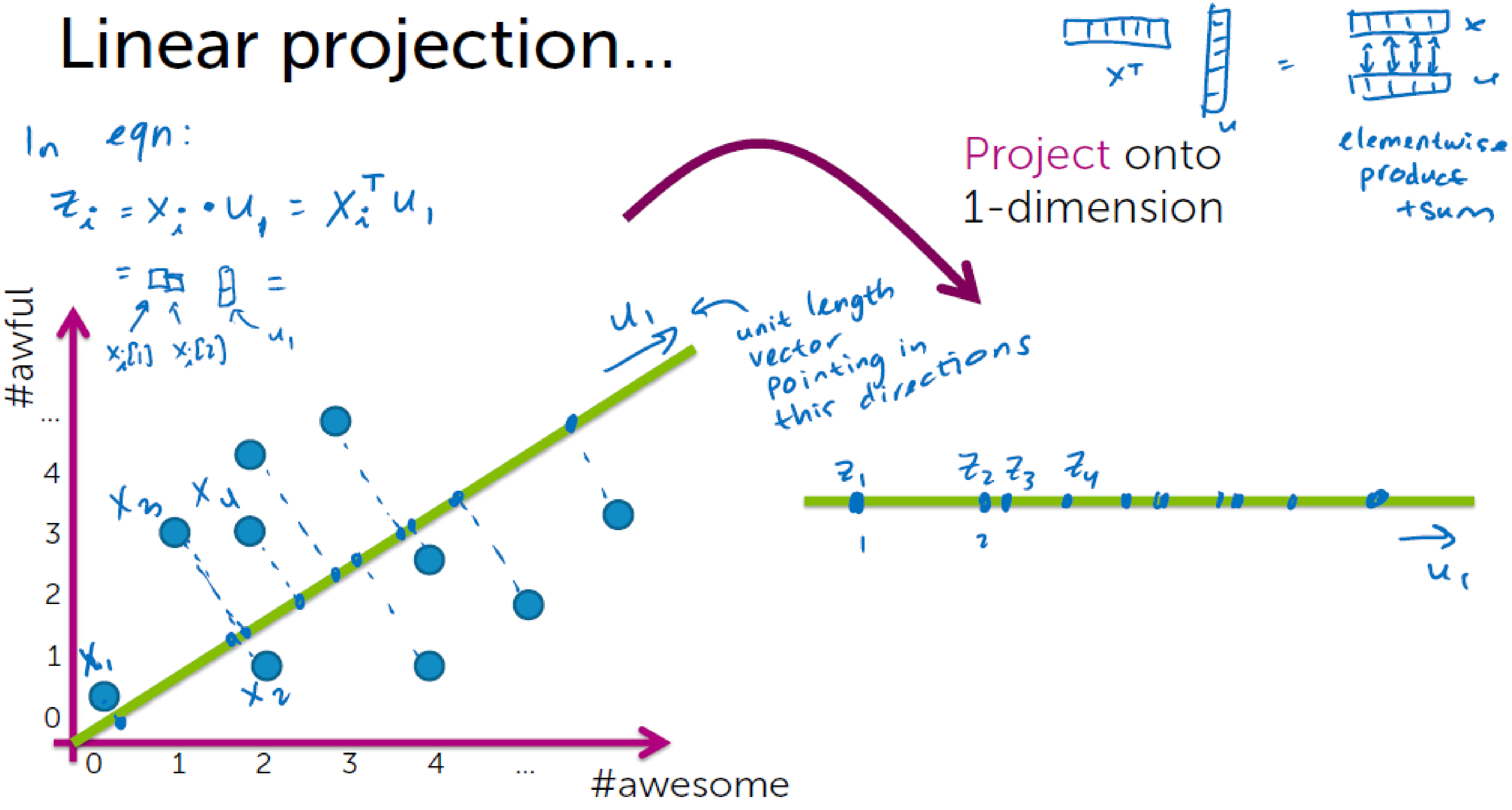
- Let's see this in the unsupervised setting
  - just **x**, but no y

Lowering dimensionality projection

Linear projection…

In eqn:

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



Project onto 1-dimension

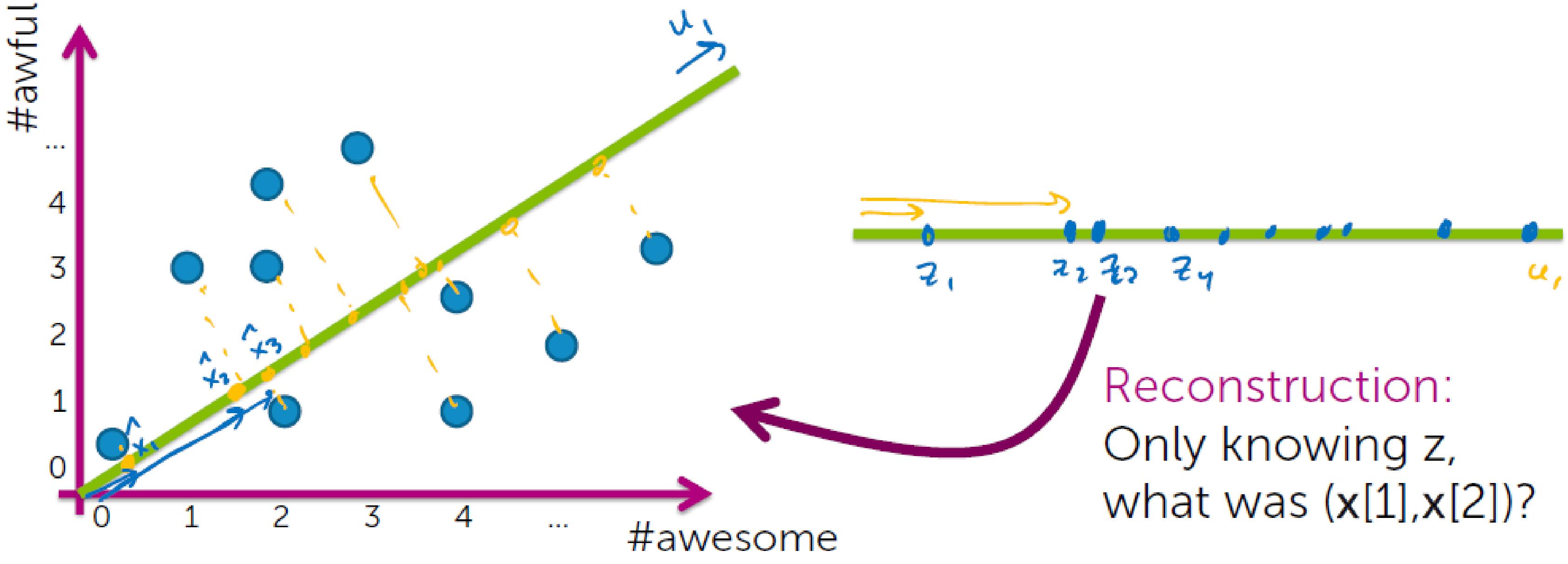
Lowering dimensionality projection

Linear projection and reconstruction

In eqn:

$\hat{x}_i = z_i u_1$

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



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

Lowering dimensionality projection

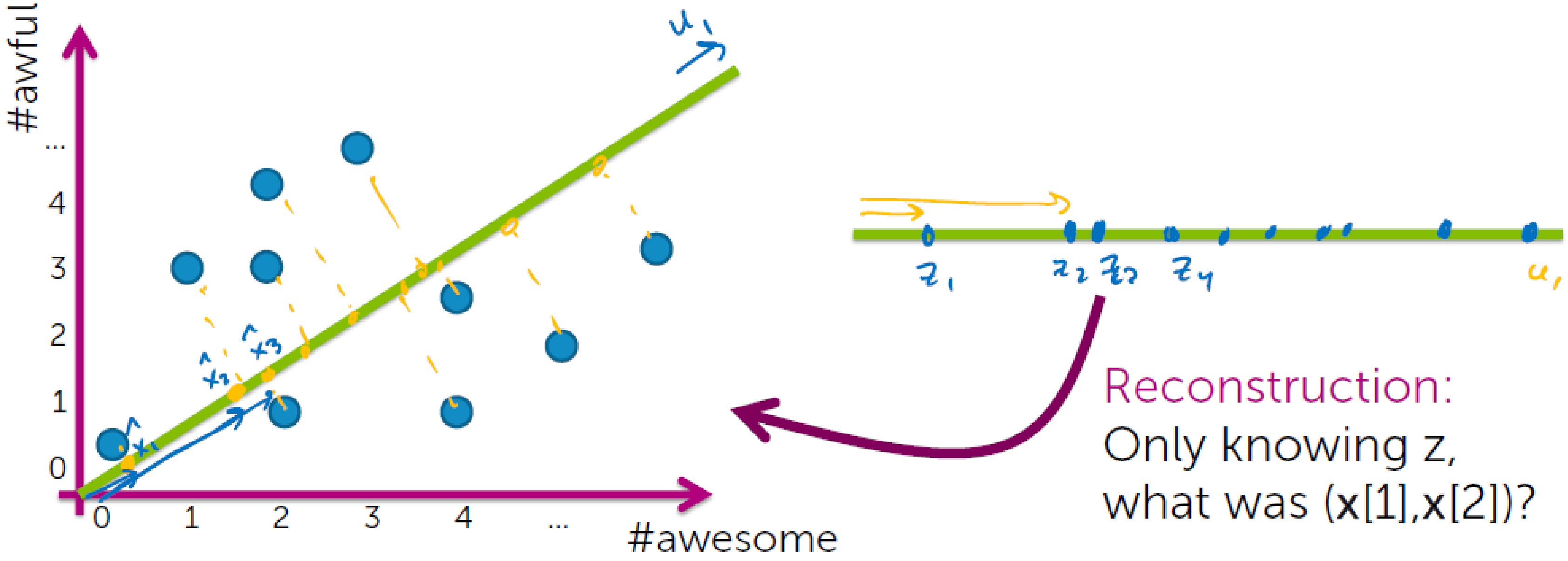
Linear projection and reconstruction

In eqn:

$\hat{x}_i = z_i u_1$

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

and



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

Lowering dimensionality projection

What if we project onto d orthogonal vectors?

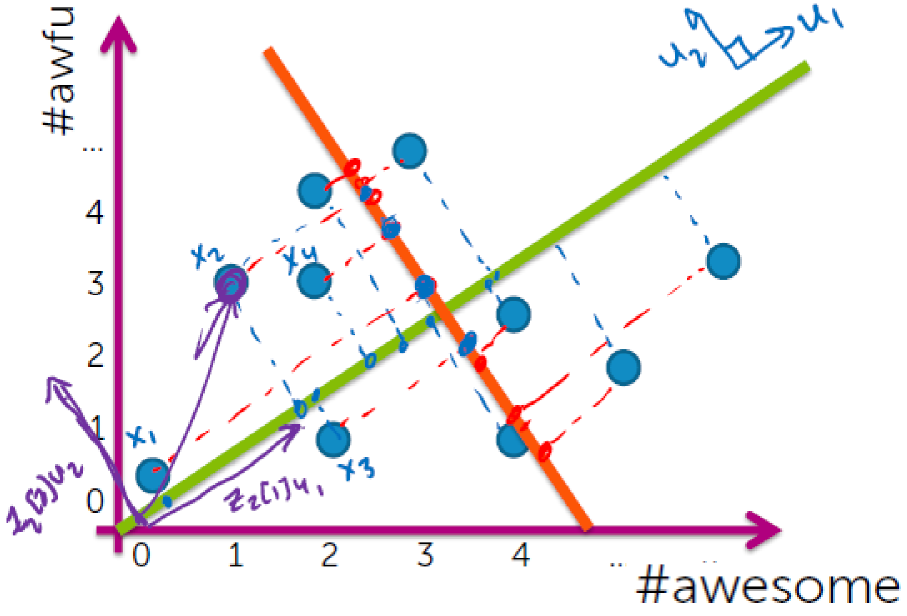
In eqns:

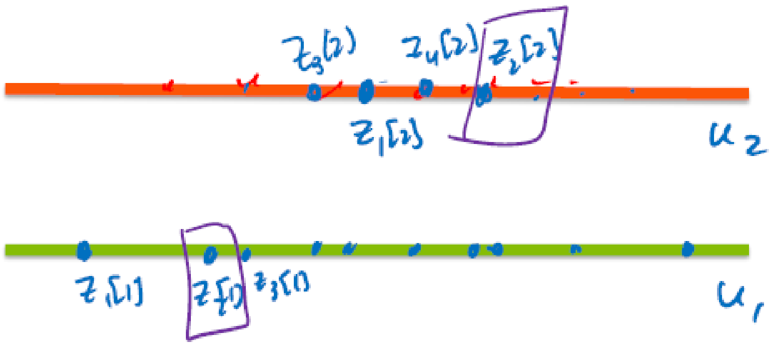
$$\hat{x}_i[1:2) = z_i[1]u_1 + z_i[2]u_2$$

$$= \# \square + \# \square = \square$$

$z_i[1] \leftarrow$ projection onto u_1

$z_i[2] \leftarrow$ projection onto u_2

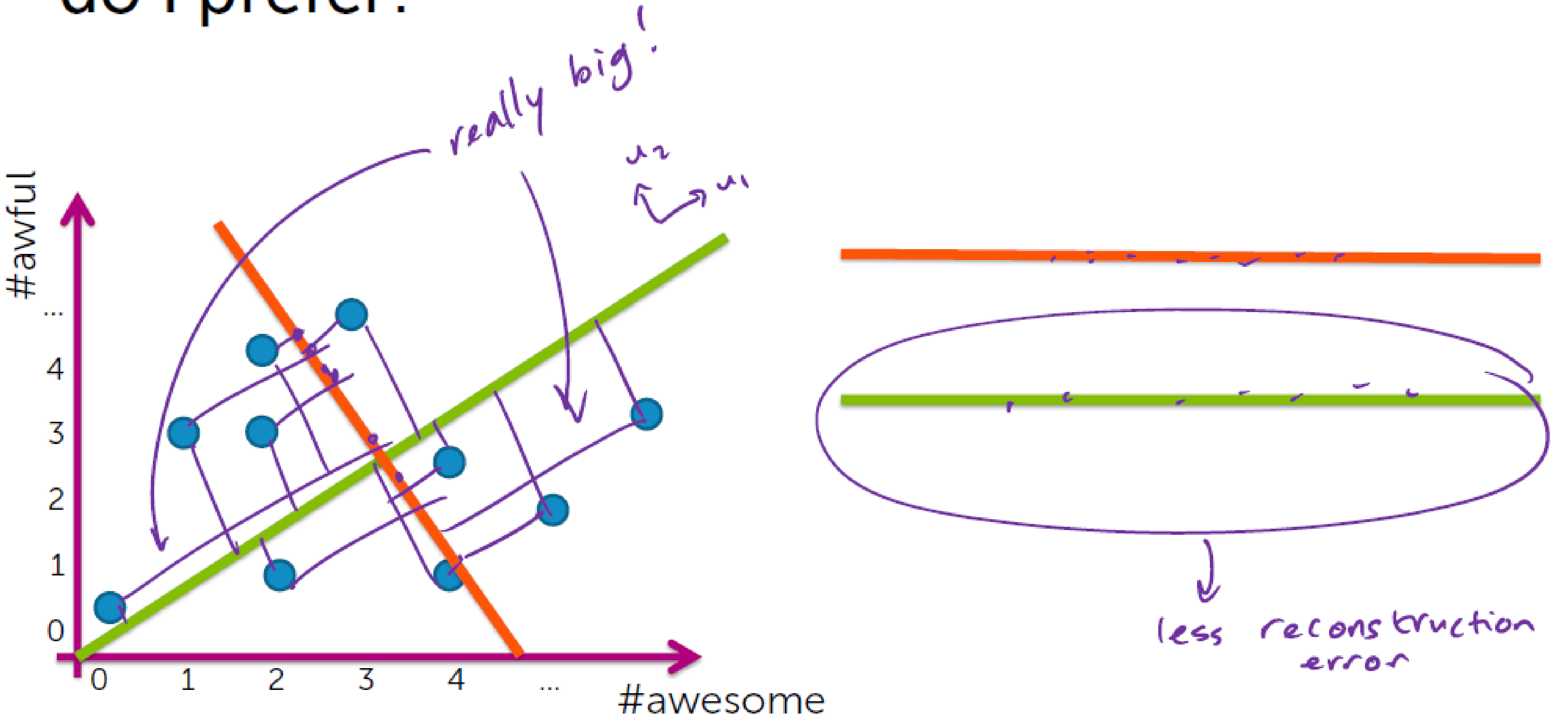




Perfect reconstruction!

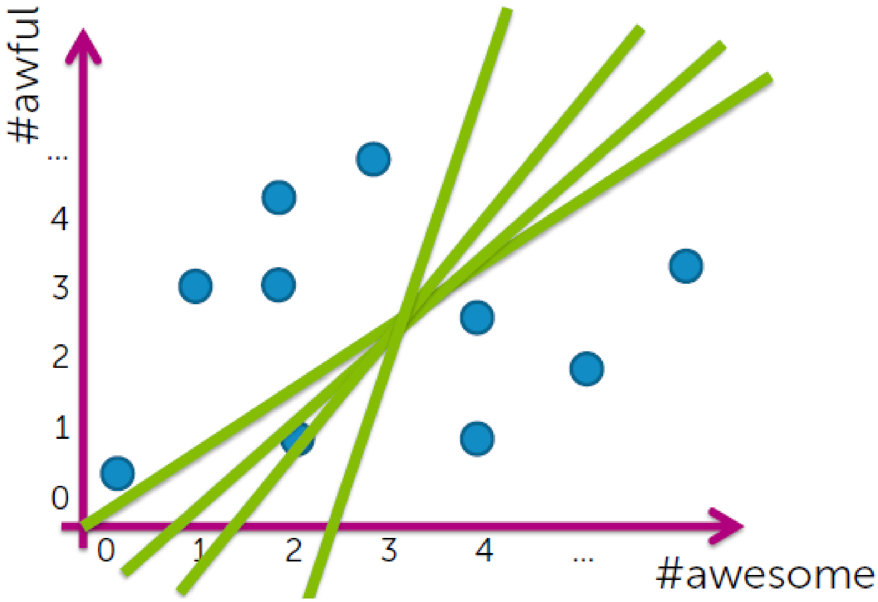
Lowering dimensionality projection

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

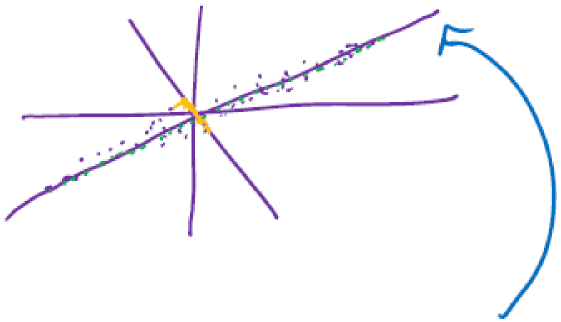


Lowering dimensionality projection

What about over all single vectors?



Consider extreme data example:



choose direction of greatest variations

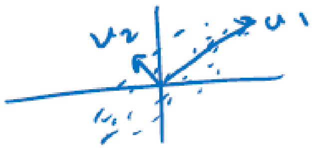
Principal component analysis (PCA)

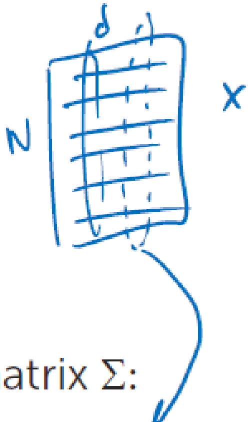
Basic idea

- 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 → 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],...,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

$x_i[1:d] \longrightarrow z_i[1:k]$ $k \le d$

Principal component analysis (PCA)

Reconstruction

Using our principal components, reconstruct observation in original domain:

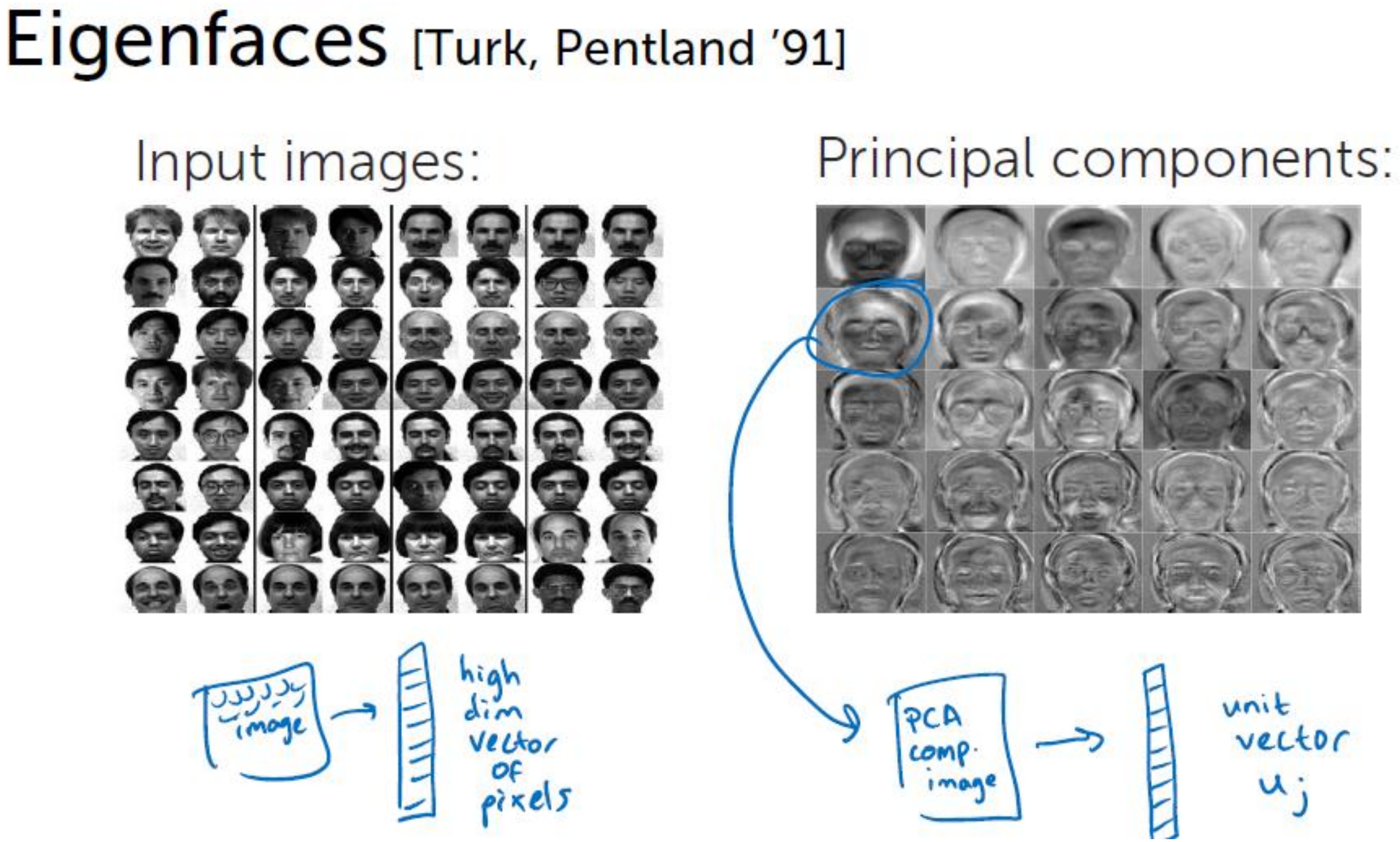
amount

in this direction

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

recenter... add back subtracted mean

Principal component analysis (PCA)



Eigenfaces [Turk, Pentland '91]

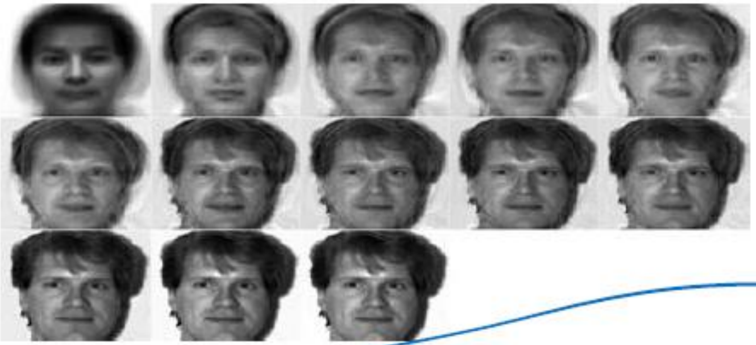
Input images:

Principal components:

Principal component analysis (PCA)

Eigenfaces reconstruction

Each image corresponds to adding 8 principal components:



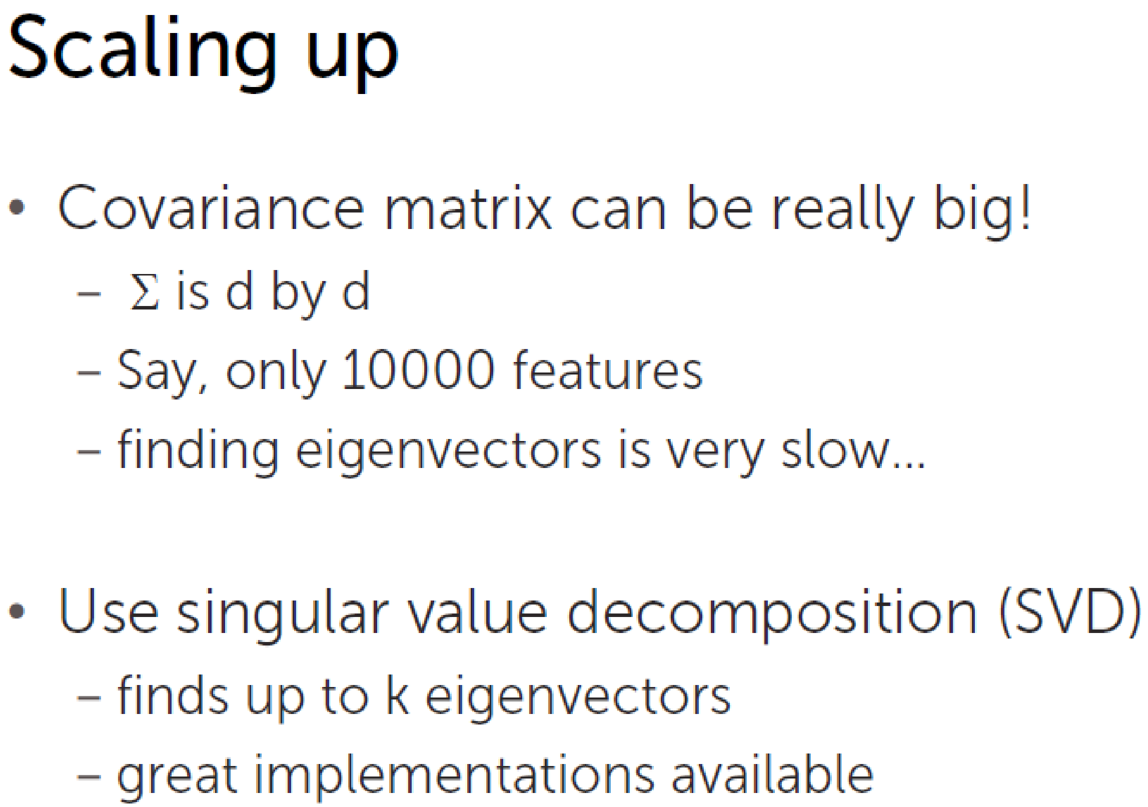
$$\boxed{\text{reconst. image}}_i = \sum_{j=1}^{k} z_i(j)\, u_j + \bar{x}$$

proj. of x_i onto u_j

PCA comp. images

image dataset mean image

Principal component analysis (PCA)



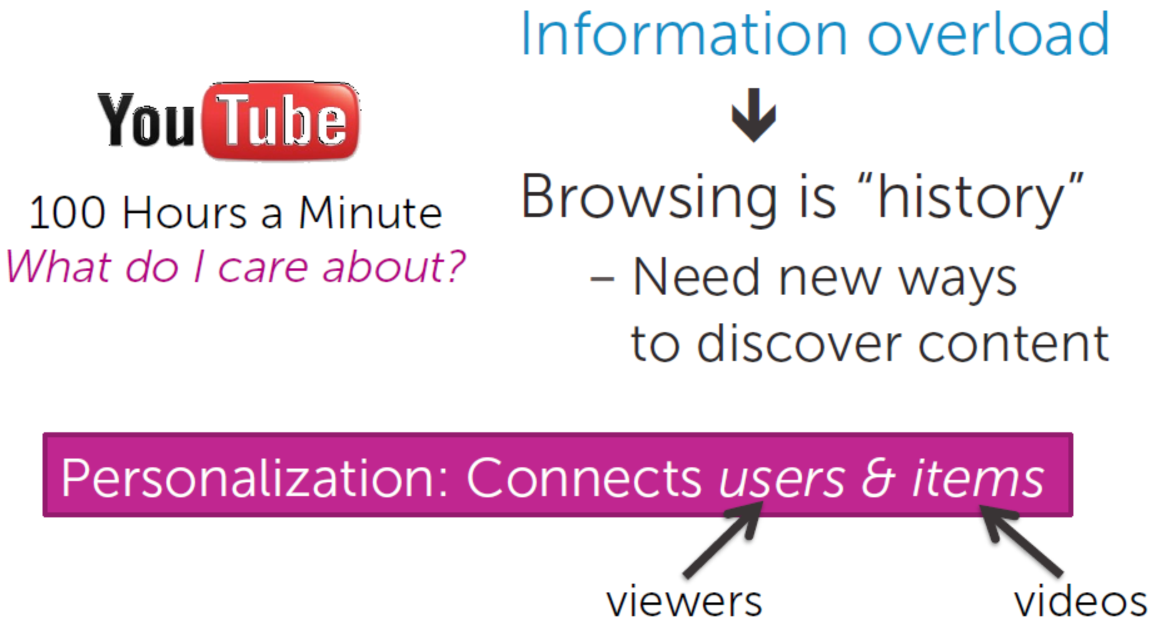
Scaling up

- 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



Information overload

↓

Browsing is "history"

– Need new ways to discover content

100 Hours a Minute

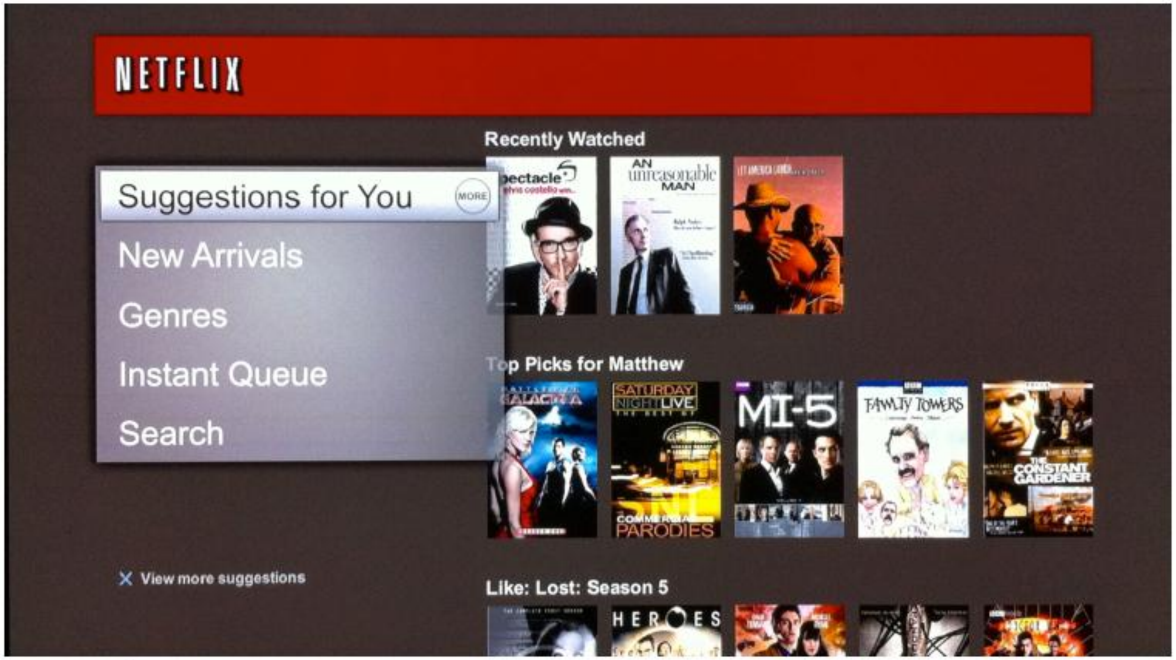
What do I care about?

Personalization: Connects *users & items*

viewers

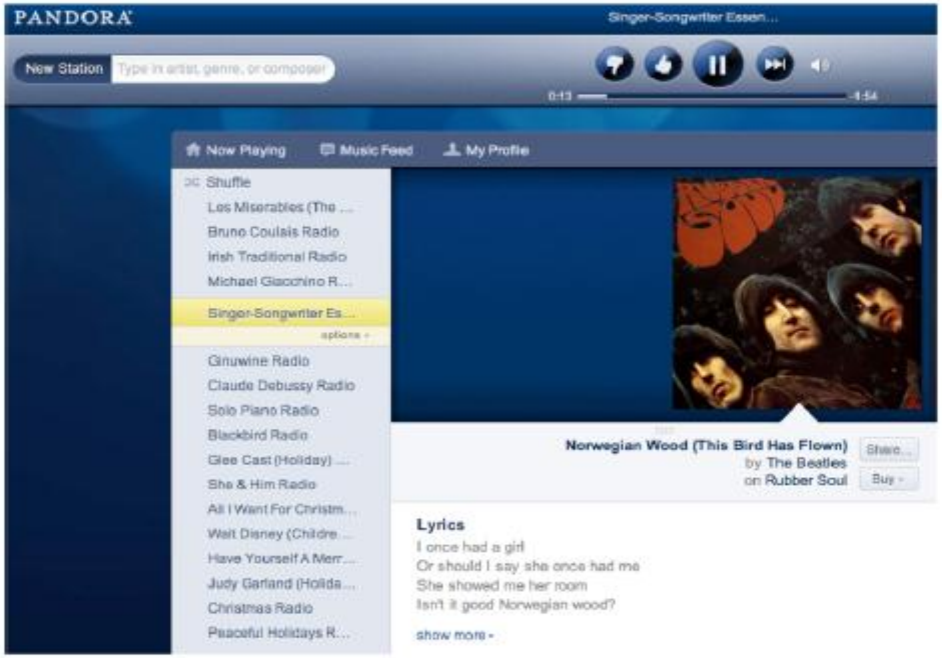
videos

Recomender system: films



Connect users with movies they may want to watch

Recomender system: music



Norwegian Wood (This Bird Has Flown)
by The Beatles
on Rubber Soul

Lyrics

I once had a girl
Or should I say she once had me
She showed me her room
Isn't it good Norwegian wood?

