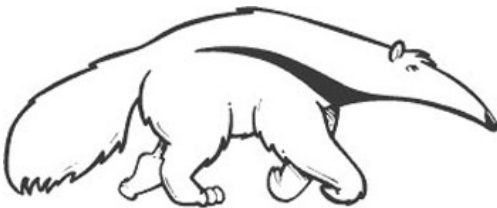


+

# Machine Learning and Data Mining

## Dimensionality Reduction; PCA & SVD

Prof. Alexander Ihler

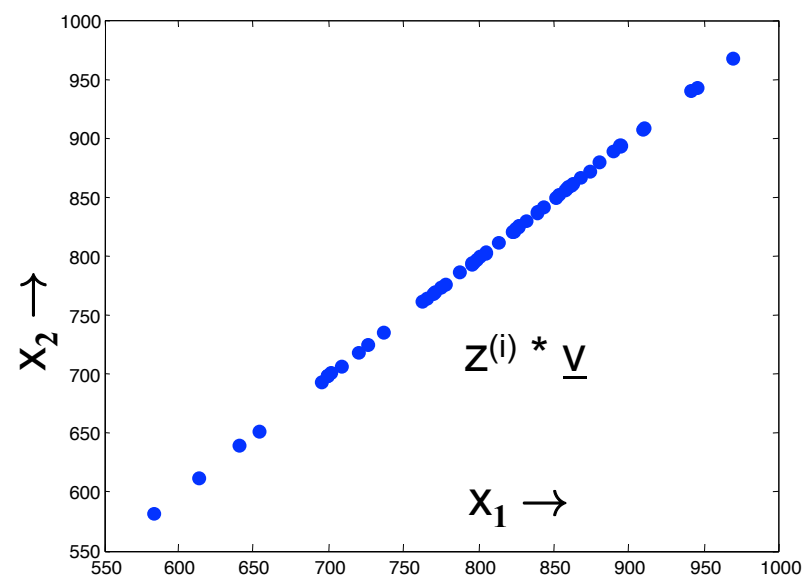
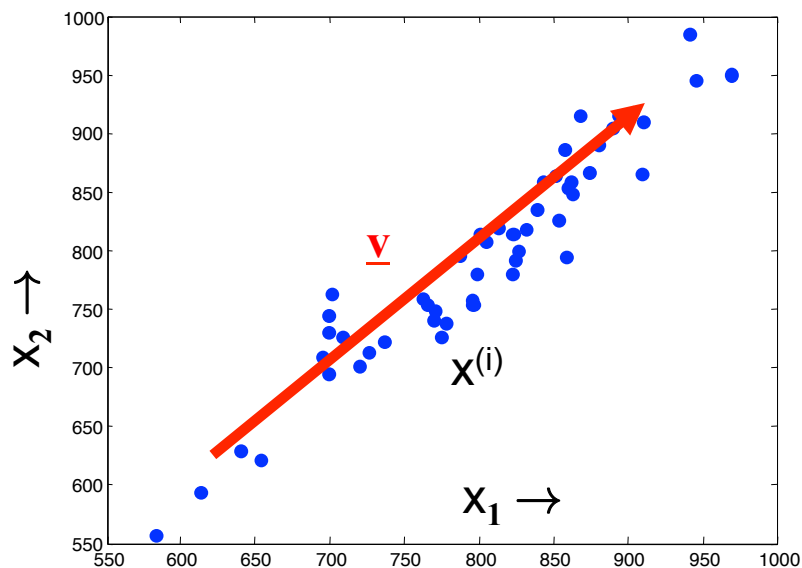


# Motivation

- High-dimensional data
  - Images of faces
  - Text from articles
  - All S&P 500 stocks
- Can we describe them in a “simpler” way?
- 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  $f(z)$ :  $[x_1, x_2] = z * \underline{v} = z * [v_1, v_2]$
- $\underline{v}$  is the same for all data points (communicate once)
- $z$  tells us the closest point on  $v$  to the original point  $[x_1, x_2]$

