

Machine Learning & Data Mining

CS/CNS/EE 155

Lecture 10:
Latent Factor Models &
Non-Negative Matrix Factorization

SOCAL MACHINE LEARNING SYMPOSIUM

UC San Diego, April 8, 2020

REGISTER

INTRODUCTION

The Southern California Machine Learning Symposium brings together students and faculty to promote machine learning in the southern California region. The workshop serves as a forum for researchers from a variety of fields working on machine learning to share and discuss their latest findings.

Topics to be covered at the SoCal ML symposium include but are not limited to:

- Machine learning with graphs, social networks, and structured data.
- Active learning, reinforcement learning, crowdsourcing.
- Learning with images and natural language.
- Learning with high-dimensional data.
- Neural networks, deep learning, and graphical models.
- Learning dynamic and streaming data.
- Applications to interesting new domains.
- Addressing each of these issues at scale.
- New, open, or unsolved problems in machine learning theory or applications.

The majority of workshop will be focused on student and postdoc contributions, in the form of contributed talks and posters.

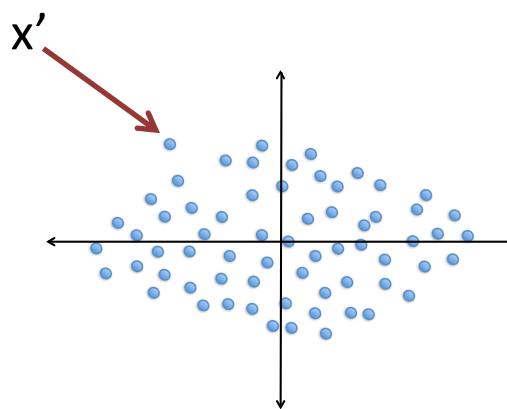
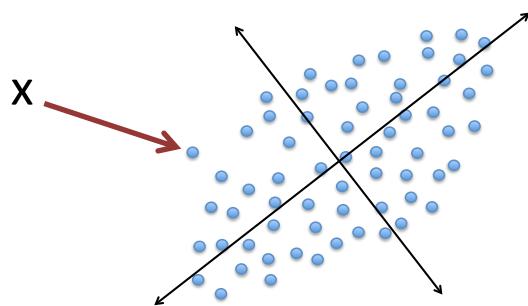
<https://cseweb.ucsd.edu/~jmcauley/workshops/scmls20/>

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

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

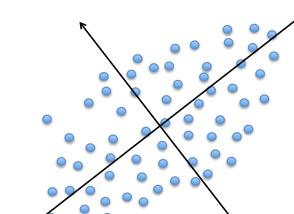
Transform into new coordinates

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

$$\text{proj}_{U_{1:K}}(x) = U_{1:K} U_{1:K}^T x$$

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

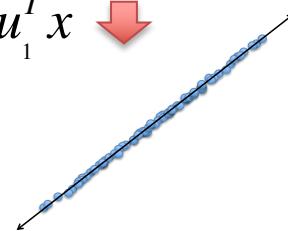
“Low Rank” Subspace



$$u_1^T x \downarrow$$



$$u_1 u_1^T x \downarrow$$



Recap: Singular Value Decomposition

$$X = [x_1, \dots, x_N] \in \mathbb{R}^{D \times N}$$

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

↑ ↑ ←
Orthogonal Diagonal Orthogonal

SVD

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

“Residual”

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

Recap: SVD & PCA

$$XX^T = U \Lambda U^T$$

PCA

Orthogonal Diagonal

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

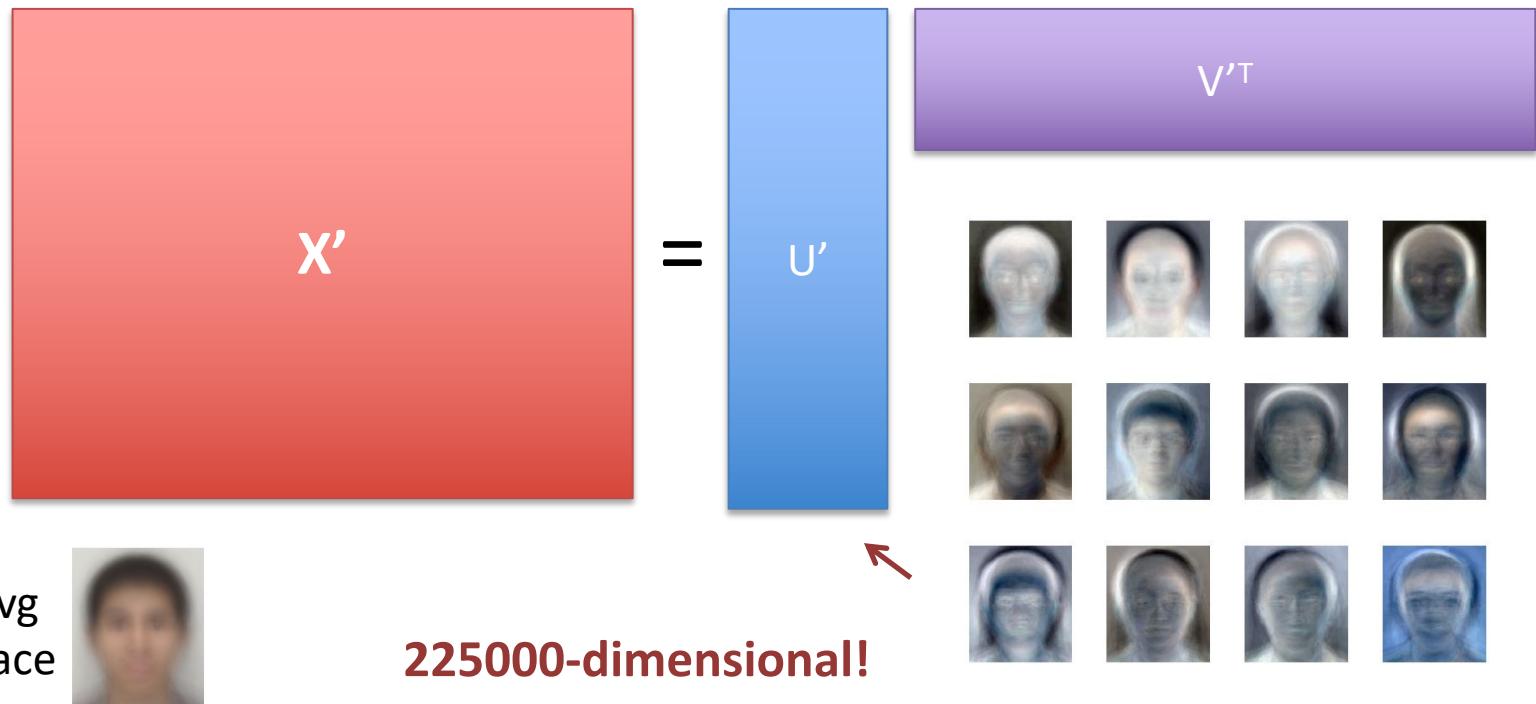
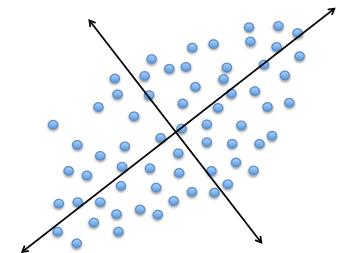
SVD