Recommendations form coherent & diverse sequence

Recomender system: friends

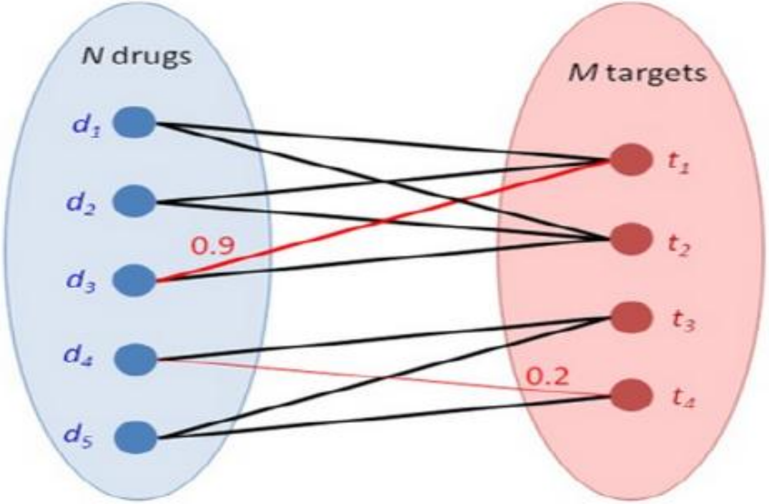
Friend recommendations

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

Recomender system: medicine

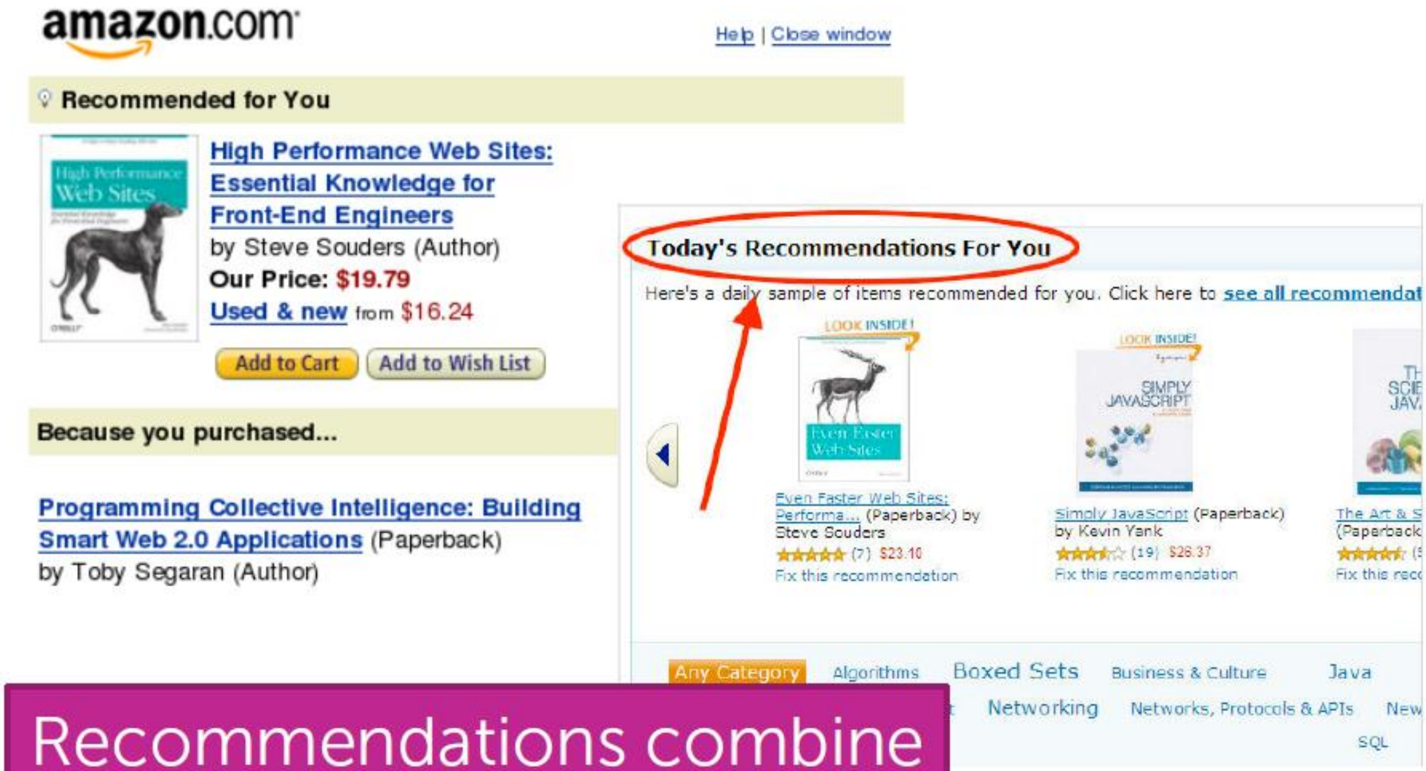
Drug-target interactions

Cobanoglu et al. '13



What drug should we "repurpose" for some disease?

Recomender system:



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

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 recommendat**

Even Faster Web Sites: Performa... (Paperback) by Steve Souders
(7) $23.10
Fix this recommendation

Simply JavaScript (Paperback) by Kevin Yank
(19) $26.37
Fix this recommendation

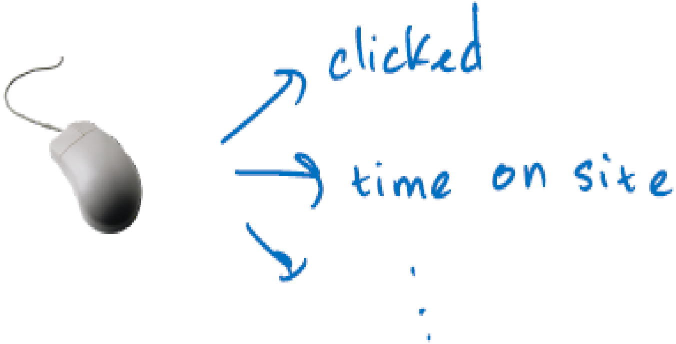
The Art & S
(Paperback
Fix this reco

Recommendations combine global & session interests

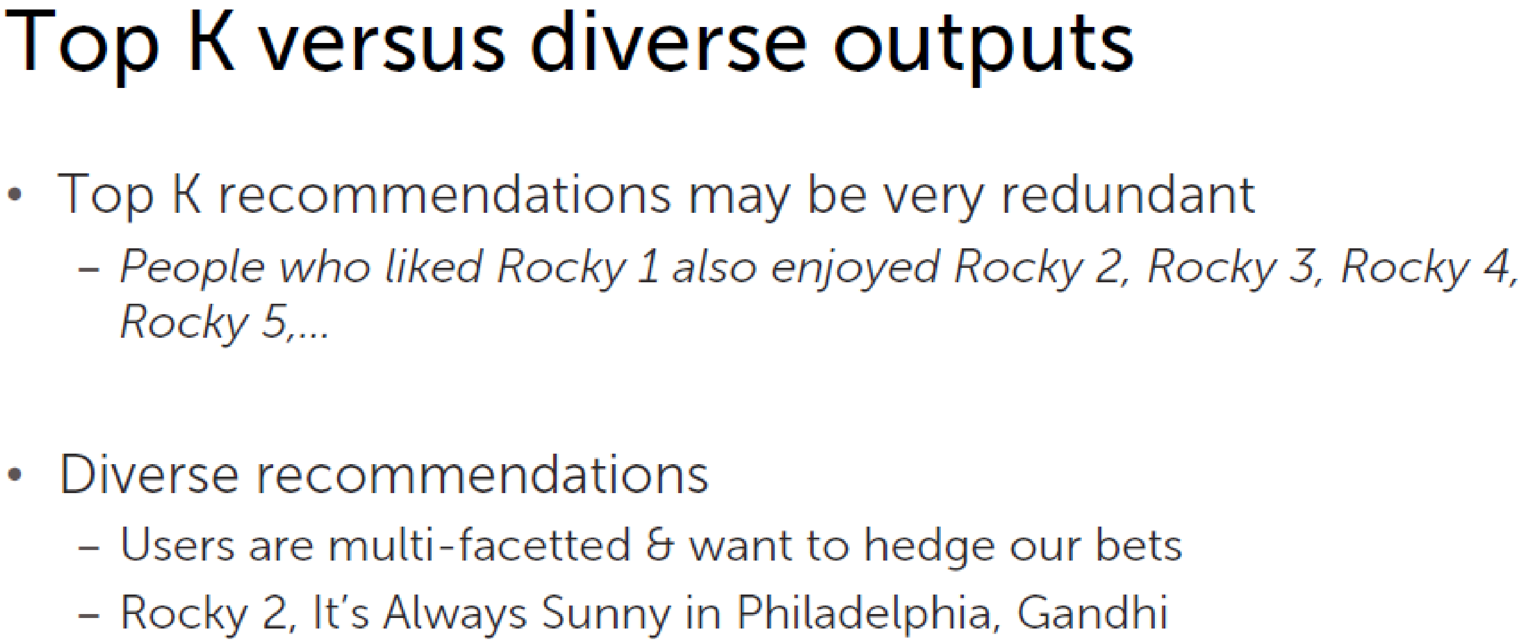
Challenges: Type of feedback

- 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-facetted & 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?

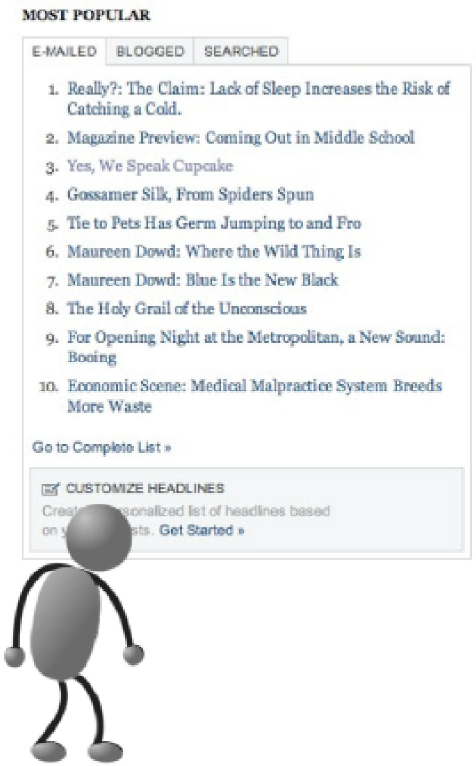
Simplest approach: Popularity

What are people viewing now?

- Rank by global popularity

Limitation:

- No personalization



MOST POPULAR

E-MAILED | BLOGGED | SEARCHED

1. Really?: The Claim: Lack of Sleep Increases the Risk of Catching a Cold.
2. Magazine Preview: Coming Out in Middle School
3. Yes, We Speak Cupcake
4. Gossamer Silk, From Spiders Spun
5. Tie to Pets Has Germ Jumping to and Fro
6. Maureen Dowd: Where the Wild Thing Is
7. Maureen Dowd: Blue Is the New Black
8. The Holy Grail of the Unconscious
9. For Opening Night at the Metropolitan, a New Sound: Booing
10. Economic Scene: Medical Malpractice System Breeds More Waste

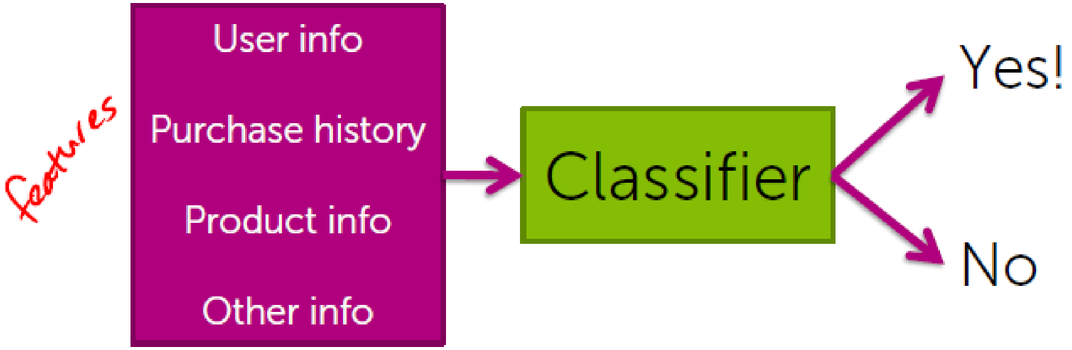
Go to Complete List »

CUSTOMIZE HEADLINES

Creat sonalized list of headlines based on y sts. Get Started »

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, …

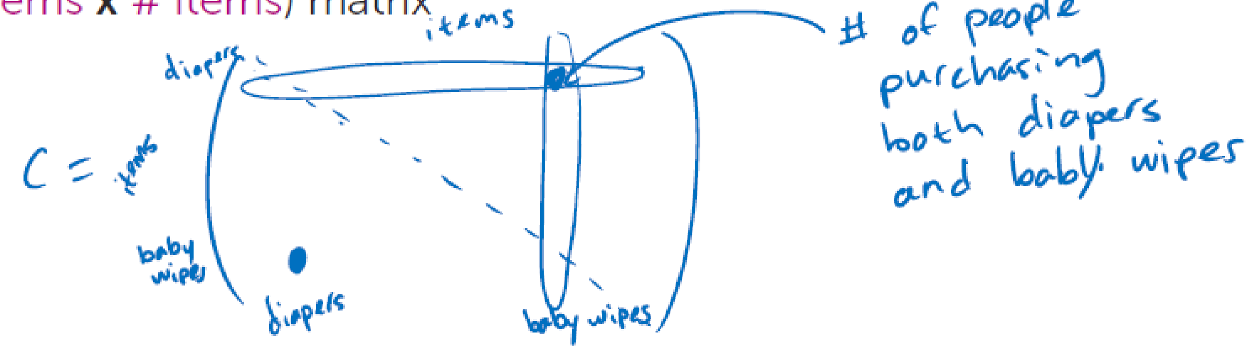
Cons:

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

Recommender system: co-occurence

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-occurence

Making recommendations using co-occurences

User  purchased *diapers*

1. Look at *diapers* row of matrix

$$\begin{array}{cccccc} [\,0 & \cdots & 4 & \cdots & 100 & \cdots\,] \\ \text{DVD} & & \text{pacifiers} & & \text{baby wipes} & \cdots \end{array}$$

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*

sophie [0 . . . 1 million . . .]

