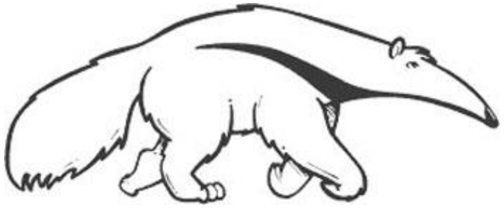
Machine Learning and Data Mining

Dimensionality Reduction; PCA & SVD

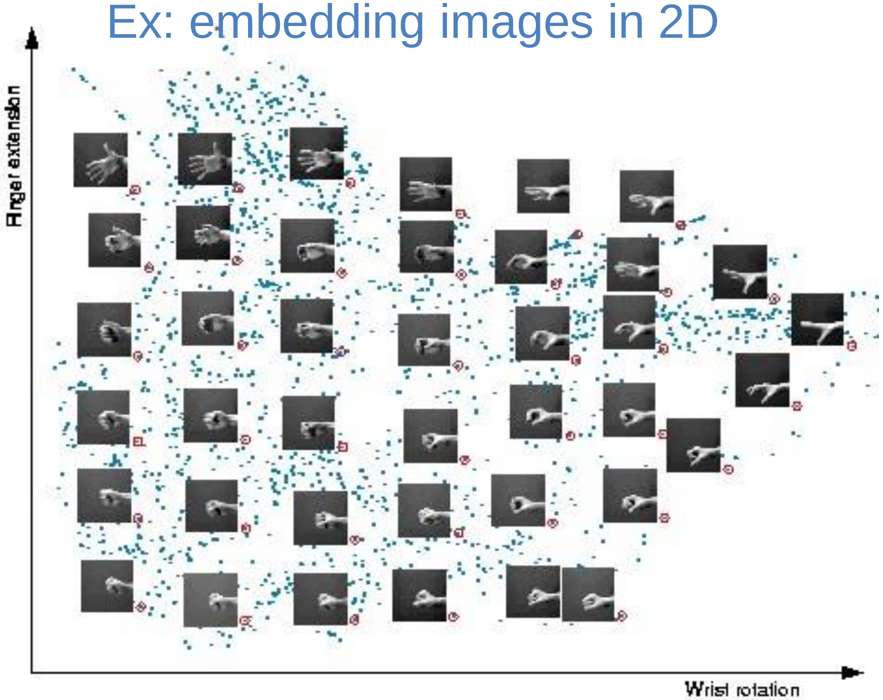
Kalev Kask



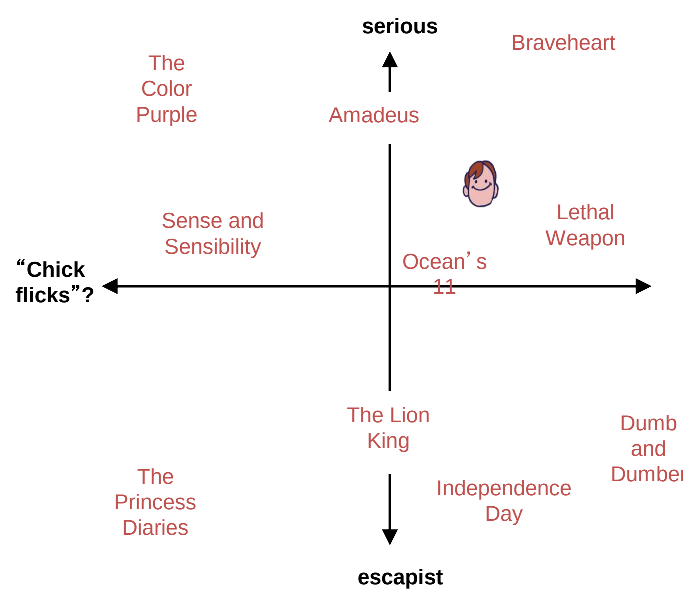
Motivation

- High-dimensional data
  - Images of faces
  - Text from articles
  - All S&P 500 stocks
- Can we describe them in a "simpler" way?
  - Embedding: place data in R^d, such that "similar" data are close

Ex: embedding images in 2D



Ex: embedding movies in 2D



Motivation

- High-dimensional data
  - Images of faces
  - Text from articles
  - All S&P 500 stocks
- Can we describe them in a "simpler" way?
  - Embedding: place data in R^d, such that "similar" data are close

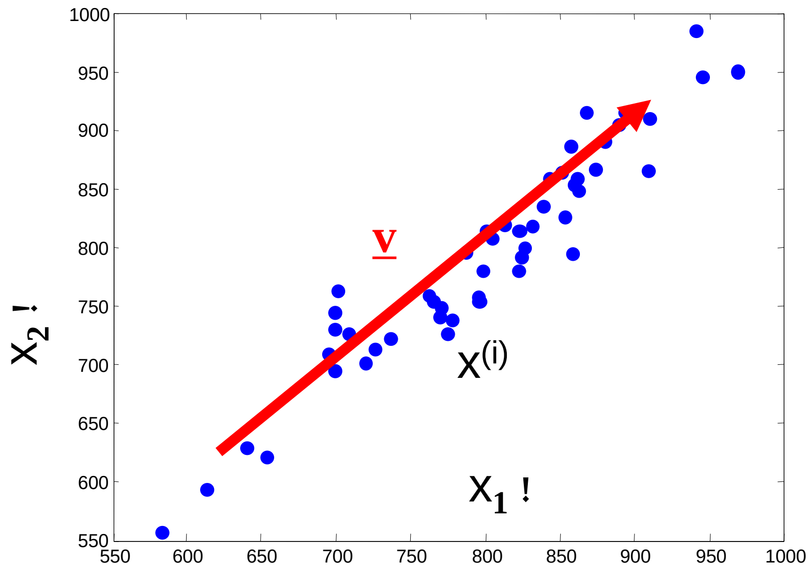
- Ex: S&P 500 – vector of 500 (change in) values per day
  - But, lots of structure
  - Some elements tend to "change together"
  - Maybe we only need a few values to approximate it?
  - "Tech stocks up 2x, manufacturing up 1.5x, …" ?

