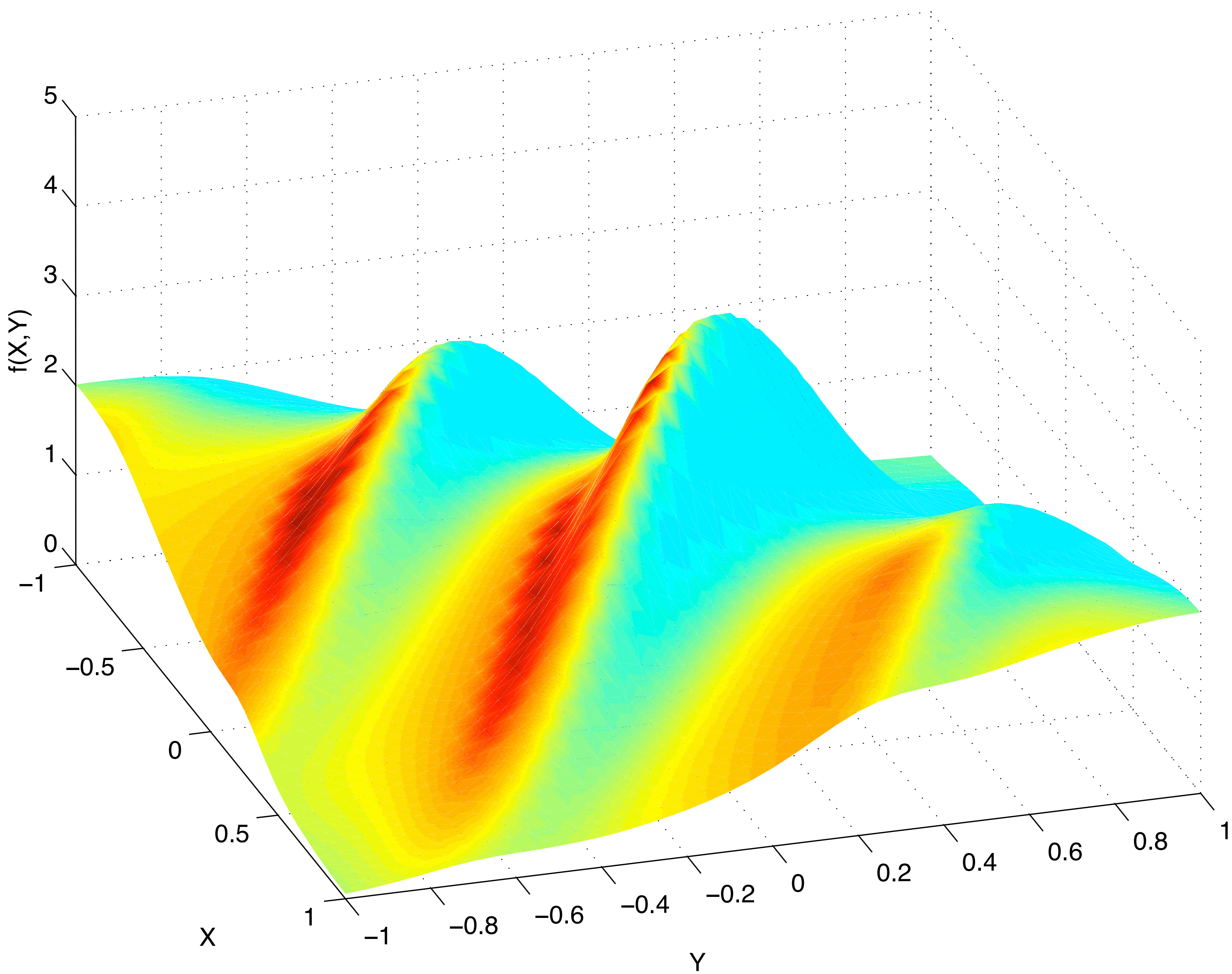
Review

- Parallel importance sampling
  - bias due to 1/normalizer
  - particle filter = recursive parallel IS
- MCMC
  - randomized search for high P(x)
  - burn-in, mixing
  - approx. iid: $\{ X_t, X_{t+\Delta}, X_{t+2\Delta}, X_{t+3\Delta}, \ldots \}$
  - use to construct estimator of $E_P(g(X))$

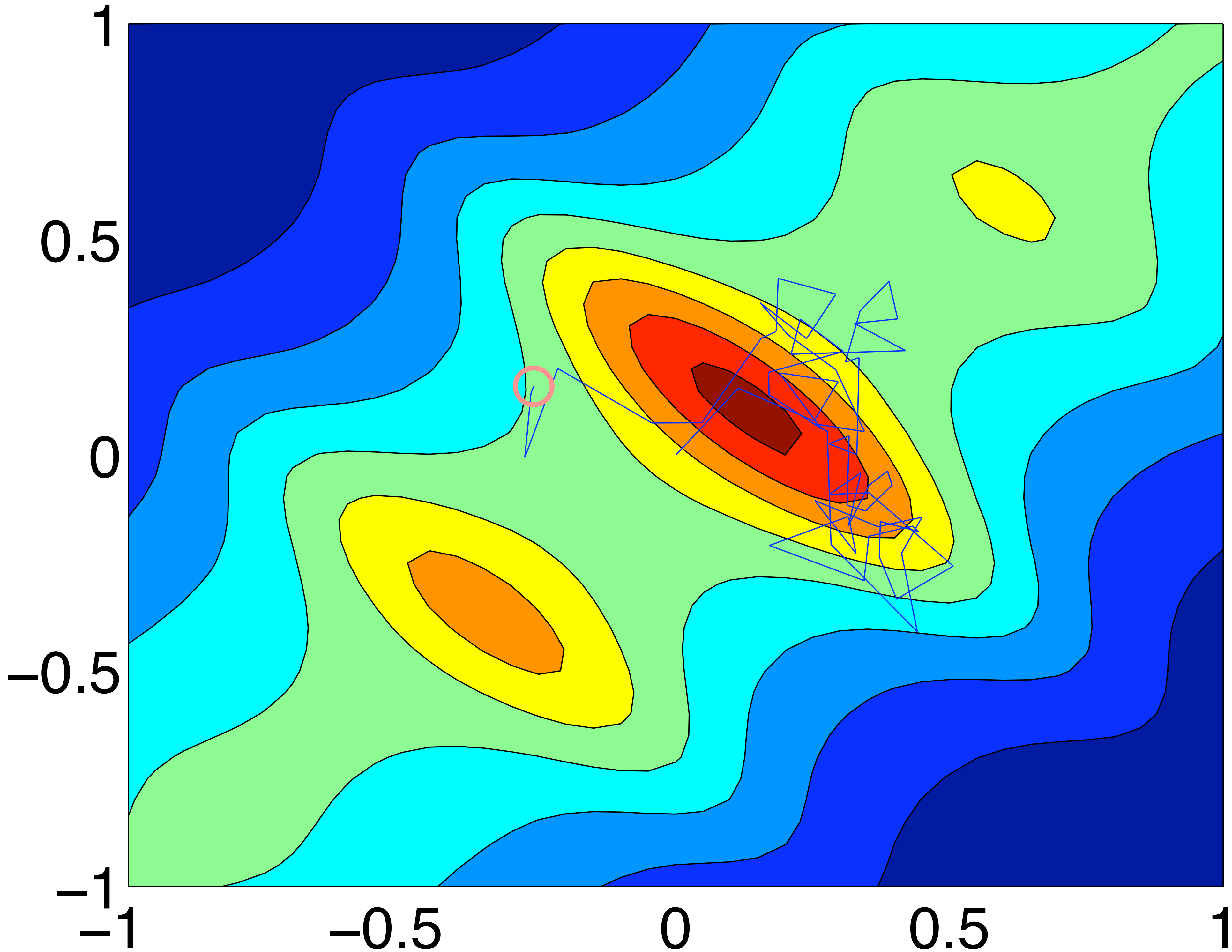
Review

- Metropolis-Hastings
  - way to design chain w/ stationary dist'n P(X)
  - proposal distribution Q(X' | X)
    - e.g., random walk N(X' | X, σ^2I)
  - accept w.p. min(1, $\frac{P(x')}{P(x_t)} \frac{Q(x_t \mid x')}{Q(x' \mid x_t)}$)
  - tension btwn long moves, high accept rate

