# CPSC 340: Machine Learning and Data Mining

Recommender Systems

Fall 2015

# Admin

- Assignment 4 posted:
  - Due Friday of next week.
- Midterm being marked now.

# Non-Negativity vs. L1-Regularization

- Last time we discussed how non-negativity leads to sparsity

$$\underset{w \geq 0}{\text{argmin}} \; \frac{1}{2}\|y - Xw\|^2$$

- Which is an alternative to L1-regularization.

$$\underset{w}{\text{argmin}} \; \frac{1}{2}\|y - Xw\|^2 + \lambda\|w\|_1$$

- Sparsity level is fixed with non-negative, variable with L1 (via λ).
- You can do both:

$$\underset{w \geq 0}{\text{argmin}} \; \frac{1}{2}\|y - Xw\|^2 + \lambda\sum_{j=1}^{d} w_j$$

- This enforces non-negativity, and you can control sparsity level.
- Can be solve with projected-gradient, since it's differentiable:
  - Some of the best methods for L1-regularization use this.

# PCA for Compression

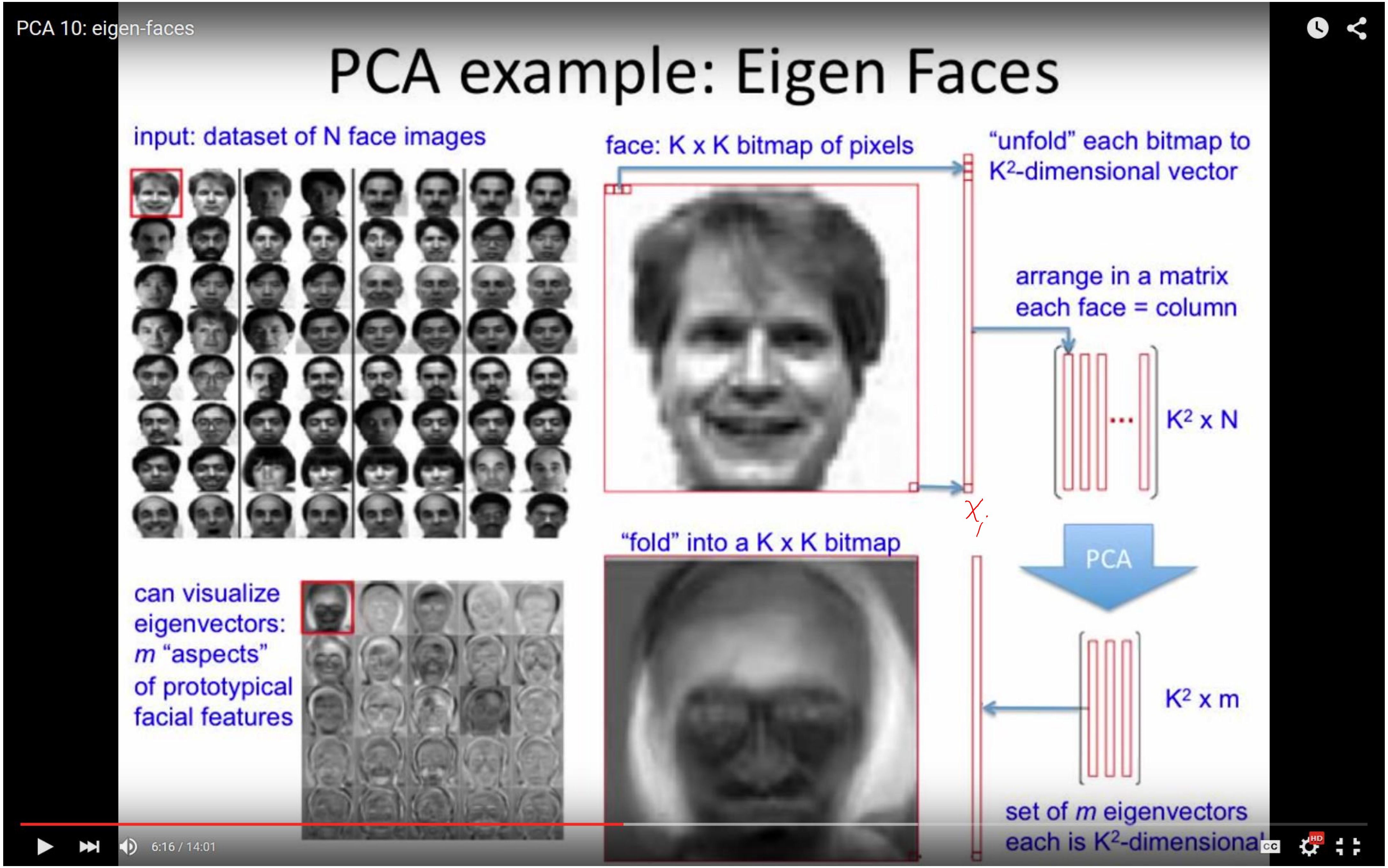
- Generalization of Euclidean norm to matrices is 'Frobenius' norm:

$$\|X\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{d} x_{ij}^2}$$

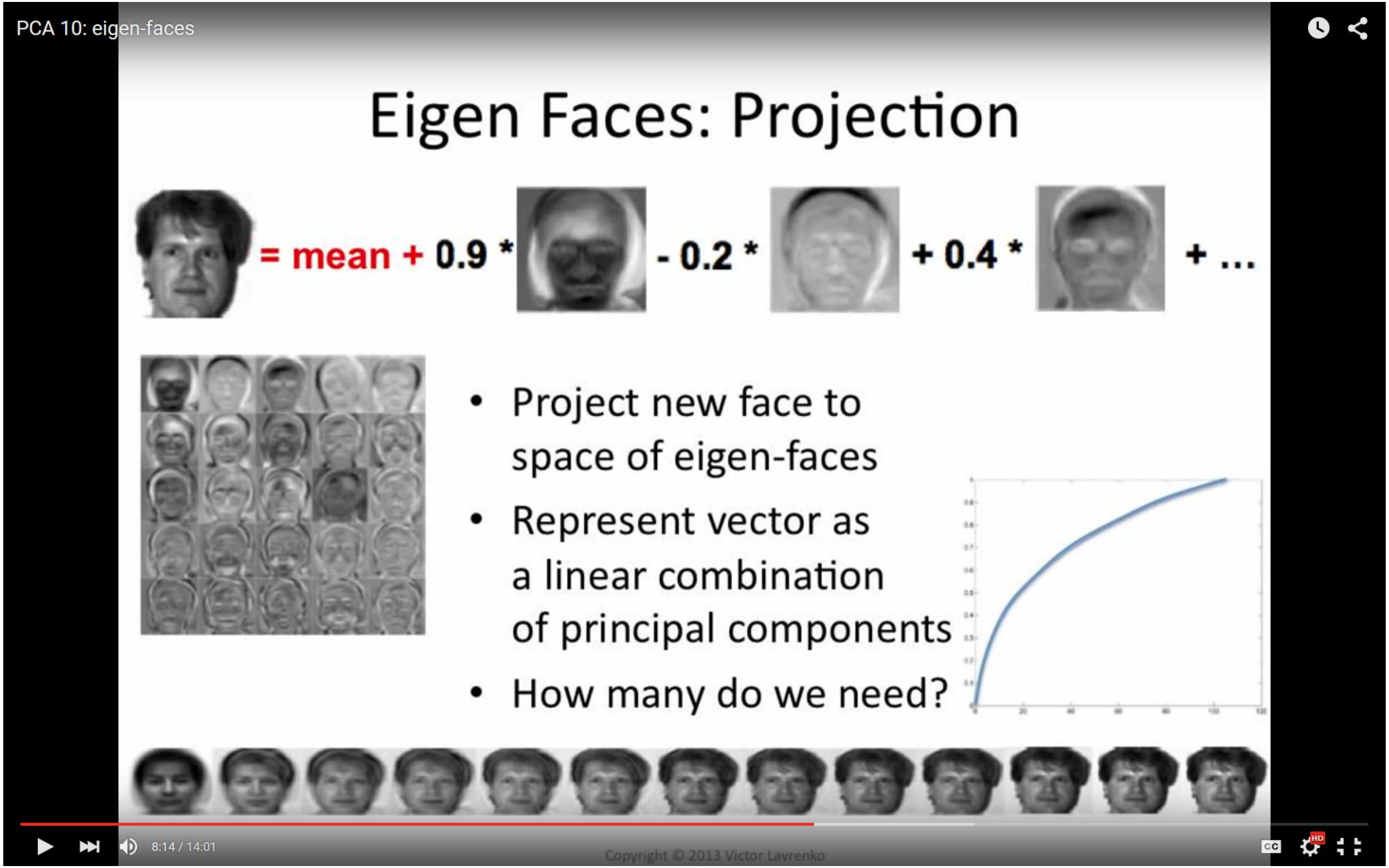
- Viewing latent-factor model as approximation, $X \approx ZW$
- Standard latent-factor model minimizes Frobenius norm:

$$\sum_{i=1}^{n} \sum_{j=1}^{d} (x_{ij} - w_j^T z_i)^2 = \|X - ZW\|_F^2$$

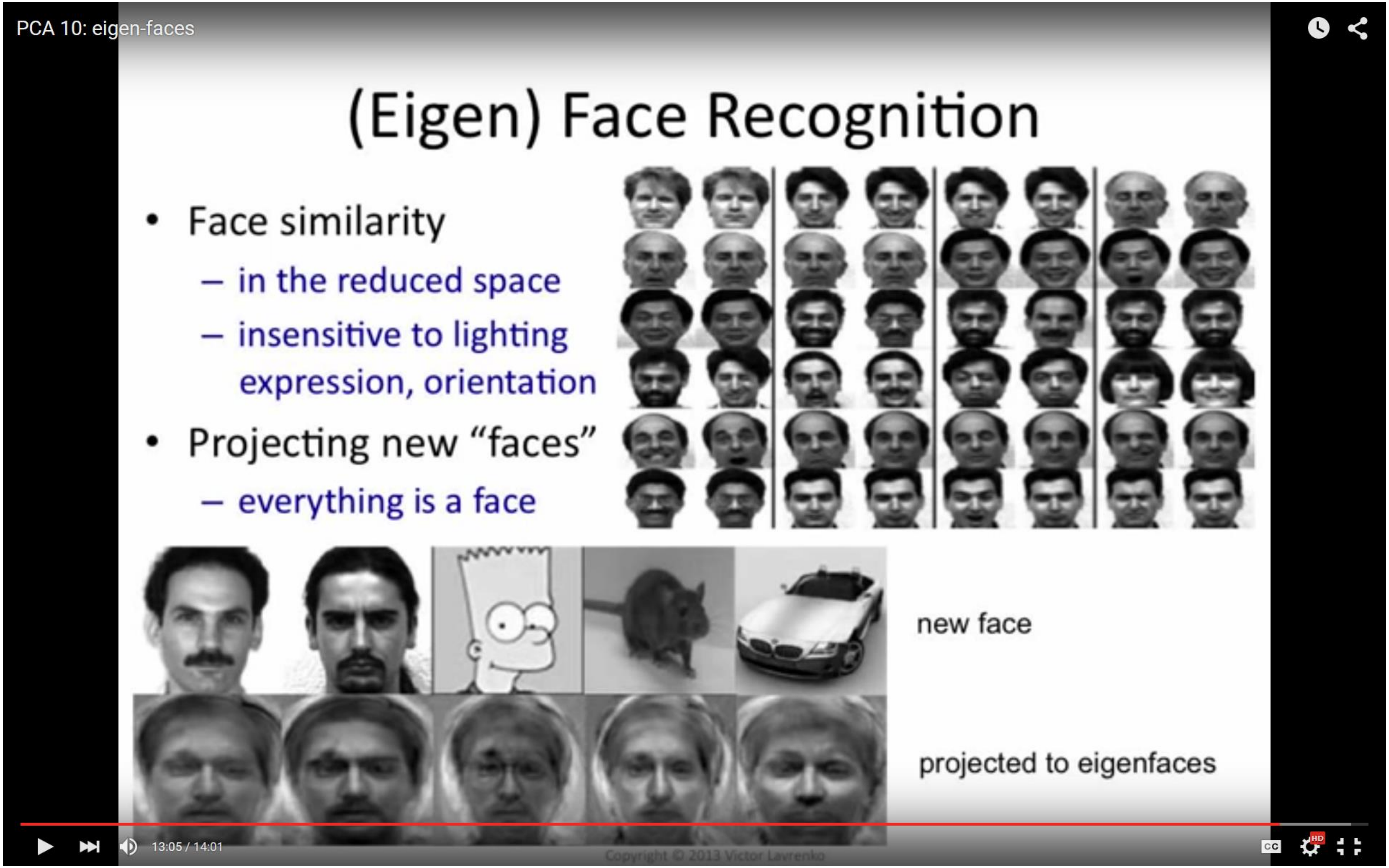
- For fixed 'k', PCA optimally compresses in terms of 'W' and 'Z'.
  – Though NMF can be even smaller due to sparsity in 'W' and 'Z'.