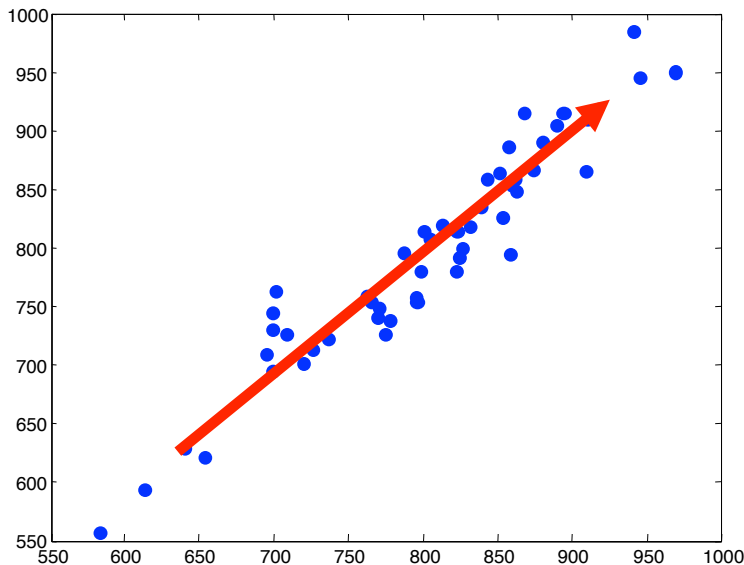


# Principal Components Analysis

- What is the vector that would most closely reconstruct X?

$$\min_{a,v} \sum_i (x^{(i)} - a^{(i)}v)^2$$

- Given  $v$ :  $a^{(i)}$  is the projection of each point  $x^{(i)}$  onto  $v$
- $v$  chosen to minimize the residual variance
- Equivalently,  $v$  is the direction of maximum variance
- Extensions: best two dimensions:  $x_i = a_i v + b_i w + m$



# Geometry of the Gaussian

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

Oval shows constant  $\Delta^2$  value...

$$\boldsymbol{\Sigma} = \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^T$$

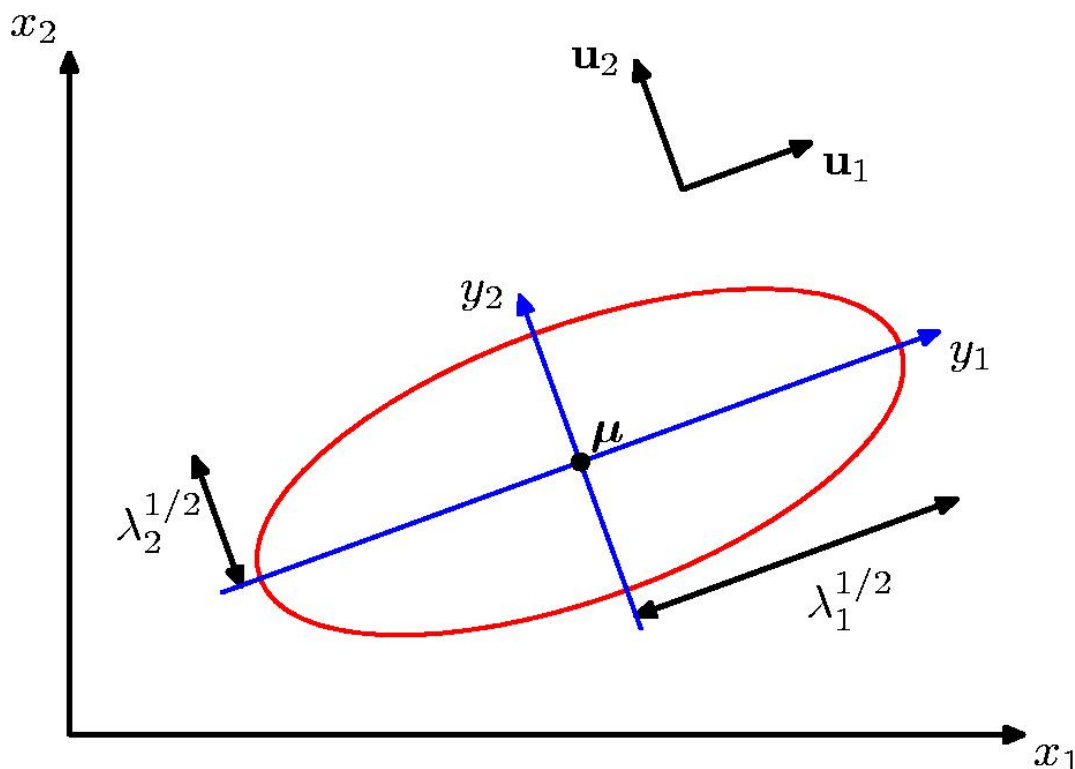
Write  $\boldsymbol{\Sigma}$  in terms of  
eigenvectors...