DVD diaper . . . baby wipes . . .

Result:

- Drowns out other effects
- Recommend based on popularity

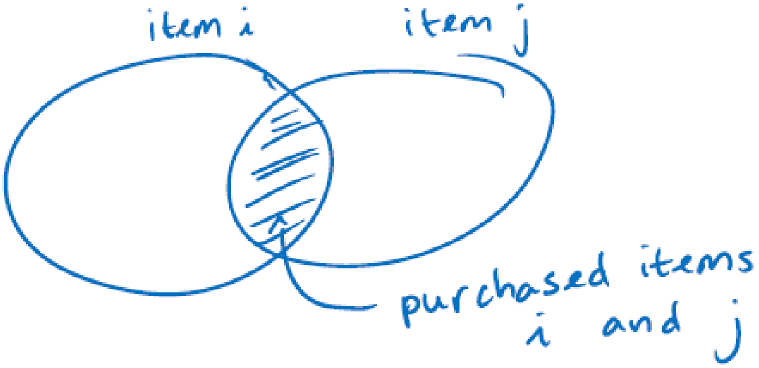
Recommender system: co-occurence

Normalize co-occurrences: Similarity matrix

Jaccard similarity: normalizes by popularity

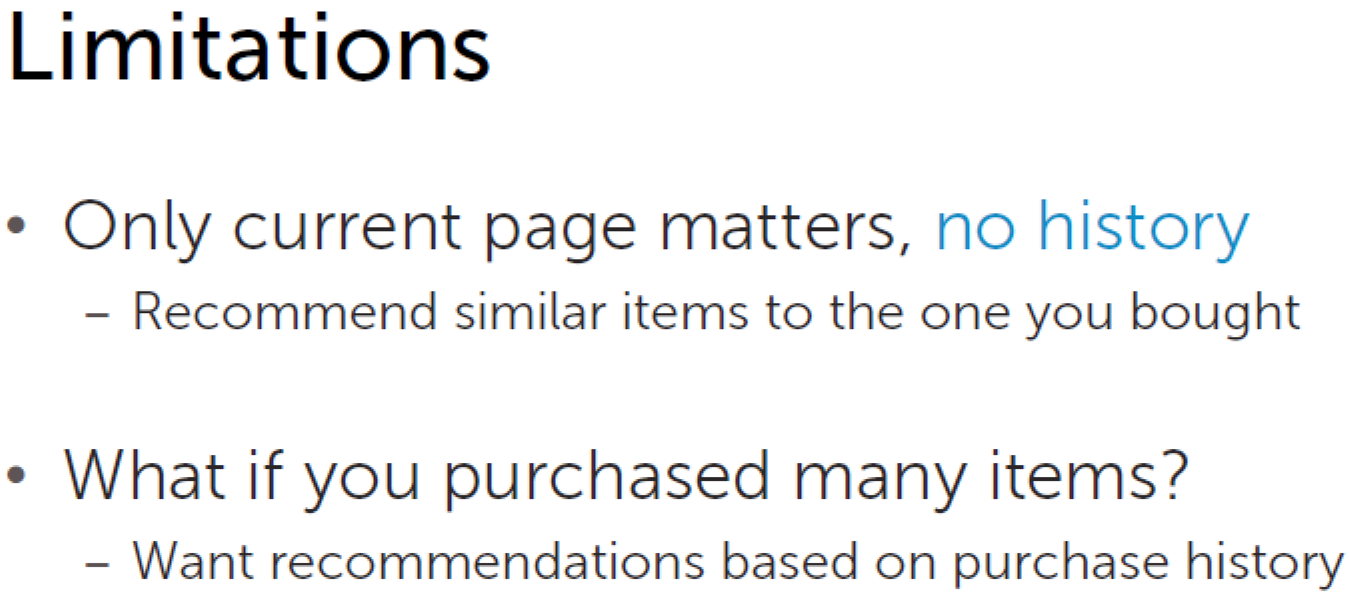
- Who purchased *i* and *j* *divided by* who purchased *i* or *j*

$$\frac{\#\ \text{purchased } i \text{ and } j}{\#\ \text{purchased } i\ \underline{\underline{\text{or}}}\ j}$$



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

Recommender system: co-occurence



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-occurence

(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?

Score( , *baby wipes*) = $\frac{1}{2}\left(S_{\text{baby wipes},\ \underline{\text{diapers}}} + S_{\text{baby wipes},\ \underline{\text{milk}}}\right)$

- Could also weight recent purchases more

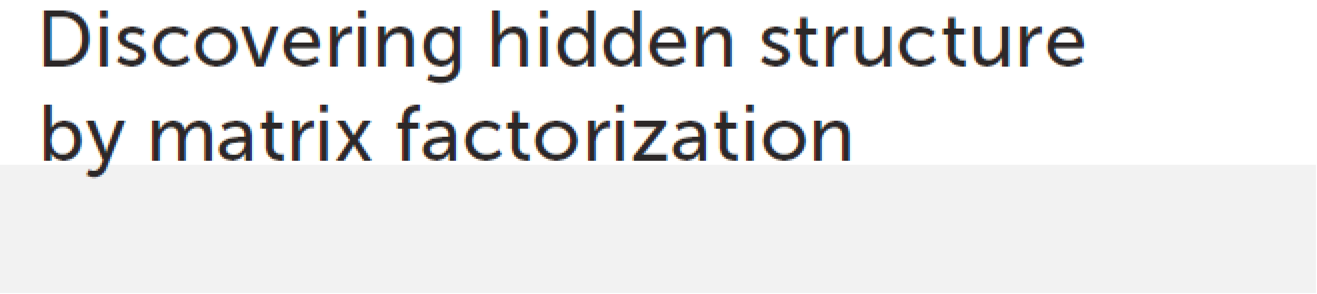
Sort *Score*( , *j*) and find item *j* with highest similarity

Recommender system: co-occurence

Limitations

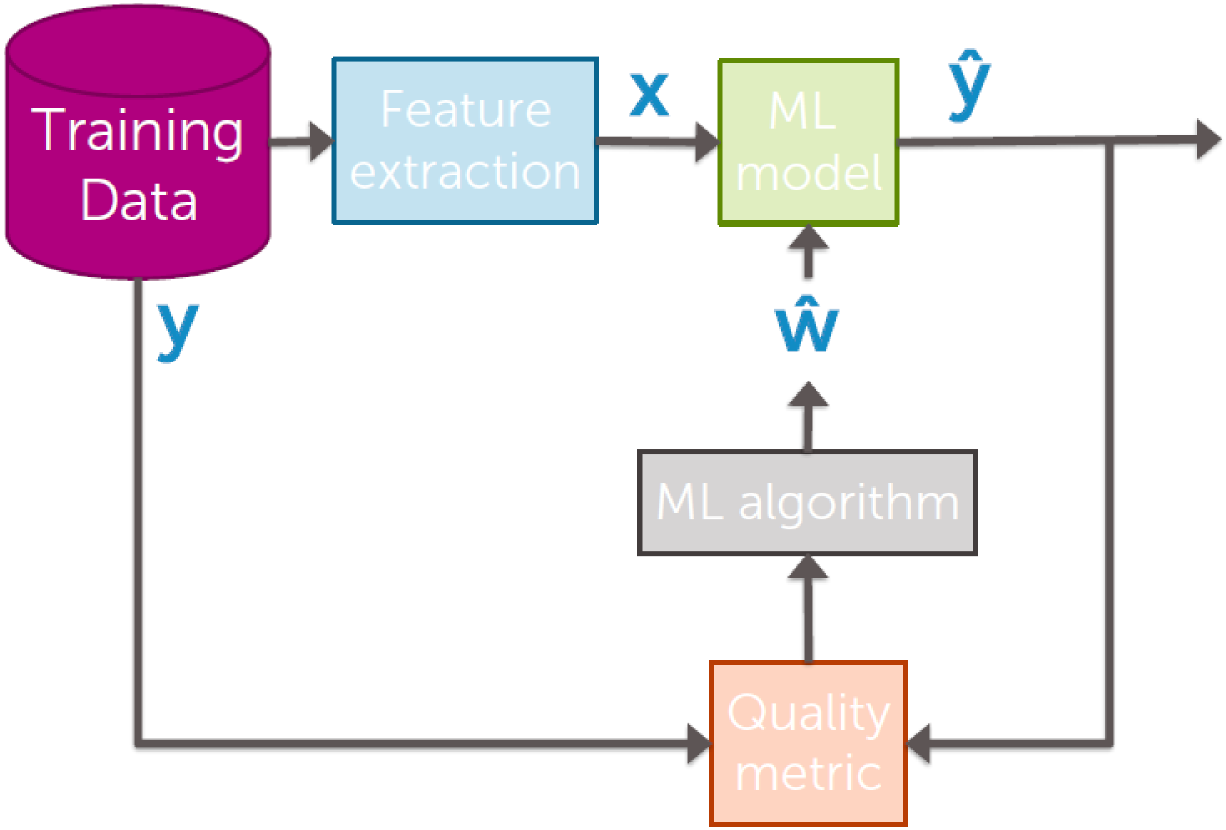
- Does **not** utilize:
  - context (e.g., time of day)
  - user features (e.g., age)
  - product features (e.g., baby vs. electronics)
- Scalability – similarity matrix M^2 size
- Cold start problem
  - What if a new user or product arrives?

Recommender system: matrix factorization



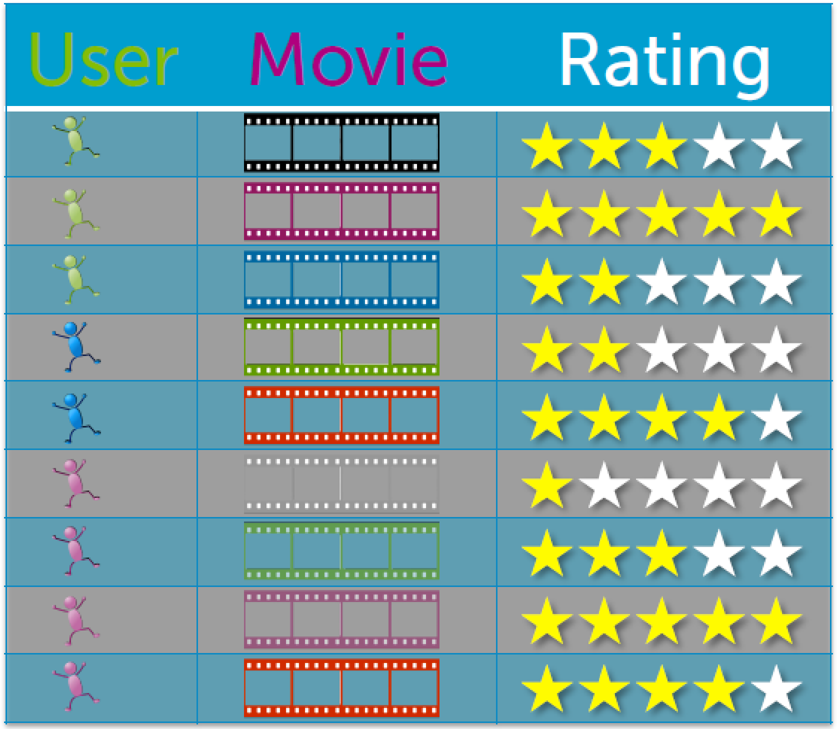
Discovering hidden structure by matrix factorization

