Caltech

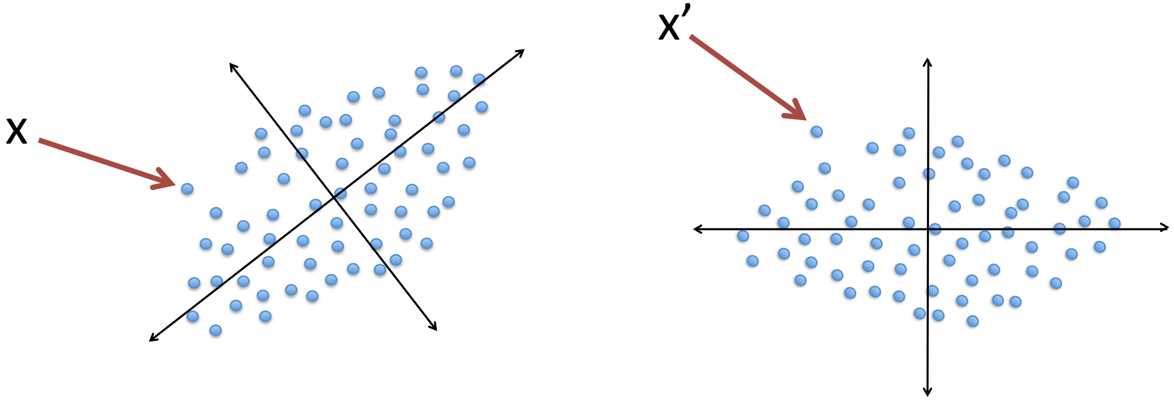
Machine Learning & Data Mining

CS/CNS/EE 155

Lecture 13:
Latent Factor Models &
Non-Negative Matrix Factorization

Recap: Orthogonal Matrix

- A matrix U is orthogonal if $UU^T = U^TU = I$
  - For any column u: $u^Tu = 1$
  - For any two columns u, u': $u^Tu' = 0$
  - U is a rotation matrix, and U^T is the inverse rotation
  - If $x' = U^Tx$, then $x = Ux'$



Recap: Orthogonal Matrix

- Any subset of columns of U defines a subspace

$$x' = U_{1:K}^T x$$

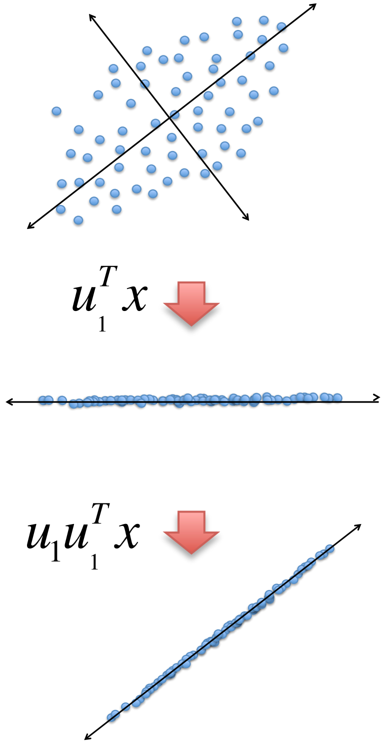
Transform into new coordinates

Treat $U_{1:K}$ as new axes

$$proj_{U_{1:K}}(x) = U_{1:K} U_{1:K}^T x$$

Project x onto $U_{1:K}$ in original space

“Low Rank” Subspace



Recap: Singular Value Decomposition

$$X = \left[x_1, \ldots, x_N \right] \in \mathrm{Re}^{D \times N}$$

$$X = U \Sigma V^T$$

Orthogonal

Orthogonal Diagonal

SVD

$$\sum_{i=1}^{N} \left\| x_i - U_{1:K} U_{1:K}^T x_i \right\|^2$$

“Residual”

$U_{1:K}$ is the K-dim subspace with smallest residual

Recap: SVD & PCA

$$XX^T = \underset{\text{Orthogonal}}{U}\,\underset{\text{Diagonal}}{\Lambda}\,U^T$$

PCA

$$X = \underset{\text{Orthogonal}}{U}\,\underset{\text{Diagonal}}{\Sigma}\,\underset{\text{Orthogonal}}{V^T}$$

SVD

$$XX^T = \left(U\Sigma V^T\right)\left(U\Sigma V^T\right)^T = U\Sigma V^T V\Sigma U^T = U\Sigma^2 U^T$$

Matrix Norms

- Frobenius Norm

$$\|X\|_{Fro} = \sqrt{\sum_{ij} X_{ij}^2} = \sqrt{\sum_d \sigma_d^2}$$

- Trace Norm

$$\|X\|_* = \sum_d \sigma_d = \mathrm{trace}\left(\sqrt{X^T X}\right)$$

$$X = U\Sigma V^T$$

Each σ_d is guaranteed to be non-negative
By convention: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_D \geq 0$

$$\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_D \end{bmatrix}$$

Properties of Matrix Norms

$$\|X\|_* = \text{trace}\left(\sqrt{\left(U\Sigma V^T\right)^T U\Sigma V^T}\right) = \text{trace}\left(\sqrt{V\Sigma U^T U\Sigma V^T}\right)$$

$$= \text{trace}\left(\sqrt{V\Sigma\Sigma V^T}\right) = \text{trace}\left(\sqrt{V\Sigma^2 V^T}\right) = \text{trace}\left(V\Sigma V^T\right)$$

$$= \text{trace}\left(\Sigma V^T V\right) = \text{trace}\left(\Sigma\right) = \sum_d \sigma_d$$

$$X = U\Sigma V^T$$

Each σ_d is guaranteed to be non-negative
By convention: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_D \geq 0$

$$\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_D \end{bmatrix}$$

$$\text{trace}(ABC) = \text{trace}(BCA) = \text{trace}(CAB)$$

Properties of Matrix Norms

$$\|X\|_{Fro}^2 = \text{trace}\left(X^T X\right) = \text{trace}\left(\left(U\Sigma V^T\right)^T U\Sigma V^T\right)$$

$$= \text{trace}\left(V\Sigma^2 V^T\right) = \text{trace}\left(\Sigma^2 V^T V\right)$$

$$= \text{trace}\left(\Sigma^2\right) = \sum_d \sigma_d^2$$

$$X = U\Sigma V^T$$

Each σ_d is guaranteed to be non-negative
By convention: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_D \geq 0$

$$\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_D \end{bmatrix}$$

$$\text{trace}(ABC) = \text{trace}(BCA) = \text{trace}(CAB)$$

Other Useful Properties

- Cauchy Schwarz:

$$\langle A,B\rangle^2 = \mathrm{trace}(A^T B)^2 \leq \langle A,A\rangle\langle B,B\rangle = \mathrm{trace}(A^T A)\mathrm{trace}(B^T B) = \|A\|_F^2 \|B\|_F^2$$

- AM-GM Inequality:

$$\|A\|\|B\| = \sqrt{\|A\|^2 \|B\|^2} \leq \frac{1}{2}\left(\|A\|^2 + \|B\|^2\right)$$