MH example



MH example



In example

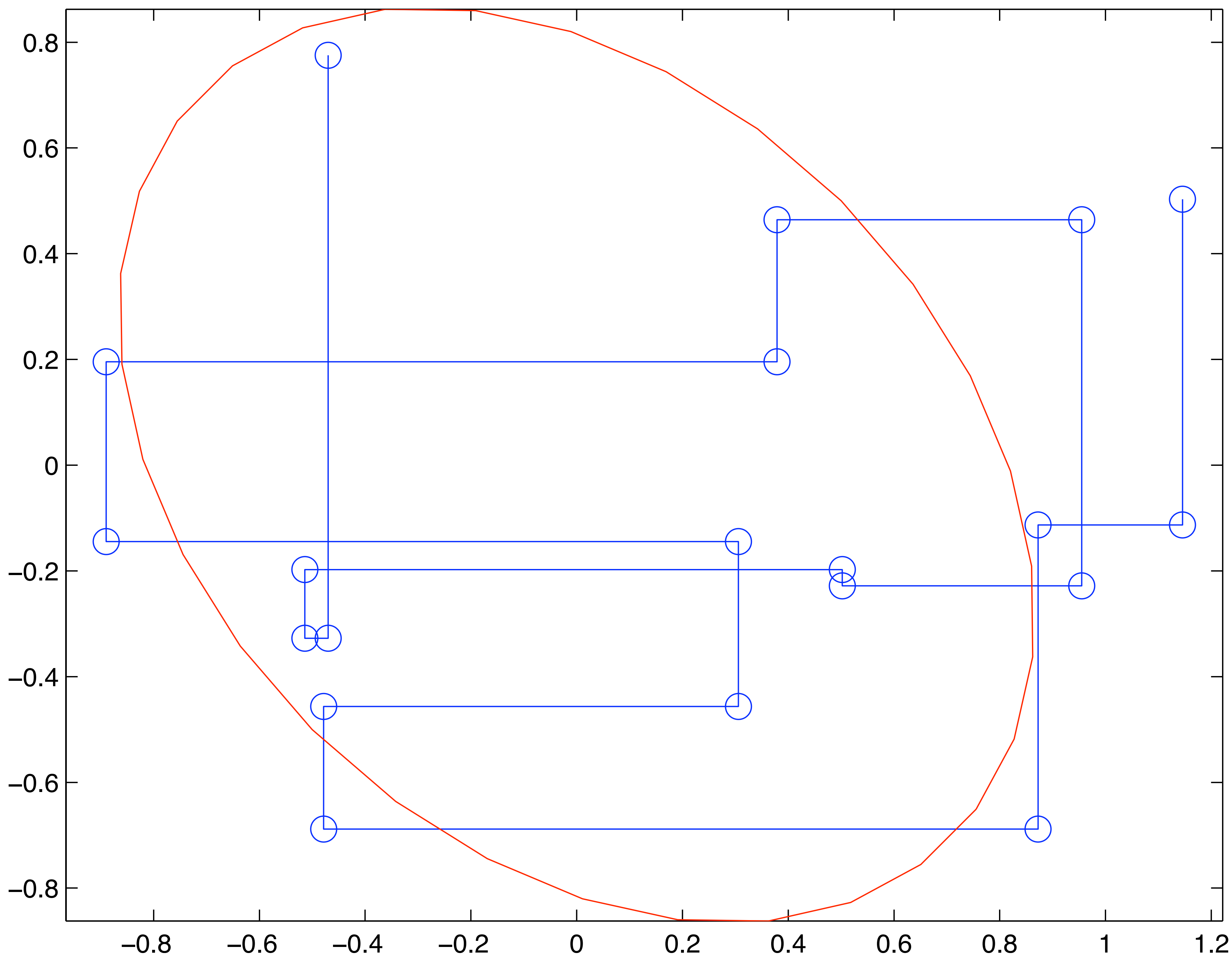
- $g(x) = x^2$
- True E(g(X)) = 0.28…
- Proposal: $Q(x' \mid x) = N(x' \mid x, 0.25^2 I)$
- Acceptance rate 55–60%
- After 1000 samples, minus burn-in of 100:

```
final estimate 0.282361
final estimate 0.271167
final estimate 0.322270
final estimate 0.306541
final estimate 0.308716
```

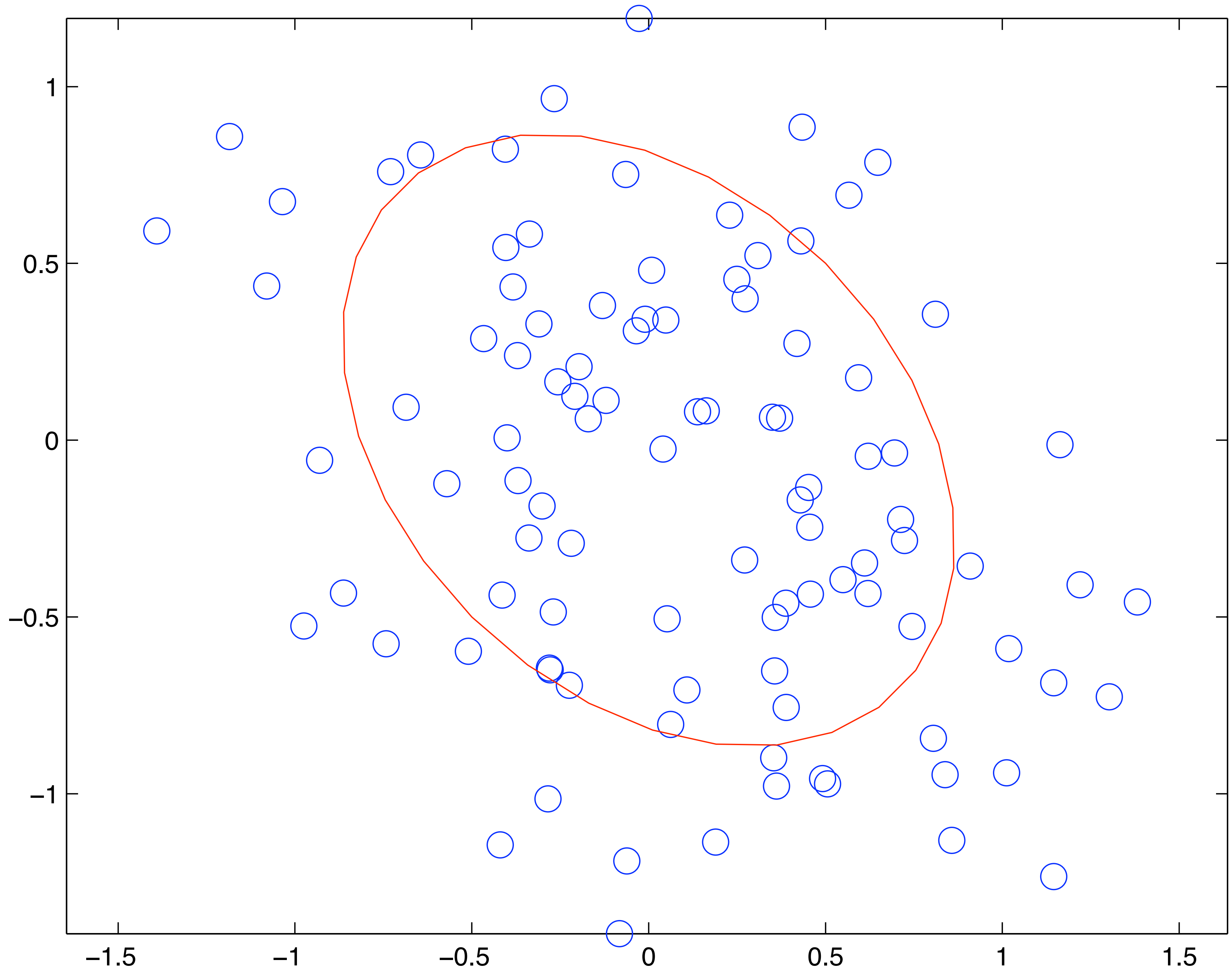
Gibbs sampler

- Special case of MH
- Divide $\mathbf{X}$ into blocks of r.v.s B(1), B(2), …
- Proposal Q:
  - pick a block i uniformly
  - sample $\mathbf{X}_{B(i)} \sim P(\mathbf{X}_{B(i)} \mid \mathbf{X}_{\neg B(i)})$
- Useful property: acceptance rate p = 1