Orthogonal Diagonal Orthogonal

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

Recap: Eigenfaces

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



Matrix Norms

- Frobenius Norm

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

- Trace Norm

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

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

Each σ_d is guaranteed to be non-negative

By convention: $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_D \geq 0$

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

Properties of Matrix Norms

$$\begin{aligned}\|X\|_{Fro}^2 &= \text{trace}(X^T X) = \text{trace}\left(\left(U\Sigma V^T\right)^T U\Sigma V^T\right) \\ &= \text{trace}(V\Sigma^2 V^T) = \text{trace}(\Sigma^2 V^T V) \\ &= \text{trace}(\Sigma^2) = \sum_d \sigma_d^2\end{aligned}$$

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

Each σ_d is guaranteed to be non-negative

By convention: $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_D \geq 0$

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

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

Properties of Matrix Norms

$$\begin{aligned}\|X\|_* &= \text{trace}\left(\sqrt{\left(U\Sigma V^T\right)^T U\Sigma V^T}\right) = \text{trace}\left(\sqrt{V\Sigma U^T U\Sigma V^T}\right) \\ &= \text{trace}\left(\sqrt{V\Sigma\Sigma V^T}\right) = \text{trace}\left(\sqrt{V\Sigma^2 V^T}\right) = \text{trace}\left(V\Sigma V^T\right) \\ &= \text{trace}\left(\Sigma V^T V\right) = \text{trace}\left(\Sigma\right) = \sum_d \sigma_d\end{aligned}$$

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

Each σ_d is guaranteed to be non-negative

By convention: $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_D \geq 0$

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

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

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

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

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!

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

- L1 induces sparsity
 - And is continuous!

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

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!

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

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:

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

- AM-GM Inequality:

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

True for any norm

- Orthogonal Transformation Invariance of Norms:

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

If U is a full-rank orthogonal matrix

- Trace Norm of Diagonals

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

If A is a square diagonal matrix

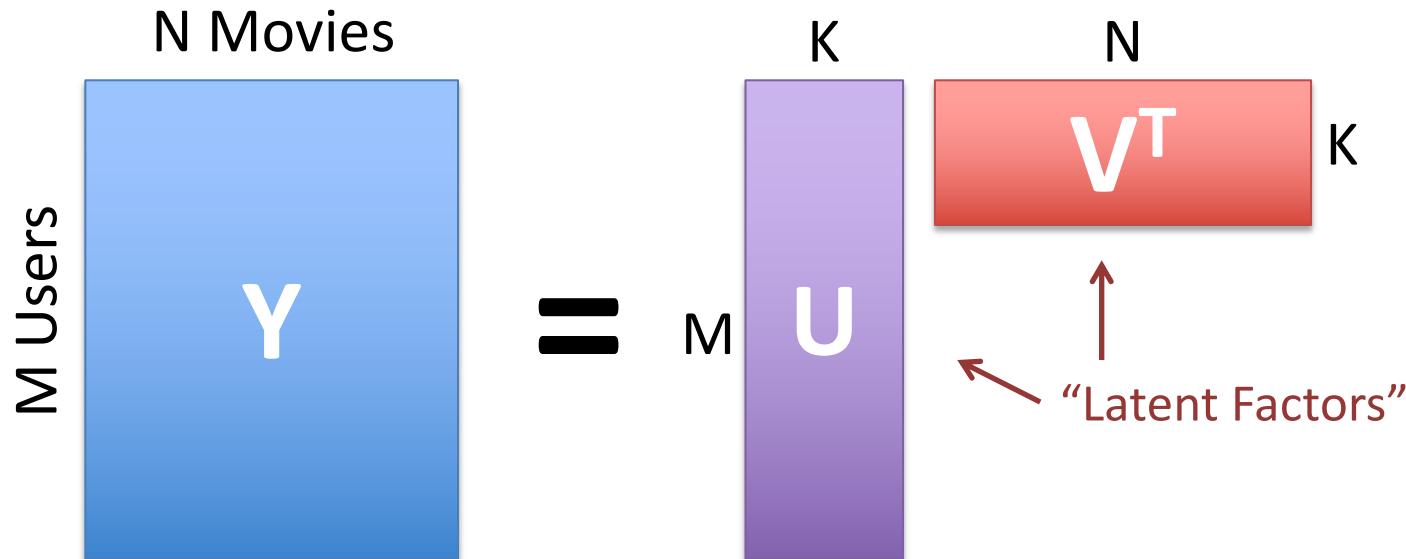
Recap: SVD & PCA

- 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:

$$\|X - U_{1:K}U_{1:K}^TX\|_{Fro}^2$$

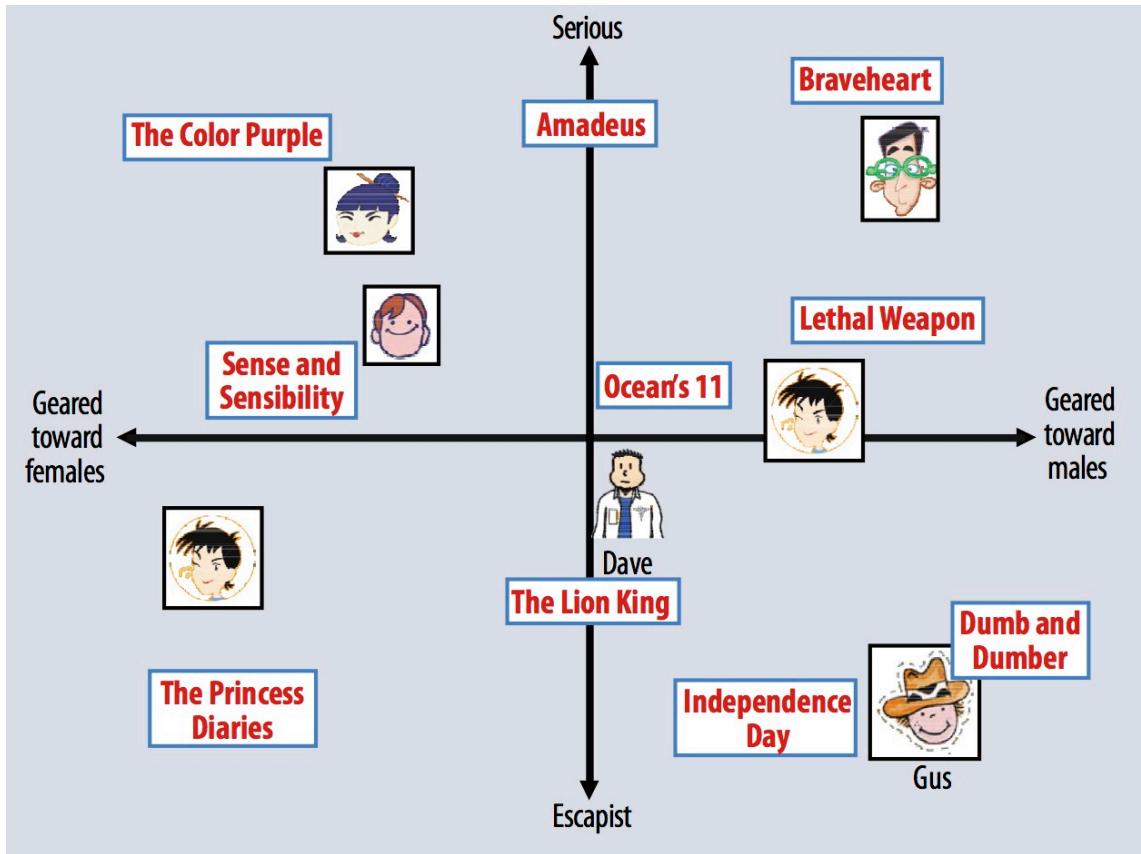
Latent Factor Models

Netflix Problem



- Y_{ij} = rating user i gives to movie j
$$y_{ij} \approx u_i^T v_j$$
- Solve using SVD!