https://www.youtube.com/watch?v=_IY74pXWlS8




https://www.youtube.com/watch?v=_lY74pXWlS8



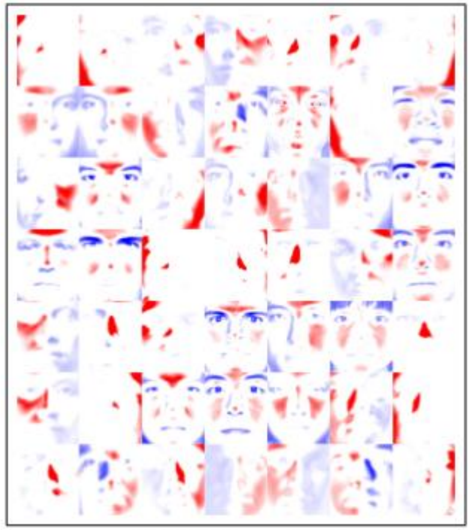

https://www.youtube.com/watch?v=_lY74pXWlS8

# Last time: Sparse Latent-Factor Models

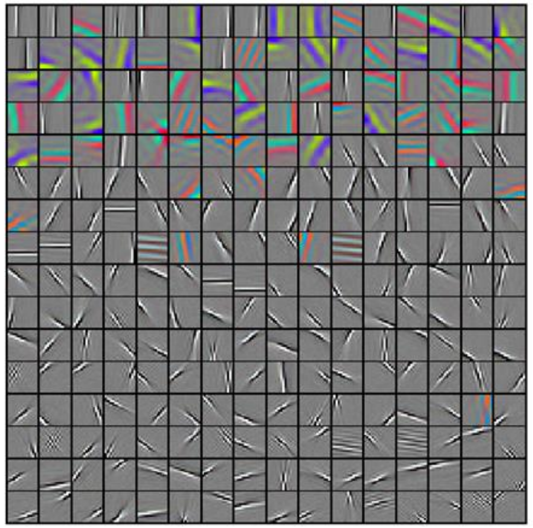
- The latent-factor model framework we've been looking at:

$$f(W,Z) = \sum_{i=1}^{n}\sum_{j=1}^{d}(x_{ij} - w_j^T z_i)^2$$

- The $w_c$ are 'latent factors', and $z_i$ is low-dimensional representation.
- Last time we consider ways to encourage sparsity in W or Z.
- Leads to representations of faces as sum of face 'parts'.
- Biologically-plausible image patch representations (depends on sensors).



(d) SPCA, $\tau = 30\%$



http://www.jmlr.org/papers/volume11/mairal10a/mairal10a.pdf
http://lear.inrialpes.fr/people/mairal/resources/pdf/review_sparse_arxiv.pdf

# Recommender System Motivation: Netflix Prize

- Netflix Prize:
  - 100M ratings from 0.5M users on 18k movies.
  - Grand prize was $1M for first team to reduce error by 10%.
  - Started on October 2nd, 2006.
  - Netflix's system was first beat October 8th.
  - 1% error reduction achieved on October 15th.
  - Steady improvement after that.
    - ML methods soon dominated.
  - One obstacle was 'Napolean Dynamite' problem:
    - Some movie ratings seem very difficult to predict.
    - Should only be recommended to certain groups.