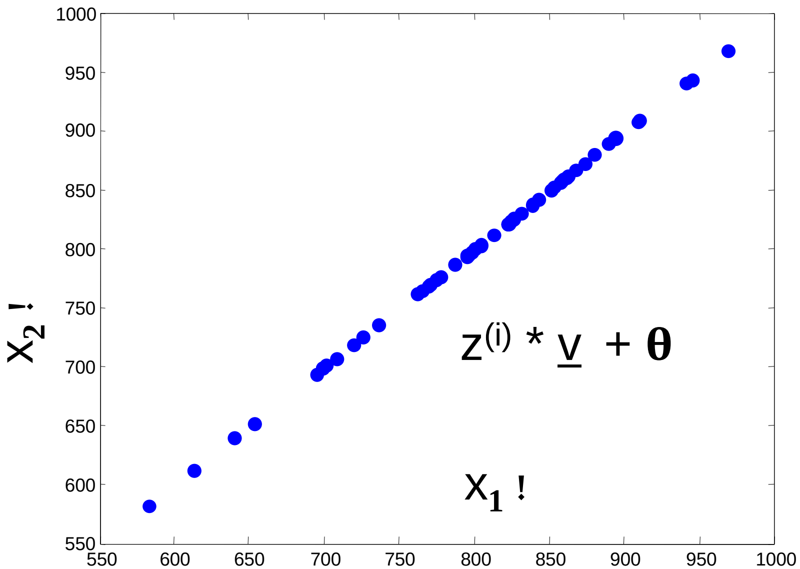
- How can we access that structure?

Dimensionality reduction

- Ex: data with two real values $[x_1,x_2]$
- We'd like to describe each point using only one value $[z_1]$
- We'll communicate a "model" to convert: $[x_1,x_2] \sim f(z_1)$

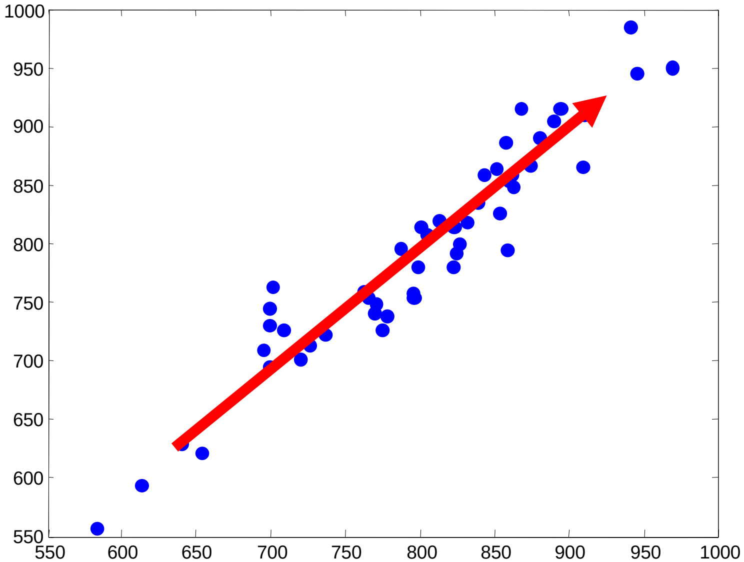
- Ex: linear function f(z): $[x_1,x_2] = \theta + z * \underline{v} = \theta + z * [v_1,v_2]$
- θ, $\underline{v}$ are the same for all data points (communicate once)
- z tells us the closest point on v to the original point $[x_1,x_2]$





Principal Components Analysis

- How should we find v?
  - Assume X is zero mean, or $\tilde{X} = X - \mu$
  - Pick v such that MSE(X,$\tilde{X}$) is min - the smallest residual variance! ("error")
  - Equivalent: Find "v" as the direction of maximum "spread" (variance)
  - Solution is the eigenvector (of covariance of $\tilde{X}$) with largest eigenvalue



Project X to v: $z = \tilde{X} \cdot v$

Variance of projected points:

$$\sum_i (z^{(i)})^2 = z^T z = v^T \tilde{X}^T \tilde{X} v$$

Best "direction" v:

$$\max_v \; v^T \tilde{X}^T \tilde{X} v \quad s.t. \; ||v|| = 1$$

$\rightarrow$ largest eigenvector of $X^T X$

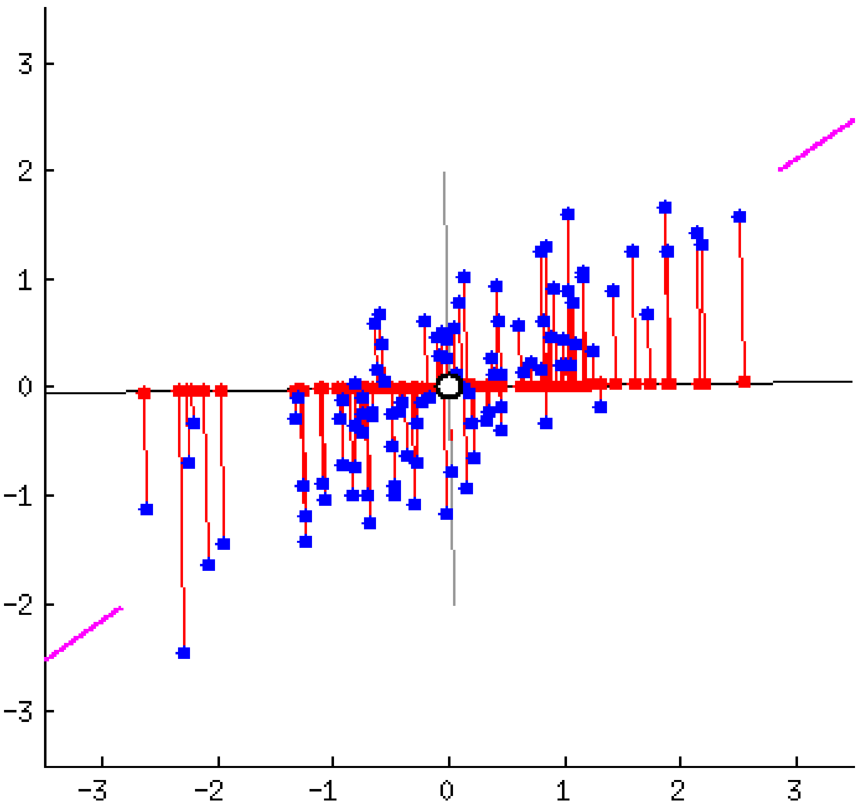
Principal Components Analysis

- How should we find v?
  - Assume X is zero mean, or $\tilde{X} = X - \mu$
  - Find "v" as the direction of maximum "spread" (variance)
  - Solution is the eigenvector (of covariance of $\tilde{X}$) with largest eigenvalue