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

Then...

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

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



# PCA representation

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

$$S = V D V^T$$

```
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

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

$$\boxed{\begin{matrix} \mathbf{X} \\ \mathbf{m \times n} \end{matrix}} \approx \boxed{\begin{matrix} \mathbf{U} \\ \mathbf{m \times k} \end{matrix}} \boxed{\begin{matrix} \mathbf{S} \\ \mathbf{k \times k} \end{matrix}} \boxed{\begin{matrix} \mathbf{V^T} \\ \mathbf{k \times n} \end{matrix}}$$

# 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
```



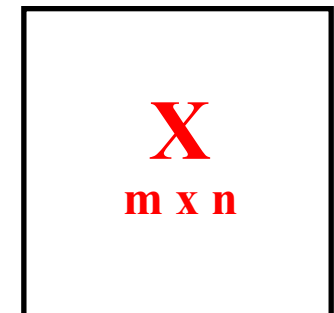
# “Eigen-faces”

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



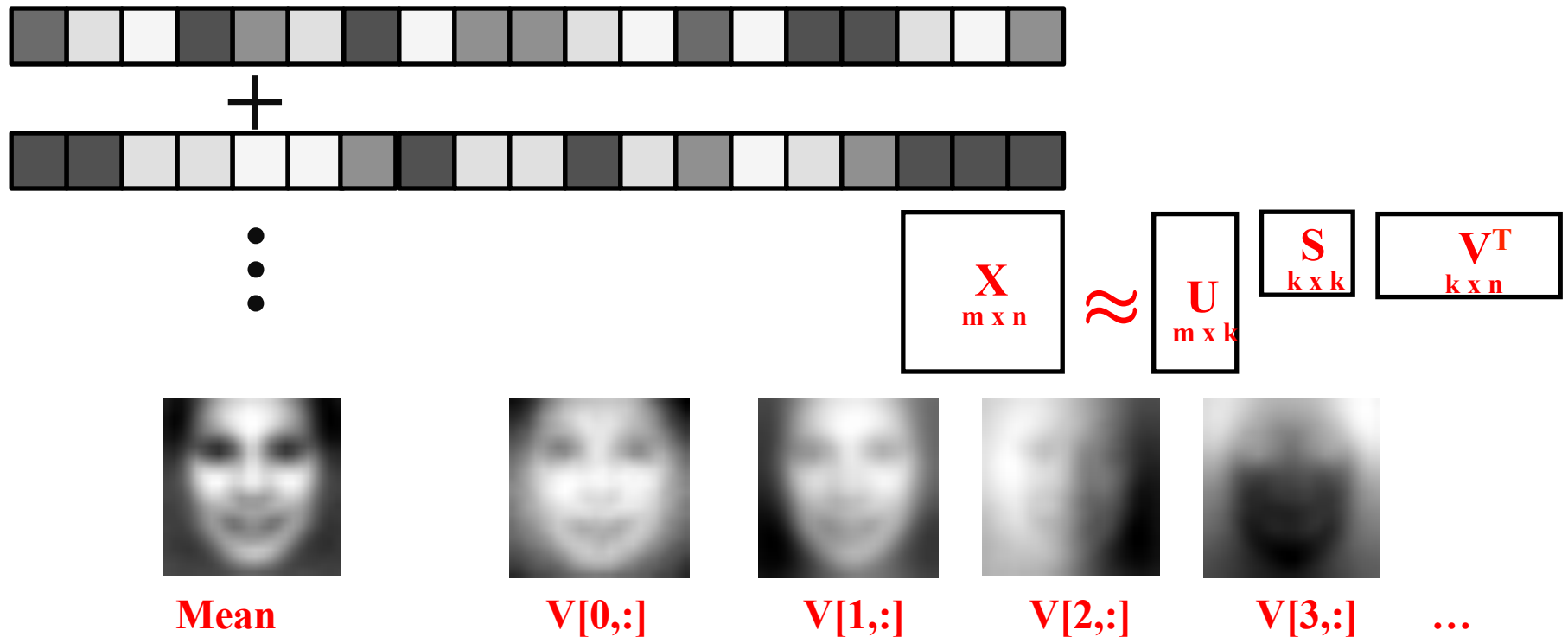
⋮

⋮



# “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



# “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



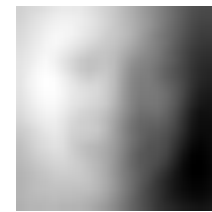
**Mean**



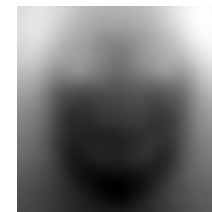
**Dir 1**



**Dir 2**

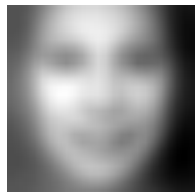


**Dir 3**

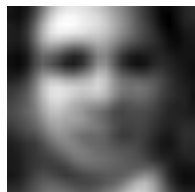


**Dir 4**

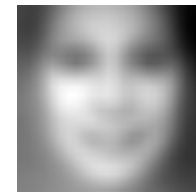
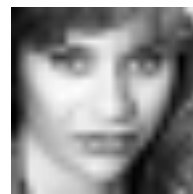
...



**K=4**



**K=50**



**K=4**



**K=50**

# Text representations

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