Flow chart


Training Data
Feature extraction
x
ML model
ŷ
y
ŵ
ML algorithm
Quality metric

Training Data

- Users watch movies and rate them

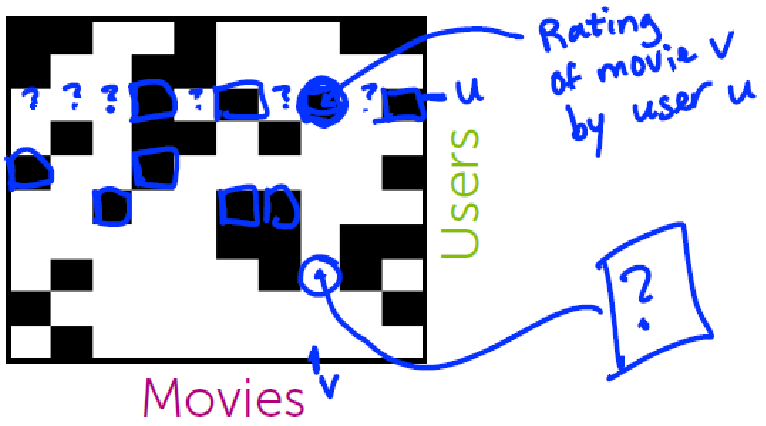


User	Movie	Rating
		★★★☆☆
		★★★★★
		★★☆☆☆
		★★☆☆☆
		★★★★☆
		★☆☆☆☆
		★★★☆☆
		★★★★★
		★★★★☆

Each user only watches a few of the available movies

Training Data: matrix completion

Rating =



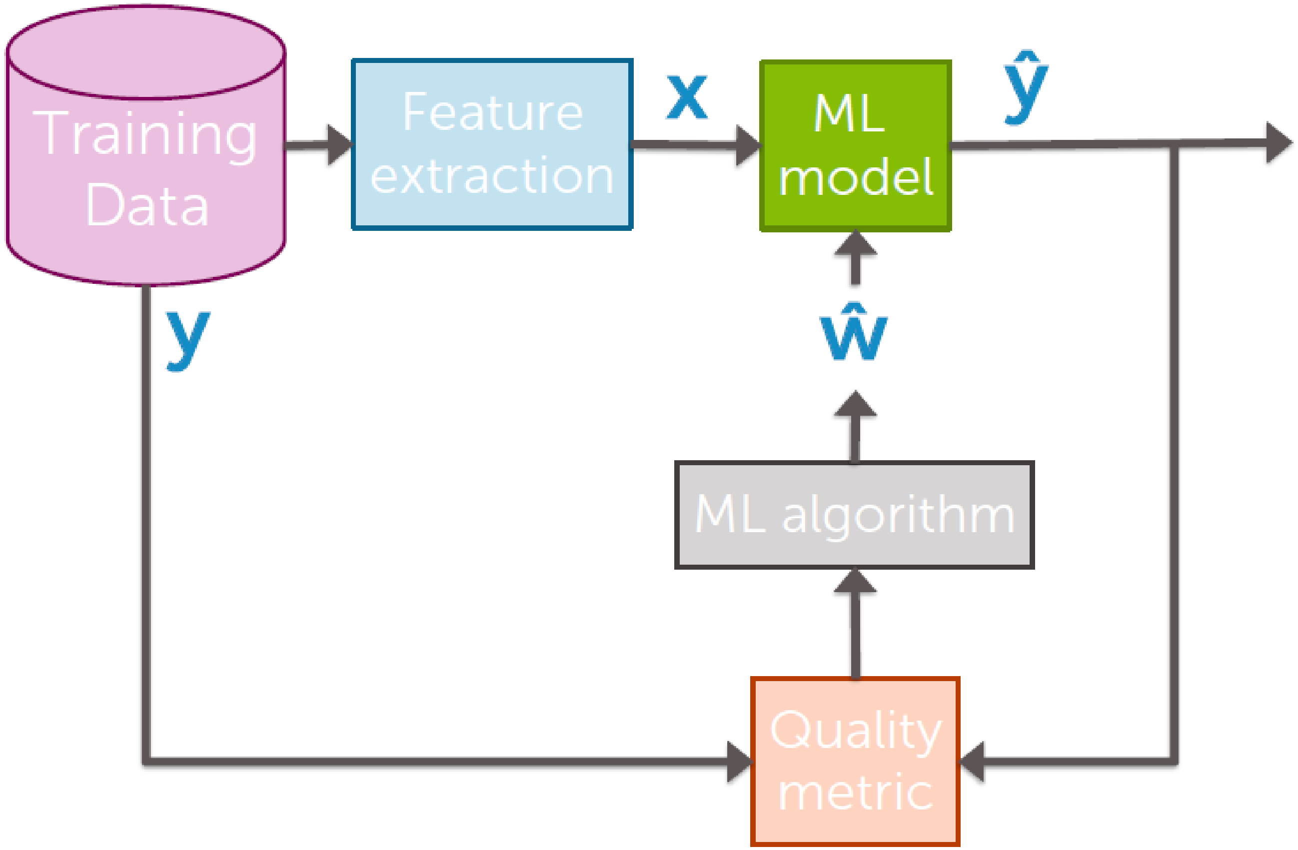
- **Data:** Users score some movies

  Rating(u,v) known for black cells

  Rating(u,v) unknown for white cells

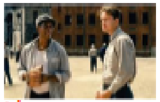
- **Goal:** Filling missing data?

Flow chart


Training Data
Feature extraction
x
ML model
ŷ
y
ŵ
ML algorithm
Quality metric

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,…

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

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

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

not 0,…,5 star

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

$$R_v = [0.3 \quad 0.01 \quad 1.5 \quad \ldots] \longrightarrow 0.3*2.5 + 0 + 1.5*0.8 + \ldots = \boxed{7.2}$$

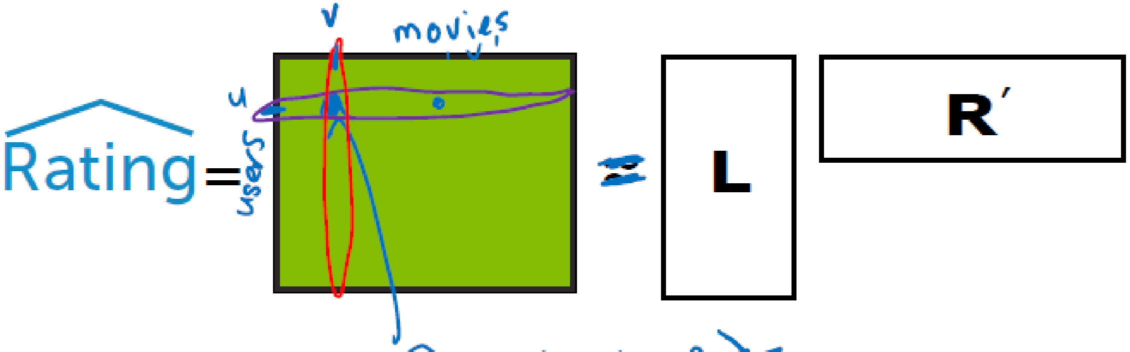
$$L_u = [2.5 \quad 0 \quad 0.8 \quad \ldots]$$

$$L_u' = [0 \quad 3.5 \quad 0.01 \quad \ldots] \longrightarrow 0 + 0.01*3.5 + 1.5*0.01 + \ldots \approx 0.8$$

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

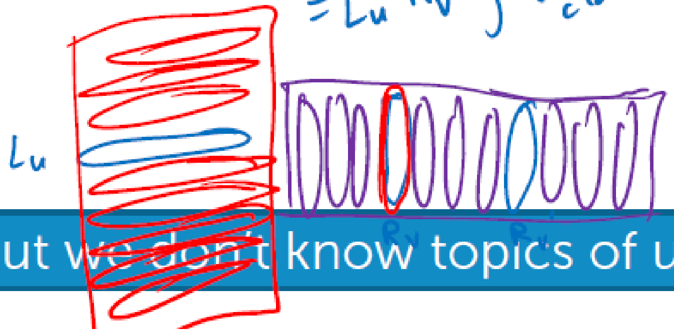
ML model

Predictions in matrix form



$$\widehat{Rating}(u,v) = \langle L_u, R_v \rangle = L_u \cdot R_v = L_u^T R_v$$

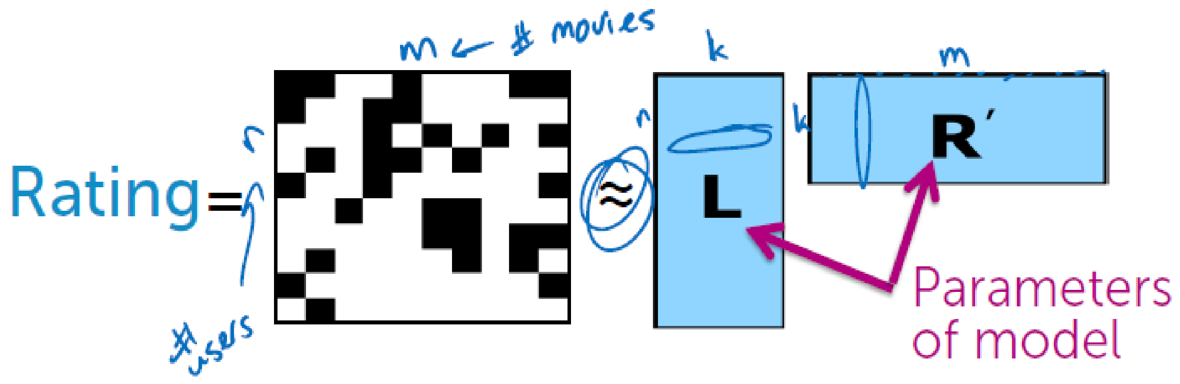
eg. notation for this class



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

Matrix factorisation model

Matrix factorization model: Discovering topics from data



\# params for model w/ k topics:

$$\underbrace{nk + k.m}_{\text{learn this using only black squares}} \ll \underbrace{n.m}_{\text{predict this}}$$

↑ full matrix

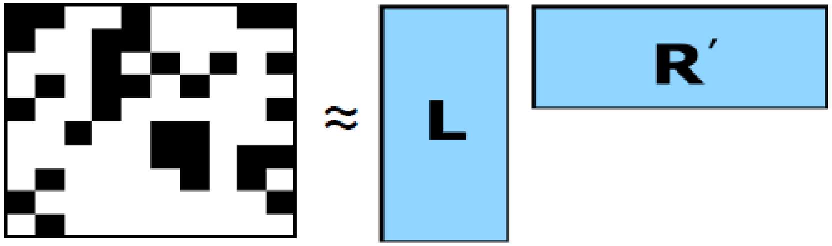
- Only use observed values to estimate "topic" vectors $\hat{L}_u$ and $\hat{R}_v$
- Use estimated $\hat{L}_u$ and $\hat{R}_v$ for recommendations

Many efficient algorithms for factorization

Matrix factorisation model

Is the problem well posed?

Can we uniquely identify the latent factors?



shorthand for Rating(u,v)