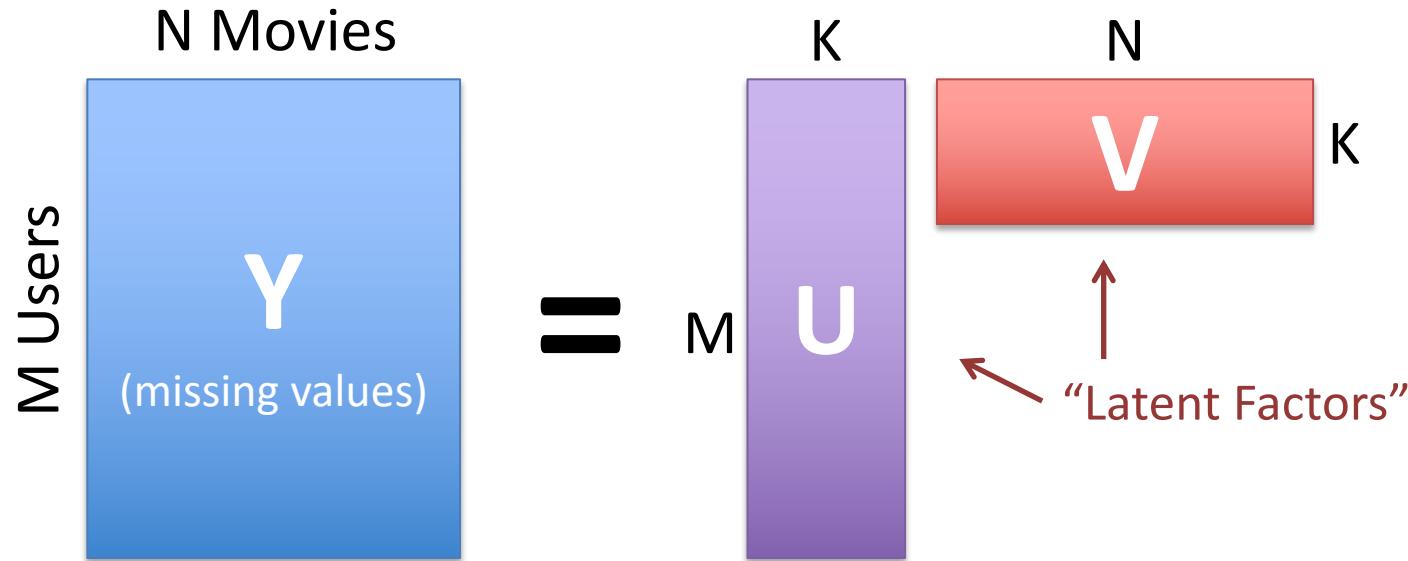
Example



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

<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:

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

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

Then:

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

Interpreting Model

- Latent-Factor Model Objective

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

- Related to:

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

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 $U, V = \text{rank of } W$

User/Movie Symmetry

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

- 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

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

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

Gradient Calculation

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

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

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

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

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

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)

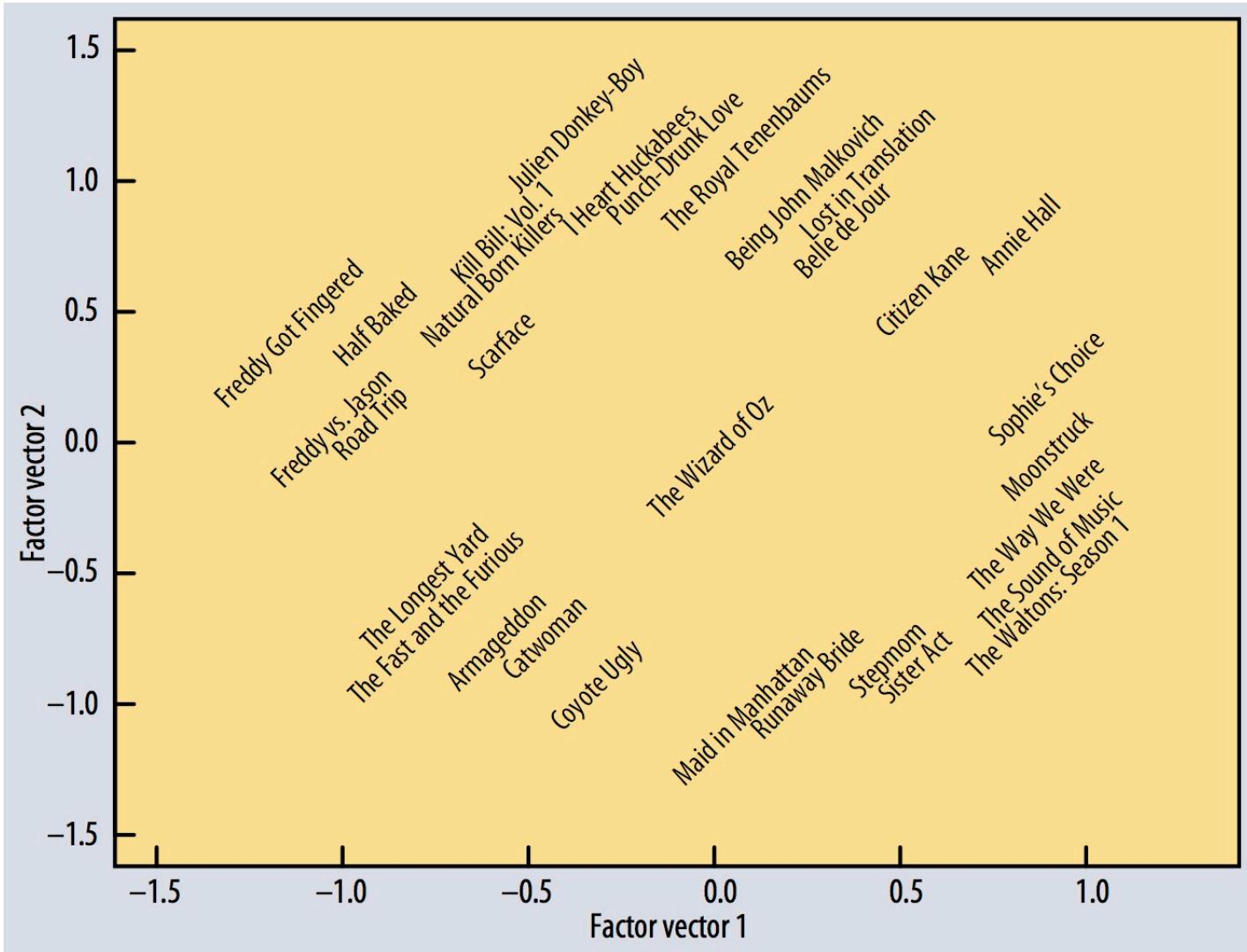
Tradeoffs

- 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

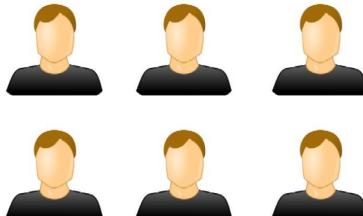
- M Tasks:

$$\underset{w}{\operatorname{argmin}} \frac{\lambda}{2} R(W) + \frac{1}{2} \sum_m \sum_i (y_i - w_m^T x_i)^2$$

↑
Regularizer

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

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



How to Regularize?