**Rain and chilly weather didn't keep thousands of parade-goers 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 parade-goers from camping out Friday night for the Tournament of Roses.

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

“I want to party all night,” said Tyne Gaudielle, 11, of Glendale, who spent the last night of the year alone on Hollywood Boulevard with a group of friends.

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

### example1/20000101.0015.txt

rain

chilly

weather

didn

keep

thousands

parade-goers

camping

out

friday

night

11th

tournament

roses

spirits

high

among

# Text representations

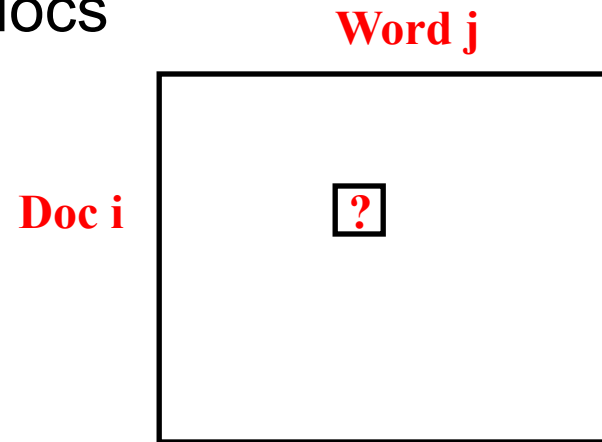
- “Bag of words”
  - Remember word counts but not order
- Example:

## **VOCABULARY:**

	<b>DOC #</b>	<b>WORD #</b>	<b>COUNT</b>
<b>0001 ability</b>	<b>1</b>	<b>29</b>	<b>1</b>
<b>0002 able</b>	<b>1</b>	<b>56</b>	<b>1</b>
<b>0003 accept</b>	<b>1</b>	<b>127</b>	<b>1</b>
<b>0004 accepted</b>	<b>1</b>	<b>166</b>	<b>1</b>
<b>0005 according</b>	<b>1</b>	<b>176</b>	<b>1</b>
<b>0006 account</b>	<b>1</b>	<b>187</b>	<b>1</b>
<b>0007 accounts</b>	<b>1</b>	<b>192</b>	<b>1</b>
<b>0008 accused</b>	<b>1</b>	<b>198</b>	<b>2</b>
<b>0009 act</b>	<b>1</b>	<b>356</b>	<b>1</b>
<b>0010 acting</b>	<b>1</b>	<b>374</b>	<b>1</b>
<b>0011 action</b>	<b>1</b>	<b>381</b>	<b>2</b>
<b>0012 active</b>	<b>...</b>		
<b>....</b>			