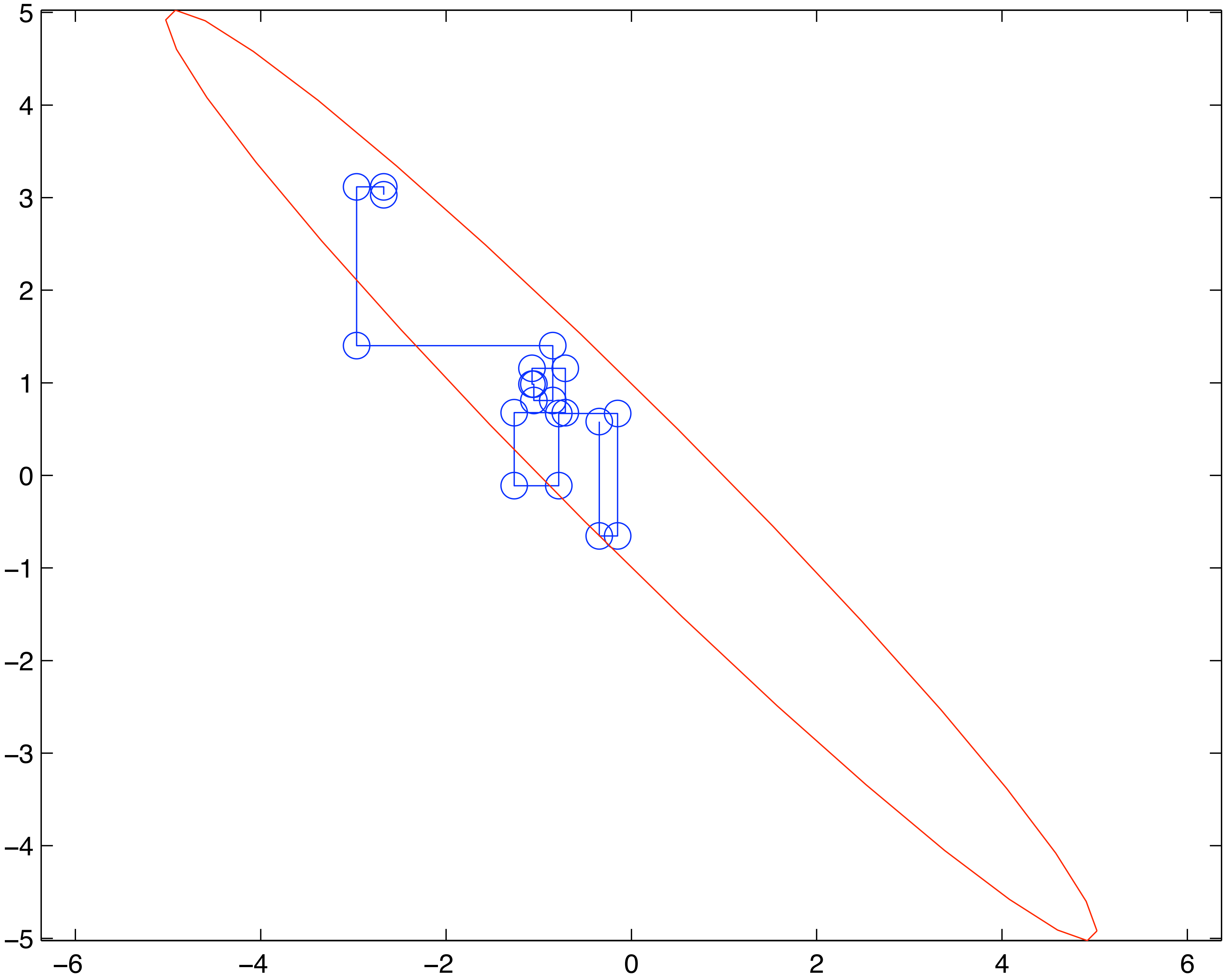
Gibbs example



Gibbs example



Gibbs failure example



Relational learning

- Linear regression, logistic regression: ***attribute-value*** learning
  - set of i.i.d. samples from P(X,Y)
- Not all data is like this
  - an attribute is a property of a single entity
  - what about properties of sets of entities?

Application: document clustering

10-601 Machine Learning Fall 2009

Geoff Gordon and Miroslav Dudik
School of Computer Science, Carnegie Mellon University

About | People | Lectures | Recitations | Homework | Exams | Projects

- Mailing lists
- Textbooks
- Grading
- Auditing
- Homework policy
- Collaboration policy
- Late policy
- Regrade policy
- Final project

Class lectures: Mondays and Wednesdays 10:30-11:50 in Newell Simon Hall 1305

Recitations: Wednesday, 6:00-8:00 pm GHC 8102

HW3 is out! It's due on Wednesday Oct 7, 10:30 am

Machine Learning is concerned with computer programs that learn to make better predictions or take better actions given increasing numbers of observations (e.g., programs that learn to spot high-risk medical patients, recognize human faces, recommend music and movies, or drive autonomous robots). This course covers theory and practical algorithms for machine learning from a variety of perspectives. We cover topics such as Bayesian networks, boosting, support-vector machines, dimensionality reduction, and reinforcement learning. The course also covers theoretical concepts such as bias-variance trade-off, PAC learning, margin-based generalization bounds, and Occam's Razor. Short programming assignments include hands-on experiments with various learning algorithms. Typical assignments include learning to automatically classify email by topic, and learning to automatically classify the mental state of a person from brain image data. The course will include a term project where the students will have opportunity to explore some of the class topics on a real-world data set in more detail.

Students entering the class with a pre-existing working knowledge of probability, statistics and algorithms will be at an advantage, but the class has been designed so that anyone with a strong numerate background can catch up and fully participate. This class is intended for Masters students and advanced undergraduates.

