Citation Key

for more information see: http://open.umich.edu/wiki/CitationPolicy

Use + Share + Adapt

{ Content the copyright holder, author, or law permits you to use, share and adapt. }

PD-GOV **Public Domain – Government**: Works that are produced by the U.S. Government. (USC 17 § 105)

PD-EXP **Public Domain – Expired**: Works that are no longer protected due to an expired copyright term.

PD-SELF **Public Domain – Self Dedicated**: Works that a copyright holder has dedicated to the public domain.

CC ZERO **Creative Commons – Zero Waiver**

CC BY **Creative Commons – Attribution License**

CC BY-SA **Creative Commons – Attribution Share Alike License**

CC BY-NC **Creative Commons – Attribution Noncommercial License**

CC BY-NC-SA **Creative Commons – Attribution Noncommercial Share Alike License**

GNU-FDL **GNU – Free Documentation License**

Make Your Own Assessment

{ Content Open.Michigan believes can be used, shared, and adapted because it is ineligible for copyright. }

PD-INEL **Public Domain – Ineligible**: Works that are ineligible for copyright protection in the U.S. (USC 17 § 102(b)) *laws in your jurisdiction may differ

{ Content Open.Michigan has used under a Fair Use determination. }

FAIR USE **Fair Use**: Use of works that is determined to be Fair consistent with the U.S. Copyright Act. (USC 17 § 107) *laws in your jurisdiction may differ

Our determination **DOES NOT** mean that all uses of this 3rd-party content are Fair Uses and we **DO NOT** guarantee that your use of the content is Fair.

To use this content you should **do your own independent analysis** to determine whether or not your use will be Fair.

Lecture 9: Page Rank; Singular Value Decomposition

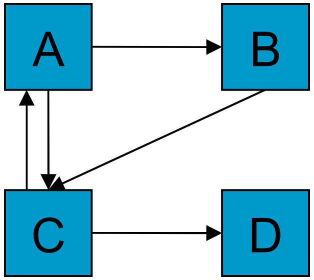
SI583: Recommender Systems

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

Recap: PageRank

- Google's big original idea [Brin &Page, 1998]
- Idea: ranking is based on "random web surfer":
  - start from any page at random
  - pick a random link from the page, and follow it
  - repeat!
  - ultimately, this process will converge to a <u>stable distribution</u> over pages (with some tricks...)
  - most likely page in this stable distribution is ranked highest
- Strong points:
  - Pages linked to by many pages *tend* to be ranked higher (not always)
  - A link ("vote") from a highly-ranked page carries more weight
  - Relatively hard to manipulate

Some Intuitions



- Will D's Rank be more or less than ¼?
- Will C's Rank be more or less than B's?
- How will A's Rank compare to D's?

Third Iteration

- AR+E

r1 .2879845
r2 .21046512
r3 .39263566
r4 .2879845

- Normalized (divide by 1.18)

r1 .24424721
r2 .17850099
r3 .3330046
r4 .24424721

Personalized PageRank

- Pick E to be some sites that I like
  - My bookmarks
  - Links from my home page
- Rank flows more from these initial links than from other pages
  - But much of it may still flow to the popular sites, and from them to others that are not part of my initial set

Other applications for pagerank?

Another method: Singular Value Decomposition (SVD)

- Back to product recommendation setting
- SVD-based collaborative filtering often used in place of User-user / Item-Item

- Two different advantages:
  – Accuracy benefits: identifies “latent features” of items that are useful for predictions
  – Scalability: Easier to compute when ratings are sparse

- Related terms: Principal Component Analysis, Latent Semantic Indexing,

Motivating SVD

- Consider the following scenario
  - Joe rates items A,B,C,D; likes AC, dislikes BD
  - Sue rates items C,D,E,F; likes CE, dislikes DF
  - John rates items E,F,G,H; likes EG, dislikes FH
- Will Joe like item G?