True for any norm

- Orthogonal Transformation Invariance of Norms:

$$\|UA\|_F = \|A\|_F \qquad \|UA\|_* = \|A\|_*$$

If U is a full-rank orthogonal matrix

- Trace Norm of Diagonals

$$\|A\|_* = \sum_i |A_{ii}|$$

If A is a square diagonal matrix

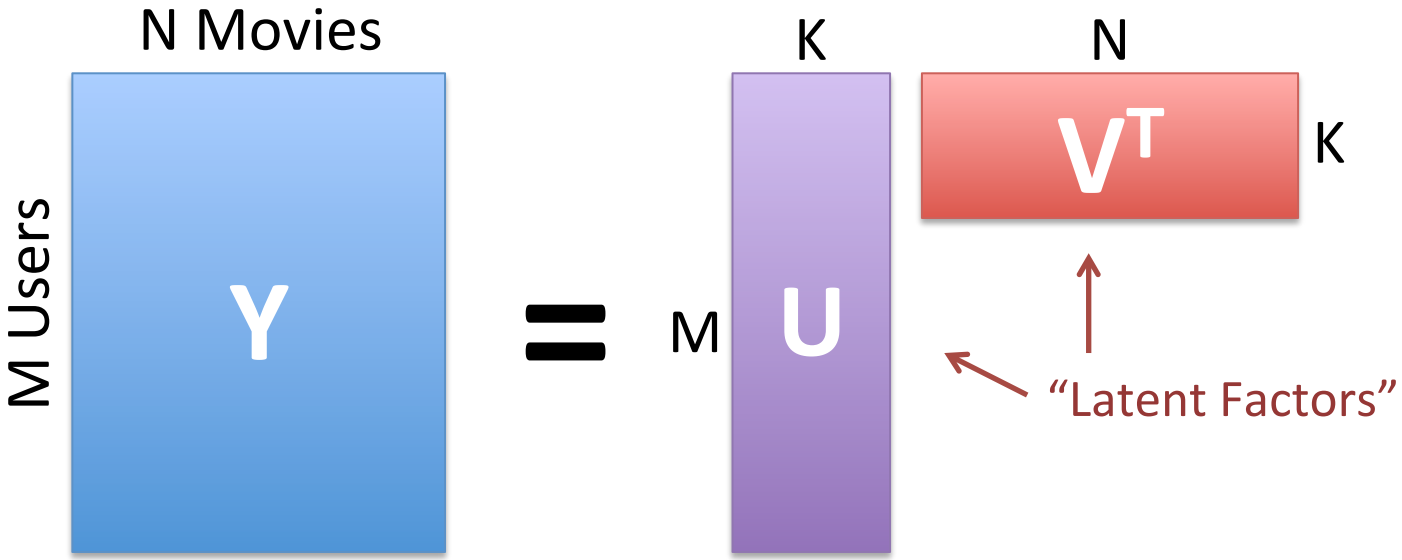
Recap: SVD & PCA

- SVD: $X = U\Sigma V^T$
- PCA: $XX^T = U\Sigma^2 U^2$
- The first K columns of U are the best rank-K subspace that minimizes the Frobenius norm residual:

$$\left\| X - U_{1:K} U_{1:K}^T X \right\|_{Fro}^2$$

Latent Factor Models

Netflix Problem



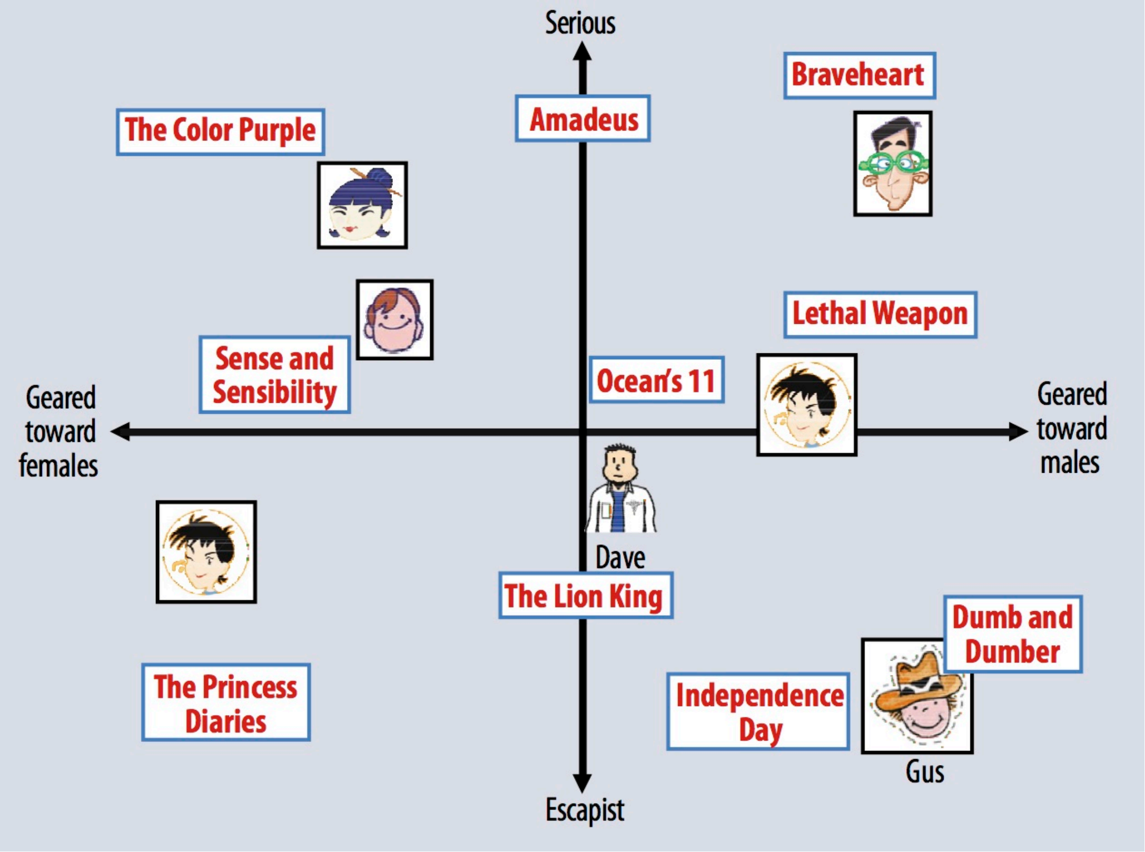
"Latent Factors"

- Y_{ij} = rating user i gives to movie j

$$y_{ij} \approx u_i^T v_j$$

- Solve using SVD!