Announcement Emails

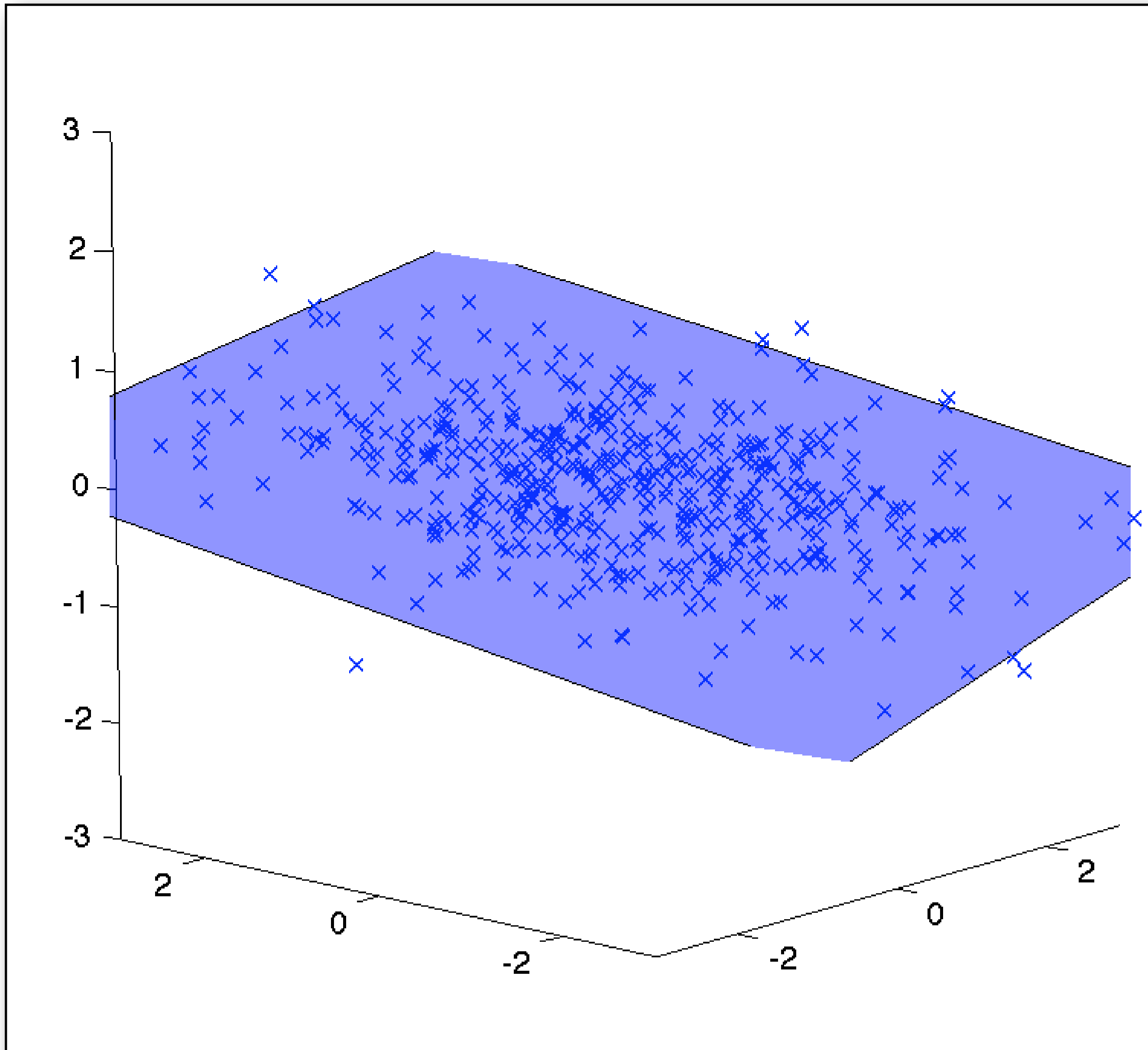
Application: recommendations

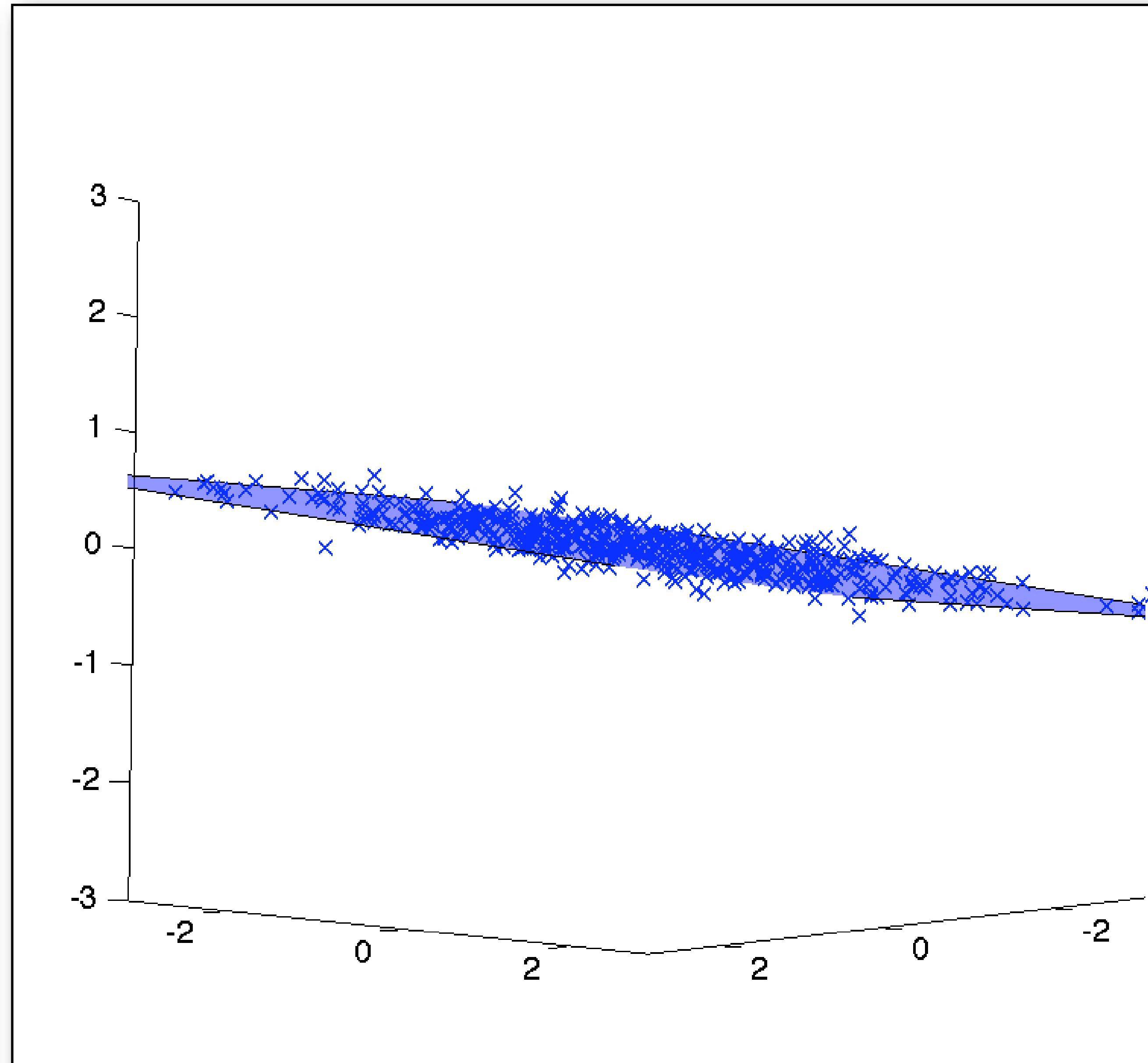
Latent-variable models

Best-known LVM: PCA

- Suppose X_{ij}, U_{ik}, V_{jk} all ~ Gaussian
  - yields ***principal components analysis***
  - or ***probabilistic PCA***
  - or ***Bayesian PCA***

PCA: the picture





Mean subtraction

- $U_{ik} \sim N(0, \nu^2)$
- $V_{jk} \sim N(0, \nu^2)$
- $X_{ij} \sim N(U_i \cdot V_j, \sigma^2)$

```
>> mu = mean(X(:));
>> colmu = mean(X - mu);
>> rowmu = mean(X' - mu)';
>> X = X - mu - repmat(colmu, size(X,1), 1) -
      repmat(rowmu, 1, size(X,2));
```

