Caltech

Machine Learning & Data Mining

CS/CNS/EE 155

Lecture 10:

Latent Factor Models &

Non-Negative Matrix Factorization

CALTECH AI4SCIENCE 2ND WORKSHOP

February 14, 2019

https://sites.google.com/view/ai-for-science-workshop/home

RSVP:

https://www.eventbrite.com/e/caltech-ai4science-workshop-registration-55391262758

Tom Miller

Machine Learning for Chemistry

Azita Emami

Brain/Computer Interfaces

David Van Valen

Deep Learning for Cellular Biology

Brandon Rothrock

Making Planetary Images Searchable

Aaron Ames

Robotics & Control

Joel Burdick

DARPA Subterranean Challenge

Animesh Garg

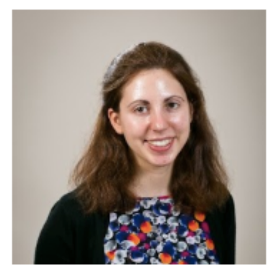
Robotics at Nvidia

Sara Beery (Perona Group)

Computer Vision in the Wild

Yuxin Chen

Active Learning using Gaussian Processes



Ellen Feldman Novoseller (Burdick Group)

Active Learning for Spinal Cord Therapy

Zach Wu (Arnold Group)

Active Learning for Protein Engineering

Yury Tokpanov (Atwater Group)

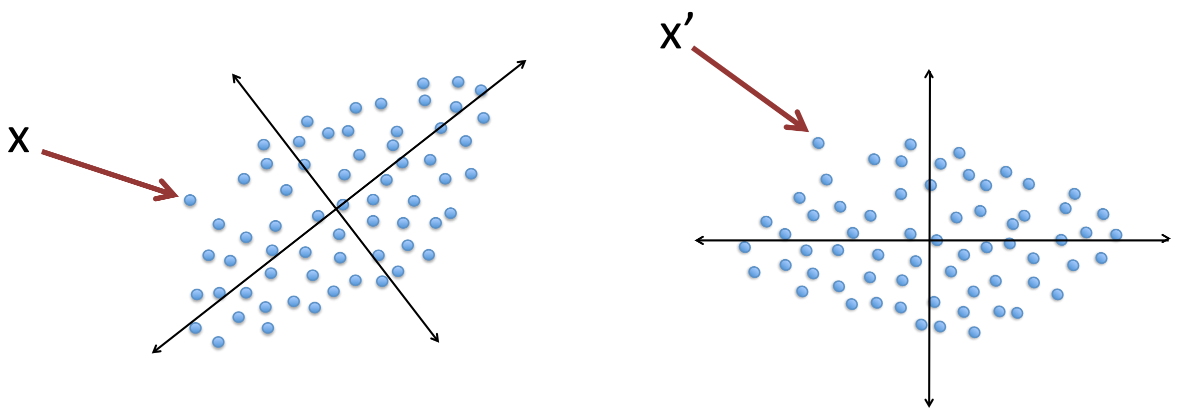
Active Learning for Nanophotonics

Today

- Some useful matrix properties
  - Useful for Homework 5
- Latent Factor Models
  - Low-rank models with missing values
- 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

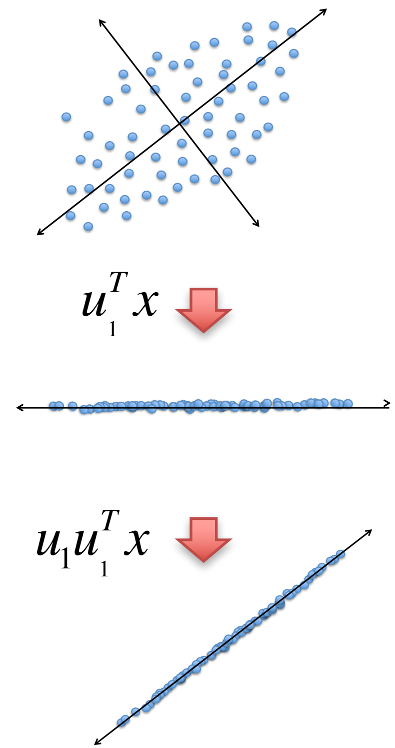
- Any subset of columns of U defines a subspace

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

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

$$\underbrace{XX^T = U}_{\text{Orthogonal}}\underbrace{\Lambda}_{\text{Diagonal}} U^T$$

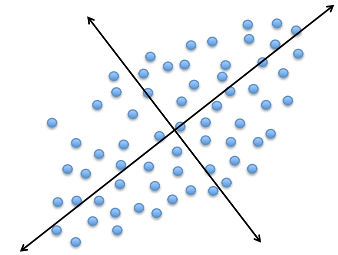
PCA

$$X = \underbrace{U}_{\text{Orthogonal}}\underbrace{\Sigma}_{\text{Diagonal}}\underbrace{V^T}_{\text{Orthogonal}}$$

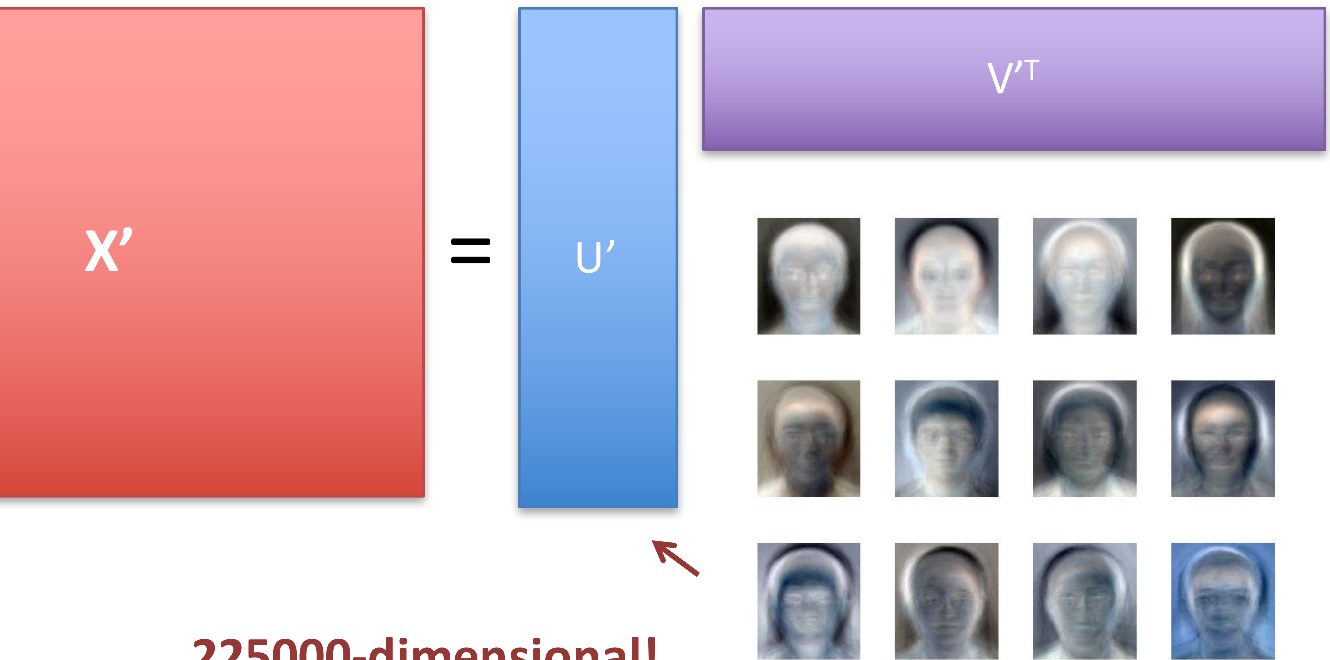
SVD

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

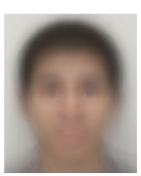
Recap: Eigenfaces



- Each col of U' is an "Eigenface"
- Each col of V'^T = coefficients of a student



Avg Face



225000-dimensional!