# Lessons Learned from Netflix Prize

- Prize awarded in 2009:
  - Ensemble method that averaged 107 models.
  - Increasing diversity of models more important than improving models.


- Winning entry (and most entries) used collaborative filtering:
  - Only look at ratings, not features of movies/users.
- You also do really well with a simple collaborative filtering model:
  - Regularized SVD is latent-factor model now adopted by many companies.

http://bits.blogs.nytimes.com/2009/09/21/netflix-awards-1-million-prize-and-starts-a-new-contest/?_r=0

# Motivation: Other Recommender Systems

- Recommender systems are now everywhere:
  - Music, news, books, jokes, experts, restaurants, friends, dates, etc.
- Main types approaches:
  1. Content-based filtering:
     - Extract features $x_i$ of users and items, building model to predict rating $y_i$ given $x_i$.
     - Usual supervised learning: allows prediction for new users/items.
     - Example: G-mail's 'important messages' (personalization with 'local' features).
  2. Collaborative filtering:
     - Try to predict $y_{ij}$ given $y_{ik}$ for other items 'k' and $y_{kj}$ for other users 'k'.
     - Needs more data about individual users/products, but doesn't need features.
     - Example: Amazon recommendation algorithm (uses $y_{kj}$ for other users 'k').

# Collaborative Filtering Problem

- Collaborative filtering is 'filling in' the user-item matrix:

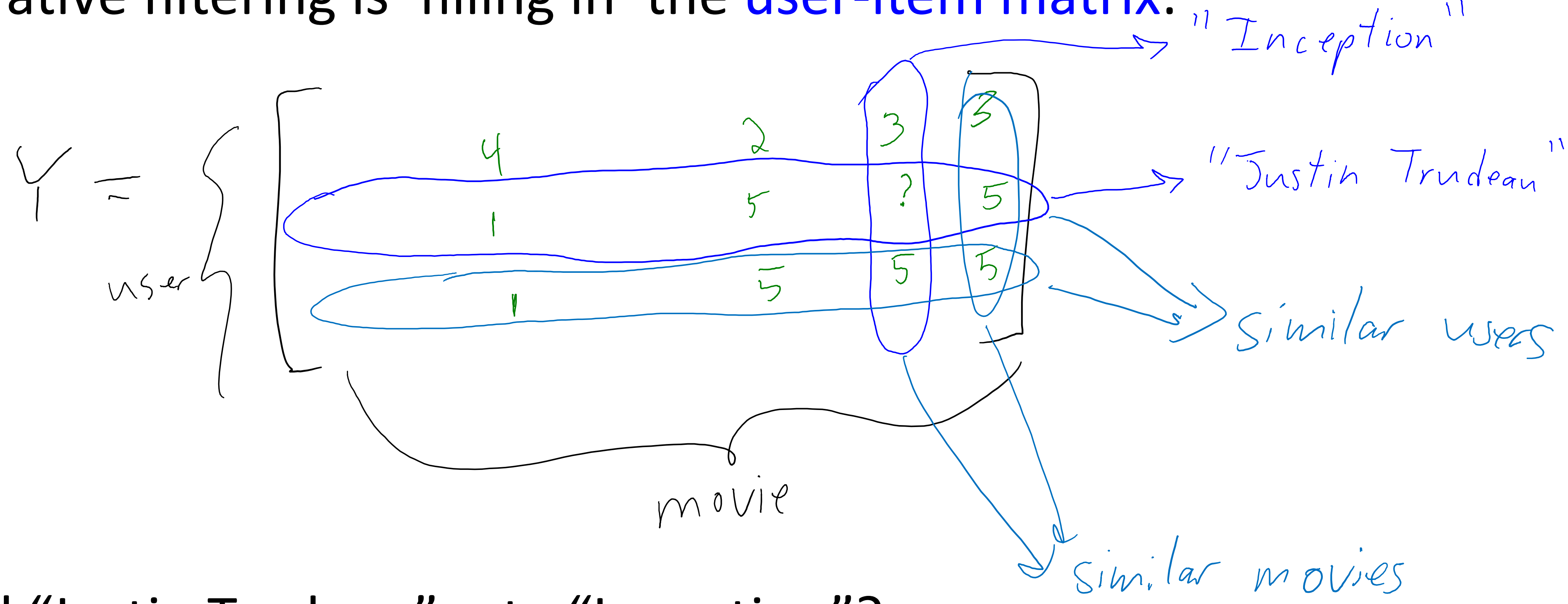
"Inception"