Data weights

- Let W_{ij} =

- Likelihood $\cdot$ prior =

- More generally, $W_{ij} \geq 0$

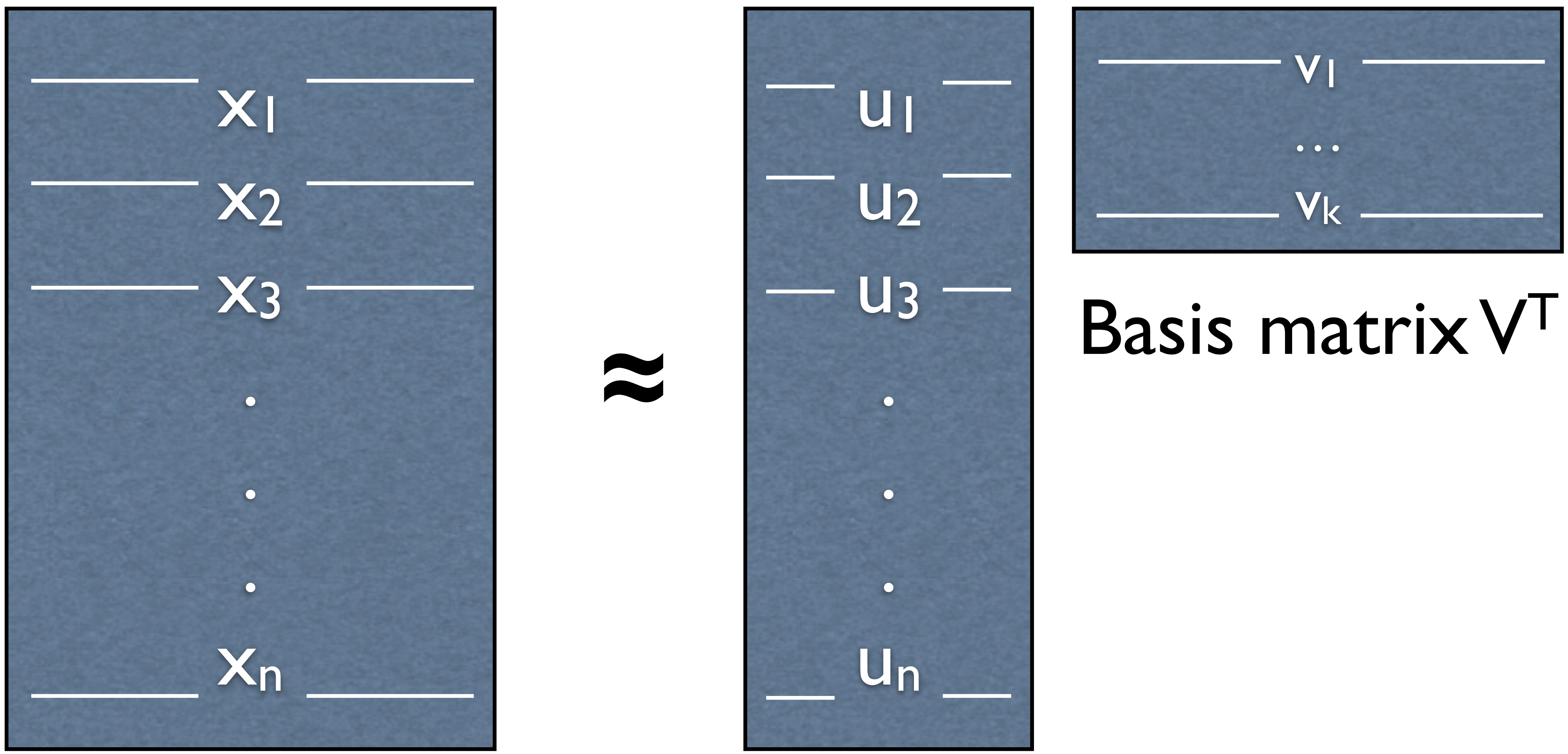
PCA: cartoon example

Movie

User

	1	2	3	4	5	6	...
A	1	1	0	0	1	0	...
B	0	1	1	0	0	0	...
C	1	1	0	1	1	0	...
D	1	0	0	1	1	0	...
E	0	1	0	1	0	0	...
F	0	1	1	1	0	1	...
...	...	...	...	...	...	...	...