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

- Standard L2 Norm:

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

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

How to Regularize?

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

- Trace Norm:

$$\operatorname{argmin}_W \frac{\lambda}{2} \|W\|_* + \sum_m \sum_i (y_i - w_m^T x_i)^2$$

- 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?

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

- Latent Factor Approach

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

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

Tradeoff

- D*N parameters:

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

- D*K + N*K parameters:

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

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

Multitask Learning

- M Tasks:

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

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

- 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

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

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

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

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

Latent Factor Multitask Learning vs Collaborative Filtering

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

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

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

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

General Bilinear Models

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

- 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?

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

U: MxK
V: NxK

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

U: MxK
V: DxK

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

U: FxK
V: DxK

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

Story So Far: Latent Factor Models

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

- **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”

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

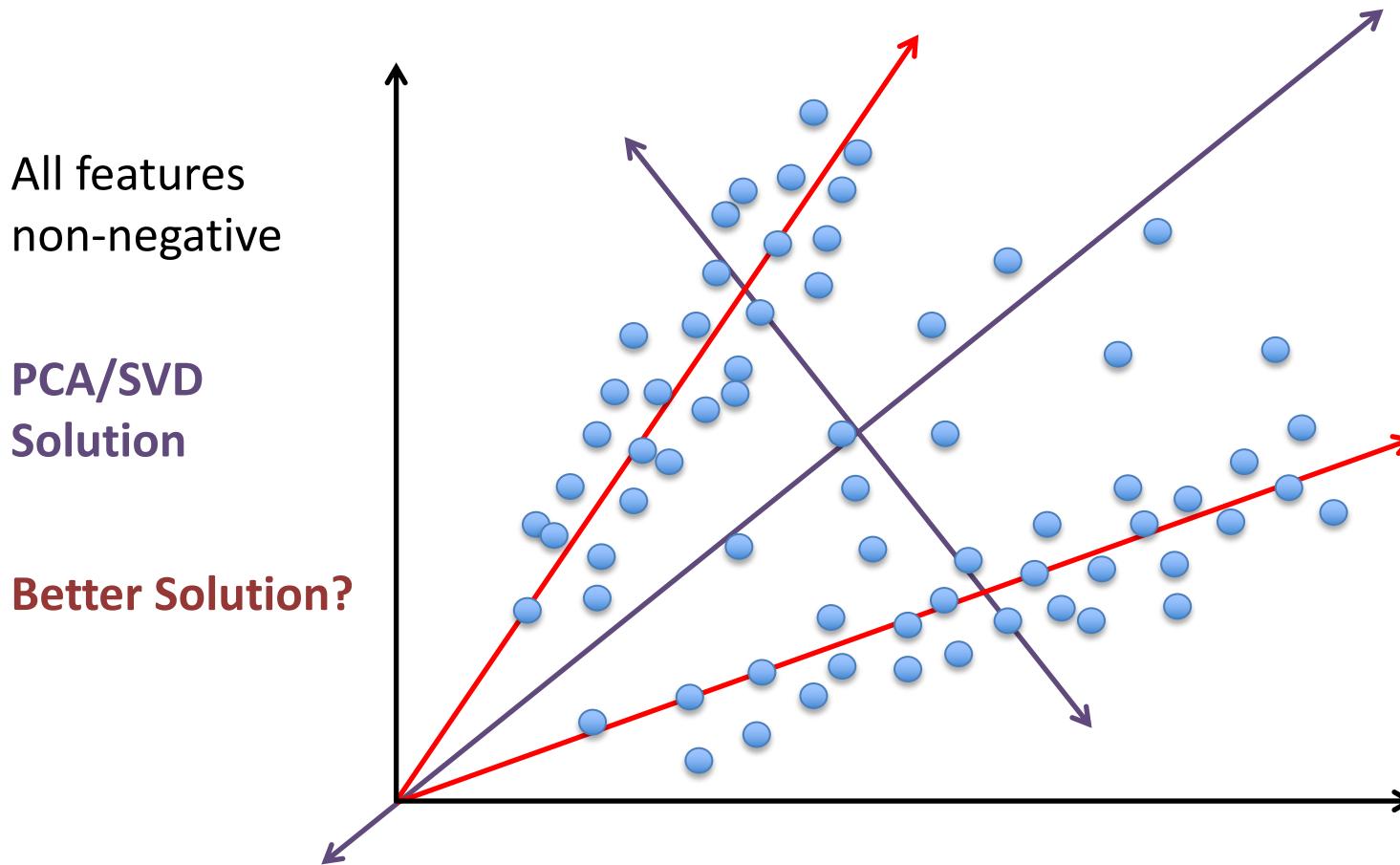
- Vx is a linear mapping
 - From x to low-dimensional space
- Can also learn non-linear mapping
 - E.g., the hidden layer activations of a neural net



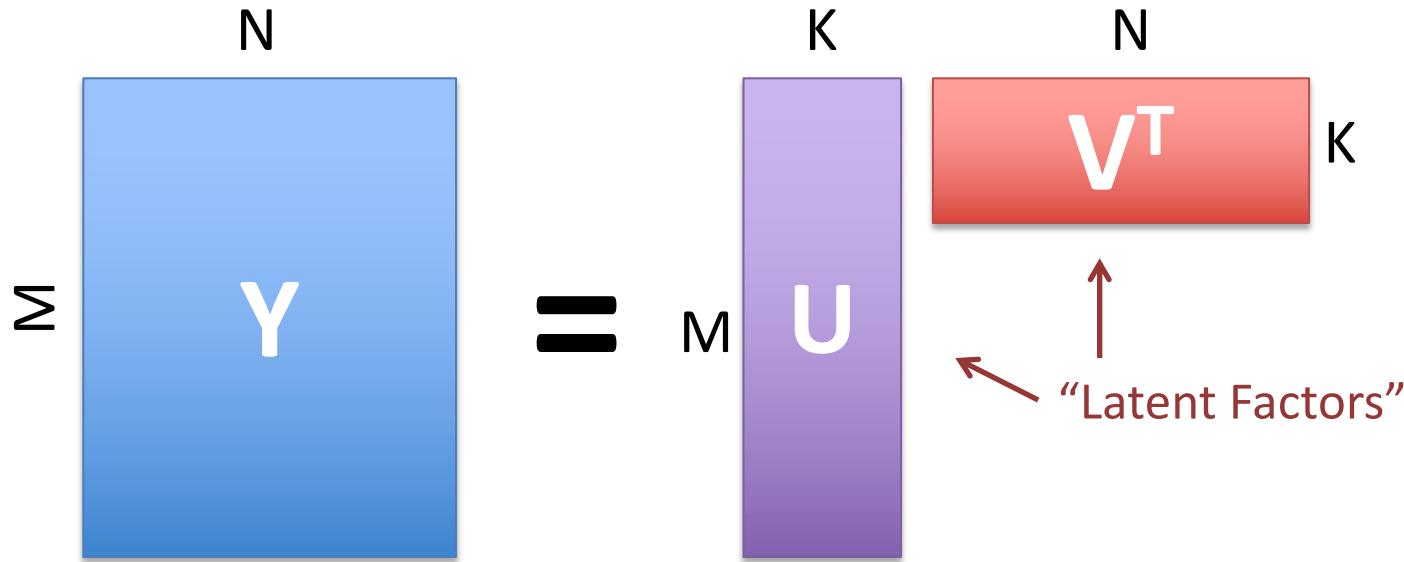
<http://www.cs.cornell.edu/~andreas/iccv15.pdf>