Example

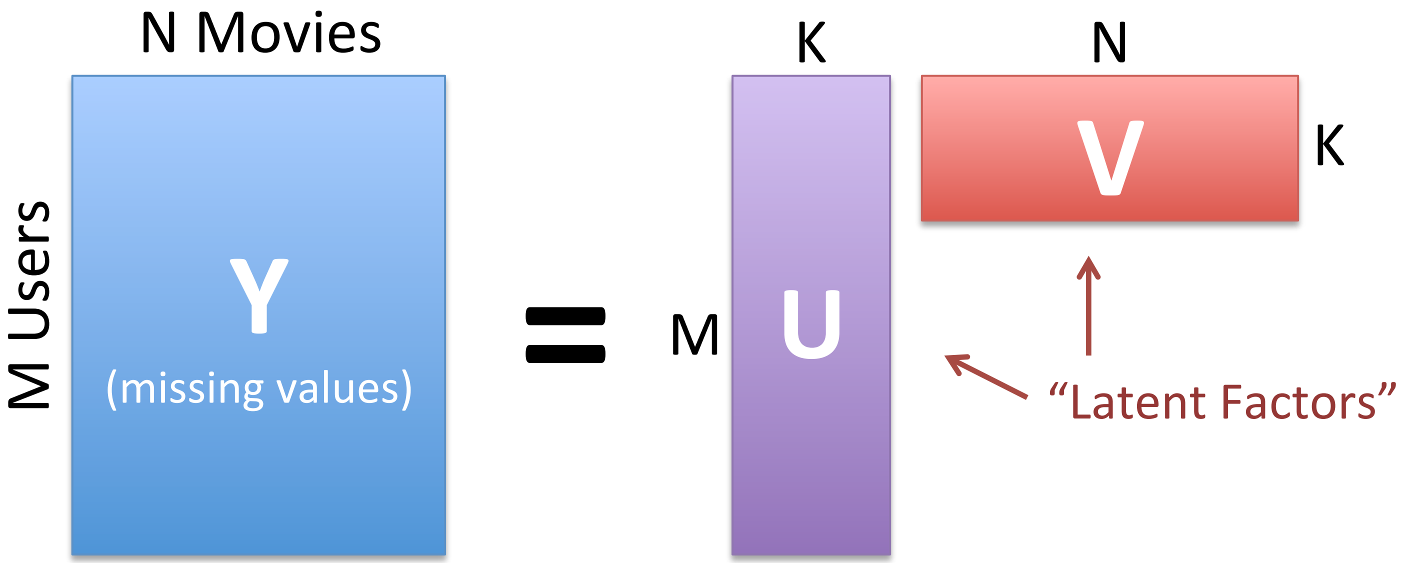


$$y_{ij} \approx u_i^T v_j$$

Miniproject 2: create your own.

http://www2.research.att.com/~volinsky/papers/ieeecomputer.pdf

Actual Netflix Problem



- Many missing values!

Collaborative Filtering

- M Users, N Items
- Small subset of user/item pairs have ratings
- Most are missing

- Applicable to any user/item rating problem
  - Amazon, Pandora, etc.

- **Goal:** Predict the missing values.

Latent Factor Formulation

- Only labels, no features
  - Labels are "structured"

$$S = \left\{ y_{ij} \right\}$$

- Learn a **latent** representation over users U and movies V such that:

$$\underset{U,V}{\text{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2 \right) + \sum_{ij} \left(y_{ij} - u_i^T v_j \right)^2$$

Connection to Trace Norm

- Suppose we consider all U,V that achieve perfect reconstruction: $Y=UV^T$

- Find U,V with lowest complexity:

$$\underset{Y=UV^T}{\operatorname{argmin}} \frac{1}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right)$$

- Complexity equivalent to trace norm:

$$\|Y\|_* = \min_{Y=UV^T} \frac{1}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right)$$

Proof (One Direction)

We will prove: $\|Y\|_* \geq \min_{Y=AB^T} \frac{1}{2}\left(\|A\|_{Fro}^2 + \|B\|_{Fro}^2\right)$

$Y = U\Sigma V^T$

SVD

Choose: $A = U\sqrt{\Sigma}, \qquad B = V\sqrt{\Sigma}$

Then:

$$\min_{Y=AB^T} \frac{1}{2}\left(\|A\|_{Fro}^2 + \|B\|_{Fro}^2\right) \leq \frac{1}{2}\left(\left\|U\sqrt{\Sigma}\right\|_{Fro}^2 + \left\|V\sqrt{\Sigma}\right\|_{Fro}^2\right)$$

$$= \frac{1}{2}\left(\text{trace}\left(\left(U\sqrt{\Sigma}\right)^T\left(U\sqrt{\Sigma}\right)\right) + \text{trace}\left(\left(V\sqrt{\Sigma}\right)^T\left(V\sqrt{\Sigma}\right)\right)\right)$$

$$= \frac{1}{2}\left(\text{trace}\left(\sqrt{\Sigma}U^TU\sqrt{\Sigma}\right) + \text{trace}\left(\sqrt{\Sigma}V^TV\sqrt{\Sigma}\right)\right)$$

$$= \frac{1}{2}\left(\text{trace}\left(\sqrt{\Sigma}\sqrt{\Sigma}\right) + \text{trace}\left(\sqrt{\Sigma}\sqrt{\Sigma}\right)\right)$$

$$= \frac{1}{2}\left(\text{trace}(\Sigma) + \text{trace}(\Sigma)\right) = \text{trace}(\Sigma) = \|Y\|_*$$

User/Movie Symmetry

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right) + \sum_{ij}\left(y_{ij} - u_i^T v_j\right)^2$$

- If we knew V, then linear regression to learn U
  - Treat V as features

- If we knew U, then linear regression to learn V
  - Treat U as features

Optimization

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right) + \sum_{ij} \omega_{ij}\left(y_{ij} - u_i^T v_j\right)^2 \qquad \omega_{ij} \in \{0,1\}$$

- Only train over observed y_{ij}
- Two ways to Optimize
  - – Gradient Descent
  - – Alternating optimization
    - Closed Form (for each sub-problem)
  - – Homework question

Gradient Calculation

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right) + \frac{1}{2}\sum_{ij}\omega_{ij}\left(y_{ij} - u_i^T v_j\right)^2$$

$$\partial_{u_i} = \lambda u_i - \sum_j \omega_{ij} v_j \left(y_{ij} - u_i^T v_j\right)^T$$

Closed Form Solution (assuming V fixed):

$$u_i = \left(\lambda I_K + \sum_j \omega_{ij} v_j v_j^T\right)^{-1}\left(\sum_j \omega_{ij} y_{ij} v_j\right)$$

Coordinate Gradient Descent

- Initialize U & V randomly
- Loop
  - Choose one u_i, v_j randomly
  - Take gradient step:

$$u_i = u_i - \eta \partial_{u_i}$$

$$\partial_{u_i} = \lambda u_i - \sum_j \omega_{ij} v_j \left(y_{ij} - u_i^T v_j \right)$$

Stochastic Gradient Descent

- Initialize U & V randomly
- Loop
  - Choose one data point (i,j) randomly
  - Compute gradient for all U & V of:

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2N}\left(\|U\|_{Fro}^{2}+\|V\|_{Fro}^{2}\right)+\frac{1}{2}\left(y_{ij}-u_i^T v_j\right)^2$$

N = #observed entries

  - Take a gradient step for all U & V

Alternating Optimization

- Initialize U & V randomly
- Loop
  - – Choose next u_i or v_j
  - – Solve optimally:

$$u_i = \left(\lambda I_K + \sum_j \omega_{ij} v_j v_j^T \right)^{-1} \left(\sum_j \omega_{ij} y_{ij} v_j \right)$$