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

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

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

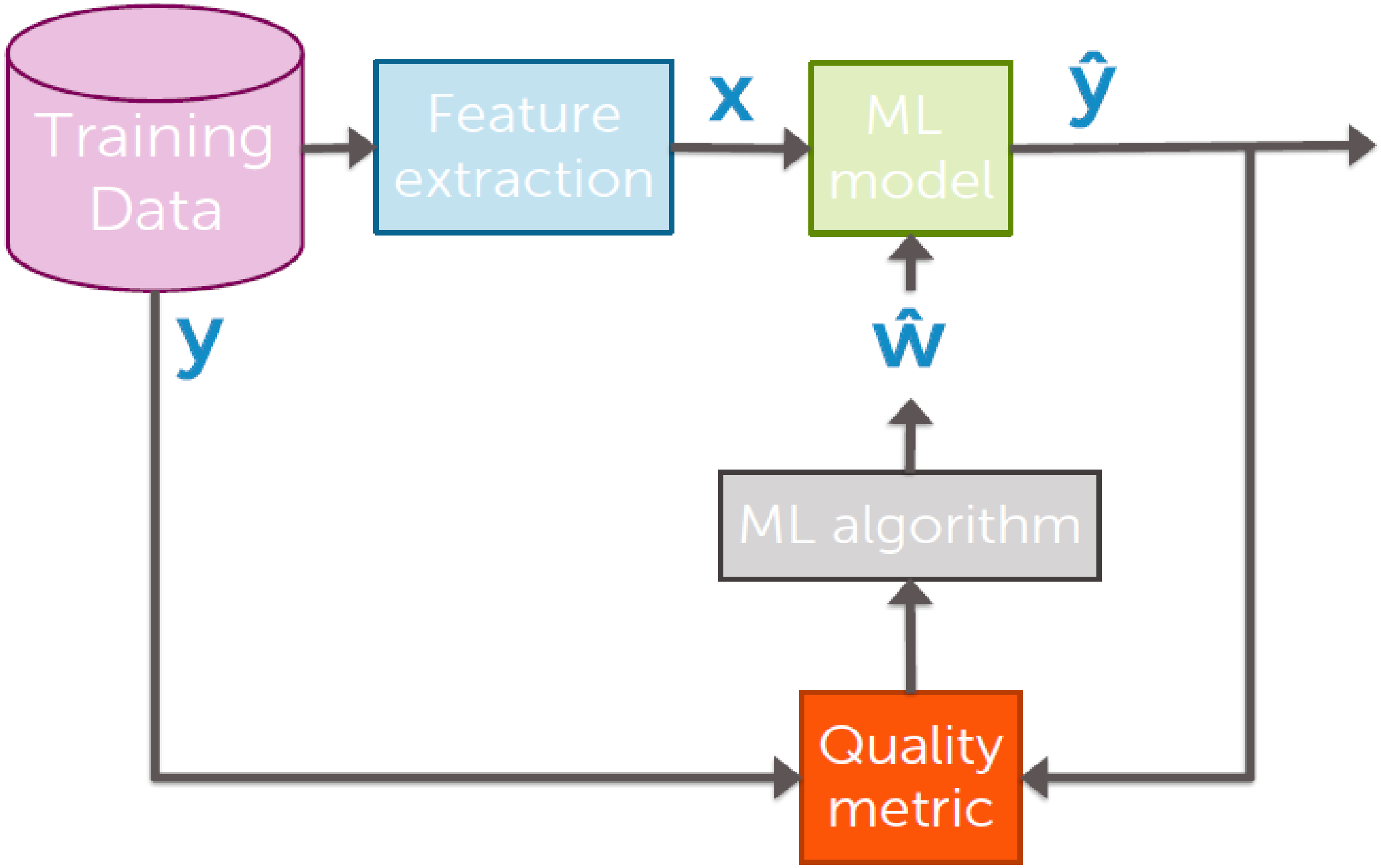
Then, $\tilde{L}_u \cdot \tilde{R}_v = c L_u \cdot \frac{1}{c} R_v = c \frac{1}{c} (L_u \cdot R_v) = L_u \cdot R_v \approx r_{uv}$

(other trans. have same effect... orthonormal trans)

Other (orthonormal) transformations can have the same effect.

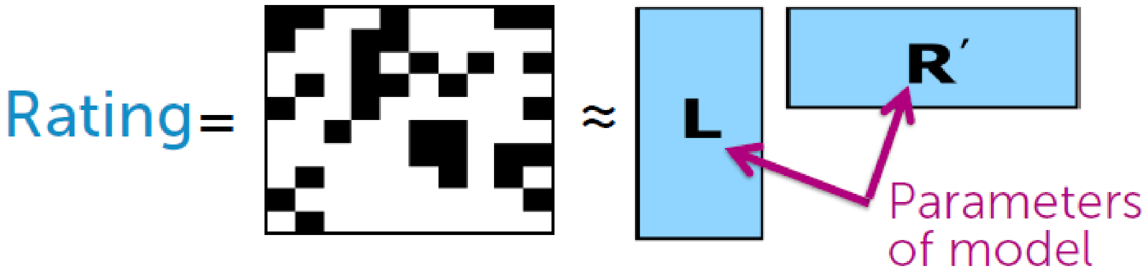
can't uniquely identify $L_u, R_v \Rightarrow$ don't interpret individually, only product

Flow chart


Training Data
Feature extraction
$\mathbf{x}$
ML model
$\hat{\mathbf{y}}$
$\mathbf{y}$
$\hat{\mathbf{w}}$
ML algorithm
Quality metric

Matrix factorisation model

Matrix factorization objective



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

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

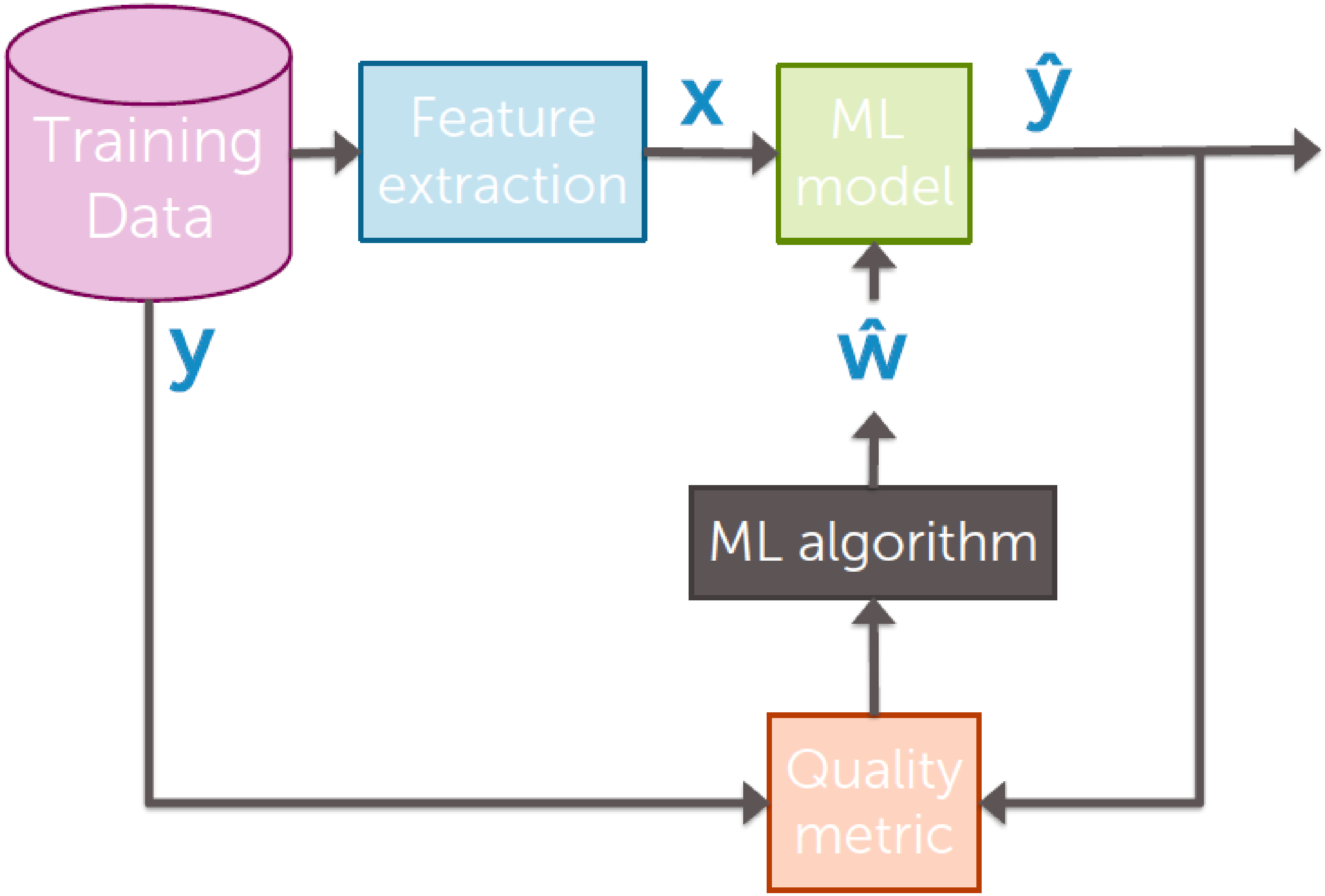
actual rating

only over obs. values

- Non-convex objective

subject to convergence to local mode

Flow chart


Training Data
Feature extraction
$\mathbf{x}$
ML model
$\hat{\mathbf{y}}$
$\mathbf{y}$
$\hat{\mathbf{w}}$
ML algorithm
Quality metric

ML algorithm

Coordinate descent

Goal: Minimize some function g $\min_{w} g(w)$

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

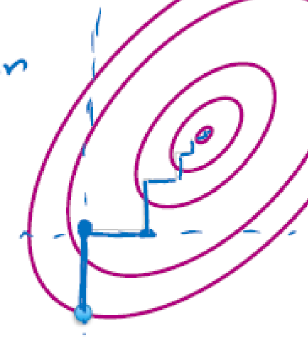
Often, hard to find minimum for all coordinates, but easy for each coordinate

Coordinate descent:

fixed from prev. iteration

```
Initialize $\hat{w}$ = 0 (or smartly...)
    while not converged
        pick a coordinate j
    $\hat{w}_j \leftarrow \min_{\omega} g(\hat{w}_0, \ldots, \hat{w}_{j-1}, \omega, \hat{w}_{j+1}, \ldots, \hat{w}_p)$
```

just min over ith 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:

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

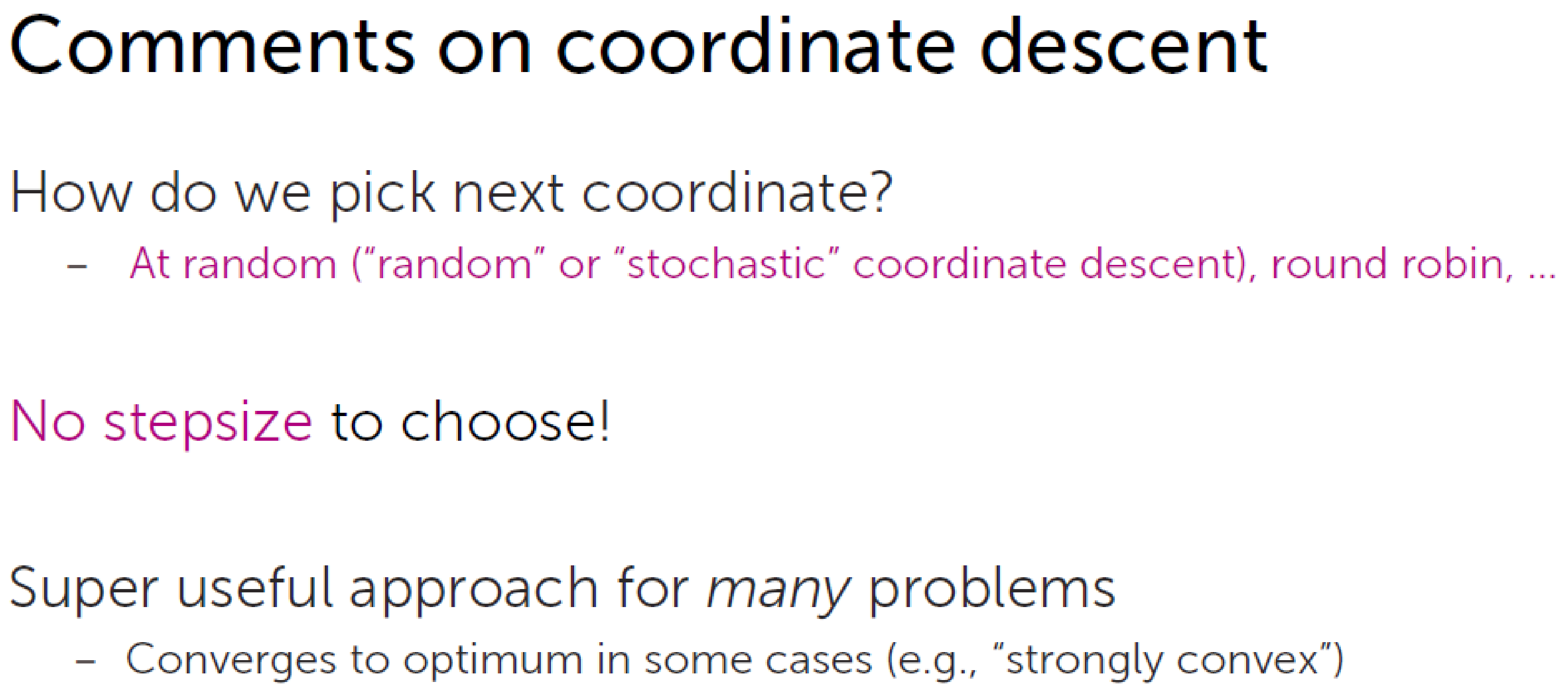
$V_u \triangleq$ set of movies user u has rated

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

← 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

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

← 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: $\min_{L_u} \sum_{v \in V_u} (\underset{\underset{\text{param vector}}{\uparrow}}{L_u} \cdot \underset{\underset{\text{fixed}}{=}}{R_v} - \underset{\nwarrow \text{ obs.}}{r_{uv}})^2$
- 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