$$Y = \text{user}\left\{ \underbrace{\begin{bmatrix} 4 & 2 & 3 & 3 \\ 1 & 5 & ? & 5 \\ 1 & 5 & 5 & 5 \end{bmatrix}}_{\text{movie}} \right. \quad \text{"Justin Trudeau"}$$

- How will "Justin Trudeau" rate "Inception"?

# Collaborative Filtering Problem

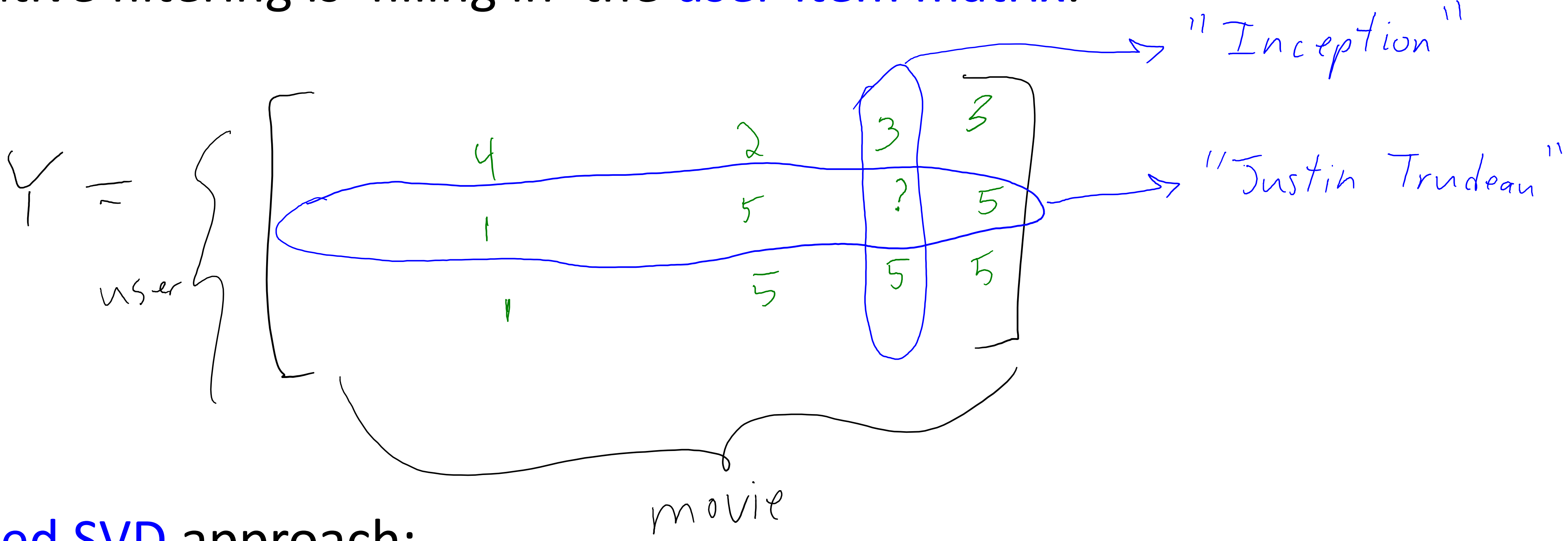
- Collaborative filtering is ‘filling in’ the user-item matrix:




- How will “Justin Trudeau” rate “Inception”?

# Collaborative Filtering Problem

- Collaborative filtering is 'filling in' the user-item matrix:




- Regularized SVD approach:
  - Assume each user 'i' has latent features $z_i$.
  - Assume each item 'j' has latent features $w_j$.
  - Learn these features from the available entries.
  - Use regularization to improve test error.