    - (assuming all other variables fixed)

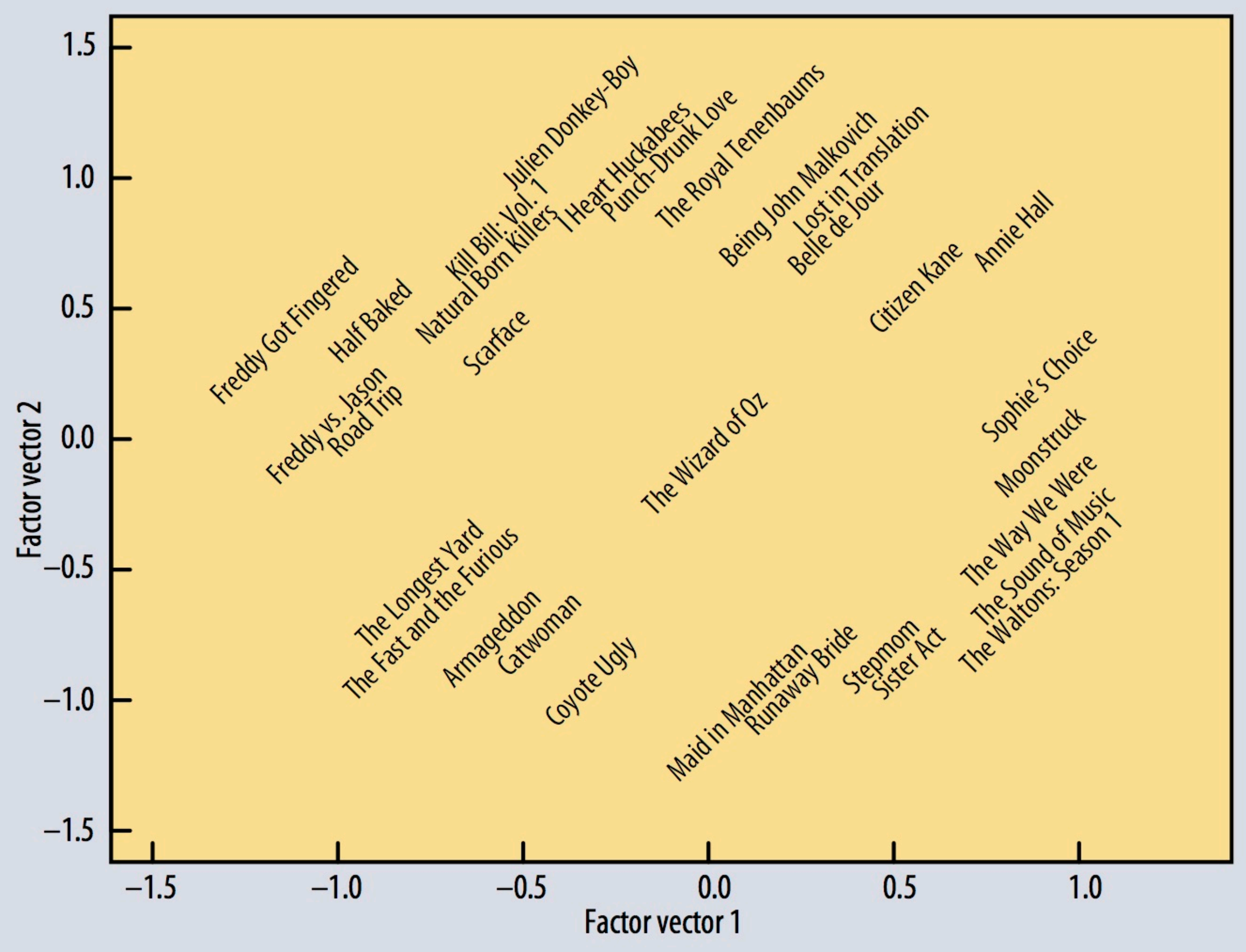
Tradeoffs

- Alternating optimization much faster in terms of #iterations
  - But requires inverting a matrix:

$$u_i = \left(\lambda I_K + \sum_j \omega_{ij} v_j v_j^T\right)^{-1} \left(\sum_j \omega_{ij} y_{ij} v_j\right)$$

- Gradient descent faster for high-dim problems
  - Also allows for streaming data

$$u_i = u_i - \eta \partial_{u_i}$$



Miniproject 2: create your own.

http://www2.research.att.com/~volinsky/papers/ieeecomputer.pdf

Recap: Collaborative Filtering

- **Goal:** predict every user/item rating

- **Challenge:** only a small subset observed

- **Assumption:** there exists a low-rank subspace that captures all the variability in describing different users and items

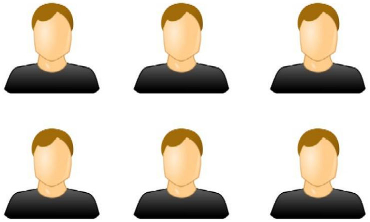
Multitask Learning

- M Tasks:

$$S^m = \left\{(x_i, y_i^m)\right\}_{i=1}^N$$

$$\underset{W}{\operatorname{argmin}} \frac{\lambda}{2} \underset{\substack{\uparrow \\ \text{Regularizer}}}{R(W)} + \frac{1}{2} \sum_m \sum_i \left(y_i - w_m^T x_i\right)^2$$

- Example: personalized recommender system
  - One task per user:



How to Regularize?

$$\underset{W}{\operatorname{argmin}} \frac{\lambda}{2} R(W) + \frac{1}{2} \sum_m \sum_i \left(y_i - w_m^T x_i\right)^2 \qquad S^m = \left\{(x_i, y_i^m)\right\}_{i=1}^N$$

- Standard L2 Norm:

$$\underset{W}{\operatorname{argmin}} \frac{\lambda}{2} \|W\|^2 + \sum_m \sum_i \left(y_i - w_m^T x_i\right)^2 = \sum_m \left[\frac{\lambda}{2} \|w_m\|^2 + \sum_i \left(y_i - w_m^T x_i\right)^2 \right]$$

- Decomposes to independent tasks
  - For each task, learn D parameters

How to Regularize?

$$\underset{W}{\operatorname{argmin}} \frac{\lambda}{2} R(W) + \frac{1}{2} \sum_{m} \sum_{i} \left(y_i - w_m^T x_i \right)^2 \qquad S^m = \left\{ (x_i, y_i^m) \right\}_{i=1}^{N}$$

- Trace Norm:

$$\underset{W}{\operatorname{argmin}} \frac{\lambda}{2} \|W\|_* + \sum_{m} \sum_{i} \left(y_i - w_m^T x_i \right)^2$$

- Induces W to have low rank across all task

Recall: Trace Norm & Latent Factor Models

- Suppose we consider all U,V that achieve perfect reconstruction: $W=UV^T$

- Find U,V with lowest complexity:

$$\underset{W=UV^T}{\operatorname{argmin}} \frac{1}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right)$$