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

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

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

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

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

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

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

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

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: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_D \geq 0$

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

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

Frobenius Norm = Squared Norm

- Matrix version of L2 Norm:

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

- Useful for regularizing matrix models

$$X = U \Sigma V^T \qquad \Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_D \end{bmatrix}$$

Recall: L1 & Sparsity

- w is sparse if mostly 0's:
  - Small L0 Norm

$$\|w\|_0 = \sum_d 1_{[w_d \neq 0]}$$

- Why not L0 Regularization?
  - **Not continuous!**

$$\underset{w}{\operatorname{argmin}} \, \lambda \|w\|_0 + \sum_{i=1}^{N} \left(y_i - w^T x_i\right)^2$$

- L1 induces sparsity
  - And is continuous!

$$\underset{w}{\operatorname{argmin}} \, \lambda |w| + \sum_{i=1}^{N} \left(y_i - w^T x_i\right)^2$$

Omitting b &
for simplicity

Trace Norm = L1 of Eigenvalues

- A matrix X is considered low rank if it has few non-zero singular values:

$$\|X\|_{Rank} = \sum_d 1_{[\sigma_d > 0]}$$

Not continuous!

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

aka "spectral sparsity"

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

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

Other Useful Properties

- Cauchy Schwarz:

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

- AM-GM Inequality:

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

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

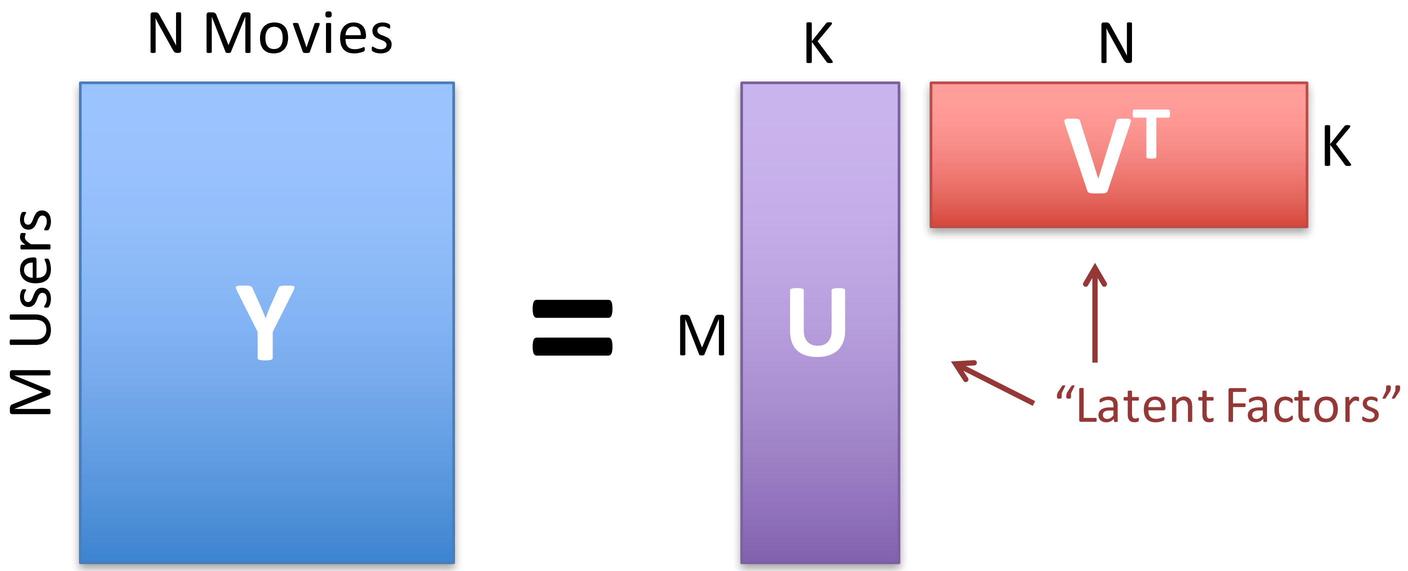
Recap: SVD & PCA

- SVD: $$X = U \Sigma V^T$$
- PCA: $$XX^T = U \Sigma^2 U^T$$
- 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

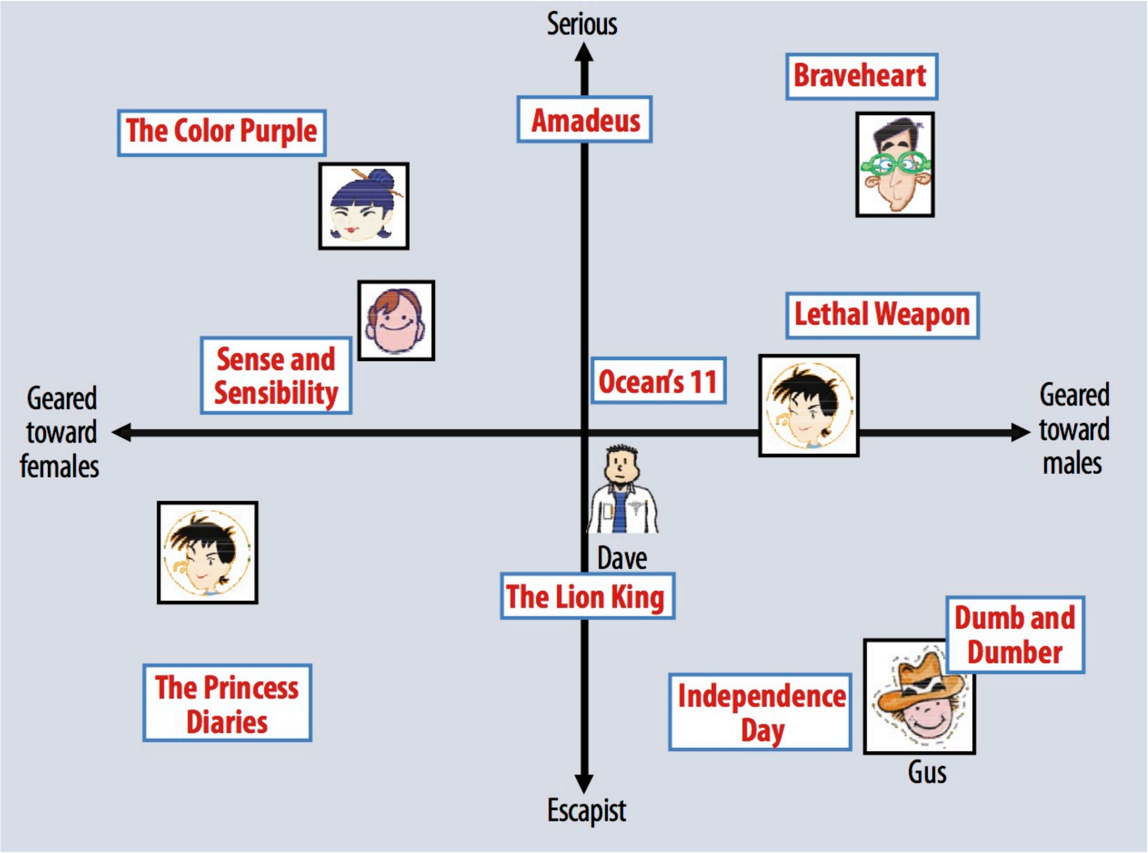


- Y_{ij} = rating user i gives to movie j

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

- Solve using SVD!

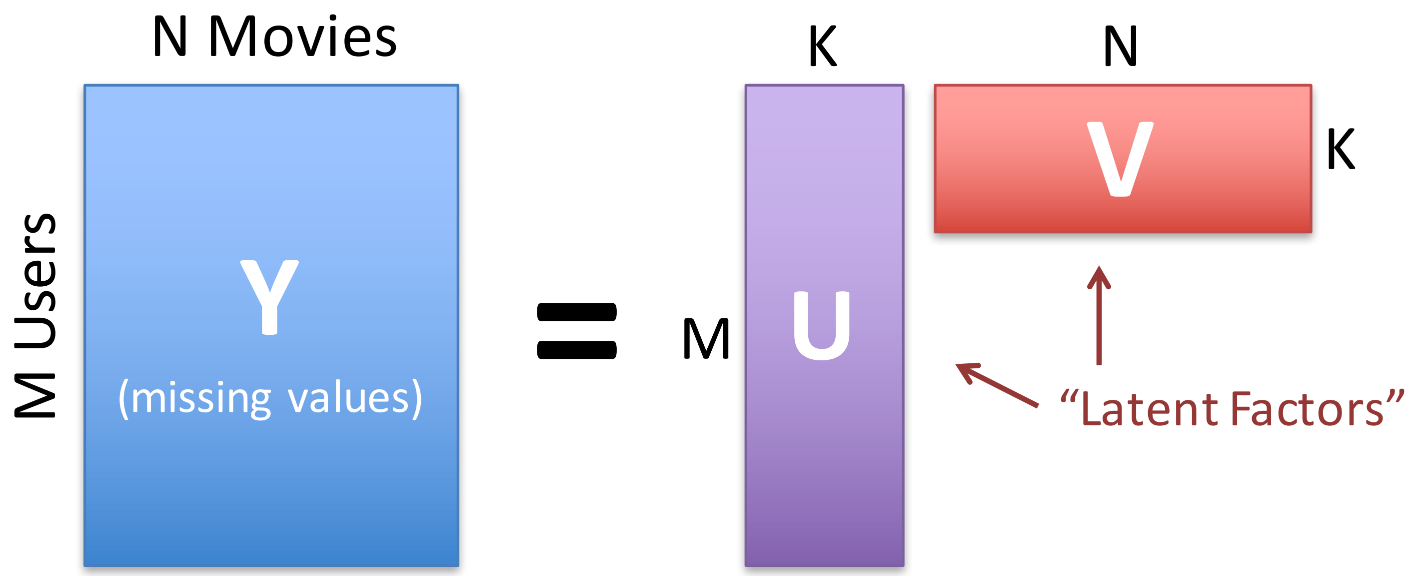
Example



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

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

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

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

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

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:

$$\begin{aligned}
\min_{Y=AB^T} \frac{1}{2}\left(\|A\|_{Fro}^2 + \|B\|_{Fro}^2\right) &\leq \frac{1}{2}\left(\left\|U\sqrt{\Sigma}\right\|_{Fro}^2 + \left\|V\sqrt{\Sigma}\right\|_{Fro}^2\right) \\
&= \frac{1}{2}\left(\text{trace}\left(\left(U\sqrt{\Sigma}\right)^T\left(U\sqrt{\Sigma}\right)\right) + \text{trace}\left(\left(V\sqrt{\Sigma}\right)^T\left(V\sqrt{\Sigma}\right)\right)\right) \\
&= \frac{1}{2}\left(\text{trace}\left(\sqrt{\Sigma}U^TU\sqrt{\Sigma}\right) + \text{trace}\left(\sqrt{\Sigma}V^TV\sqrt{\Sigma}\right)\right) \\
&= \frac{1}{2}\left(\text{trace}\left(\sqrt{\Sigma}\sqrt{\Sigma}\right) + \text{trace}\left(\sqrt{\Sigma}\sqrt{\Sigma}\right)\right) \\
&= \frac{1}{2}\left(\text{trace}(\Sigma) + \text{trace}(\Sigma)\right) = \text{trace}(\Sigma) = \|Y\|_*
\end{aligned}$$

Interpreting Model

- Latent-Factor Model Objective

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

- Related to:

$$\underset{W}{\operatorname{argmin}} \lambda \|W\|_* + \sum_{ij}\left(y_{ij} - w_{ij}\right)^2$$

Find the best low-rank approximation to Y!

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

Equivalent when U,V = rank of W

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):

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

Gradient Descent Options

- Stochastic Gradient Descent
  - Update all model parameters for single data point

- Alternating SGD:
  - Update a single column of parameters at a time

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

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

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)

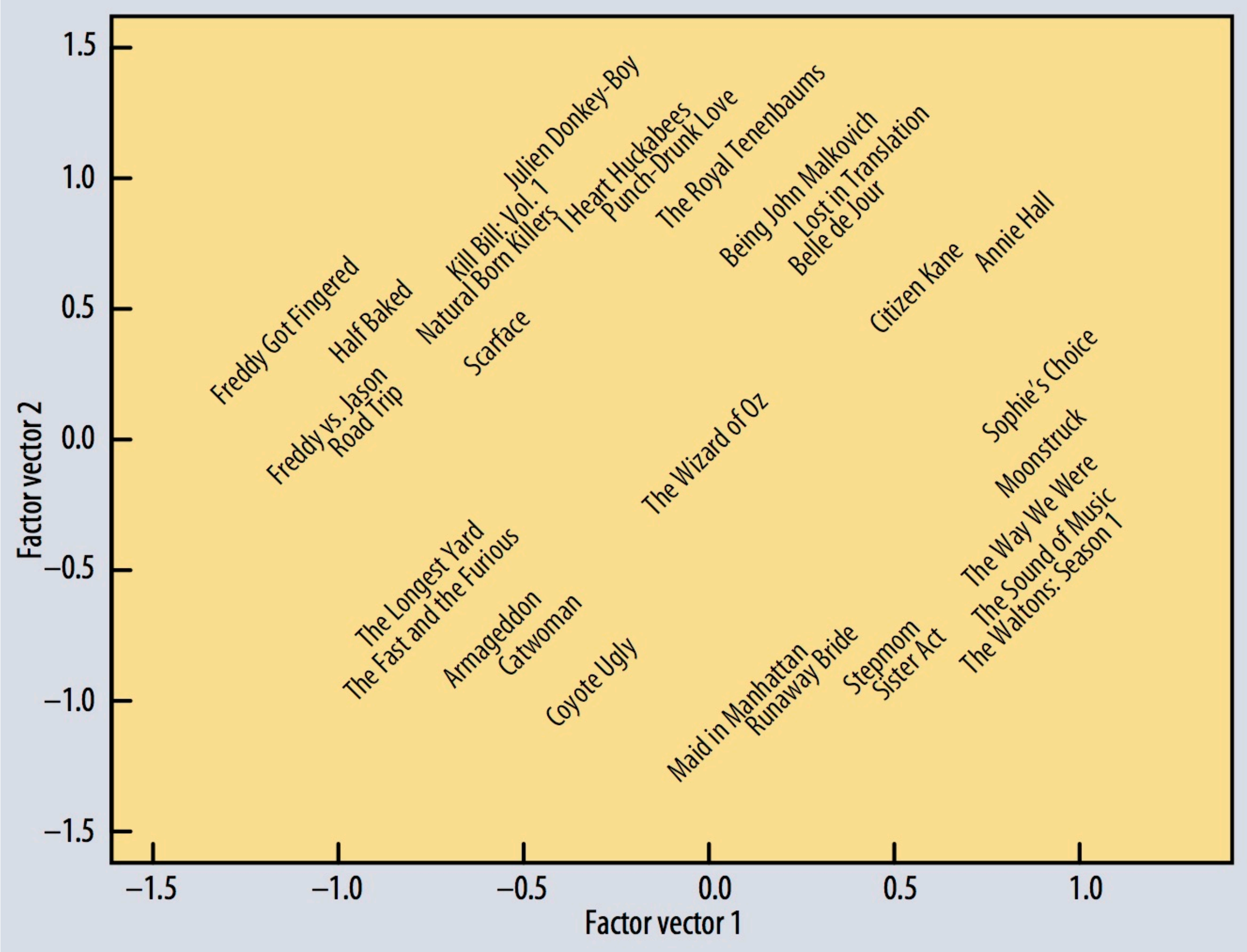
Tradeoffs

- 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

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



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

Recap: Collaborative Filtering

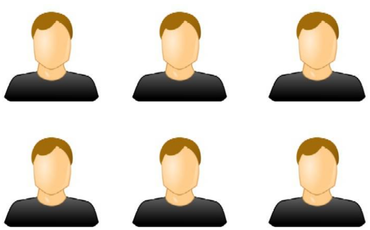
- **Goal:** predict every user/item rating
- **Challenge:** only a small subset observed
- **Assumption:** there exists a low-rank subspace that captures all the variability in describing different users and items

Aside: Multitask Learning

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

$$\underset{W}{\operatorname{argmin}} \frac{\lambda}{2} \underbrace{R(W)}_{\text{Regularizer}} + \frac{1}{2}\sum_{m}\sum_{i}\left(y_i - w_m^T x_i\right)^2$$

- 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:

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

- 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:

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

- D*K + N*K parameters:

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

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

Multitask Learning

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

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

- 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

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

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

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

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

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

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

- 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

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

$$S = \{(x_i, z_i, y_i)\}$$

U: FxK
V: DxK

Story So Far: Latent Factor Models

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

- **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

Aside: Non-Linear "Projections"