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

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

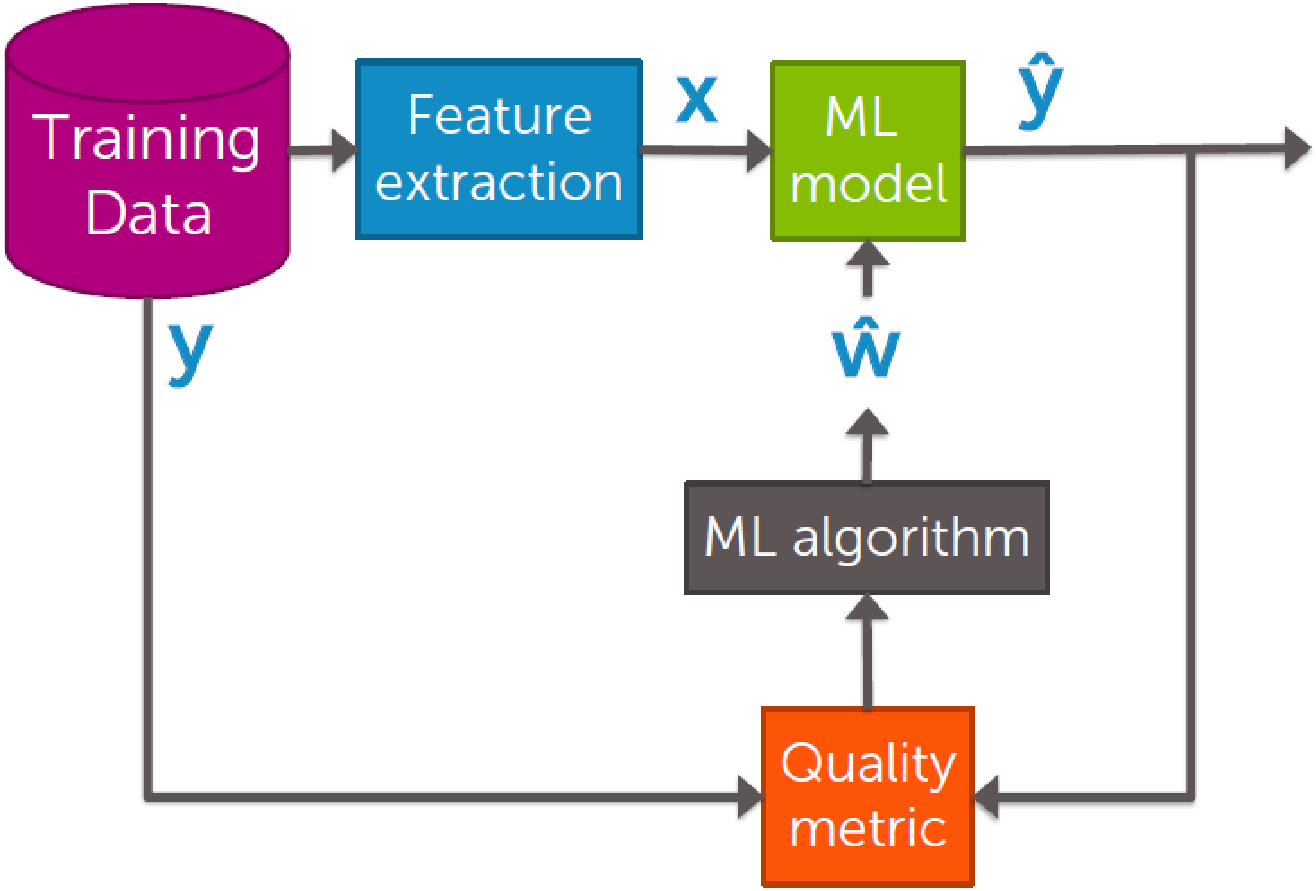
$$\min_{R_v} \sum_{u\in U_v} (L_u \cdot R_v - r_{uv})^2 \quad + \lambda \|R\|$$

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

L_2: $\sum_u \|L_u\|_2^2 = \|L\|_F^2$ → ridge

L_1: $\sum_u \|L_u\|_1$ → lasso

Flow chart


Training Data
Feature extraction
$\mathbf{x}$
ML model
$\hat{y}$
y
$\hat{w}$
ML algorithm
Quality metric

Recommendation

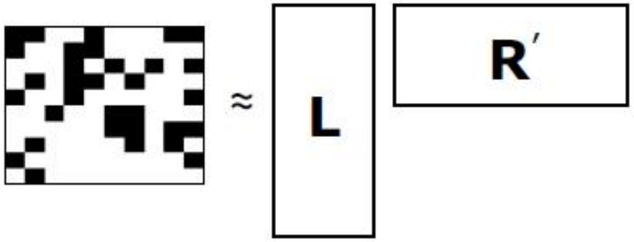
Using the results of matrix factorization

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

recommend movies v w/ highest score (u,v)

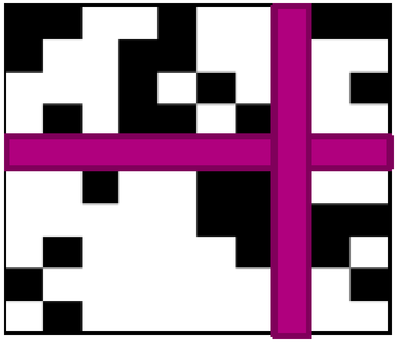
Example topics discovered from Wikipedia

Application to text data:



Limitations of matrix factorisation

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



Rating =

Cold-start problem more formally

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

- No previous observations

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

- Objective considered so far:

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

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

only term that appears

$L_{u'}$

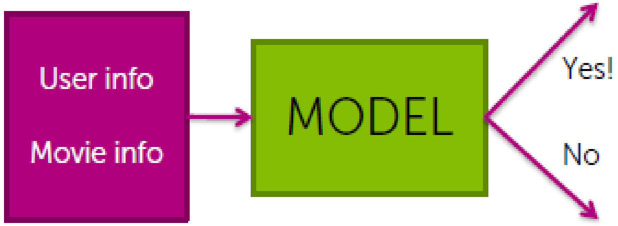
- Optimal user factor:

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

- Predicted user ratings:

always predict: $r_{u'v} = 0$ $\forall v$ … 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):

$$\phi(v) = (\overset{\text{genre}}{\text{'action'}}, \overset{\text{year}}{1994}, \overset{\text{director}}{\text{Tarantino}}, \overset{\ldots}{\ldots})$$

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

$w =$ vector of same length

- Fit linear model:

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

- Minimize:

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

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

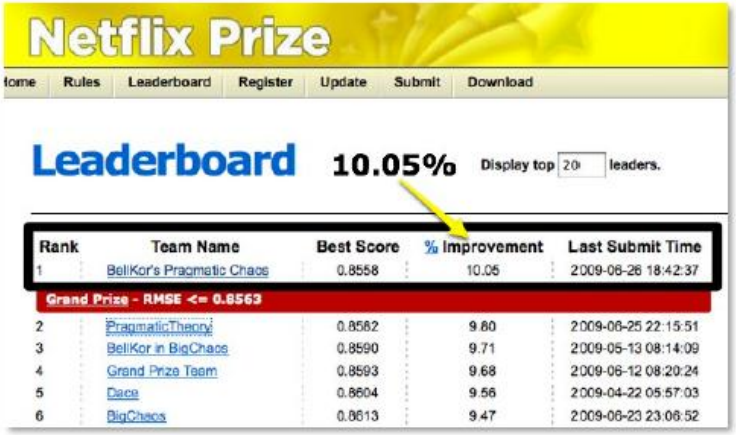
A unified model:

$$r_{uv} \approx L_u \cdot R_v + (w + w_u) \cdot \phi(u,v)$$

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



Leaderboard 10.05%

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	Dace	0.8604	9.56	2009-04-22 05:57:03
6	BigChaos	0.8613	9.47	2009-06-23 23:06:52

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

Recommending system: how effective?

How many liked items were recommended?

Recall

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

$$\boxed{= \frac{3}{5}}$$

Recommending system: how effective?

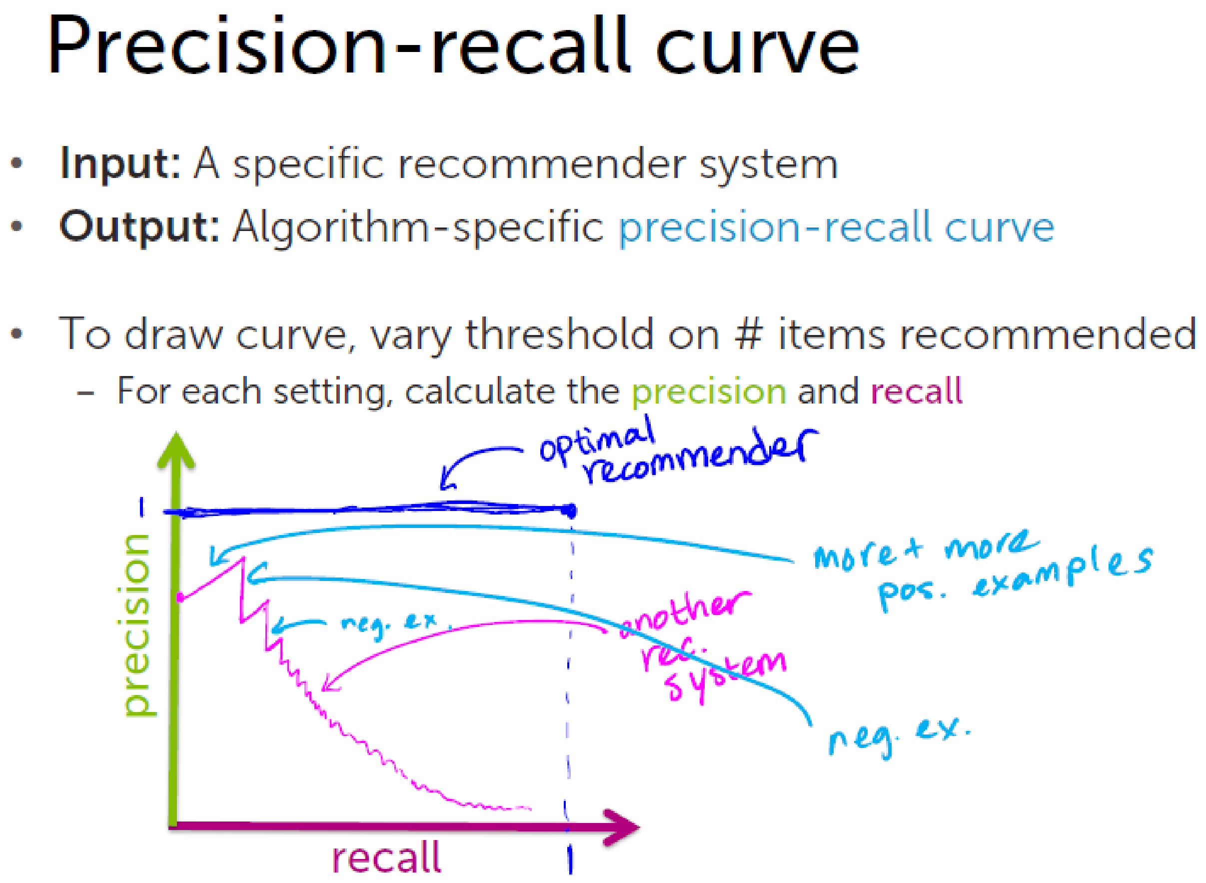
How many recommended items were liked?

Precision

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

$$\boxed{= \frac{3}{11}}$$

Recommending system: how effective?



Precision-recall curve

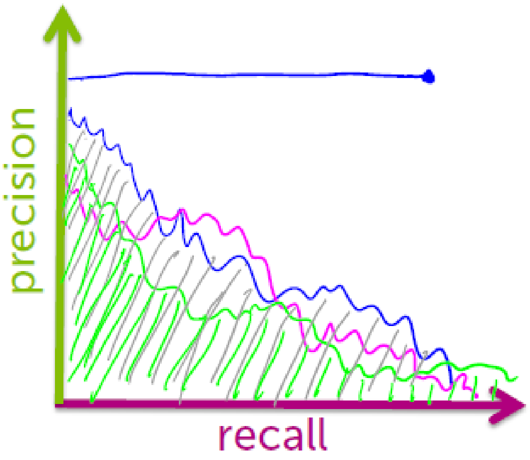
- **Input:** A specific recommender system
- **Output:** Algorithm-specific precision-recall curve

- 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