Non-Negative Matrix Factorization

Limitations of PCA & SVD



Non-Negative Matrix Factorization



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

CS 155 Non-Negative Face Basis

1.2115 , 65.8148



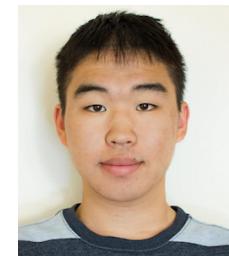
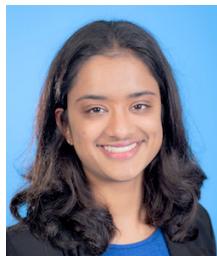
1.0074 , 25.5445



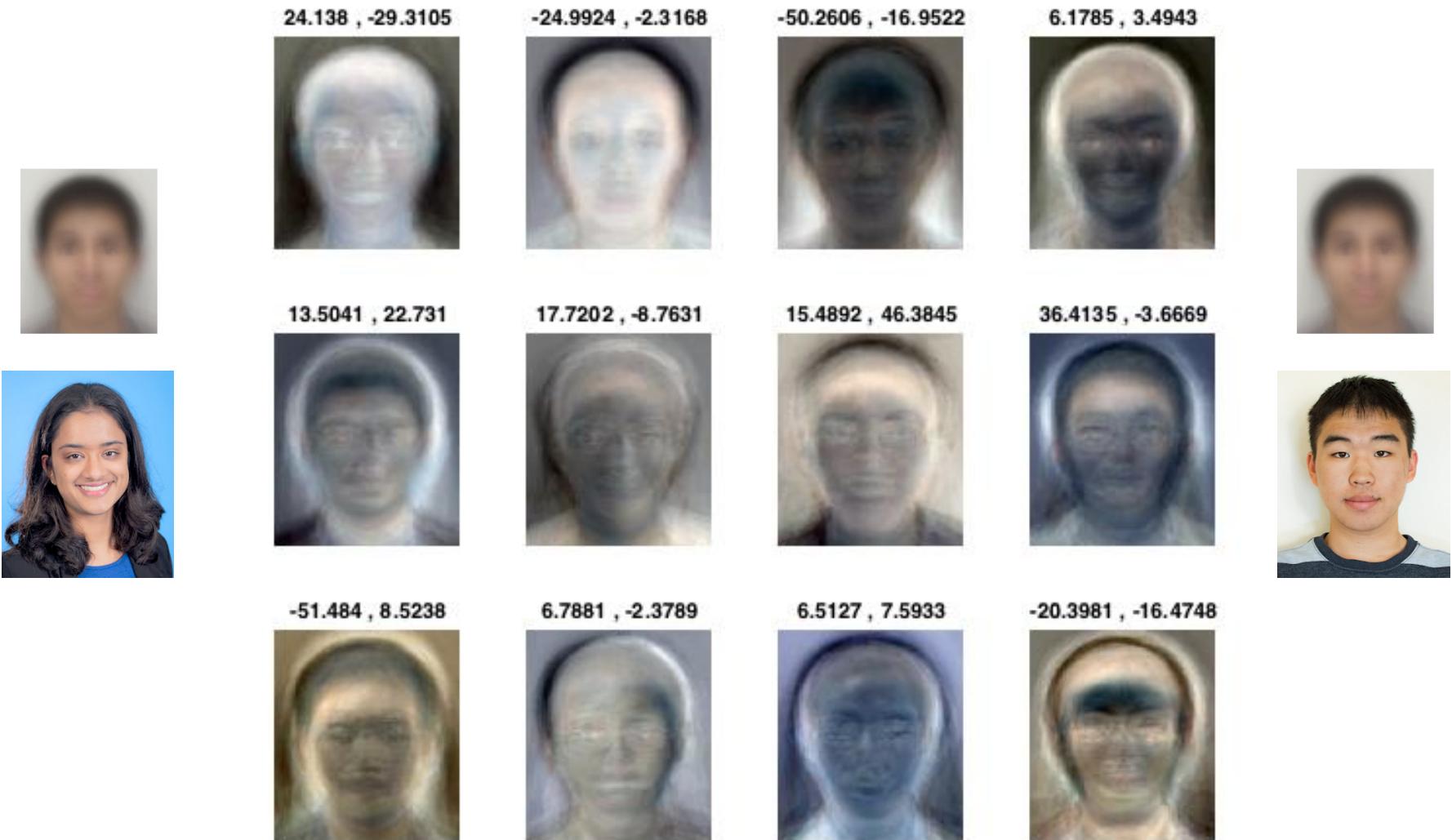
4.3367e-05 , 10.4963



1.6509e-14 , 74.2559



CS 155 Eigenfaces



Aside: Non-Orthogonal Projections

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

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

SVD

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

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

Pseudoinverse

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

$$\begin{aligned} A^{+T} A^T x &= U \Sigma^+ V^T V \Sigma U^T x \\ &= U_{1:K} U_{1:K}^T x \end{aligned}$$

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
- Generalized Relative Entropy
 - Aka, unnormalized KL divergence
 - Penalizes ratio
- Train using gradient descent