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

- Vx is a linear mapping
  - From x to low-dimensional space

- Can also learn non-linear mapping
  - E.g., the hidden layer activations of a neural net

http://www.cs.cornell.edu/~andreas/iccv15.pdf

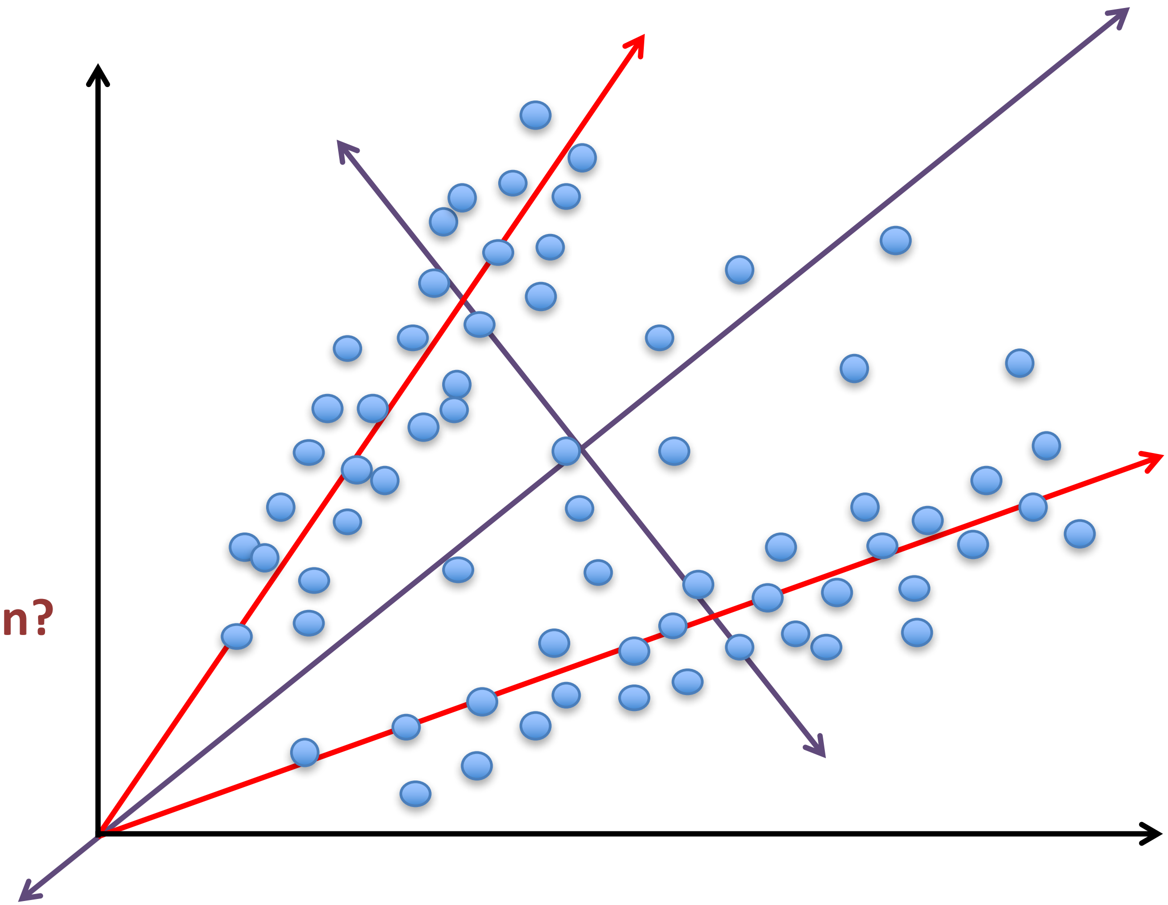
Non-Negative Matrix Factorization

Limitations of PCA & SVD

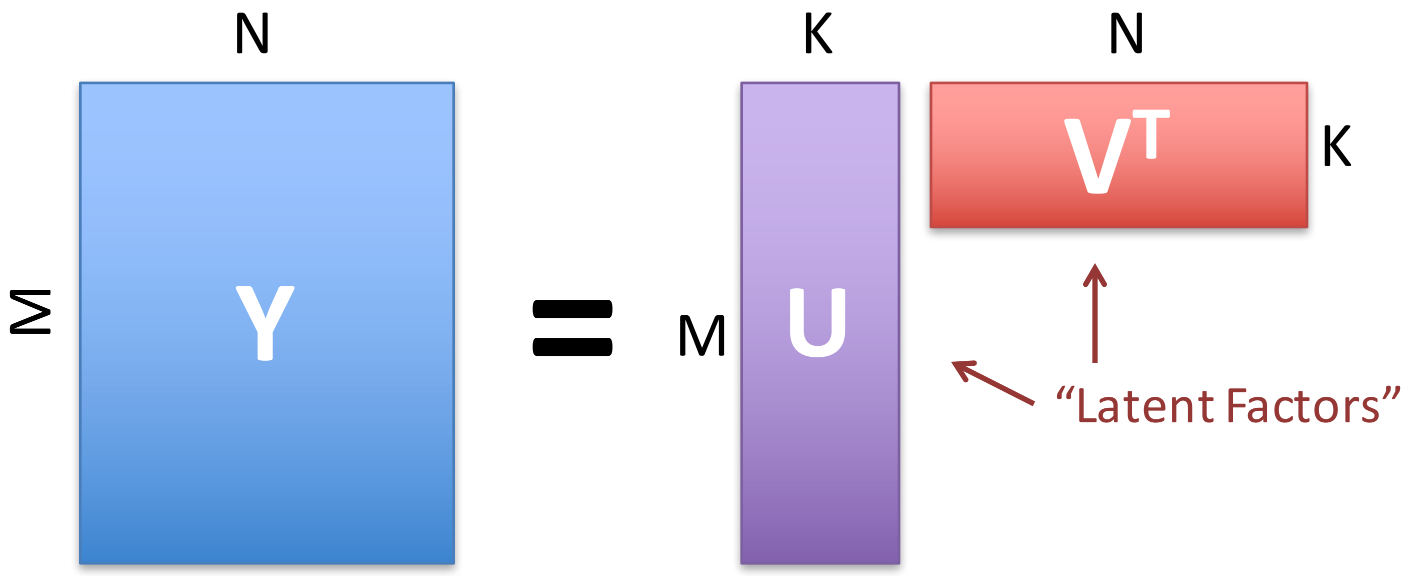
All features non-negative

PCA/SVD Solution

Better Solution?