PCA: cartoon example

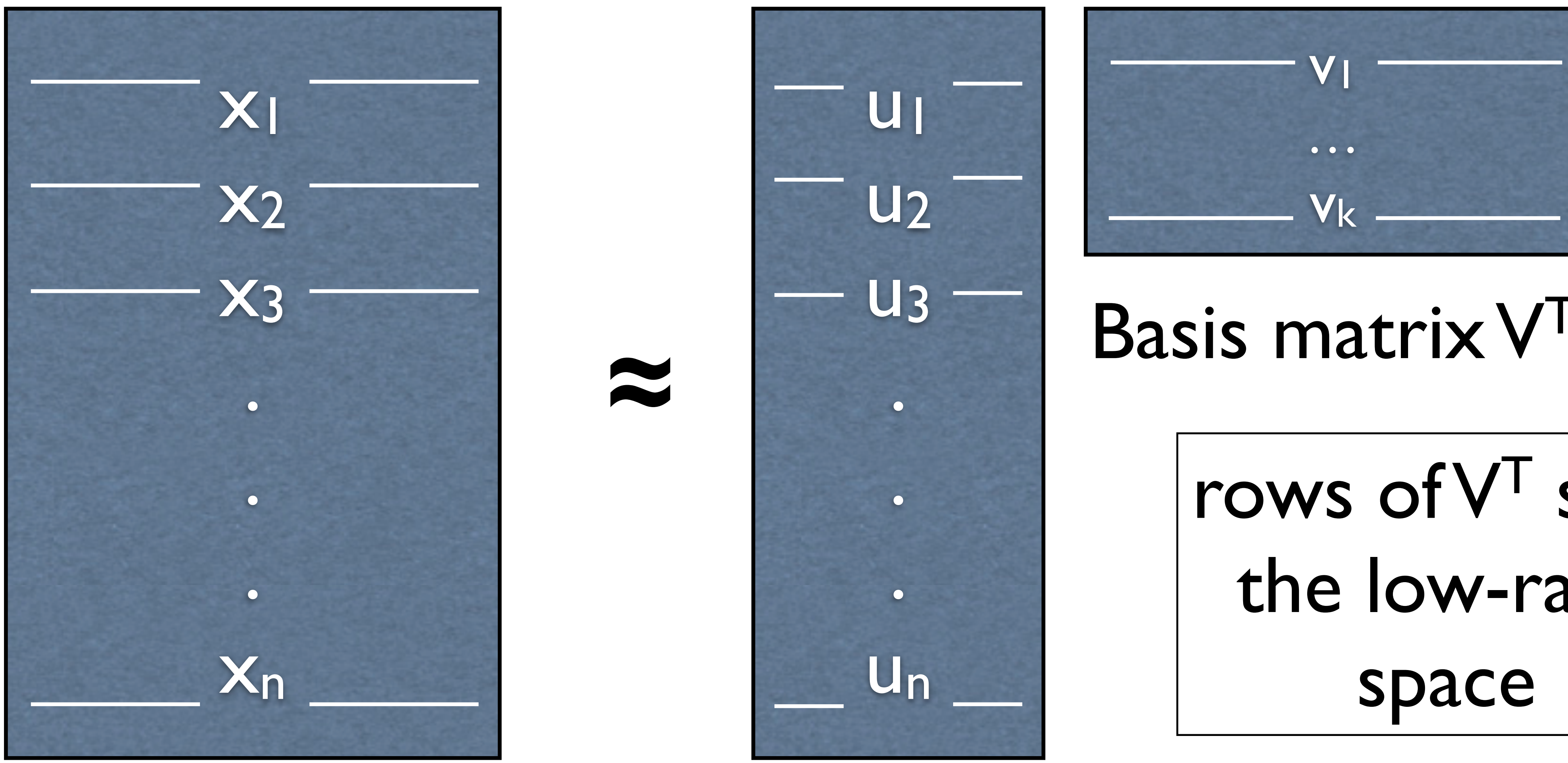


Basis matrix V^T

Data matrix X

Compressed matrix U

PCA: cartoon example



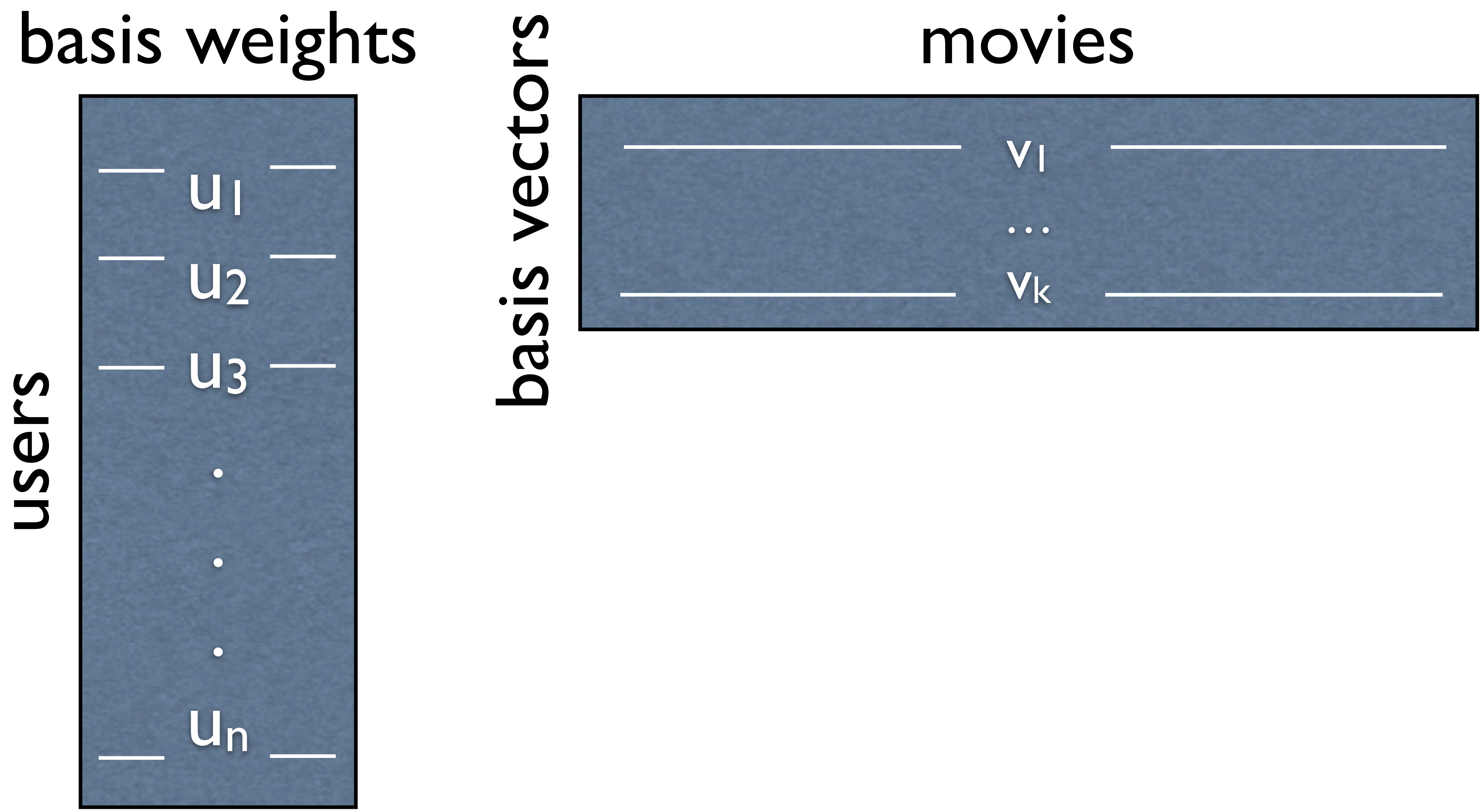
Basis matrix V^T

rows of V^T span the low-rank space

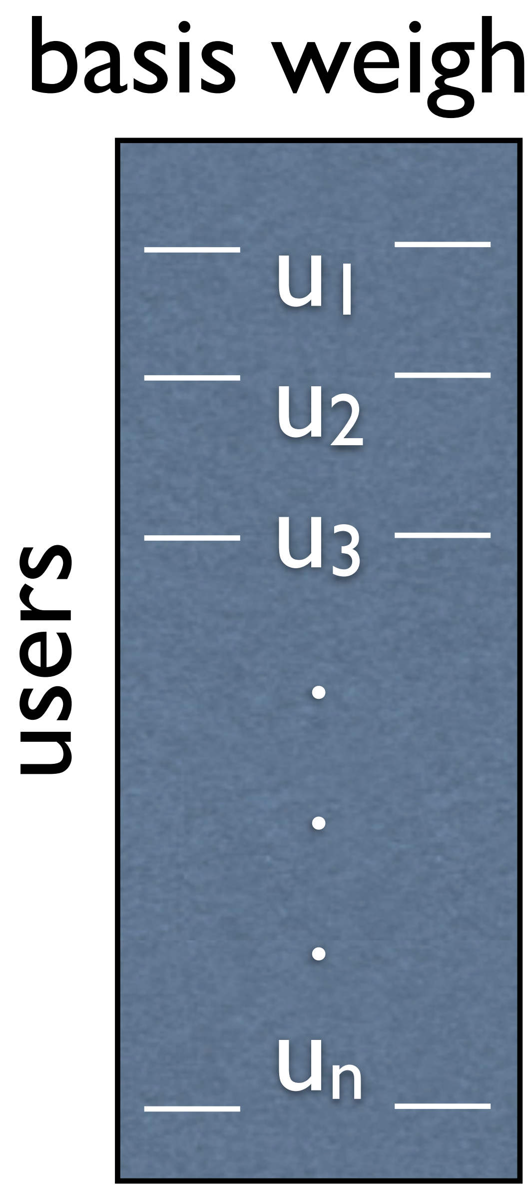
Data matrix X

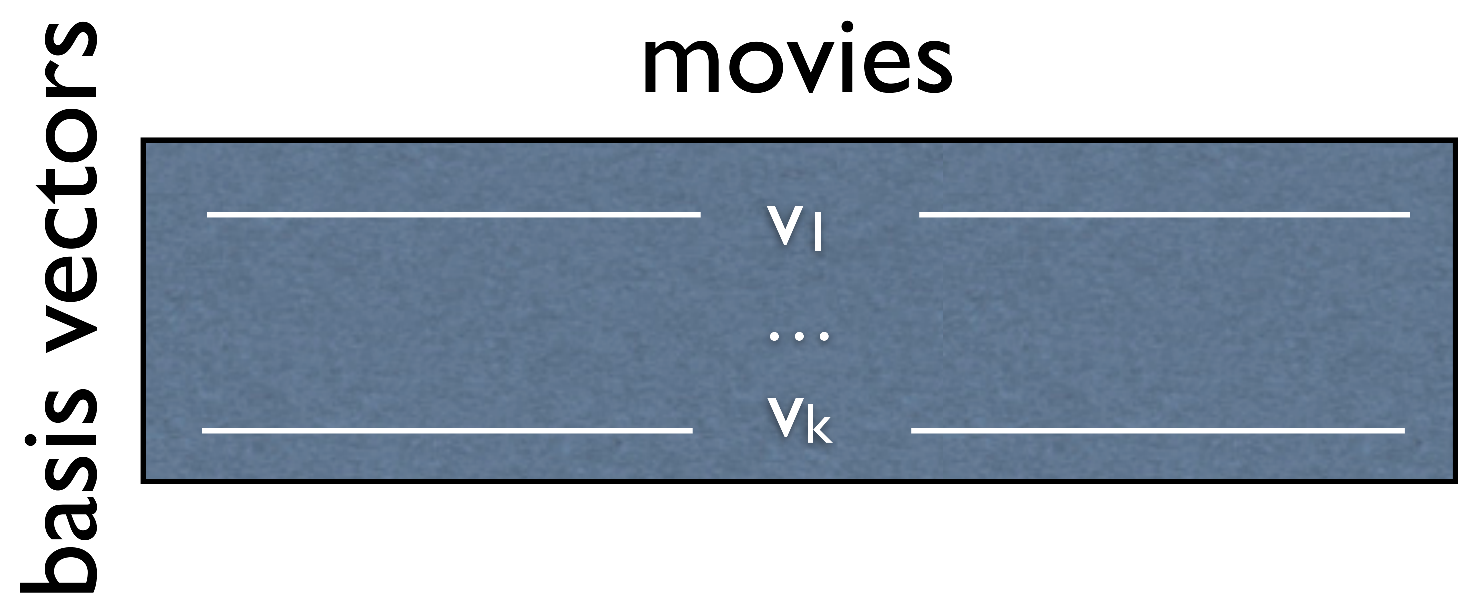
Compressed matrix U

Interpreting PCA



Interpreting PCA