  - General : x~ = $z_1 * v_1 + z_2 * v_2 + \ldots + z_k * v_k + \mu$

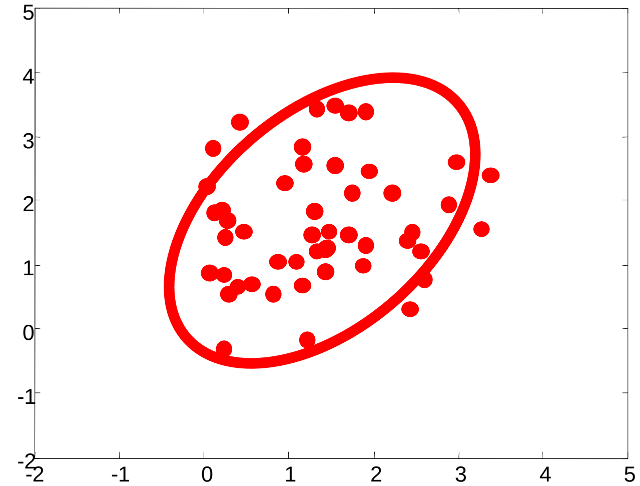
Dim Reduction Demo

https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues



Another interpretation

- Data covariance: $\Sigma = \frac{1}{m}\tilde{X}^T\tilde{X}$ $\quad\quad \tilde{X} = X - \mu$



  - Describes "spread" of the data
  - Draw this with an ellipse
  - Gaussian is

$$p(x) \propto \exp\left(-\frac{1}{2}\Delta^2\right)$$

$$\Delta^2 = (x-\mu)\Sigma^{-1}(x-\mu)^T$$

  - Ellipse shows the contour, Δ^2 = constant

Geometry of the Gaussian

$$\Delta^2 = (\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

Oval shows constant Δ^2 value…

$$\Sigma = U\Lambda U^T$$

Write Σ in terms of eigenvectors…

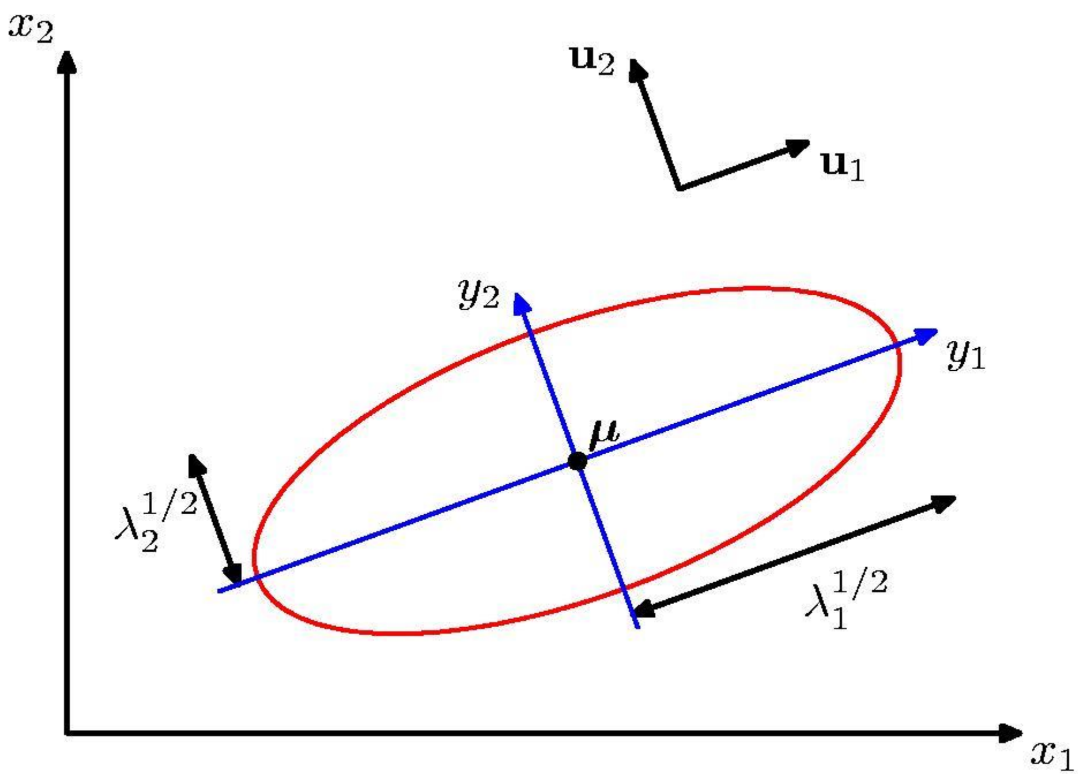
$$\Sigma = \begin{bmatrix} u_1 & u_2 \\ \downarrow & \downarrow \end{bmatrix} \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix} \begin{bmatrix} u_1 & \rightarrow \\ u_2 & \rightarrow \end{bmatrix}$$

$$\boldsymbol{\Sigma}^{-1} = \sum_{i=1}^{D} \frac{1}{\lambda_i}\mathbf{u}_i\mathbf{u}_i^{\mathrm{T}}$$

Then…

$$\Delta^2 = \sum_{i=1}^{D} \frac{y_i^2}{\lambda_i}$$

$$y_i = \mathbf{u}_i^{\mathrm{T}}(\mathbf{x}-\boldsymbol{\mu})$$



PCA representation (EVD)