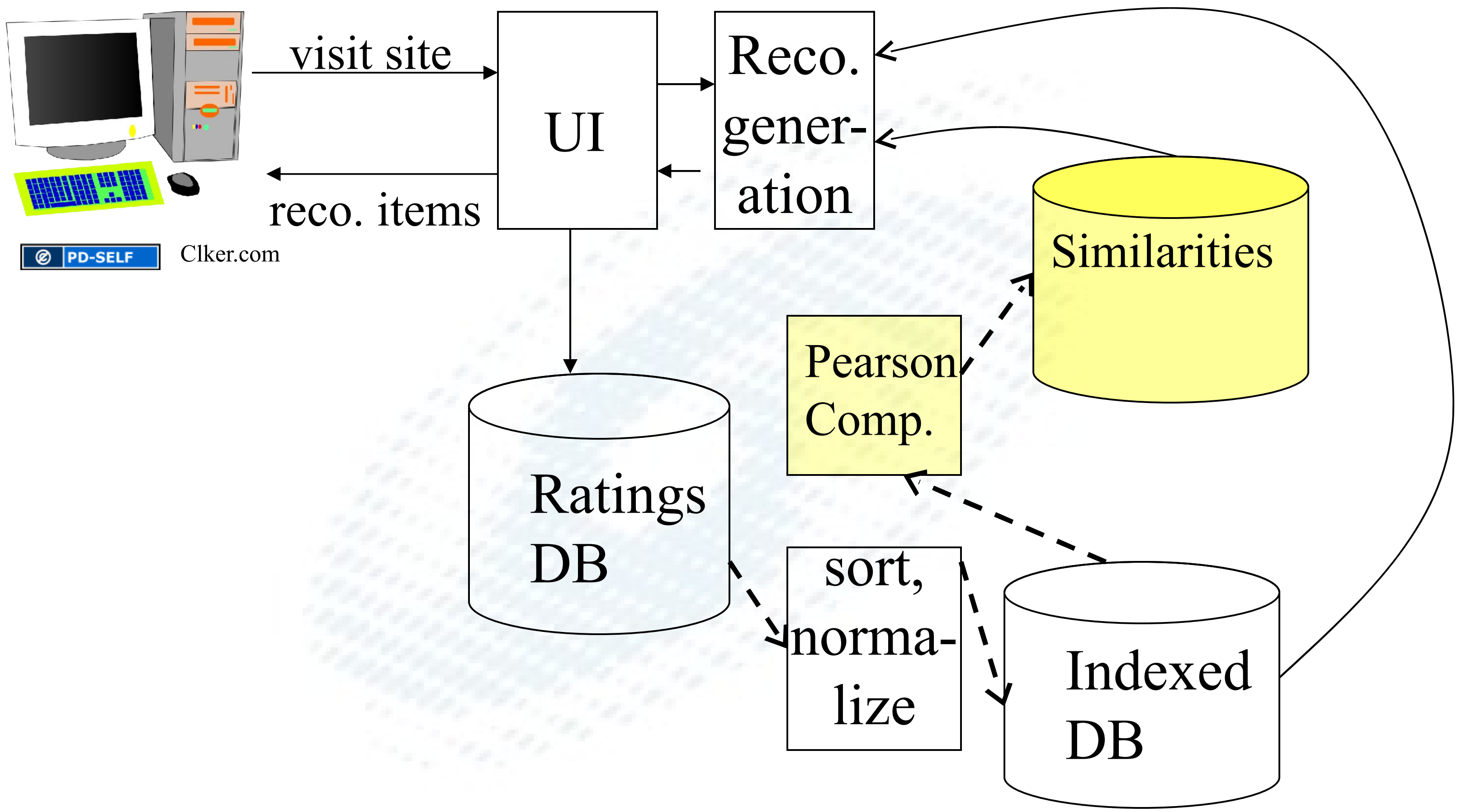
Motivating SVD

- Consider the following scenario
  - Joe rates items A,B,C,D; likes AC, dislikes BD
  - Sue rates items C,D,E,F; likes CE, dislikes DF
  - John rates items E,F,G,H; likes EG, dislikes FH
- Will Joe like item G?
  - user-user fails because Joe, John have no common ratings
  - item-item fails
  - intuitively, can argue that Joe is likely to like G..
- Idea: Capture the intuition in a CF algorithm