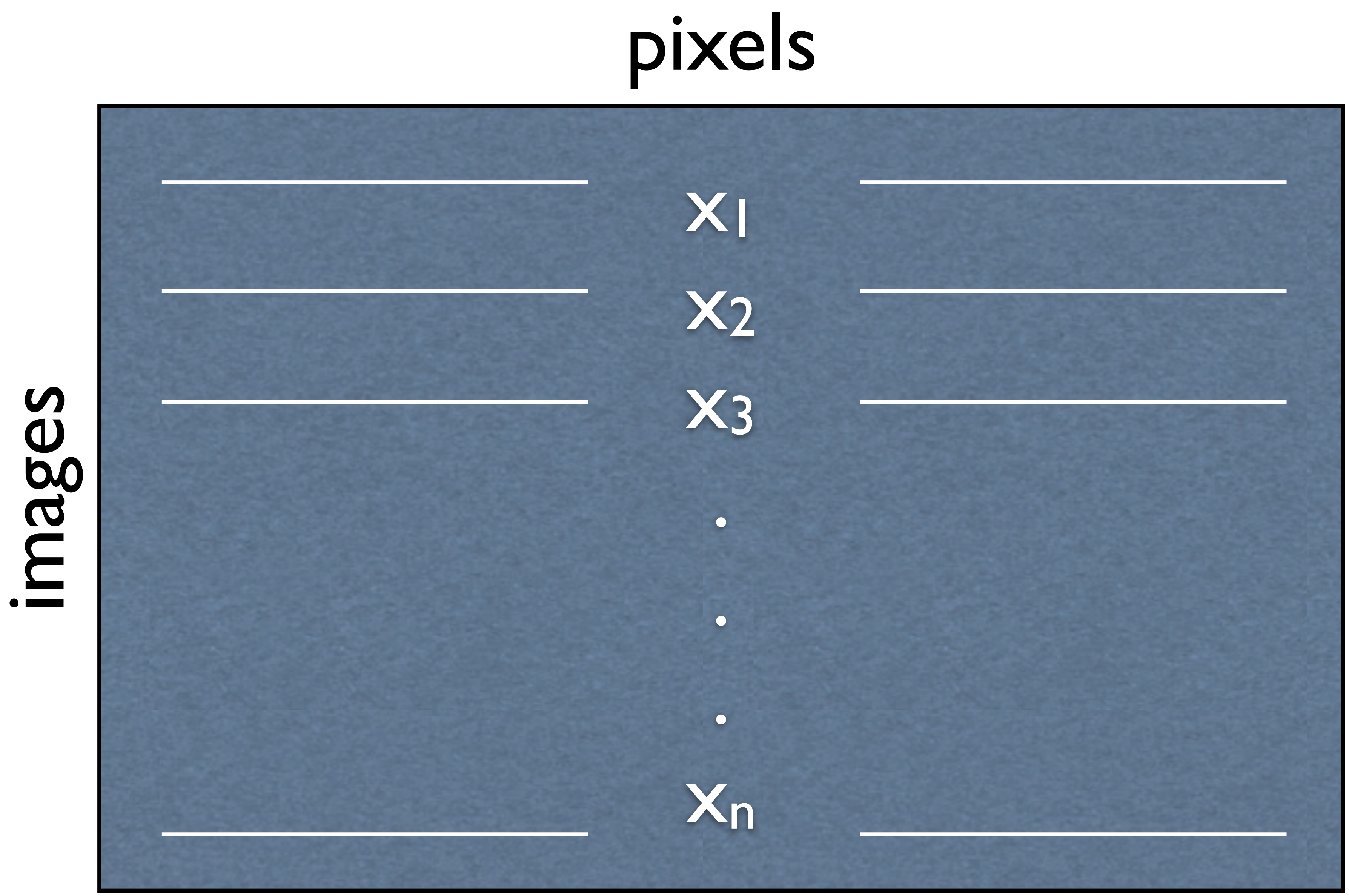
Basis vectors represent movies that ***vary together***

Weights say how much each user cares about each type of movie

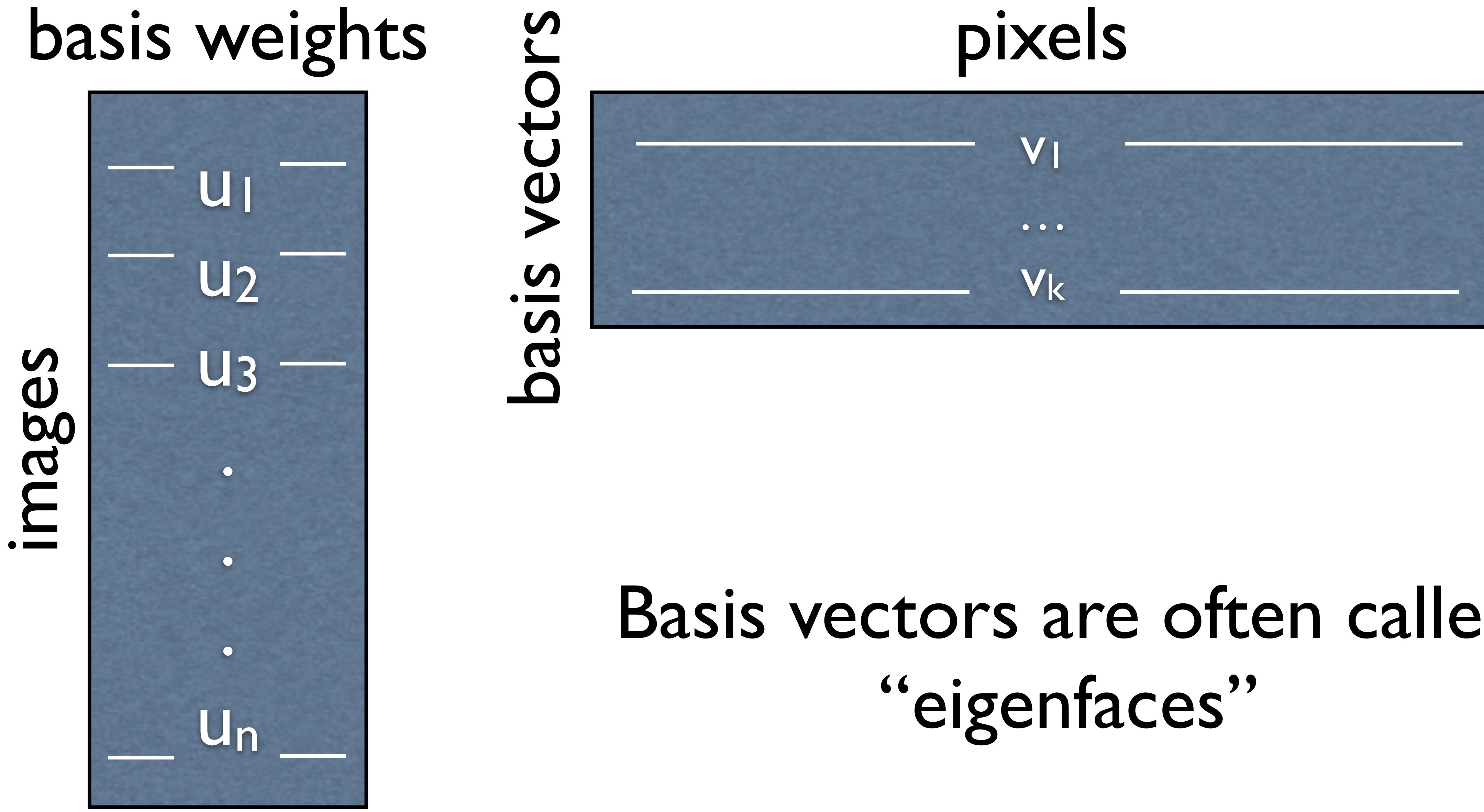
Another use of PCA

face images from Groundhog Day, extracted by Cambridge face DB project

Image matrix



Result of factoring



Basis vectors are often called "eigenfaces"