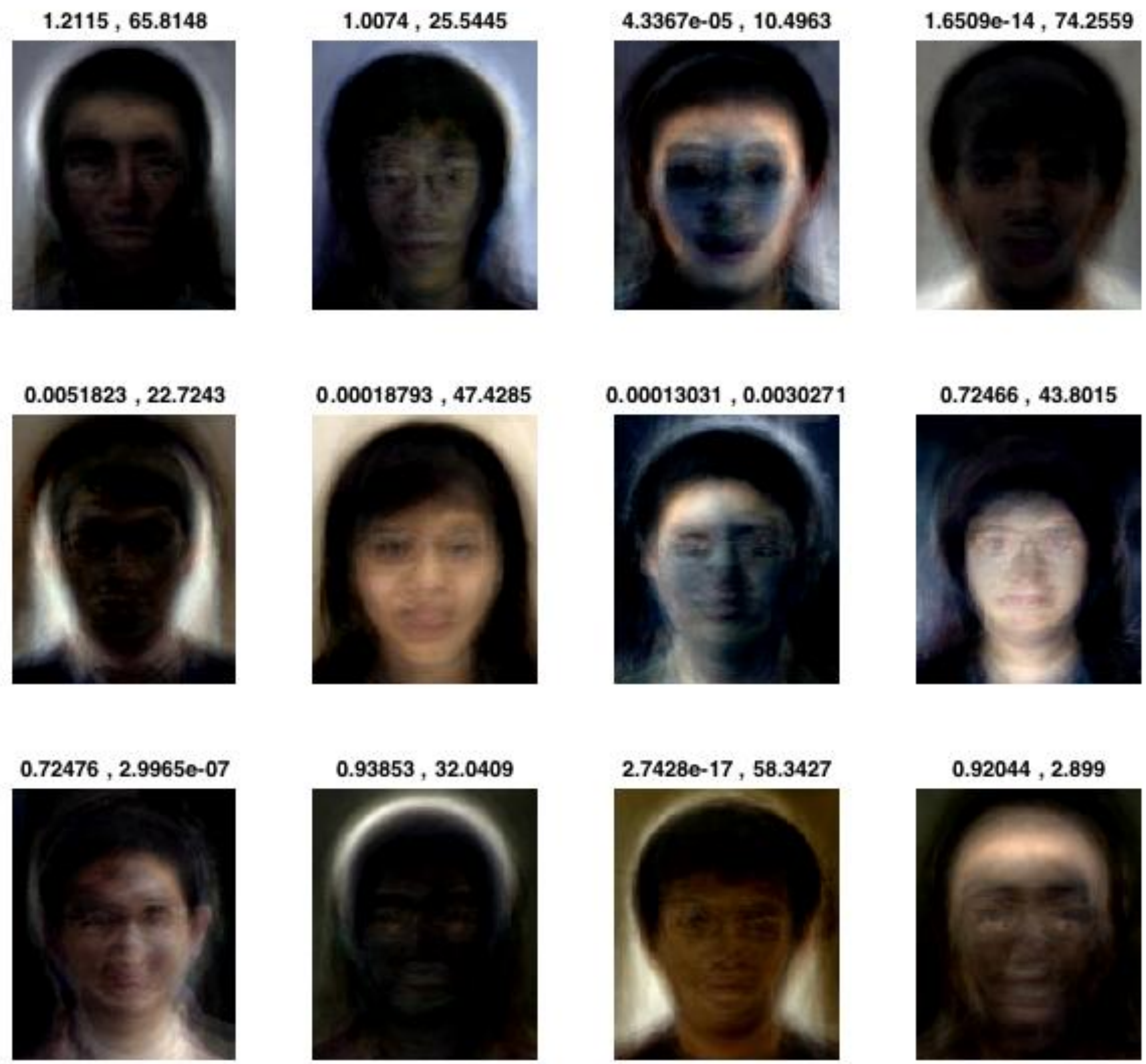


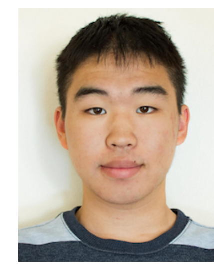
Non-Negative Matrix Factorization



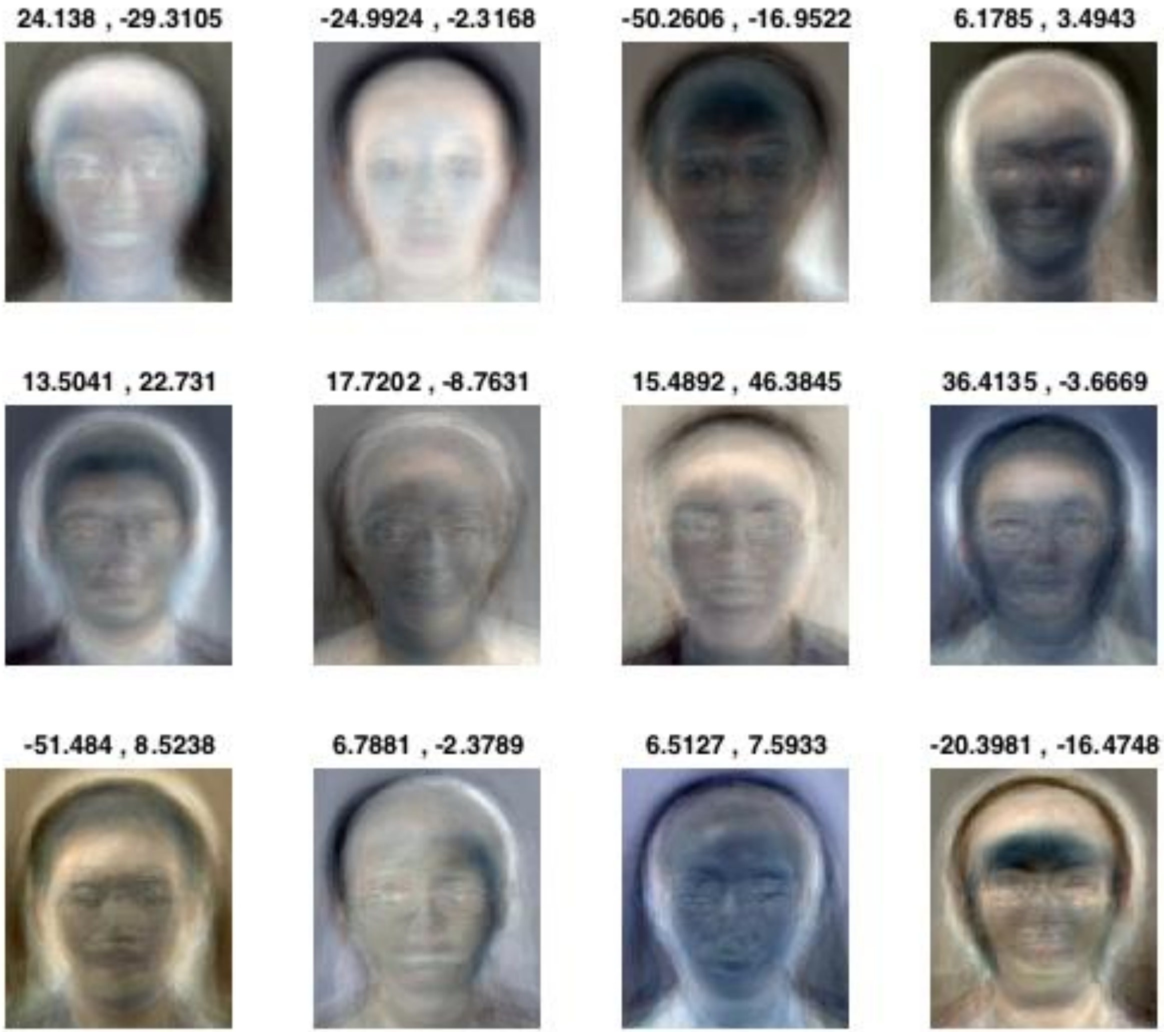
- Assume Y is non-negative
- Find non-negative U & V

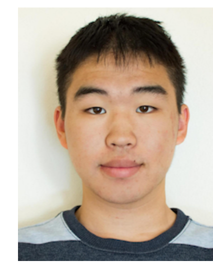
CS 155 Non-Negative Face Basis





CS 155 Eigenfaces





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.

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

$$= U_{1:K} U_{1:K}^T x$$

Objective Function

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

- 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

Non-Negative Latent Factor Models

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

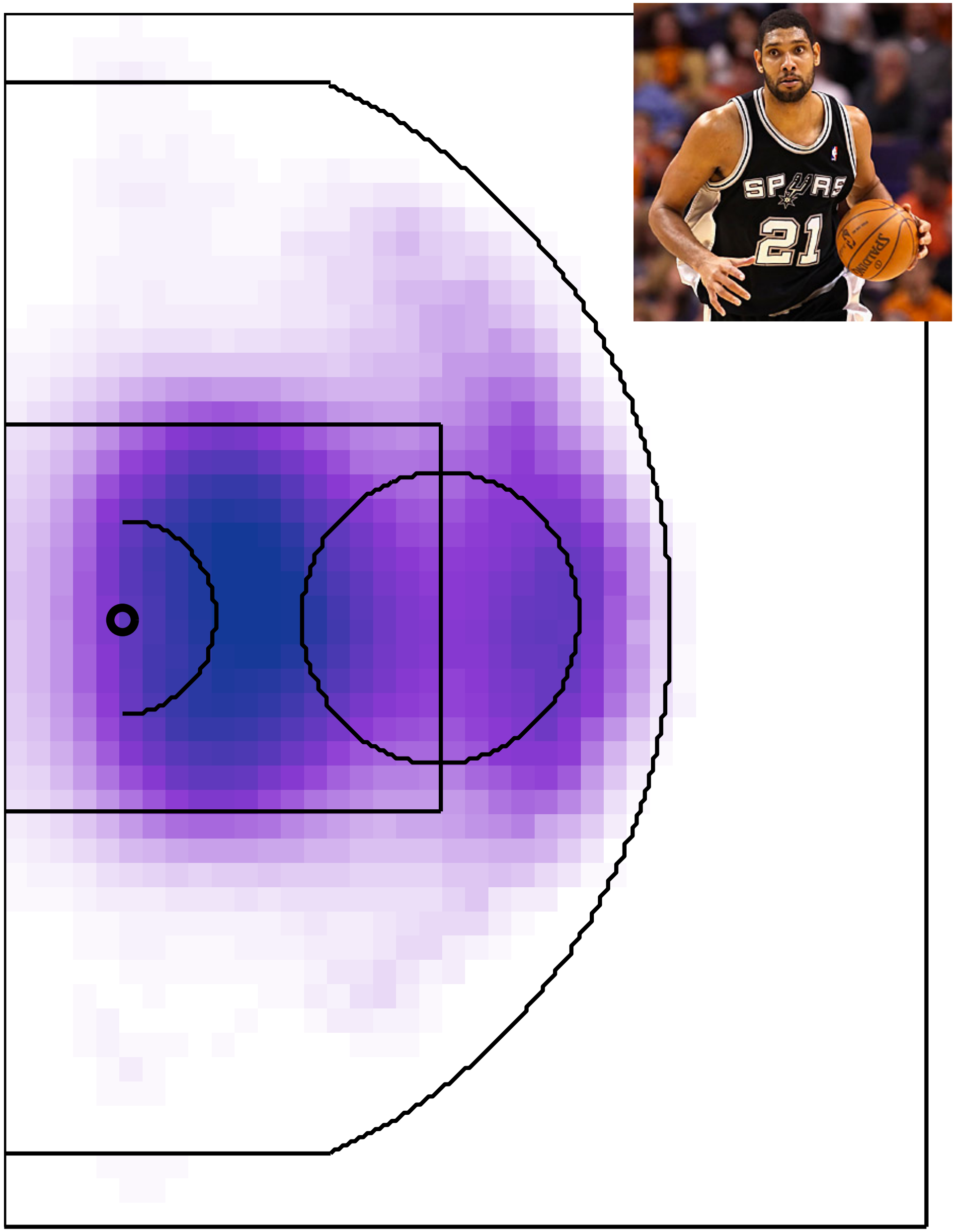
- **Simplest Case:** reduces to NNMF of matrix Y
  - No missing values
  - (z,x) indicator features
- **General Case:** projects high-dimensional non-negative features into low-dimensional non-negative linear model