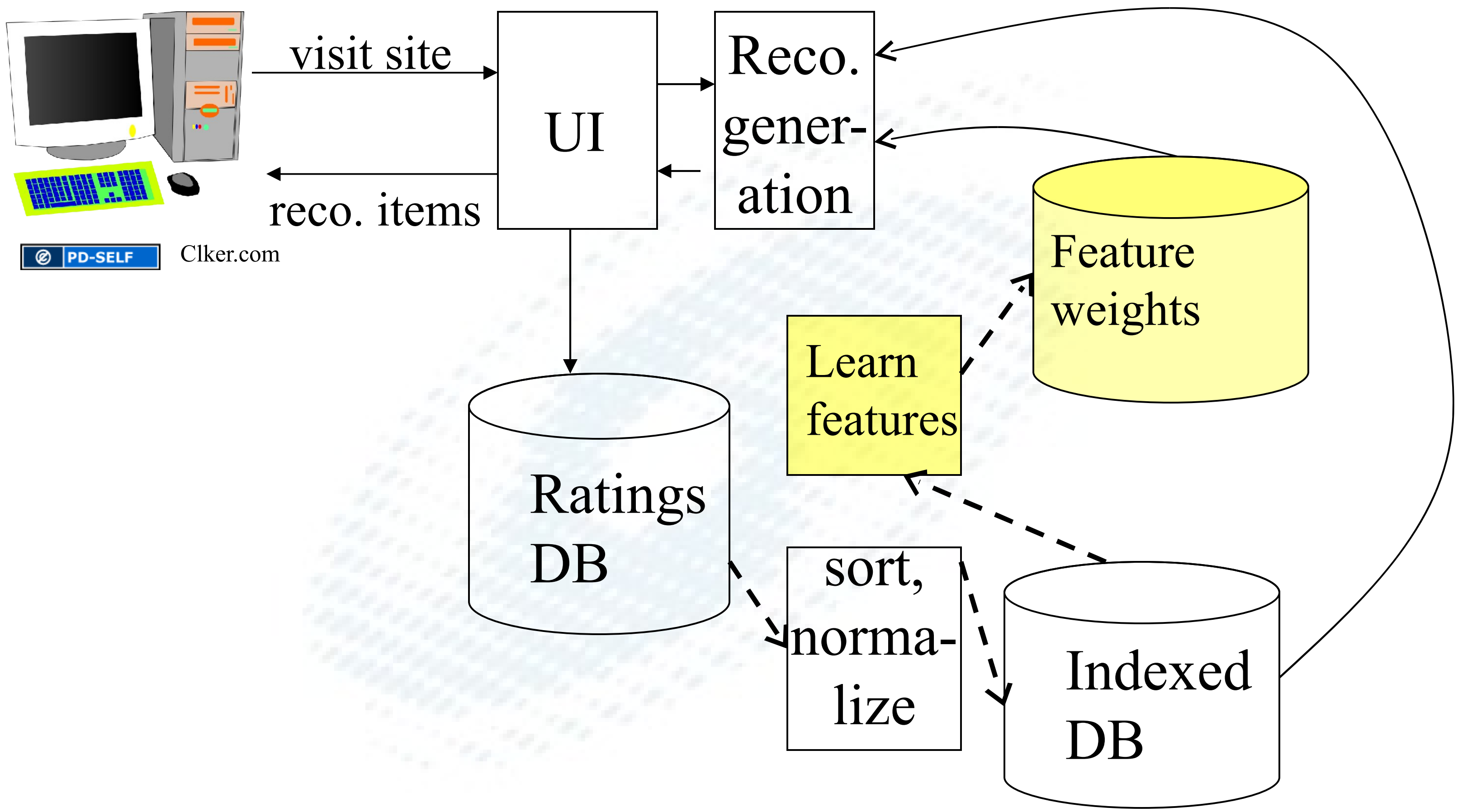
Motivating SVD..

- One intuitive explanation for why Joe might like G:
  - A,C,E,G have some common “feature”, which is why users who like one like the others
  - e.g., ACEG may be funny movies; Joe, Sue, John all like funny movies
- Generalize this idea to multiple features
- Important features have to be automatically discovered from ratings
  - or a hybrid of content and collab. filtering

Software modules: User-User

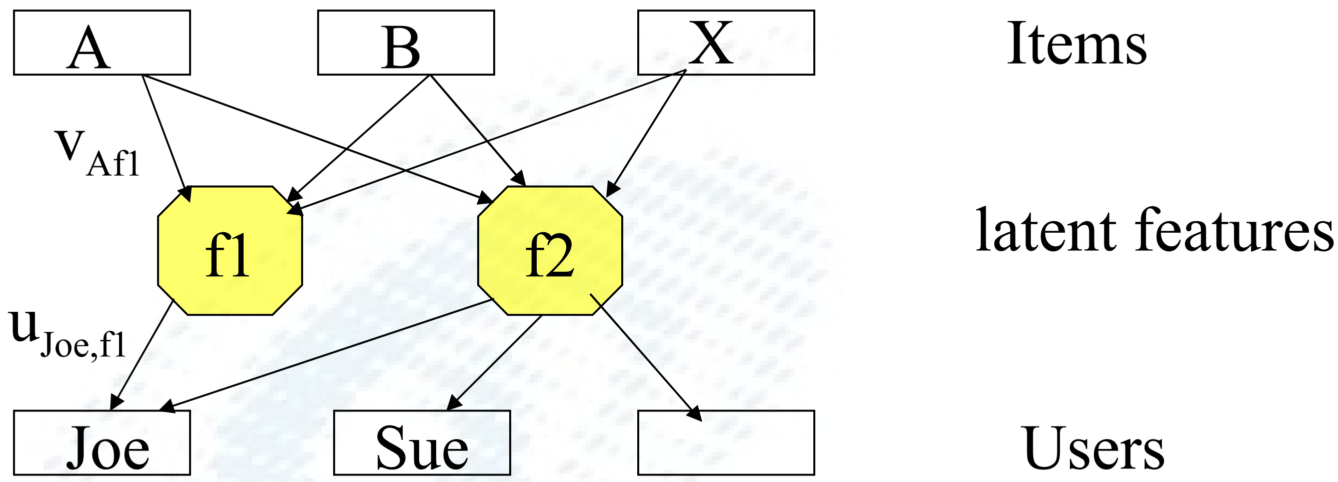


Software modules



SVD Conceptual Model

- Fit previous data to a model with k features:



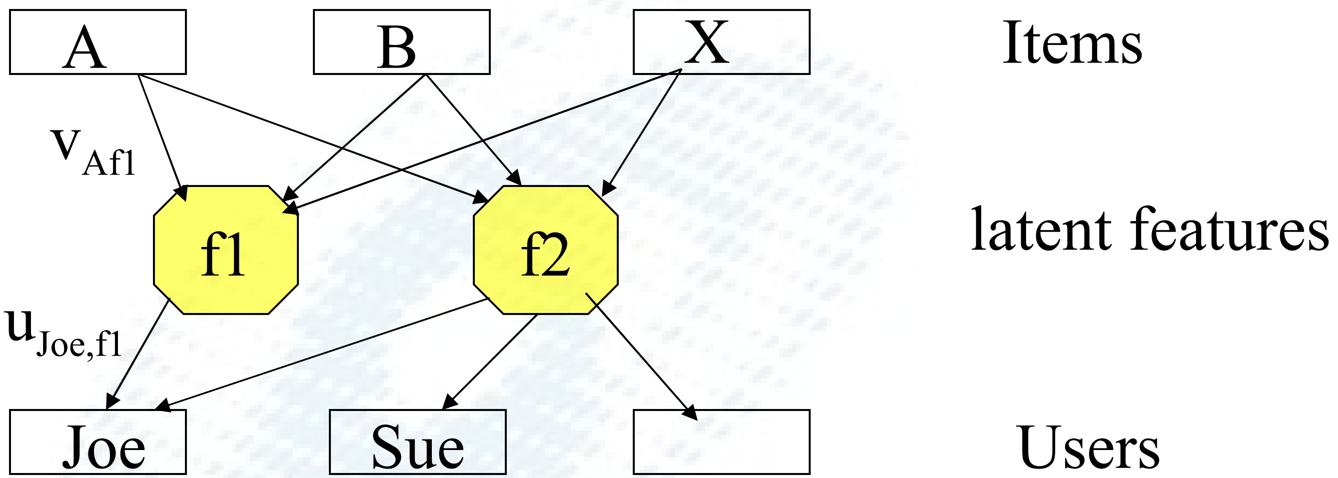
- Weights v_{Af1}, etc. indicate extent to which A has feature f1,f2
- Weights $u_{Joe,f1}$ etc. indicate extent to which Joe likes featues f1,f2
- Predict Joe's preference for X from fitted weights

Learning the weights: SVD

- start with mean-normalized rating matrix ***X***
- SVD decomposition: calculate U,S,V such that
  - U: $m \times k$, S: $k \times k$, V: $k \times n$
  - ***X = USV***
  - S is a diagonal matrix (zero on non-diag)
  - U,V are “orthogonal” => features are independent
- S indicates “intensity” of each feature
  - S_{ii}: singular value of feature i

Fitting the weights: SVD

- Model weights from SVD (U,S,V):



- Weight (item j, feature f) = $\sqrt{s_{ff}}\, V_{fj}$
- Weight (user i, feature f) = $\sqrt{s_{ff}}\, U_{if}$

Alternative: get software package to calculate weights directly..

SVD: selecting features

- More features => better fit possible
  - but also more noise in weights
  - and harder to compute (matrices are larger)
- In practice, do best fit with a small number of features (10,say)
- Which features are picked?

SVD: selecting features

- More features => better fit possible
  - but also more noise in weights
  - and harder to compute
- In practice, do best fit with a small number of features (10,say)
- Which features are picked?
  - Those with the highest singular value (intensity)
  - Small singular value => feature has negligible effect on predictions

SVD-based CF: Summary

- Pick a number of features *k*
- Normalize ratings
- Use SVD to find best fit with *k* features
- Use fitted model to predict value of Joe's normalized rating for item X
- Denormalize (add Joe's mean) to predict Joe's rating for X

SVD Practicalities

- SVD is a common mathematical operation; numerous libraries exist
- Efficient algorithms to compute SVD for the typical case of sparse ratings
- A fast, simple implementation of an SVD-based recommender (by Simon Funk/Brandyn Webb) was shown to do very well on the Netflix challenge

SVD and Content Filtering

- Similar idea: Latent Semantic Indexing used in content-filtering
  – Fit item descriptions and keywords by a set of features
  – Related words map onto the same feature
  – Similar items have the similar feature vectors

- Useful to combine content+collaborative filtering
  – Learn some features from content, some from ratings