- Claim: complexity equivalent to trace norm:

$$\|W\|_* = \min_{W=UV^T} \frac{1}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right)$$

How to Regularize?

$$\underset{W}{\operatorname{argmin}} \frac{\lambda}{2} R(W) + \frac{1}{2} \sum_m \sum_i \left(y_i - w_m^T x_i \right)^2 \qquad S^m = \left\{ (x_i, y_i^m) \right\}_{i=1}^N$$

- Latent Factor Approach

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2} \left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2 \right) + \frac{1}{2} \sum_m \sum_i \left(y_i - u_m^T V x_i \right)^2$$

- Learns a feature projection x' = Vx
- Learns a K dimensional model per task

Tradeoff

- D*N parameters:

$$\underset{W}{\operatorname{argmin}} \sum_m \left[\frac{\lambda}{2} \left\| w_m \right\|^2 + \frac{1}{2} \sum_i \left(y_i - w_m^T x_i \right)^2 \right]$$

- D*K + N*K parameters:

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2} \left(\left\| U \right\|_{Fro}^2 + \left\| V \right\|_{Fro}^2 \right) + \frac{1}{2} \sum_m \sum_i \left(y_i - u_m^T V x_i \right)^2$$

  - Statistically more efficient
  - Great if low-rank assumption is a good one

Multitask Learning

- M Tasks: $S^m = \left\{(x_i, y_i^m)\right\}_{i=1}^N$

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right) + \frac{1}{2}\sum_m \sum_i \left(y_i^m - u_m^T V x_i\right)^2$$

- Example: personalized recommender system
  - One task per user:
  - If x is topic feature representation
    - V is subspace of correlated topics
    - Projects multiple topics together

Reduction to Collaborative Filtering

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right) + \frac{1}{2}\sum_m \sum_i \left(y_i^m - u_m^T V x_i\right)^2 \qquad S^m = \left\{(x_i, y_i^m)\right\}_{i=1}^N$$

- Suppose each x_i is single indicator $x_i = e_i$
- Then: $Vx_i = v_i$
- Exactly Collaborative Filtering!

$$x_i = \begin{bmatrix} \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}$$

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right) + \frac{1}{2}\sum_m \sum_i \left(y_i^m - u_m^T v_i\right)^2$$

Latent Factor Multitask Learning vs Collaborative Filtering

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right) + \frac{1}{2}\sum_m \sum_i \left(y_i^m - u_m^T V x_i\right)^2$$

- Projects x into low-dimensional subspace Vx
- Learns low-dimensional model per task

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right) + \frac{1}{2}\sum_m \sum_i \left(y_i^m - u_m^T v_i\right)^2$$

- Creates low dimensional feature for each movie
- Learns low-dimensional model per user

General Bilinear Models

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right) + \sum_i \left(y_i - z_i^T U^T V x_i\right)^2 \quad S = \{(x_i, z_i, y_i)\}$$

- Users described by features z
- Items described by features x

- Learn a projection of z and x into common low-dimensional space
  - Linear model in low dimensional space

Why are Bilinear Models Useful?

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right) + \frac{1}{2}\sum_m \sum_i \left(y_i - u_m^T v_i\right)^2$$

U: MxK
V: NxK

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right) + \frac{1}{2}\sum_m \sum_i \left(y_i - u_m^T V x_i\right)^2$$

U: MxK
V: DxK

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right) + \frac{1}{2}\sum_i \left(y_i - z_i^T U^T V x_i\right)^2$$

$$S = \left\{(x_i, z_{i,} y_i)\right\}$$

U: FxK
V: DxK

Story So Far: Latent Factor Models

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right) + \frac{1}{2}\sum_i \left(y_i - z_i^T U^T V x_i\right)^2 \quad S = \left\{(x_i, z_i, y_i)\right\}$$

- **Simplest Case:** reduces to SVD of matrix Y
  - No missing values
  - (z,x) indicator features
- **General Case:** projects high-dimensional feature representation into low-dimensional linear model