Modeling NBA Gameplay Using Non-Negative Spatial Latent Factor Models

Fine-Grained Spatial Models

- Discretize court
  - 1x1 foot cells
  - 2000 cells
- 1 weight per cell
  - 2000 weights

$$F_s(\mathbf{x}):$$



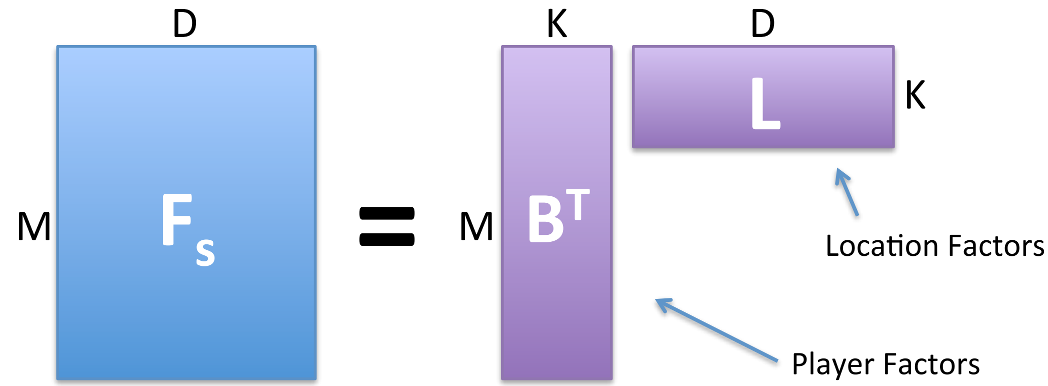
40 feet

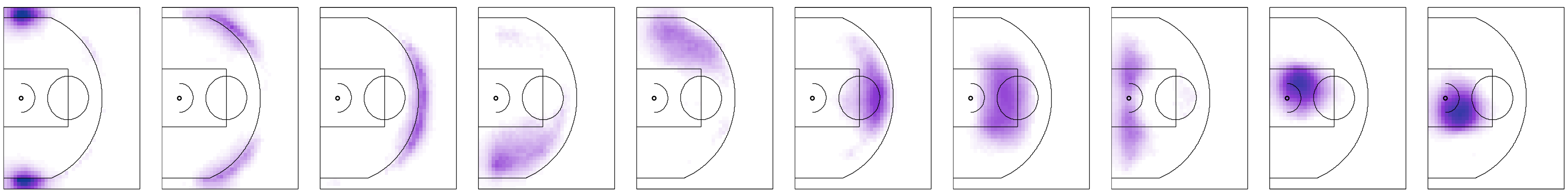
Fine-Grained Spatial Models

- Discretize court
  - 1x1 foot cells
  - 2000
- 1 weig
  - 2000 weights

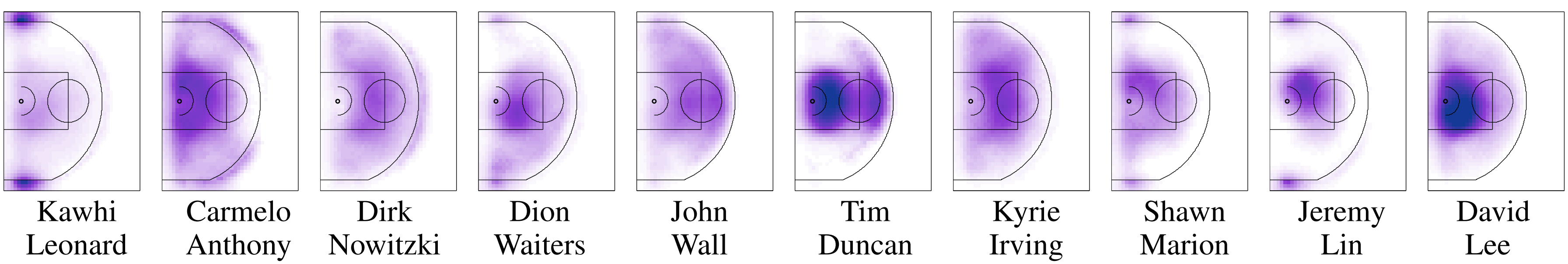
But most players haven't played that much!

$$F_s(\mathbf{x}):$$





Visualizing location factors L



Visualizing players B_bL

http://www.yisongyue.com/publications/icdm2014_bball_predict.pdf

Training Data

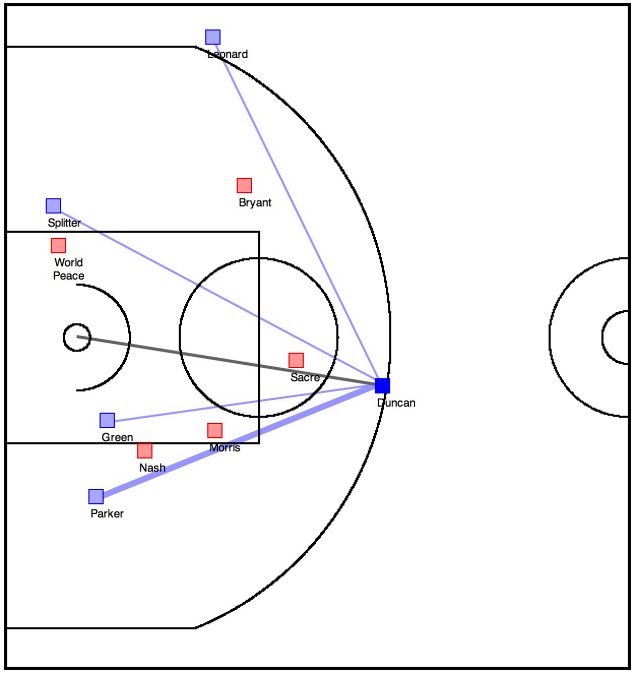
STATS SportsVU

2012/2013 Season, 630 Games,

80K Possessions, 380 frames per possession

Prediction

- Game state: **x**
  - Coordinates of all players
  - Who is the ball handler
- Event: y
  - Ball handler will shoot
  - Ball handler will pass (to whom?)
  - Ball handler will hold onto the ball
  - 6 possibilities
- **Goal:** Learn P(y|**x**)



http://www.yisongyue.com/publications/icdm2014_bball_predict.pdf

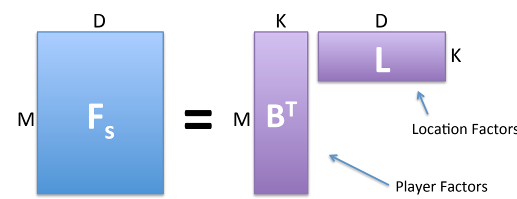
Logistic Regression

(Simple Version: Just for Shooting)

$$P(y \mid \mathbf{x}) = \frac{\exp\left\{F(y \mid \mathbf{x})\right\}}{Z(\mathbf{x} \mid F)} \qquad Z(\mathbf{x} \mid F) = \sum_{y' \in \{s, \perp\}} \exp\left\{F(y' \mid \mathbf{x})\right\}$$

$$F(y' \mid \mathbf{x}) = \begin{cases} F_s(\mathbf{x}) & y' = s \quad \text{Shot} \\ F_\perp & y' = \perp \quad \text{Hold on to ball} \end{cases}$$

Offset or bias



$$P(y = s \mid \mathbf{x}) = \frac{1}{1 + \exp\left\{-F_s(\boldsymbol{x}) + F_\perp\right\}}$$