1. Subtract data mean from each point
2. (Typically) scale each dimension by its variance
   - Helps pay less attention to magnitude of the variable
3. Compute covariance matrix, $S = 1/m \sum (x^i - \mu)' \ (x^i - \mu)$
4. Compute the eigendecomposition of S

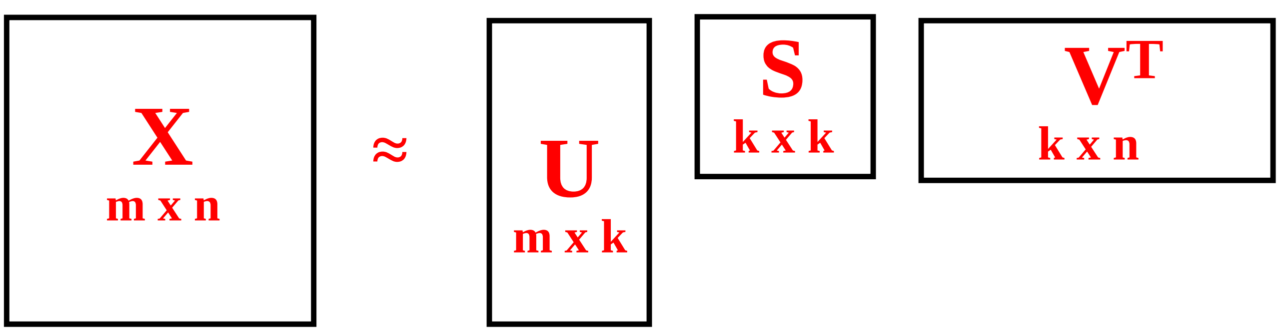
   $$S = V D V^T$$

5. Pick the k largest (by eigenvalue) eigenvectors of S

```
mu = np.mean( X, axis=0, keepdims=True )   # find mean over data points
X0 = X - mu                                 # zero-center the data
S = X0.T.dot( X0 ) / m                      # S = np.cov( X.T ), data covariance
D,V = np.linalg.eig( S )                    # find eigenvalues/vectors: can be slow!
pi = np.argsort(D)[::-1]                    # sort eigenvalues largest to smallest
D,V = D[pi], V[:,pi]                        #
D,V = D[0:k], V[:,0:k]                      # and keep the k largest
```

Singular Value Decomposition (SVD)

- Alternative method to calculate (still subtract mean 1st)
- Decompose $X = U\,S\,V^T$
  - Orthogonal: $X^T X = V\,S\,S\,V^T = V\,D\,V^T$
  - $X X^T = U\,S\,S\,U^T = U\,D\,U^T$
- U*S matrix provides coefficients
  - Example $x_i = U_{i,1}\,S_{11}\,v_1 + U_{i,2}\,S_{22}\,v_2 + \ldots$
- Gives the least-squares approximation to X of this form



SVD for PCA

- Subtract data mean from each point
- (Typically) scale each dimension by its variance
  - Helps pay less attention to magnitude of the variable
- Compute the SVD of the data matrix

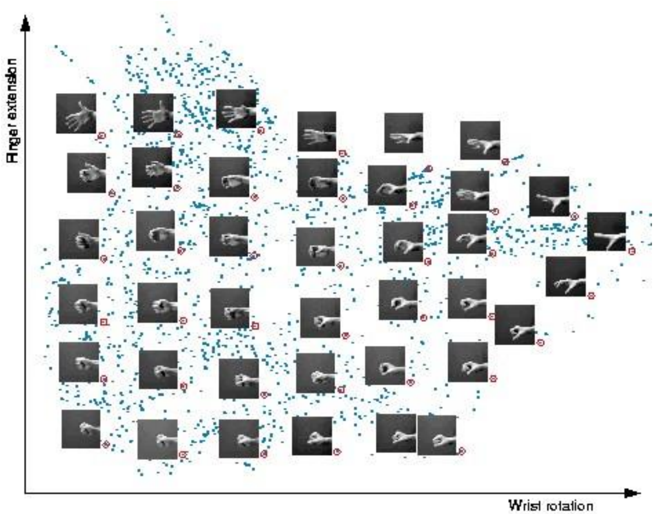
```
mu = np.mean( X, axis=0, keepdims=True )   # find mean over data points
X0 = X - mu                                # zero-center the data

U,S,Vh = scipy.linalg.svd(X0, False)       # X0 = U * diag(S) * Vh

Xhat = U[:,0:k].dot( np.diag(S[0:k]) ).dot( Vh[0:k,:] )   # approx using k largest eigendir
```

Some uses of latent spaces

- Data compression
  - Cheaper, low-dimensional representation
- Noise removal
  - Simple "true" data + noise
- Supervised learning, e.g. regression:
  - Remove colinear / nearly colinear features
  - Reduce feature dimension => combat overfitting