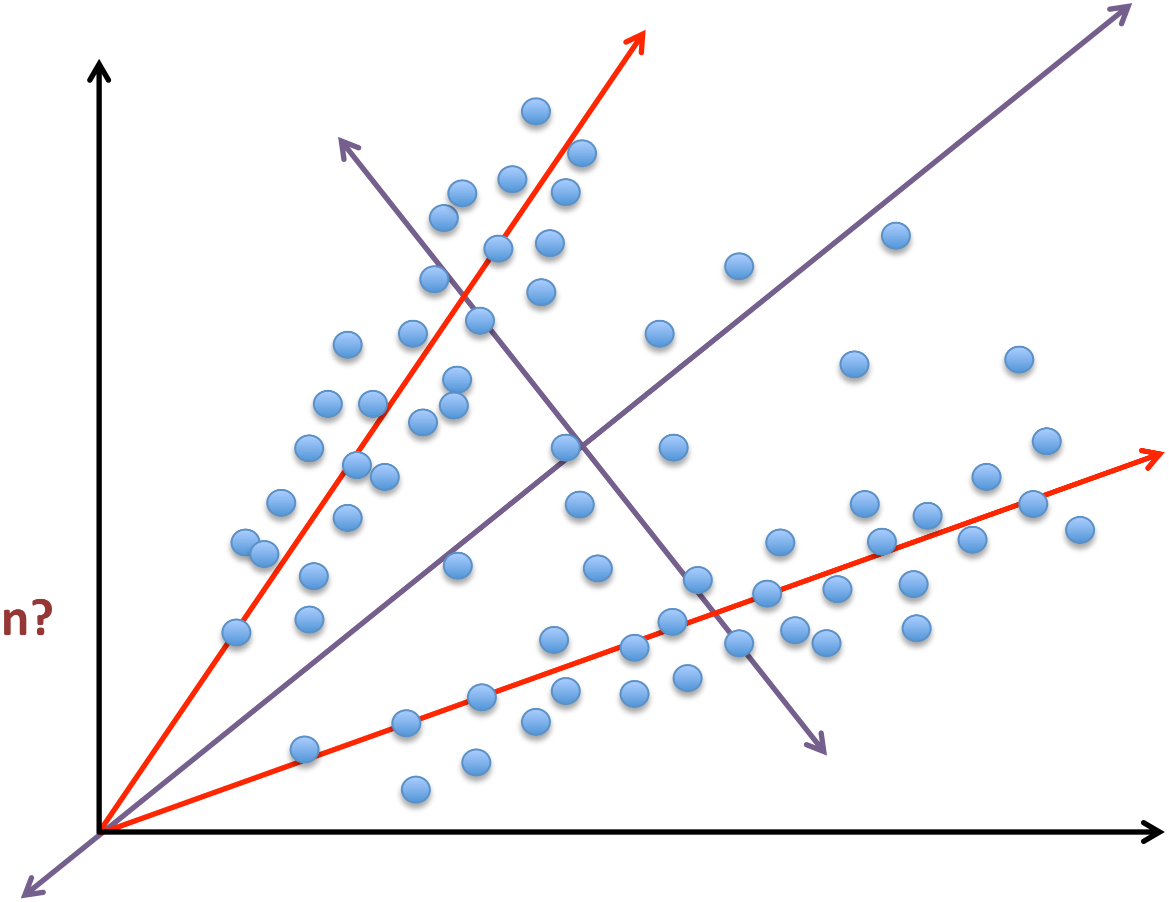
Non-Negative Matrix Factorization

Limitations of PCA & SVD

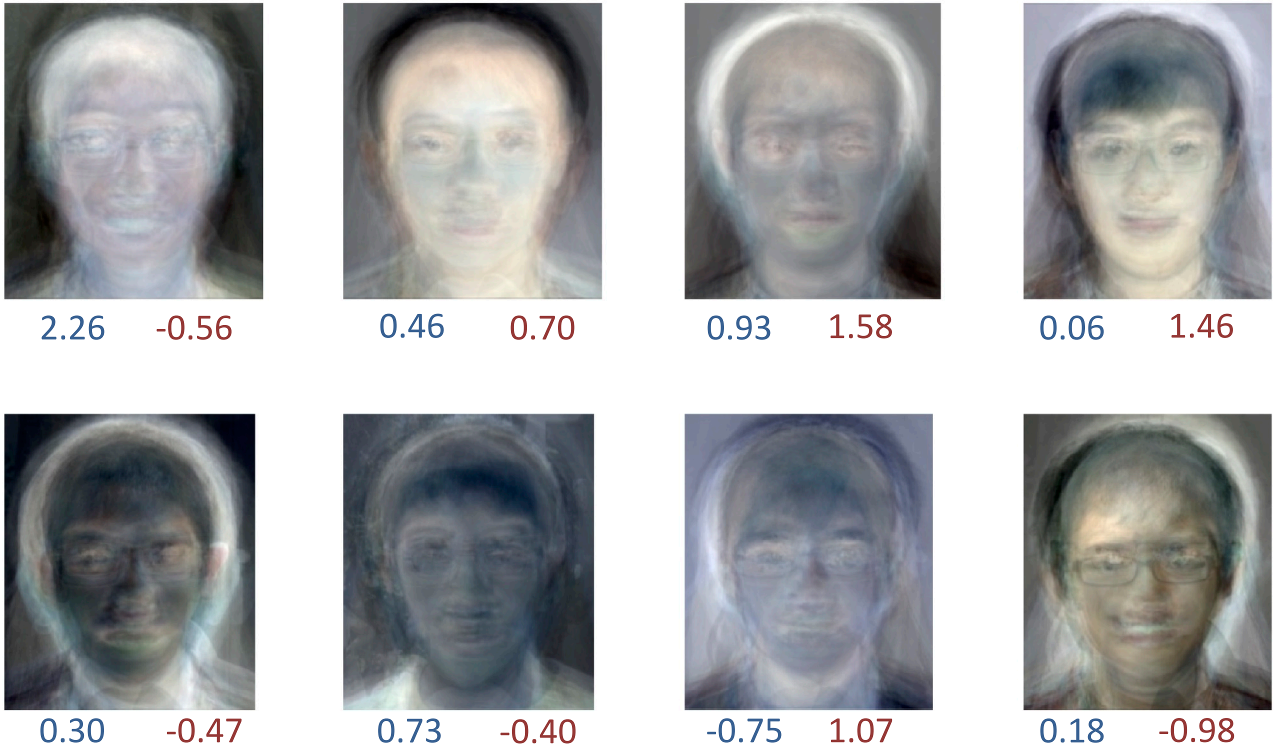
All features
non-negative

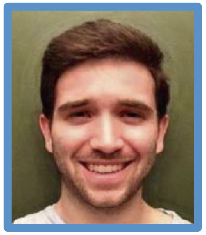
**PCA/SVD
Solution**

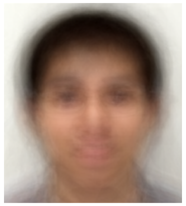
Better Solution?