# Regularized SVD

- Our standard latent-factor framework:

$$\underset{W,Z}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{d}(y_{ij}-w_j^Tz_i)^2$$

- But don't include missing entries in loss:

$$\underset{W,Z}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{d}I[y_{ij}\neq ?](y_{ij}-w_j^Tz_i)^2$$

$$I[y_{ij}=?]=\begin{cases}1 & \text{if we know } y_{ij}\\ 0 & \text{if we don't know } y_{ij}\end{cases}$$

(so we only compute error for ratings we know)

- We have a 'k' by '1' latent-vector for each user 'i' and item 'j':
  - 'k' is like the number principal components.
  - $z_i$ could reflect things like 'user likes romantic comedies'.
  - $w_j$ could reflect things like 'movie has Nicolas Cage'.
  - But you don't need explicit user/item features.

# Regularized SVD

- Add L2-regularization to improve test error:

$$\underset{W,Z}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{d} I[y_{ij} \neq ?](y_{ij} - w_j^T z_i)^2 + \frac{\lambda_W}{2}\underbrace{\sum_{j=1}^{d}\|w_j\|^2}_{=\|W\|_F^2} + \frac{\lambda_Z}{2}\underbrace{\sum_{i=1}^{n}\|z_i\|^2}_{=\|Z\|_F^2}$$

- Usually doesn't assume centered ratings.
  - So need to add user bias $\beta_i$ and item bias $\beta_j$ (also regularized):

Instead of $\hat{y}_{ij} = w_j^T z_i$ use $\hat{y}_{ij} = w_j^T z_i + \beta_i + \beta_j$

  - Could also have a global bias β reflecting average overall rating:

$$\hat{y}_{ij} = w_j^T z_i + \beta_i + \beta_j + \beta$$

  - High $\beta_j$ means movie is rated higher than average.

# Regularized SVD

- Predict rating of user ‘i‘ on movie ‘j’ using:

$$\hat{y}_{ij} = w_j^T z_i + \beta_i + \beta_j + \beta$$

- Combines:
  - Global bias β (rating for completely new user/movie).
  - User bias $\beta_i$ (rating of user ‘i’ for a new movie).
  - Item bias $\beta_j$ (rating of movie ‘j’ for a new user).
  - User latent features $z_i$ (learned features of user ‘i’).
  - Item latent features $w_j$ (learned features of item ‘j’).

# Hybrid Approach: SVDfeature

- Collaborative filtering is nice because you learn the features.
  - But needs a lot of information about each user/item.
- Hybrid approaches combine content-based/collaborative filtering:
  - SVDfeature:

$$\hat{y}_{ij} = \underbrace{w_j^T z_i + \beta_i + \beta_j + \beta}_{\text{collaborative filtering}} + \overbrace{w^T x_{ij}}^{\text{usual supervised learning}}$$

$x_{ij} \rightarrow$ features of user, movie, and/or user+movie.

  - Key component of model that won KDD Cup in 2011 and 2012.
  - For new users/items, predict using '$x_i$', 'w', and 'β' as in supervised case.
  - As you get data about user 'i', start to make personalized predictions.
  - As you get data about movie 'j', start to discover how it's rated differently.

# Beyond Accuracy in Recommender Systems

- Winning system of Netflix Challenge was never adopted.
- Other issues important in recommender systems:
  - Diversity: how different are the recommendations?
    - If you like 'Battle of Five Armies Extended Edition', recommend Battle of Five Armies?
    - Even if you really really like Star Wars, you might want non-Star-Wars suggestions.
  - Persistence: how long should recommendations last?
    - If you keep not clicking on 'Hunger Games', should it remain a recommendation?
  - Freshness: people tend to get more excited about *new/surprising* things.
  - Trust: tell user *why* you made a recommendation.
  - Social recommendation: what did your friends watch?

# Robust PCA

- Recent interest in 'robust' PCA.
- In our LFM, we allow an error $e_{ij}$ in approximating $x_{ij}$.