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

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

<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

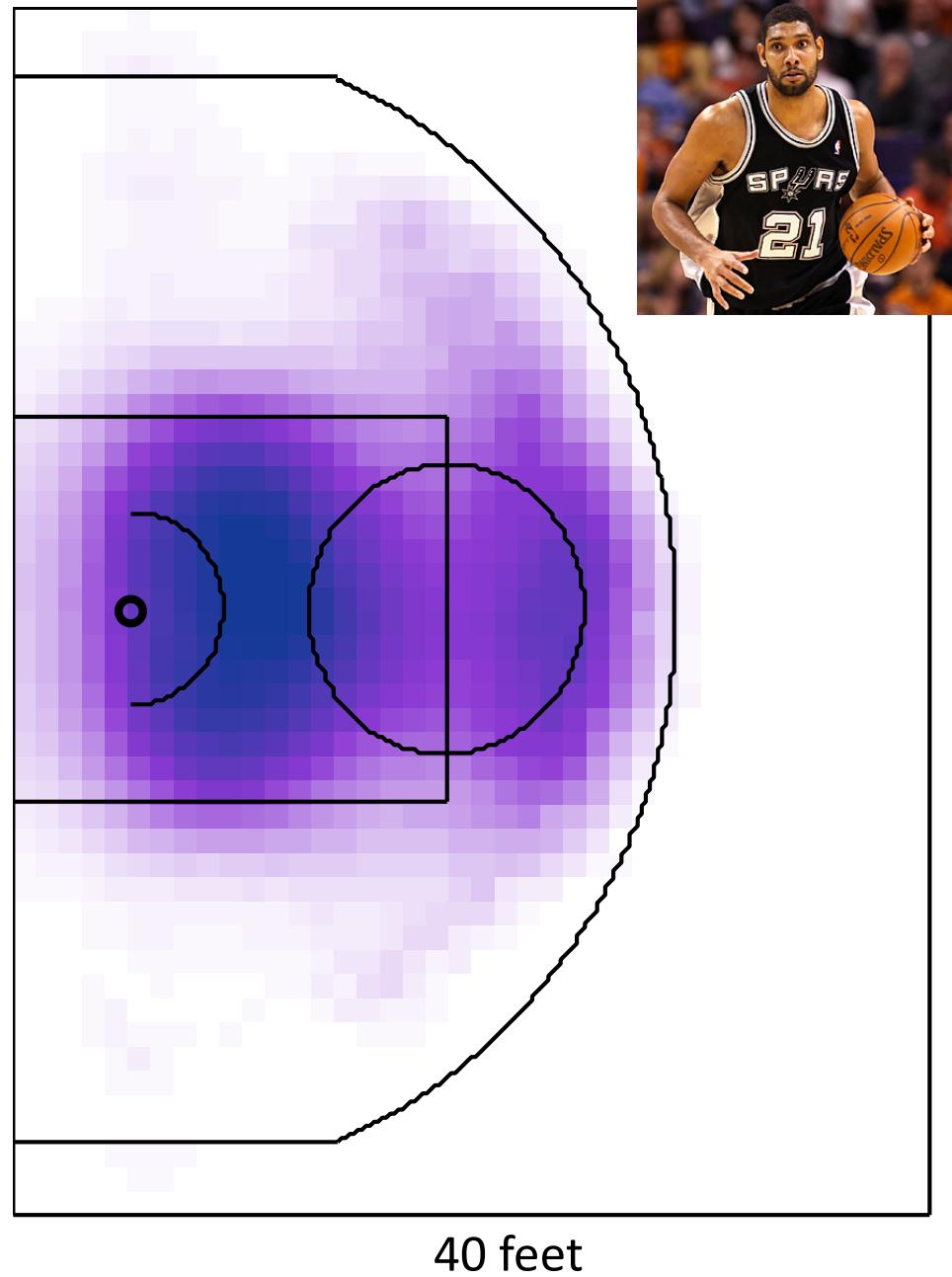
$$\operatorname{argmin}_{U,V} \frac{\lambda}{2} \left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2 \right) + \sum_i \ell(y_i, z_i^T U^T V x_i) \quad S = \{(x_i, z_i, y_i)\}$$

- **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



Fine-Grained Spatial Models

- Discretize court

- 1x1 foot cells

- 2000

- But most players haven't
played that much!

- 1 weight per cell

- 2000 weights

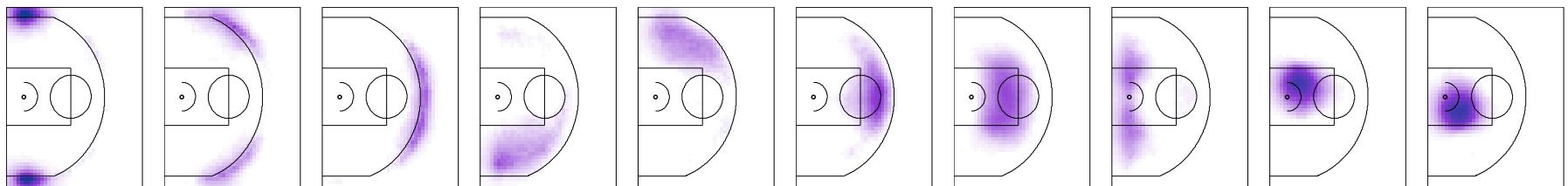
$$F_s(\mathbf{x}) :$$

40 feet

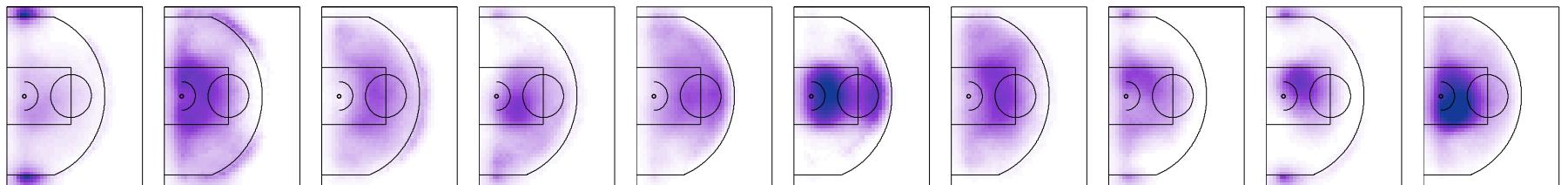
$$M \begin{matrix} D \\ F_s \end{matrix} = M \begin{matrix} K \\ B^T \end{matrix} = M \begin{matrix} L \\ K \end{matrix}$$

Location Factors

Player Factors



Visualizing location factors L



Kawhi
Leonard

Carmelo
Anthony

Dirk
Nowitzki

Dion
Waiters

John
Wall

Tim
Duncan

Kyrie
Irving

Shawn
Marion

Jeremy
Lin

David
Lee

Visualizing players $B_b L$

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

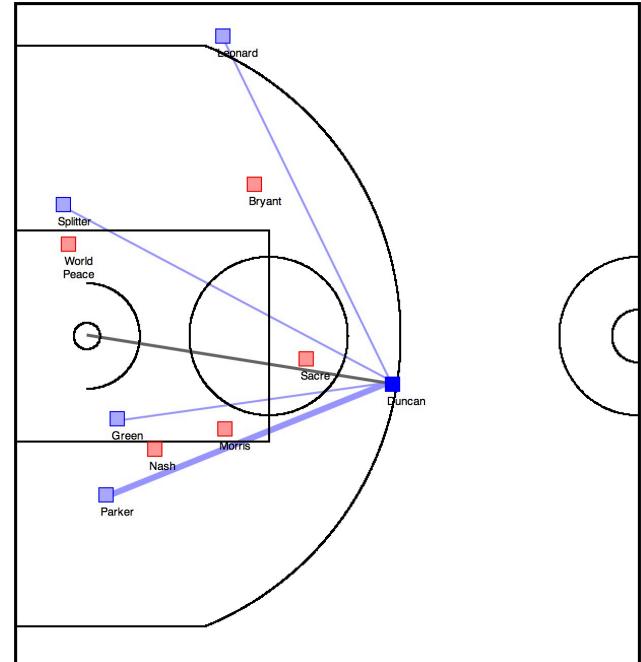
Training Data



STATS SportsVU
2012/2013 Season, 630 Games,
80K Possessions, 380 frames per possession

Prediction

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



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

Logistic Regression

(Simple Version: Just for Shooting)

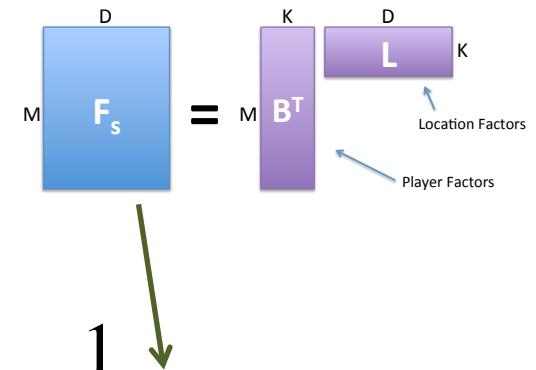
$$P(y \mid \mathbf{x}) = \frac{\exp\{F(y \mid \mathbf{x})\}}{Z(\mathbf{x} \mid F)}$$