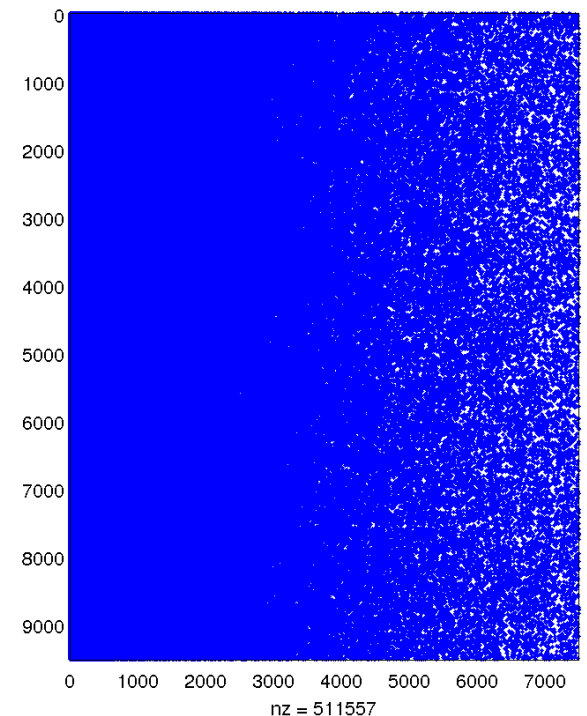
# Latent Semantic Indexing (LSI)

- 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 by e.g. sum over  $j$  to control for short docs
  - Typically don't subtract mean or normalize 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,  $D \sim 10^6$
  - Number of unique words in vocab,  $W \sim 10^5$
  - FULL Storage required  $\sim 10^{11}$
  - Sparse Storage required  $\sim 10^9$
- $D \times W$  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?

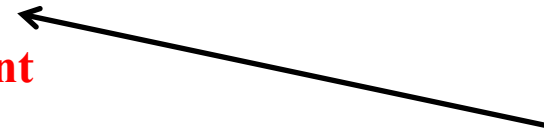
## PRINCIPAL COMPONENT 1

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

$$\begin{matrix} \boxed{\mathbf{X}} \\ \mathbf{N \times D} \end{matrix} \approx \begin{matrix} \boxed{\mathbf{U}} \\ \mathbf{N \times K} \end{matrix} \begin{matrix} \boxed{\mathbf{S}} \\ \mathbf{K \times K} \end{matrix} \begin{matrix} \boxed{\mathbf{V}^T} \\ \mathbf{K \times D} \end{matrix}$$