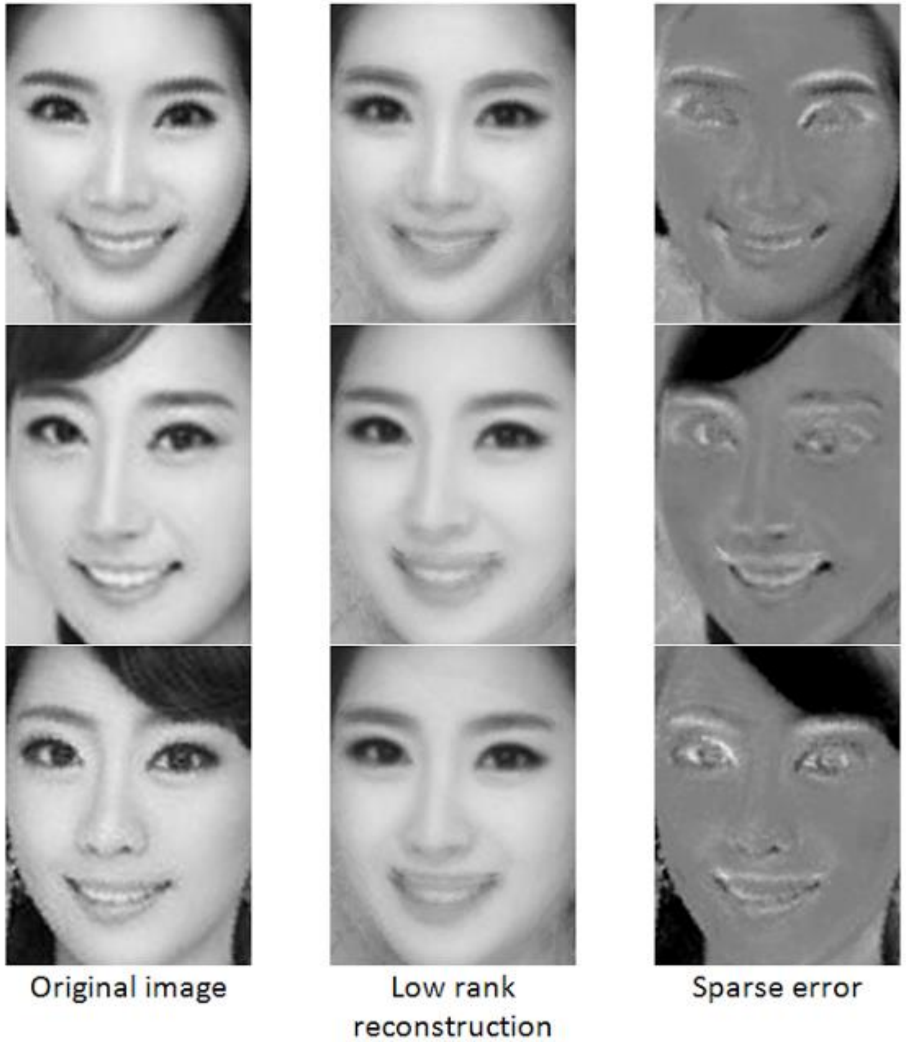
$$x_{ij} \approx w_j^T x_i + e_{ij}$$

- Use L1-regularization of $e_{ij}$:
  - Avoids degenerate solution $e_{ij} = x_{ij}$, gives sparsity in $e_{ij}$ values.
  - Will be robust to outliers in the matrix.
  - The $e_{ij}$ tell you where the outliers are.

http://statweb.stanford.edu/~candes/papers/RobustPCA.pdf

# Robust PCA

- Removing shadows/overexposure/hair with robust PCA:




http://statweb.stanford.edu/~candes/papers/RobustPCA.pdf
http://jbhuang0604.blogspot.ca/2013/04/miss-korea-2013-contestants-face.html

# Summary

- Recommender systems try to recommend products.
- Collaborative filtering tries to fill in missing values in a matrix.
- Regularized SVD is uses latent-factors for collaborative filtering.
- SVDfeature combines linear regression and regularized SVD.
- Other factors like diversity may be more important than accuracy.

- Next time: non-parametric data visualization.