Learning the Model

- Given training data: $S = \{(\boldsymbol{x}, y)\}$

Player Configuration

What Happened Next

- Learn parameters of model:

$$\underset{F_s, F_\perp}{\operatorname{argmin}} \frac{\lambda}{2} \|F_s\|^2 + \sum_{(\boldsymbol{x},y)\in S} \underbrace{\ell\left(y, F_s(x) - F_\perp\right)}_{\text{Log Loss}}$$

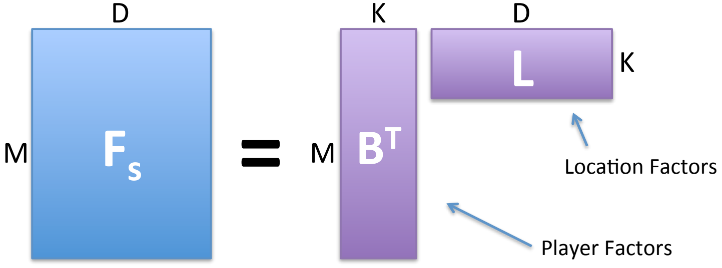
$$P(y = s \mid \mathbf{x}) = \frac{1}{1 + \exp\left\{-F_s(\boldsymbol{x}) + F_\perp\right\}}$$

Optimization via Gradient Descent

$$\underset{B\geq 0, L\geq 0, F_\perp}{\operatorname{argmin}} \frac{\lambda}{2}\left(\|B\|^2 + \|L\|^2\right) + \sum_{(\boldsymbol{x},y)} \ell\left(y, B_{b(\boldsymbol{x})}^T L_{l(\boldsymbol{x})} - F_\perp\right)$$

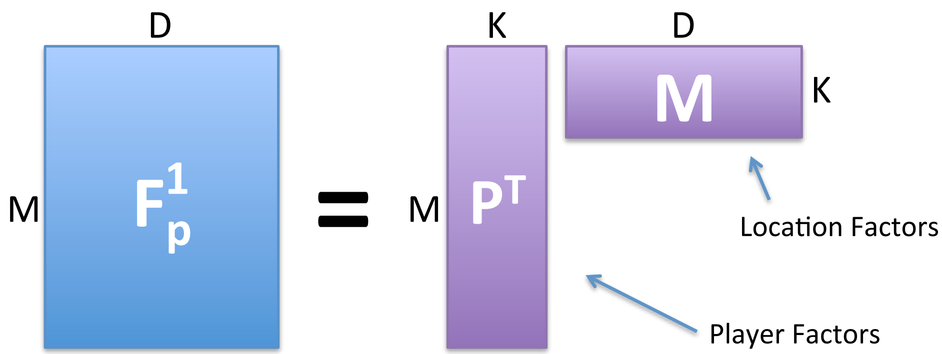
$$\partial_{L_i} = \lambda_1 L_i - \sum_{(\boldsymbol{x},y)} \frac{\partial \log P(y \mid \boldsymbol{x})}{\partial L_i}$$

$$\frac{\partial \log P(y \mid \boldsymbol{x})}{\partial L_i} = \left(1_{[y=s]} - P(s \mid \boldsymbol{x})\right) B_{b(\boldsymbol{x})}$$

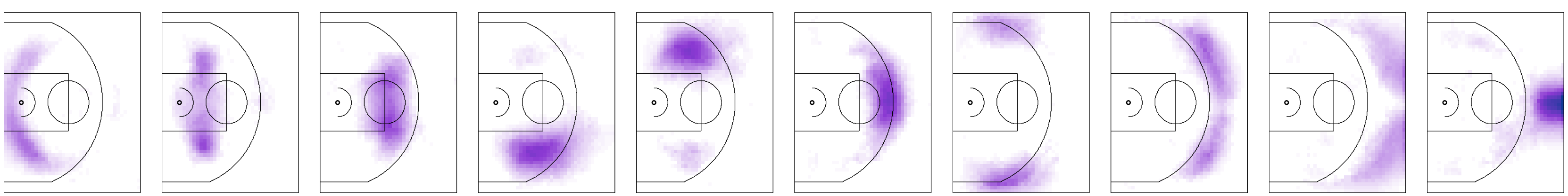


http://www.yisongyue.com/publications/icdm2014_bball_predict.pdf

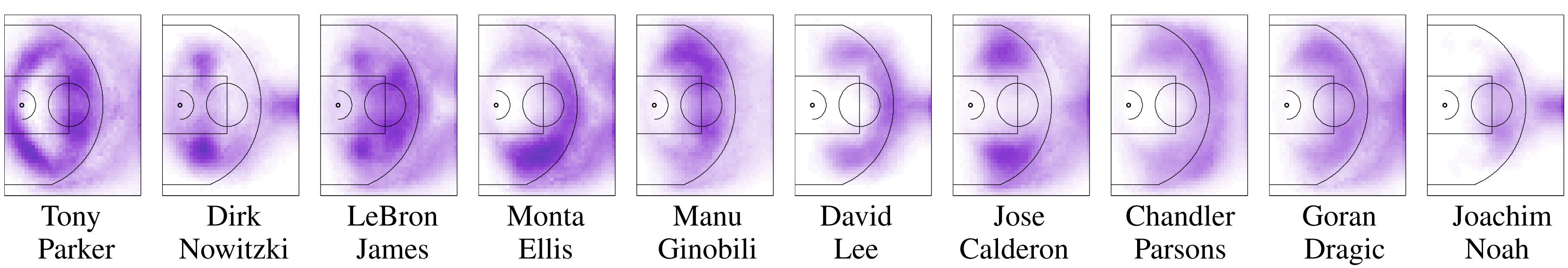
Where are Players Likely to Receive Passes?



Enforce Non-Negativity
(Accuracy Worse)
(More Interpretable)

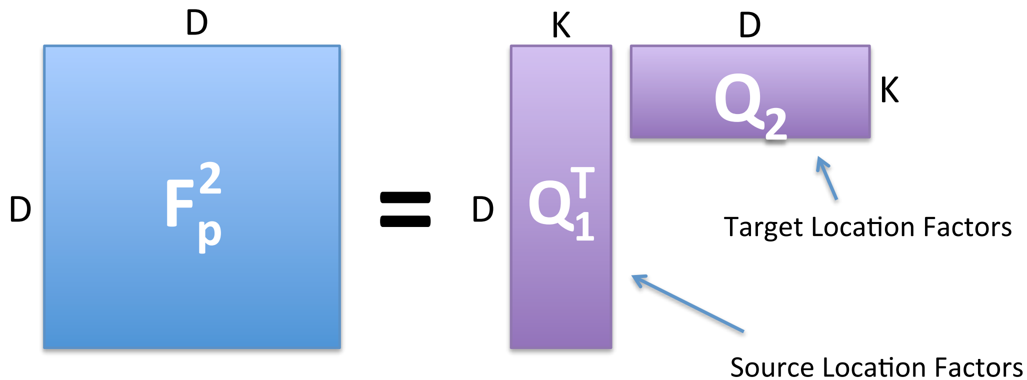


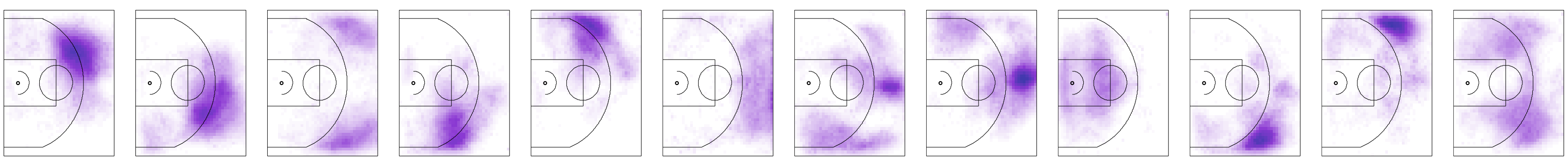
Visualizing Location Factors M



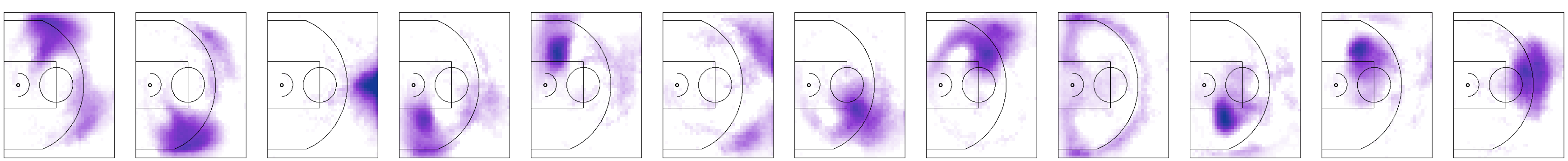
http://www.yisongyue.com/publications/icdm2014_bball_predict.pdf

How do passes tend to flow?