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

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

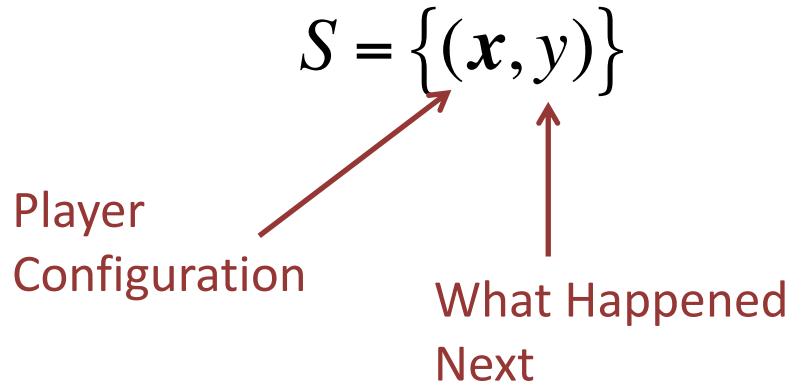
Offset or bias

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



Learning the Model

- Given training data:



- Learn parameters of model:

$$\operatorname{argmin}_{F_s, F_\perp} \frac{\lambda}{2} \|F_s\|^2 + \sum_{(x,y) \in S} \ell(y, F_s(x) - F_\perp)$$

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

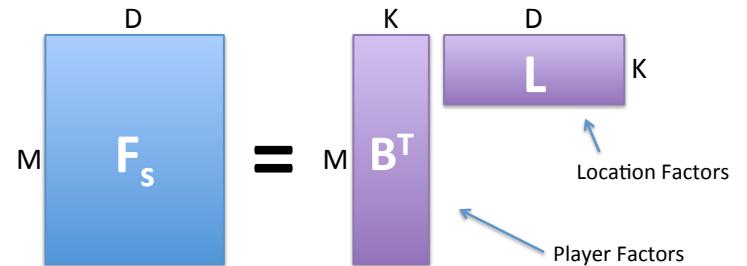
Log Loss

Optimization via Gradient Descent

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

$$\partial_{L_i} = \lambda_l L_i - \sum_{(\mathbf{x}, y)} \frac{\partial \log P(y | \mathbf{x})}{\partial L_i}$$

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



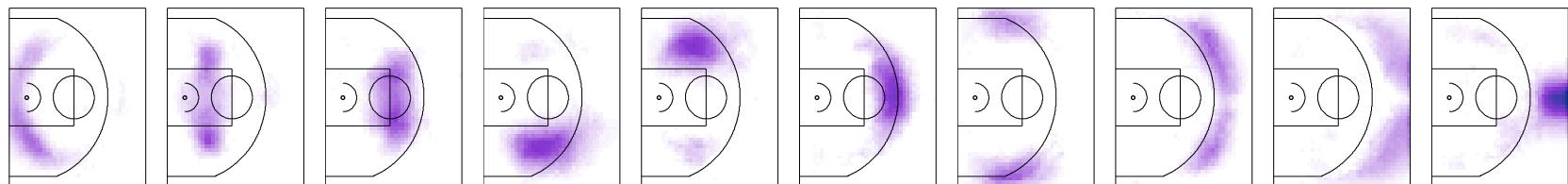
http://www.yisongyue.com/publications/icdm2014_bball_predict.pdf

Where are Players Likely to Receive Passes?

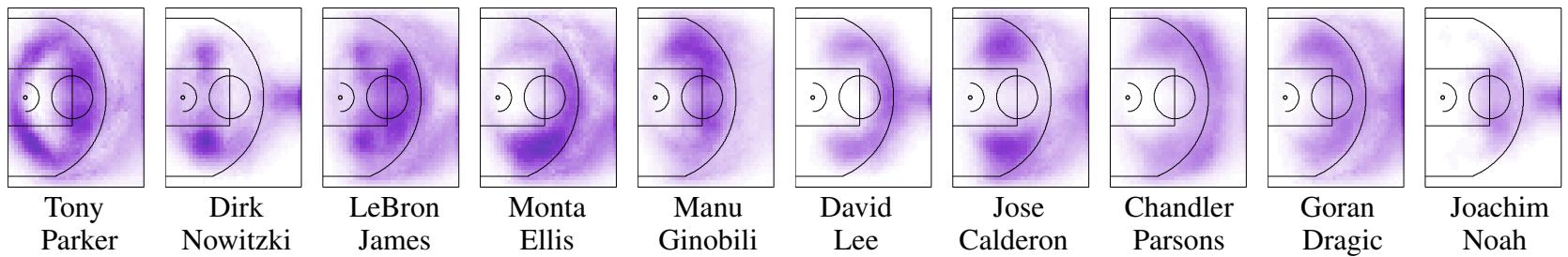
$$\begin{matrix} D \\ M \end{matrix} \mathbf{F}_p^1 = \begin{matrix} K \\ M \end{matrix} \mathbf{P}^T \quad \begin{matrix} D \\ M \end{matrix} \mathbf{M}^K$$

Location Factors
Player Factors

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



Visualizing Location Factors \mathbf{M}

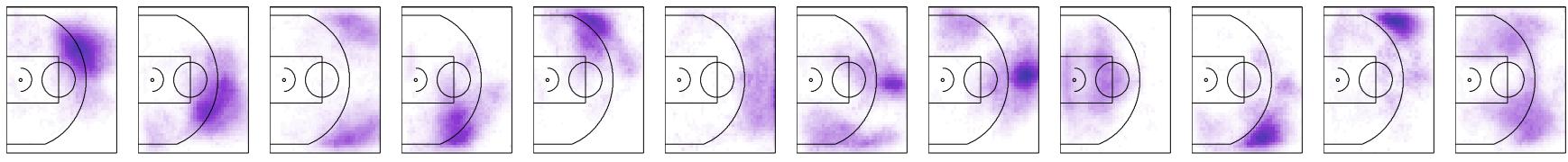


http://www.yisongyue.com/publications/icdm2014_bball_predict.pdf

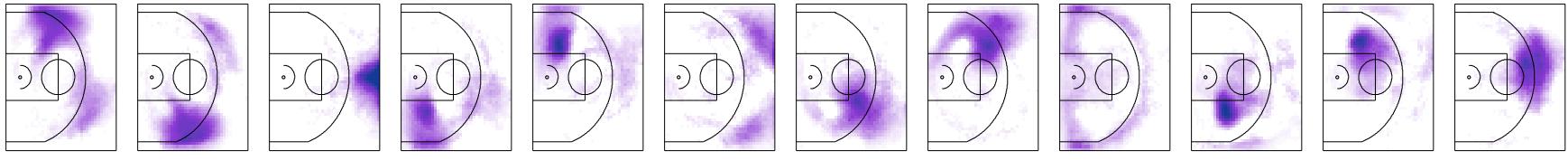
How do passes tend to flow?

$$\begin{matrix} D \\ D \end{matrix} F_p^2 = \begin{matrix} D \\ K \end{matrix} Q_1^T \begin{matrix} D \\ K \end{matrix} Q_2$$