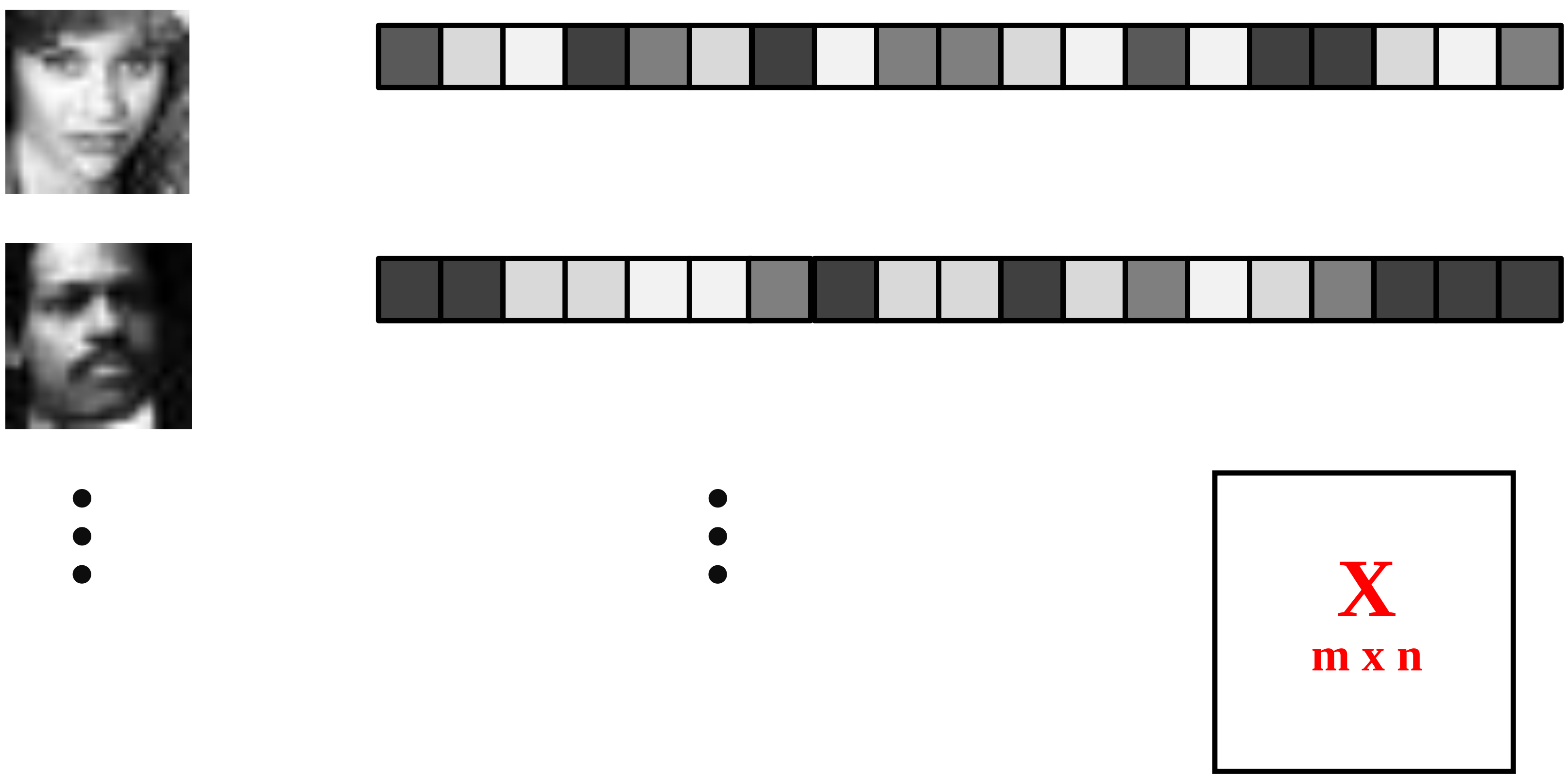
Applications of SVD

- "Eigen-faces"
  - Represent image data (faces) using PCA
- LSI / "topic models"
  - Represent text data (bag of words) using PCA
- Collaborative filtering
  - Represent rating data matrix using PCA

and more…

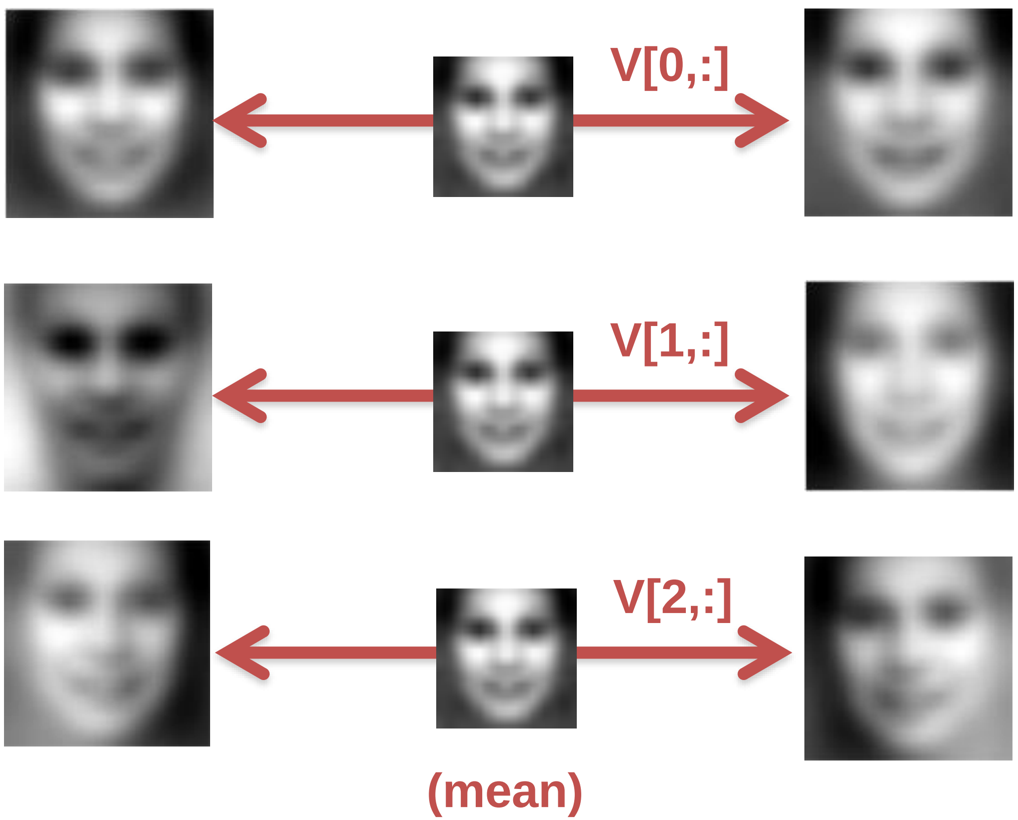
“Eigen-faces”

- “Eigen-X” = represent X using PCA
- Ex: Viola Jones data set
  - 24x24 images of faces = 576 dimensional measurements



"Eigen-faces"

- "Eigen-X" = represent X using PCA
- Ex: Viola Jones data set
  - 24x24 images of faces = 576 dimensional measurements
  - Take first K PCA components



$$\underset{m \times n}{X} \approx \underset{m \times k}{U}\ \underset{k \times k}{S}\ \underset{k \times n}{V^T}$$