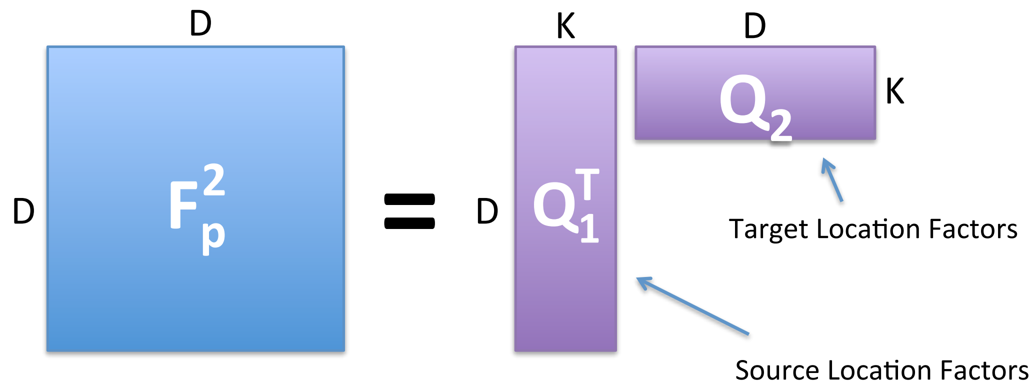
Q_1

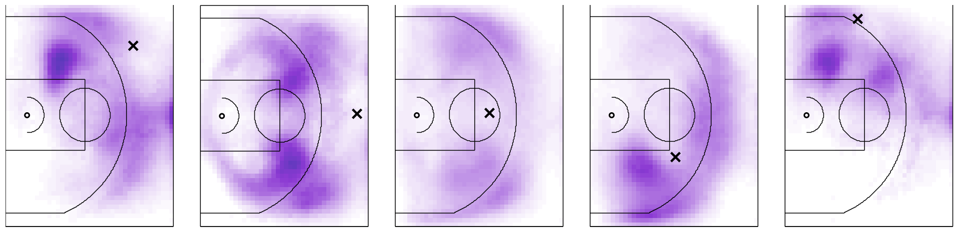


Q_2

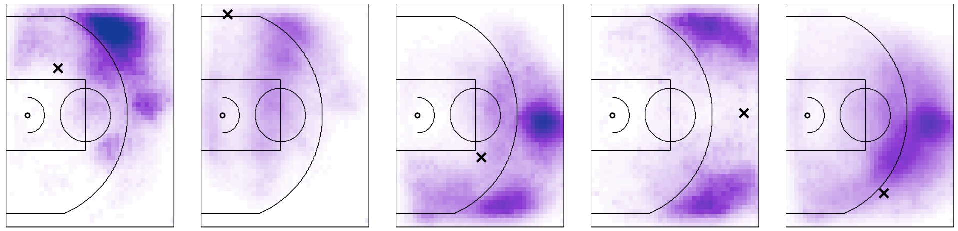
http://www.yisongyue.com/publications/icdm2014_bball_predict.pdf

How do passes tend to flow?





Passing From "X"

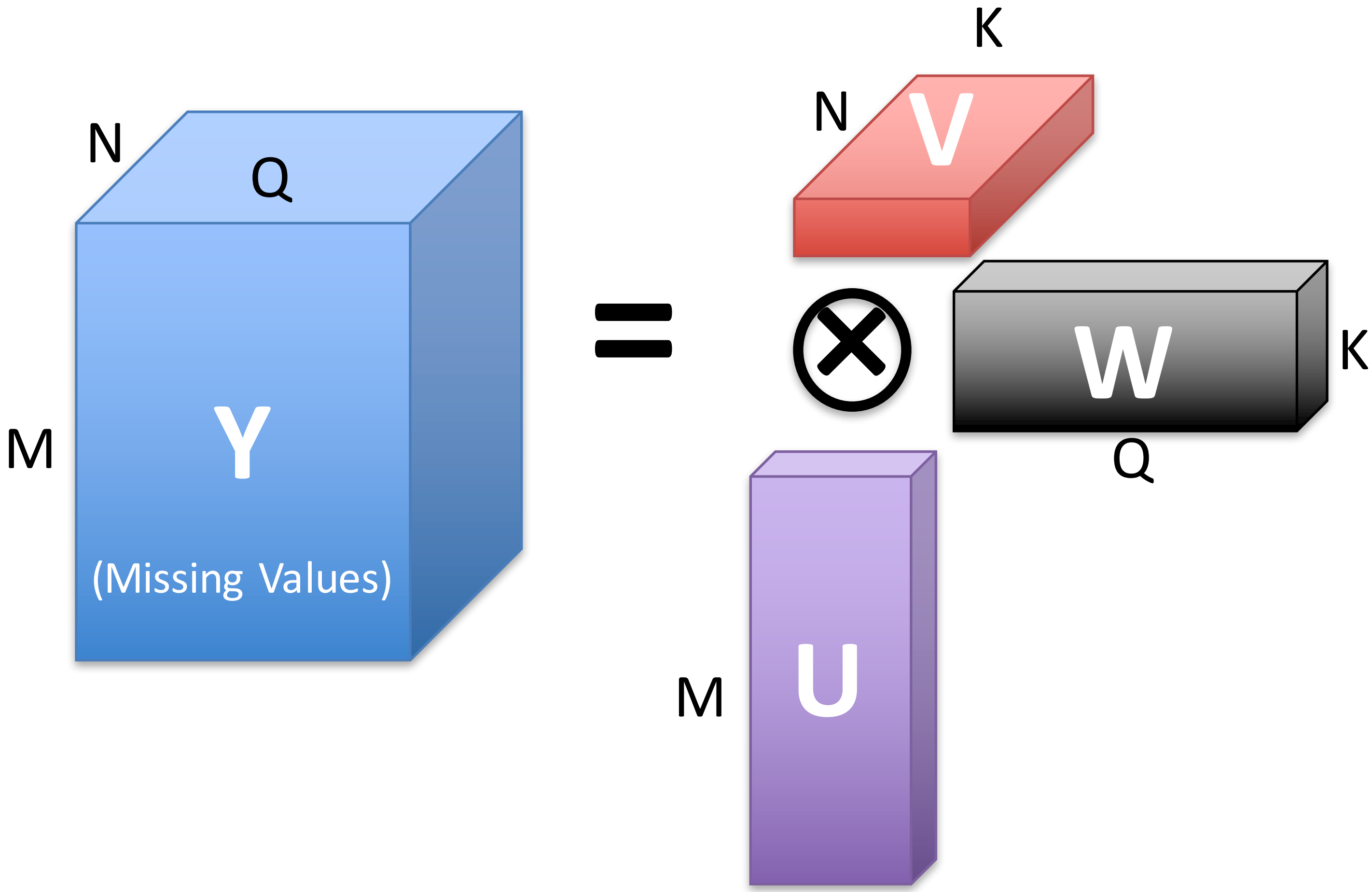


Passing To "X"

http://www.yisongyue.com/publications/icdm2014_bball_predict.pdf

Tensor Latent Factor Models

Tensor Factorization



Tri-Linear Model

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

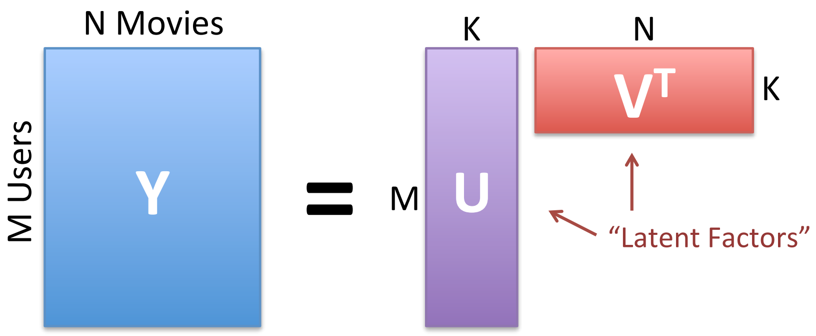
- Prediction via 3-way dot product: $\langle a,b,c\rangle = \sum_k a_k b_k c_k$
  - Related to Hadamard Product

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

Solve using Gradient Descent

Summary: Latent Factor Models

- Learns a low-rank model of a matrix of observations Y
  - Dimensions of Y can have various semantics
- Can tolerate missing values in Y
- Can also use features
- Widely used in industry



Next Lecture

- Embeddings

- Word2Vec