CS 155 Eigenface Basis



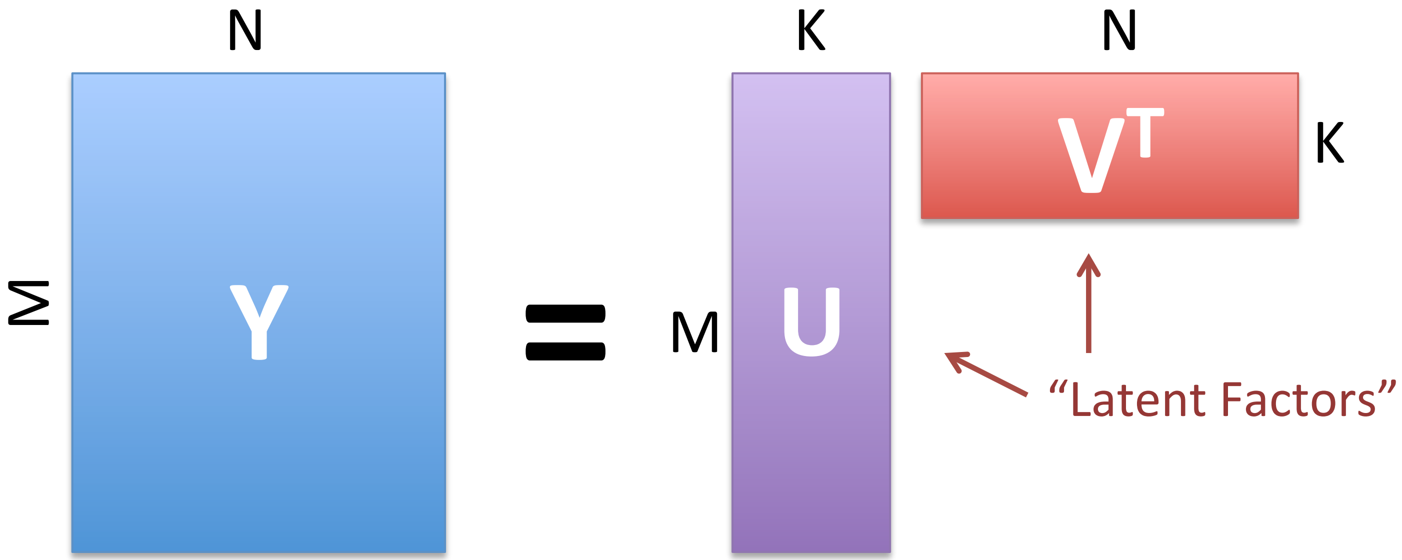




Avg Face

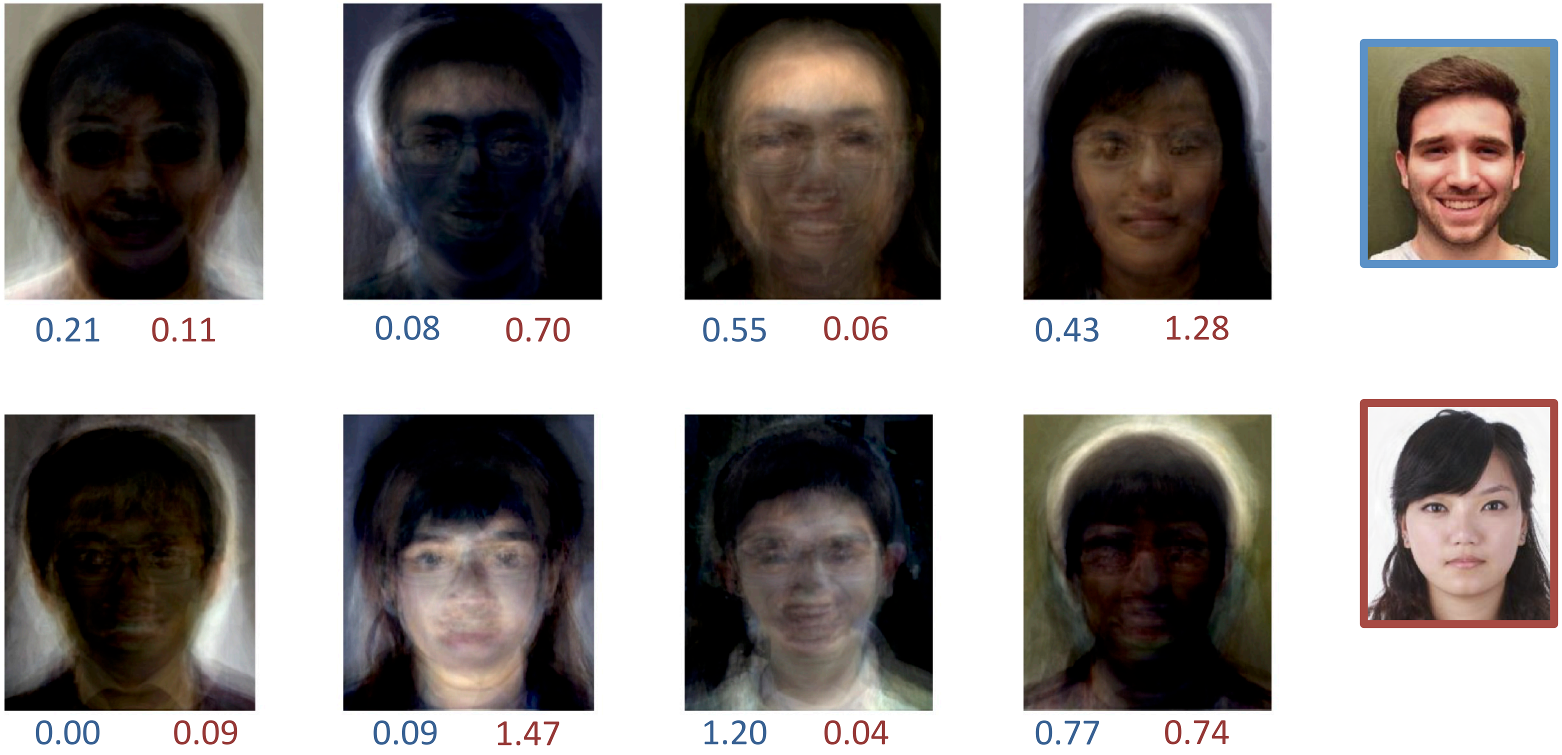
http://hebb.mit.edu/people/seung/papers/nmfconverge.pdf

Non-Negative Matrix Factorization



- Assume Y is non-negative
- Find non-negative U & V

CS 155 Non-Negative Face Basis



Aside: Non-Orthogonal Projections

- If columns of A are not orthogonal, $A^TA \neq I$
  - How to reverse transformation $x' = A^Tx$?
  - **Solution: Pseudoinverse!**

$$A = U\Sigma V^T$$

SVD

$$A^+ = V\Sigma^+ U^T$$

Pseudoinverse

$$\Sigma^+ = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_D \end{bmatrix} \qquad \sigma^+ = \begin{cases} 1/\sigma & \text{if } \sigma > 0 \\ 0 & \text{otherwise} \end{cases}$$

Intuition: use the rank-K orthogonal basis that spans A.

$$A^{+T}A^Tx = U\Sigma^+V^TV\Sigma U^Tx$$

$$= U_{1:K}U_{1:K}^Tx$$

Objective Function

$$\underset{U \geq 0, V \geq 0}{\operatorname{argmin}} \sum_{ij} \ell(y_{ij}, u_i^T v_j)$$

- Squared Loss:
  - Penalizes squared distance

$$\ell(a,b) = (a-b)^2$$

- Generalized Relative Entropy
  - Aka, unnormalized KL divergence
  - Penalizes ratio

$$\ell(a,b) = a \log \frac{a}{b} - a + b$$

- Train using gradient descent

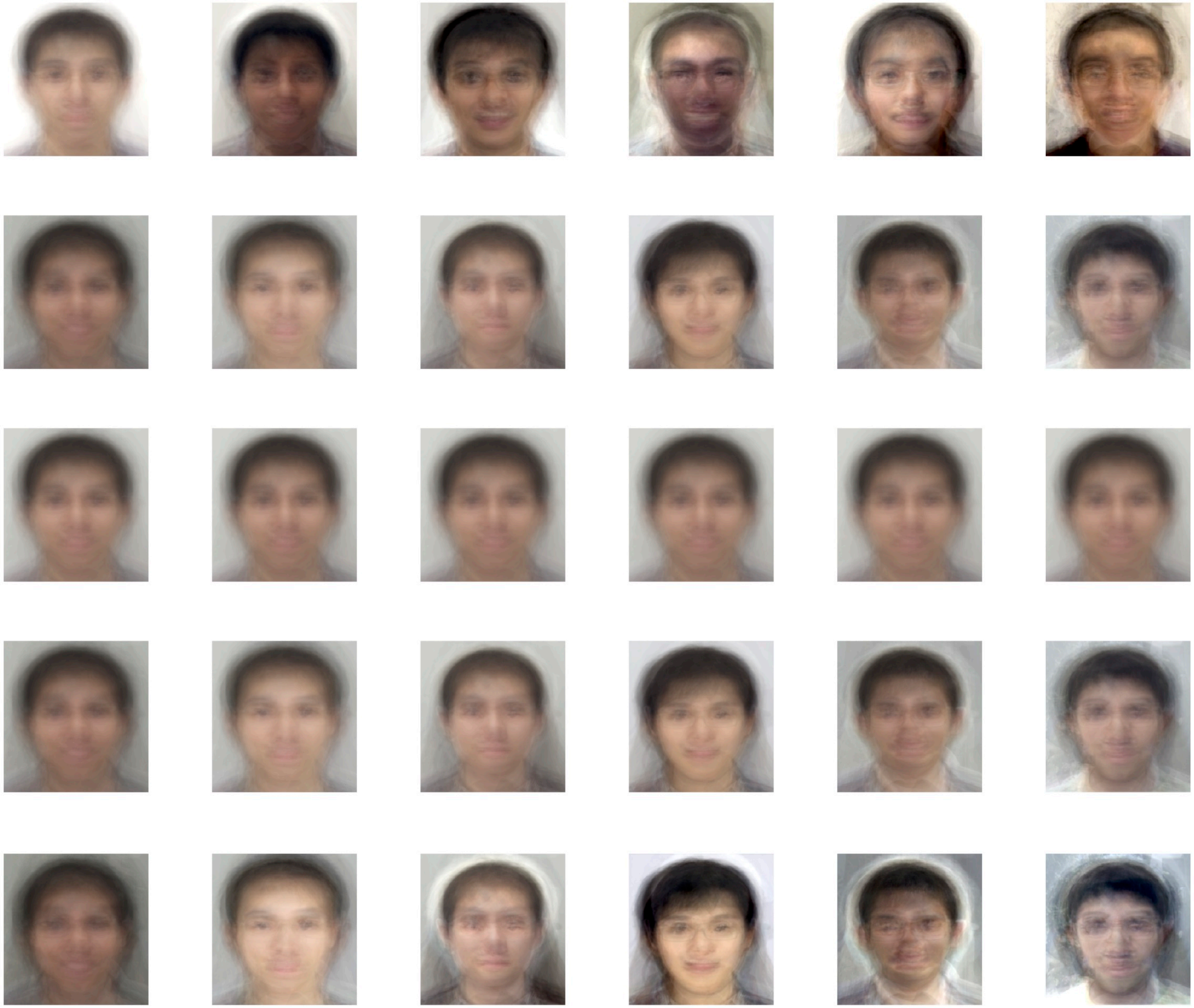
http://hebb.mit.edu/people/seung/papers/nmfconverge.pdf