"Eigen-faces"

- "Eigen-X" = represent X using PCA
- Ex: Viola Jones data set
  - 24x24 images of faces = 576 dimensional measurements
  - Take first K PCA components

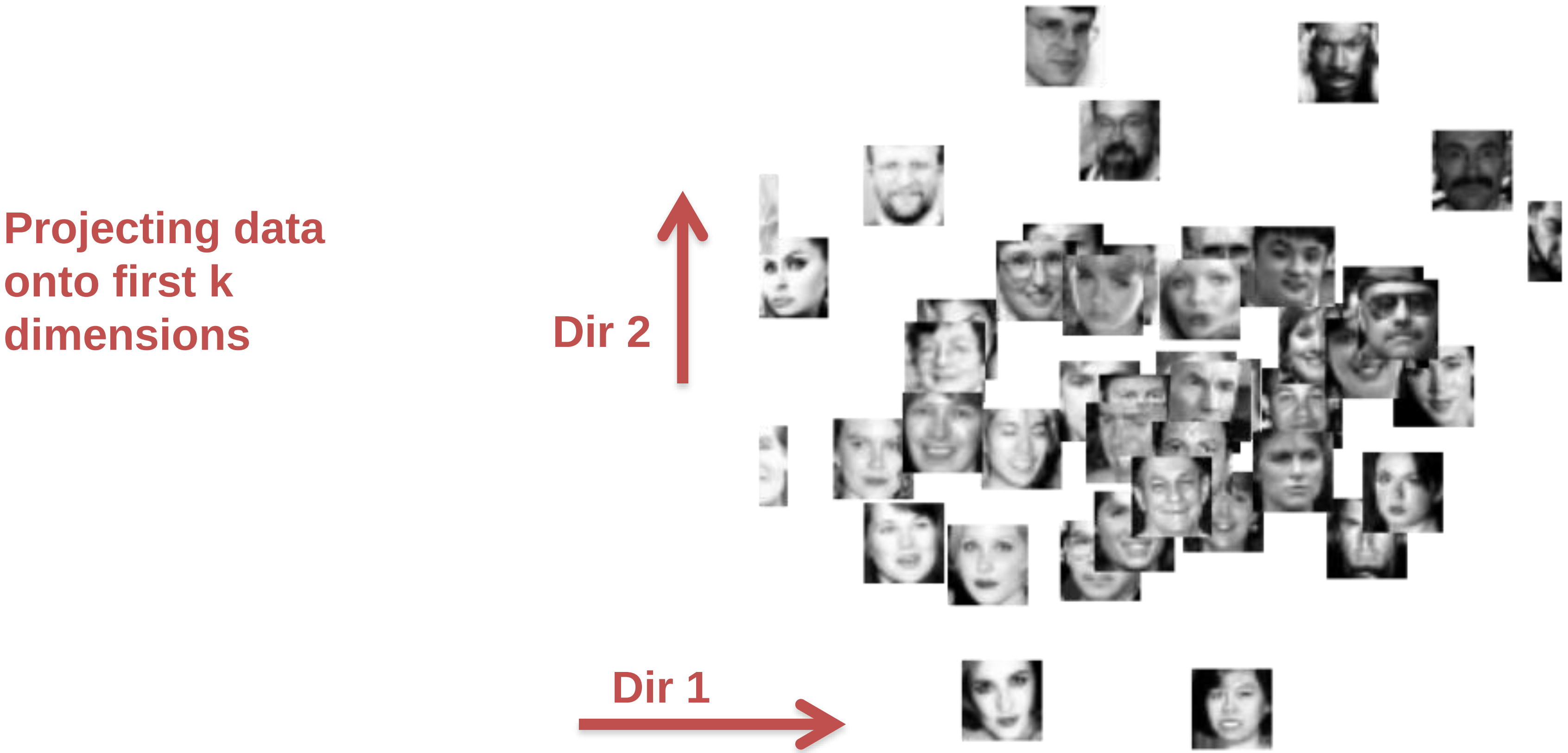
Mean Dir 1 Dir 2 Dir 3 Dir 4 …

Projecting data onto first k dimensions

Xi k=5 k=10 k=50 ….

"Eigen-faces"

- "Eigen-X" = represent X using PCA
- Ex: Viola Jones data set
  - 24x24 images of faces = 576 dimensional measurements
  - Take first K PCA components



Text representations

- "Bag of words"
  - Remember word counts but not order
- Example:

Rain and chilly weather didn't keep thousands of paradegoers from camping out Friday night for the 111th Tournament of Roses.

Spirits were high among the street party crowd as they set up for curbside seats for today's parade.

``I want to party all night,'' said Tyne Gaudielle, 15, of Glendale, who spent the last night of the year along Colorado Boulevard with a group of friends.

Whether they came for the partying or the parade, campers were in for a long night. Rain continued into the evening and temperatures were expected to dip down into the low 40s.

Text representations

- "Bag of words"
  - Remember word counts but not order
- Example:

Rain and chilly weather didn't keep thousands of
paradegoers from camping out Friday night for t
of Roses.

Spirits were high among the street party crowd a
for curbside seats for today's parade.

``I want to party all night,'' said Tyne Gaudielle,
Glendale, who spent the last night of the year alo
Boulevard with a group of friends.

Whether they came for the partying or the parad
in for a long night. Rain continued into the eveni
temperatures were expected to dip down into the

nyt/2000-01-01.0015.txt
rain
chilly
weather
didn
keep
thousands
paradegoers
camping
out
friday
night
111th
tournament
roses
spirits
high
among

Text representations

- "Bag of words"
  - Remember word counts but not order
- Example:

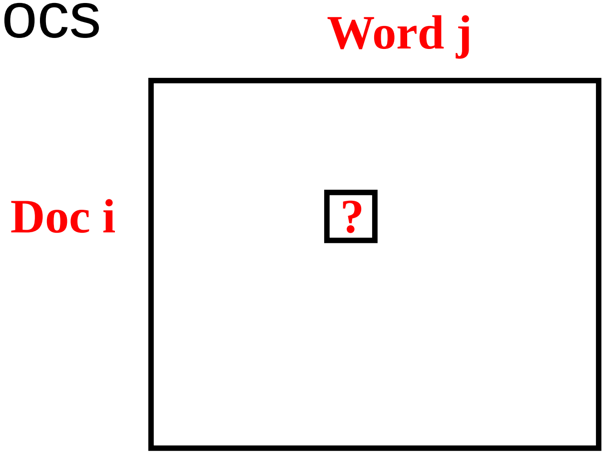
VOCABULARY:
0001 ability
0002 able
0003 accept
0004 accepted
0005 according
0006 account
0007 accounts
0008 accused
0009 act
0010 acting
0011 action
0012 active
….