# Latent space models

From Y. Koren  
of BellKor team

Model ratings matrix as  
“user” and “movie”  
positions

Infer values from known  
ratings

		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		

~

Extrapolate to unranked

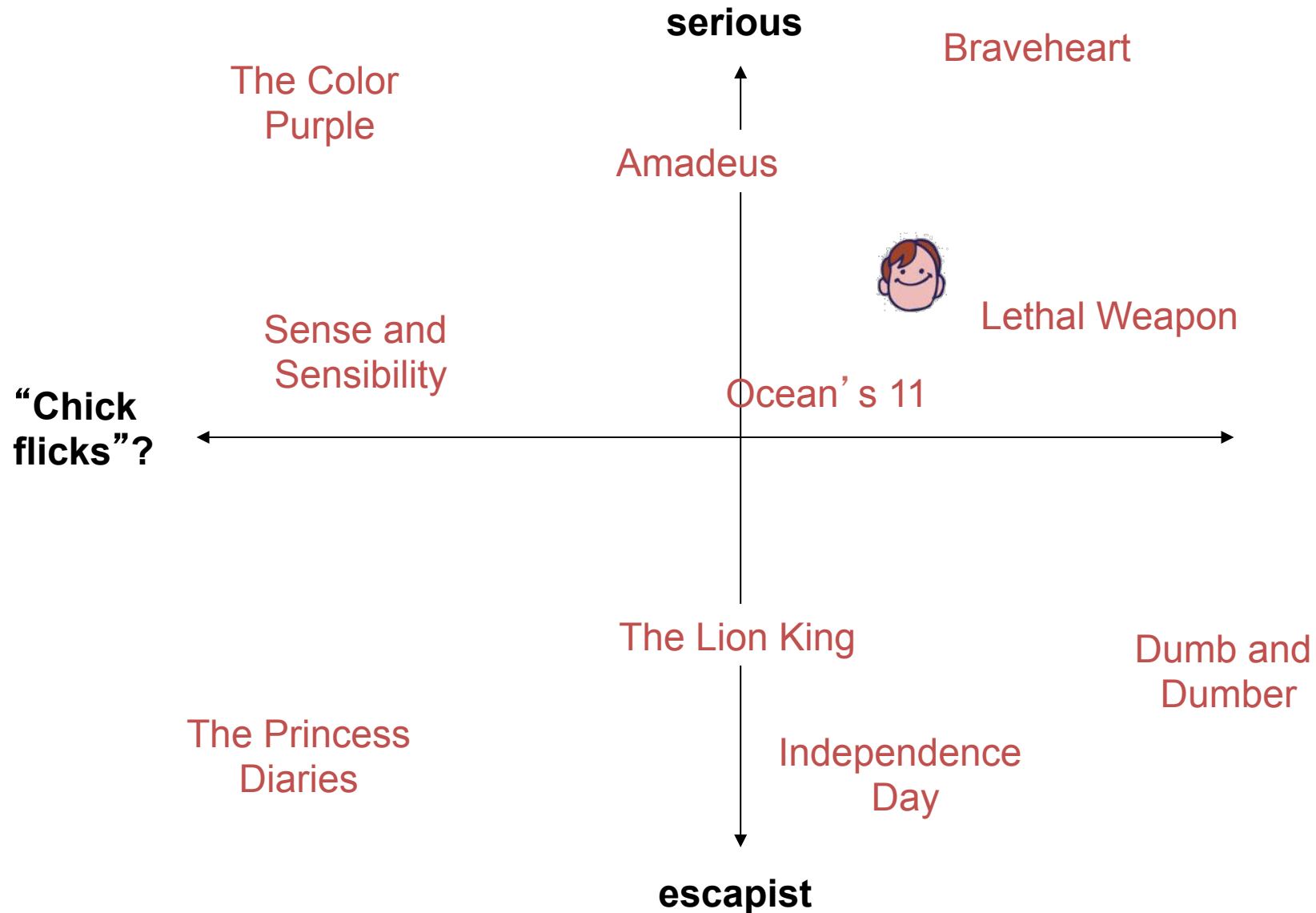
~	items	.1	-.4	.2												
		-.5	.6	.5												
		-.2	.3	.5												
		1.1	2.1	.3												
		-.7	2.1	-2												
		-1	.7	.3												

•

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](http://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 Conflict

### 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

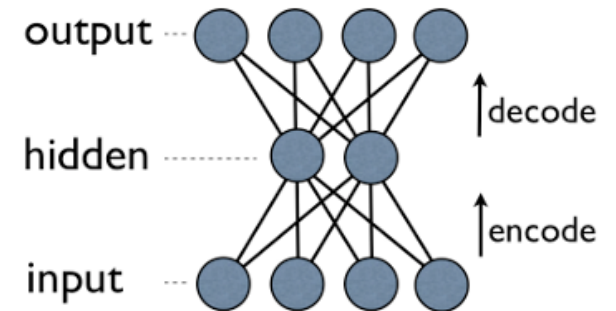
# Latent space models

- 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)

```
# 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
- 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
- Also: Boltzmann machines
  - Probabilistic mixture model to explain features “x”



[stats.stackexchange.com](https://stats.stackexchange.com)



# 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