Target Location Factors
Source Location Factors



Q_1



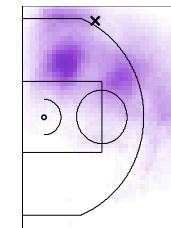
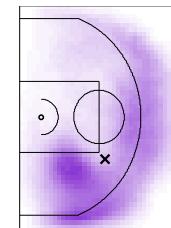
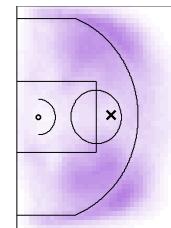
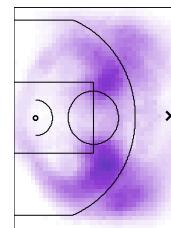
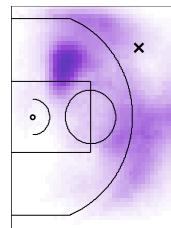
Q_2

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

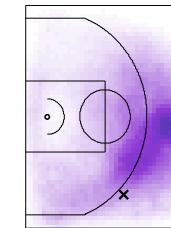
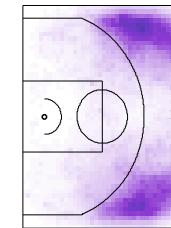
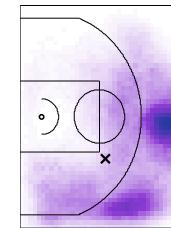
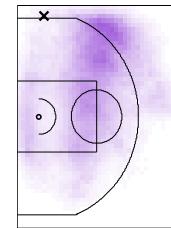
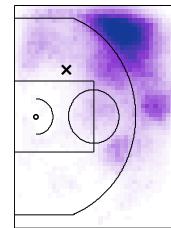
How do passes tend to flow?

$$\begin{matrix} D \\ D \end{matrix} F_p^2 = \begin{matrix} D \\ D \end{matrix} Q_1^T \begin{matrix} K \\ D \\ Q_2 \end{matrix}^K$$

Target Location Factors
Source Location Factors



Passing From “X”

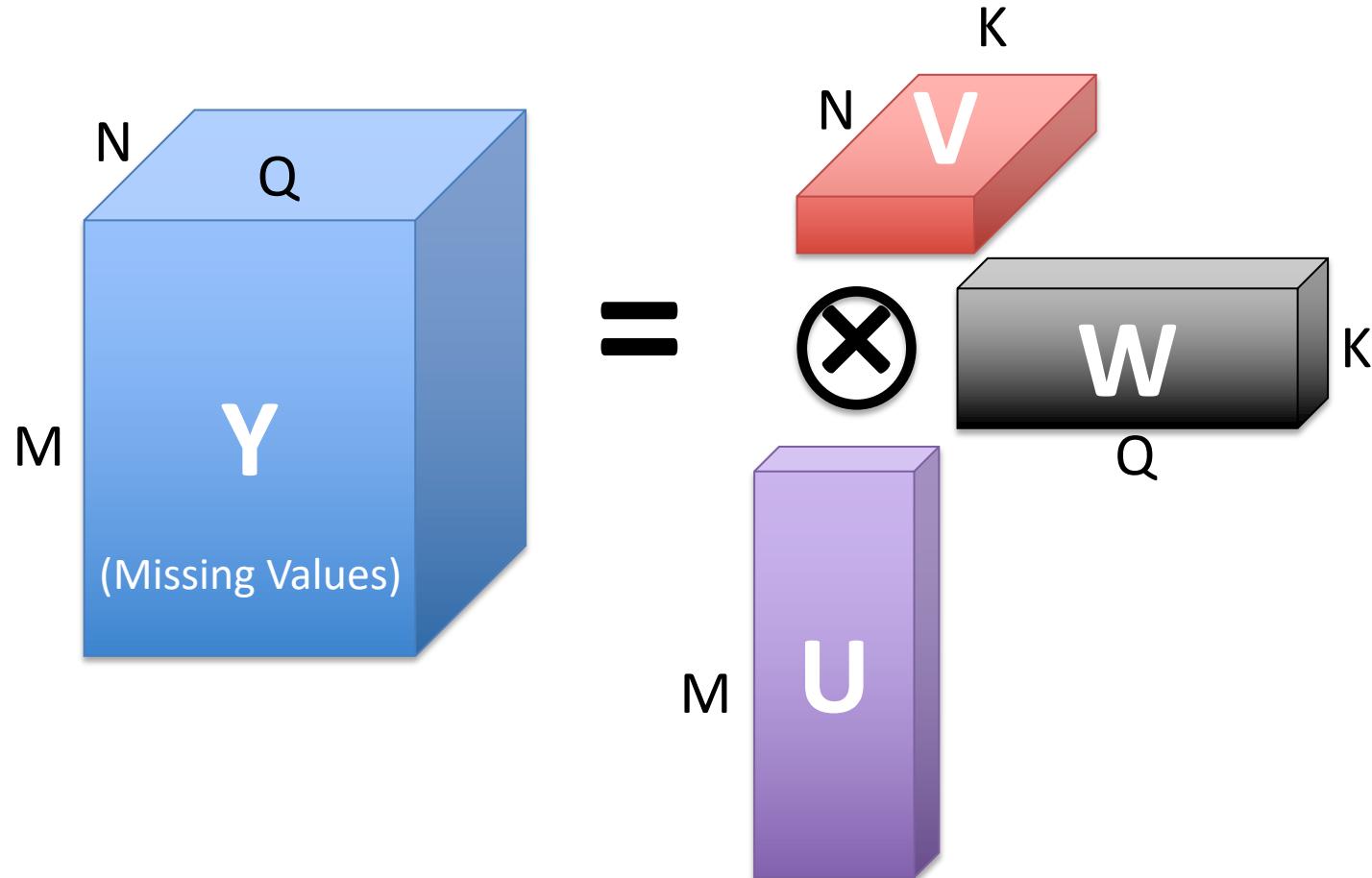


Passing To “X”

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

Tensor Latent Factor Models

Tensor Factorization



Tri-Linear Model

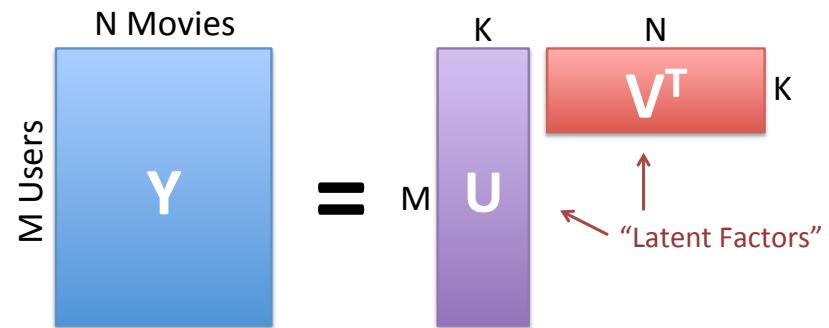
$$\operatorname{argmin}_{U,V,W} \frac{\lambda}{2} \left(\|U\|_{Fro}^2 + \|V\|_{Fro}^2 + \|W\|_{Fro}^2 \right) + \sum_i \ell \left(y_i, \langle U^T z_i, V^T x_i, W^T q_i \rangle \right)$$

- 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 \mathbf{Y}
 - Dimensions of \mathbf{Y} can have various semantics
- Can tolerate missing values in \mathbf{Y}
- Can also use features
- Widely used in industry



Next Lecture

- Embeddings
- Word2Vec