Observed Data (text docs):

DOC #	WORD #	COUNT
1	29	1
1	56	1
1	127	1
1	166	1
1	176	1
1	187	1
1	192	1
1	198	2
1	356	1
1	374	1
1	381	2
…		

Latent Semantic Indexing (LSI)

- PCA for text data
- Create a giant matrix of words in docs
  - "Word j appears" = feature x_j
  - "in document i" = data example I

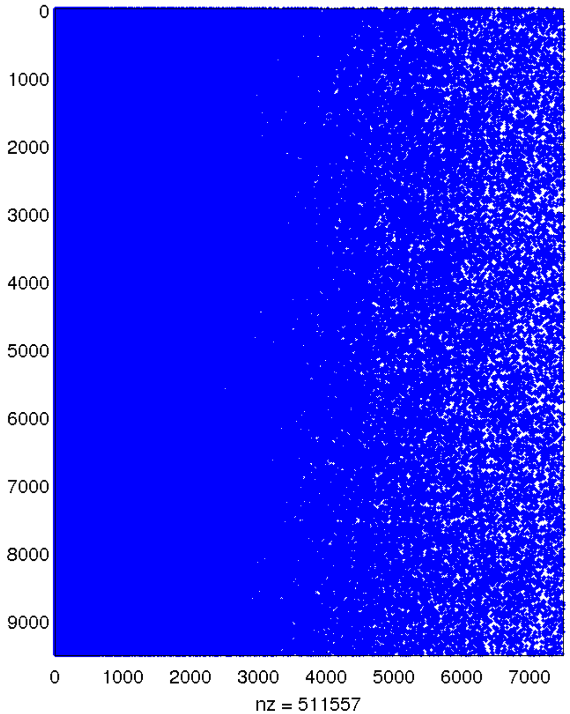


- Huge matrix (mostly zeros)
  - Typically normalize rows to sum to one, to control for short docs
  - Typically don't subtract mean or normalize columns by variance
  - Might transform counts in some way (log, etc)
- PCA on this matrix provides a new representation
  - Document comparison
  - Fuzzy search ("concept" instead of "word" matching)

Matrices are big, but data are sparse

- Typical example:
  - Number of docs, D ~ 10^6
  - Number of unique words in vocab, W ~ 10^5
  - FULL Storage required ~ 10^{11}
  - Sparse Storage required ~ 10^9

- DxW matrix (# docs x # words)
  - Looks dense, but that's just plotting
  - Each entry is non-negative
  - Typically integer / count data



Latent Semantic Indexing (LSI)

- What do the principal components look like?

PRINCIPAL COMPONENT 1

0.135 genetic
0.134 gene
0.131 snp
0.129 disease
0.126 genome_wide
0.117 cell
0.110 variant
0.109 risk
0.098 population
0.097 analysis
0.094 expression
0.093 gene_expression
0.092 gwas
0.089 control
0.088 human
0.086 cancer

Latent Semantic Indexing (LSI)

- What do the principal components look like?

PRINCIPAL COMPONENT 1

0.135 genetic
0.134 gene
0.131 snp
0.129 disease
0.126 genome_wide
0.117 cell
0.110 variant
0.109 risk
0.098 population
0.097 analysis
0.094 expression
0.093 gene_expression
0.092 gwas
0.089 control
0.088 human
0.086 cancer

PRINCIPAL COMPONENT 2

0.247 snp
-0.196 cell
0.187 variant
0.181 risk
0.180 gwas
0.162 population
0.162 genome_wide
0.155 genetic
0.130 loci
-0.116 mir
-0.116 expression
0.113 allele
0.108 schizophrenia
0.107 disease
-0.103 mirnas
-0.099 protein

Q: But what does -0.196 cell mean?

Collaborative filtering (Netflix)

From Y. Koren of BellKor team

users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

$\underset{N \times D}{X}$ ¼ $\underset{N \times K}{U}$ $\underset{K \times K}{S}$ $\underset{K \times D}{V^T}$