SVD/PCA vs NNMF

- **SVD/PCA:**
  - Finds the best orthogonal basis faces
    - Basis faces can be neg.
  - Coeffs can be negative
  - Often trickier to visualize
  - Better reconstructions with fewer basis faces
    - Basis faces capture the most variations

- **NNMF:**
  - Finds best set of non-negative basis faces
  - Non-negative coeffs
    - Often non-overlapping
  - Easier to visualize
  - Requires more basis faces for good reconstructions

Varying top 6 Eigenfaces



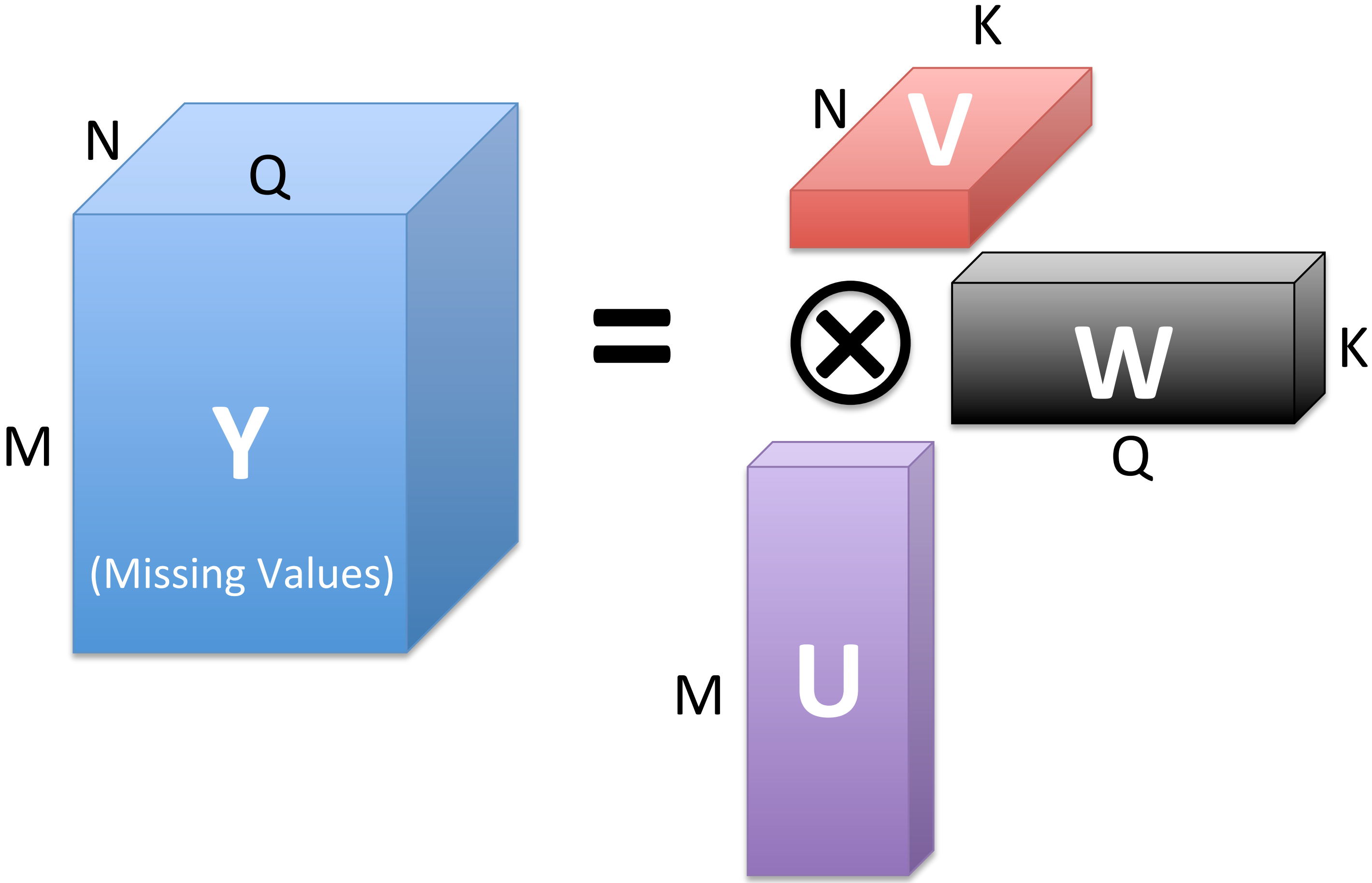
Non-Negative Latent Factor Models

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2\right) + \sum_i \ell\left(y_i, z_i^T U^T V x_i\right) \qquad S = \{(x_i, z_{i,} y_i)\}$$

- **Simplest Case:** reduces to NNMF of matrix Y
  - No missing values
  - (z,x) indicator features
- **General Case:** projects high-dimensional non-negative features into low-dimensional non-negative linear model

Tensor Latent Factor Models

Tensor Factorization



Tri-Linear Model

$$\underset{U,V,W}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2 + \|W\|_{Fro}^2\right) + \sum_i \ell\left(y_i, \left\langle U^T z_i, V^T x_i, W^T q_i \right\rangle\right)$$

- Prediction via 3-way dot product: $\langle a,b,c \rangle = \sum_k a_k b_k c_k$
  - Related to Hadamard Product

- **Example:** online advertising
  - User profile z
  - Item description x
  - Query q

Solve using
Gradient Descent

Next Week

- Embeddings

- Recent Applications of Latent Factor Models
  - Example of non-negative latent factor model
  - Example of tensor latent factor model

- Kaggle Mini-project closes Next Tuesday
  - Report due next Thursday