Latent space models

From Y. Koren
of BellKor team

Model ratings matrix as "user" and "movie" positions

Infer values from known ratings

Extrapolate to unranked

users

items

1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

~

~

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

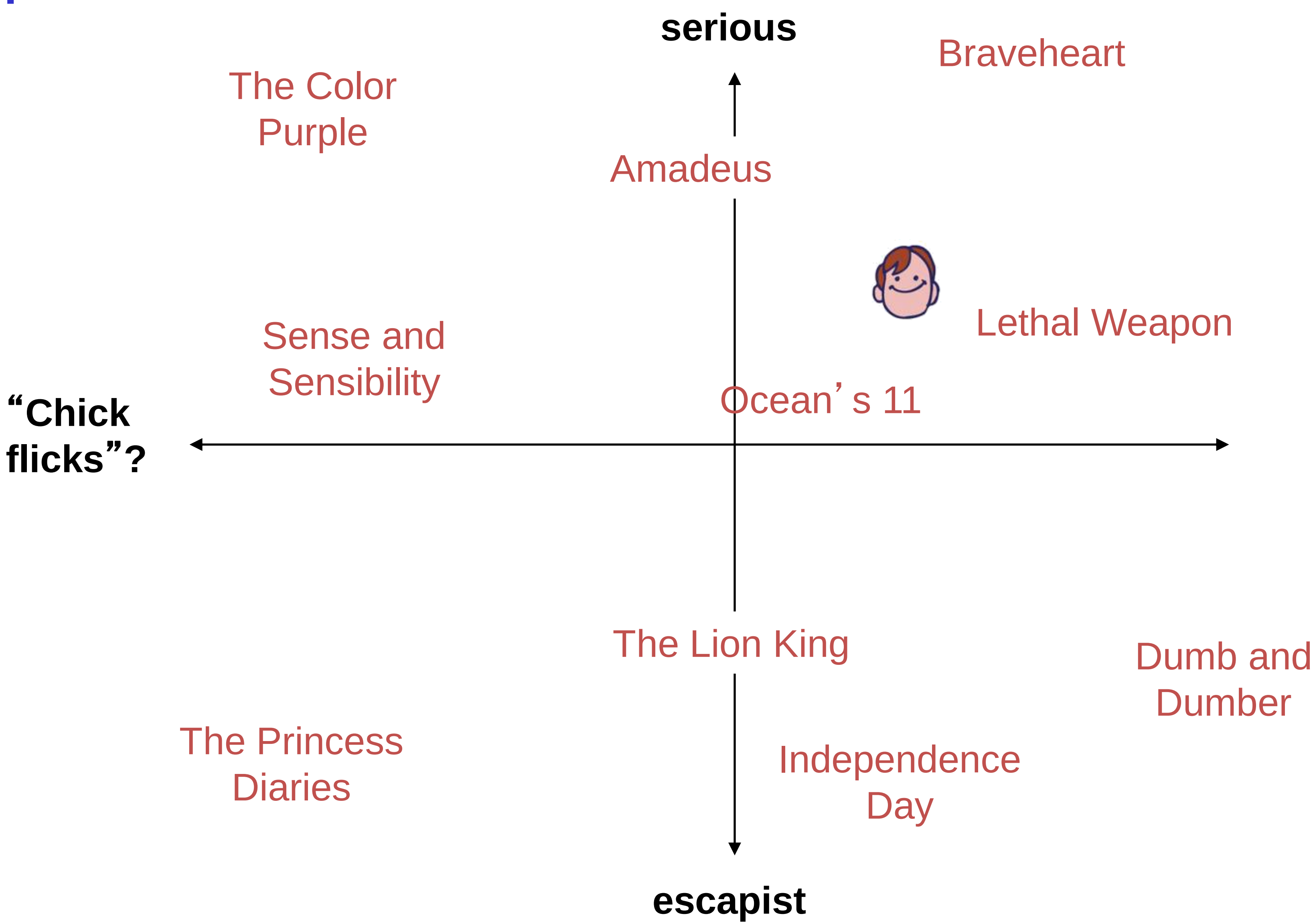
•

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

Latent space models

From Y. Koren of BellKor team



Some SVD dimensions

See timelydevelopment.com

Dimension 1

Offbeat / Dark-Comedy
Lost in Translation
The Royal Tenenbaums
Dogville
Eternal Sunshine of the Spotless Mind
Punch-Drunk Love

Mass-Market / 'Beniffer' Movies
Pearl Harbor
Armageddon
The Wedding Planner
Coyote Ugly
Miss Congeniality

Dimension 2

Good
VeggieTales: Bible Heroes: Lions
The Best of Friends: Season 3
Felicity: Season 2
Friends: Season 4
Friends: Season 5

Twisted
The Saddest Music in the World
Wake Up
I Heart Huckabees
Freddy Got Fingered
House of 1

Dimension 3

What a 10 year old boy would watch
Dragon Ball Z: Vol. 17: Super Saiyan
Battle Athletes Victory: Vol. 4: Spaceward Ho!
Battle Athletes Victory: Vol. 5: No Looking Back
Battle Athletes Victory: Vol. 7: The Last Dance
Battle Athletes Victory: Vol. 2: Doubt and Conflic

What a liberal woman would watch
Fahrenheit 9/11
The Hours
Going Upriver: The Long War of John Kerry
Sex and the City: Season 2
Bowling for Columbine

Latent space models

- Latent representation encodes some “meaning”
- What kind of movie is this? What movies is it similar to?

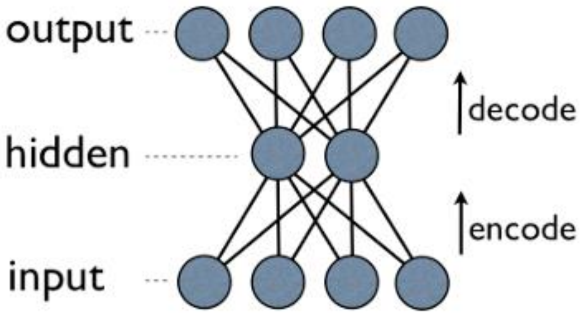
- Matrix is full of missing data
  - Hard to take SVD directly
  - Typically solve using gradient descent
  - Easy algorithm (see Netflix challenge forum)

$$J(U,V) = \sum_{u,m} (X_{mu} - \sum_k U_{mk} V_{ku})^2$$

```
# for user u, movie m, find the kth eigenvector & coefficient by iterating:
predict_um = U[m,:].dot( V[:,u] )          # predict: vector-vector product
err = ( rating[u,m] – predict_um )         # find error residual
V_ku, U_mk = V[k,u], U[m,k]                # make copies for update
U[m,k] += alpha * err * V_ku               # Update our matrices
V[k,u]  += alpha * err * U_mk              #    (compare to least-squares gradient)
```

Nonlinear latent spaces

- Latent space
  - Any alternative representation (usually smaller) from which we can (approximately) recover the data
  - Linear: "Encode" $Z = X V^T$; "Decode" X ¼ Z V



stats.stackexchange.com

- Ex: Auto-encoders
  - Use neural network with few internal nodes
  - Train to "recover" the input "x"
- Related: word2vec
  - Trains an NN to recover the context of words
  - Use internal hidden node responses as a vector representation of the word

Summary

- Dimensionality reduction
  - Representation: basis vectors & coefficients
- Linear decomposition
  - PCA / eigendecomposition
  - Singular value decomposition
- Examples and data sets
  - Face images
  - Text documents (latent semantic indexing)
  - Movie ratings