Eigenfaces

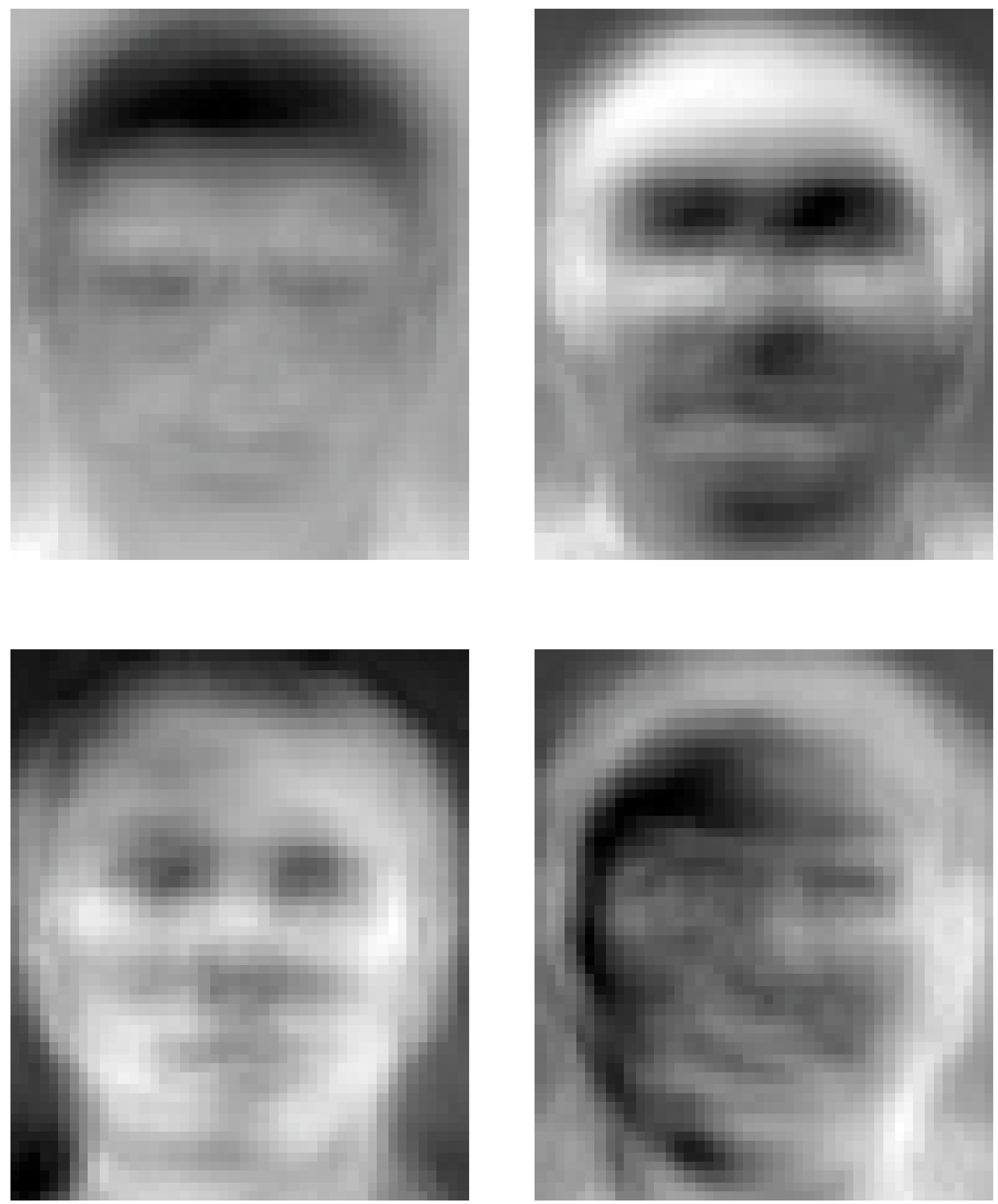


image credit: AT&T Labs Cambridge

PCA: finding the MLE

- PCA:
  - $U_{ik} \sim N(0, \nu^2)$
  - $V_{jk} \sim N(0, \nu^2)$
  - $X_{ij} \sim N(U_i \cdot V_j, \sigma^2)$
  - $\sigma/\nu \to 0$

PCA & SVD

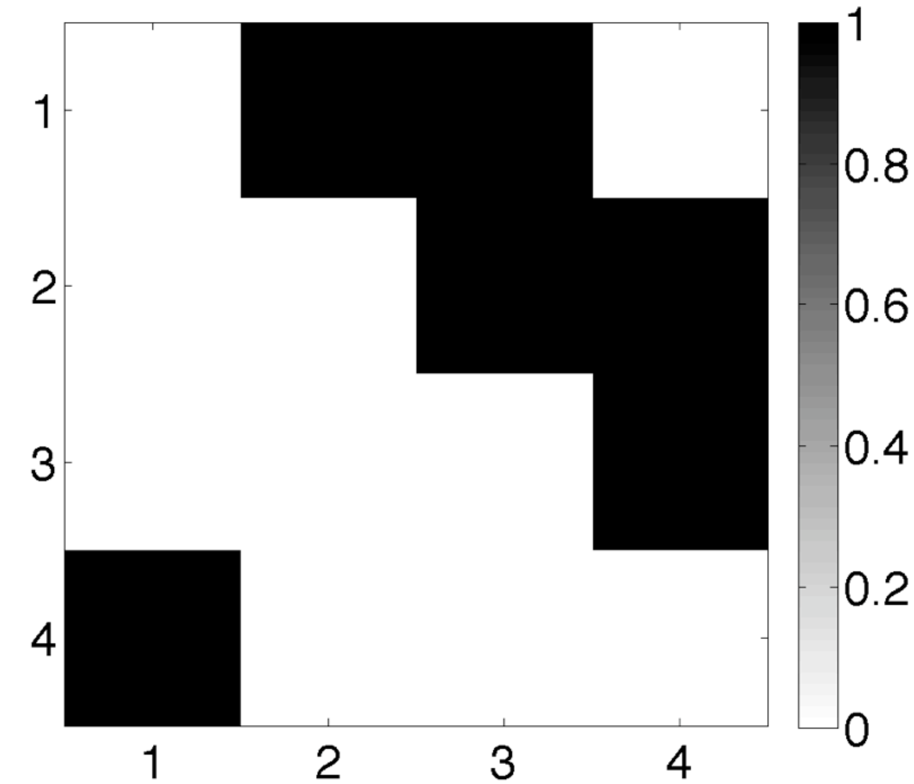
- The ***singular value decomposition*** is
  - $X = R\ \Sigma\ S^T$
  - R, S orthonormal; $\Sigma \geq 0$ diagonal
  - All matrices can be expressed this way
  - See `svd`, `svds` in Matlab

- So, PCA is U = V =

PageRank

- SVD is pretty useful: turns out to be main computational step in other models too
- A famous one: PageRank
  - Given: web graph (V, E)
  - Predict: which pages are important

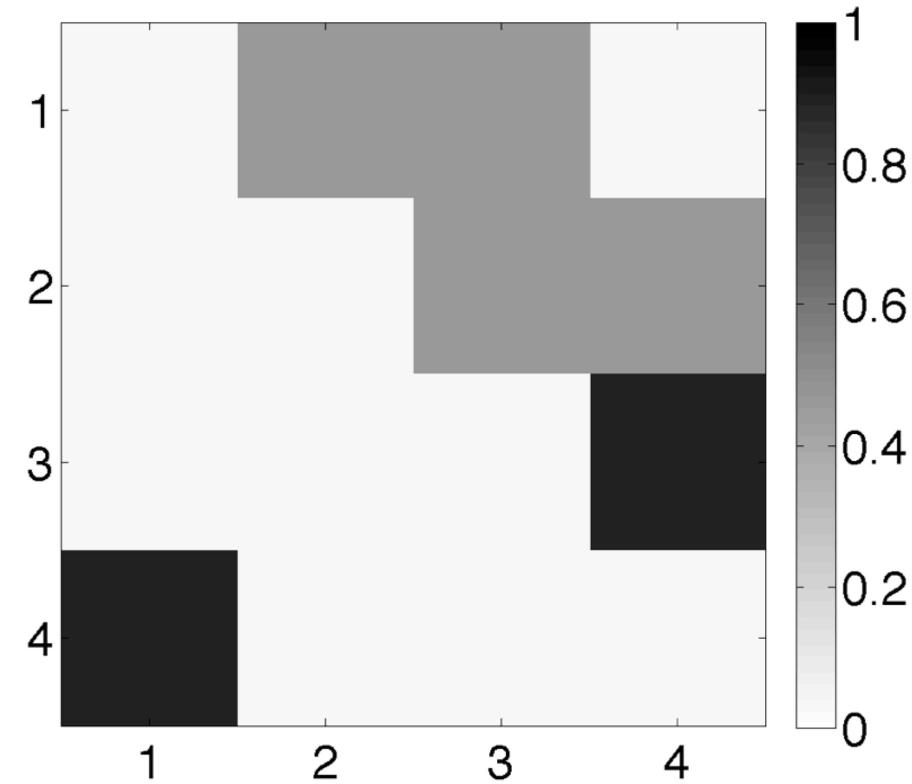
PageRank: adjacency matrix



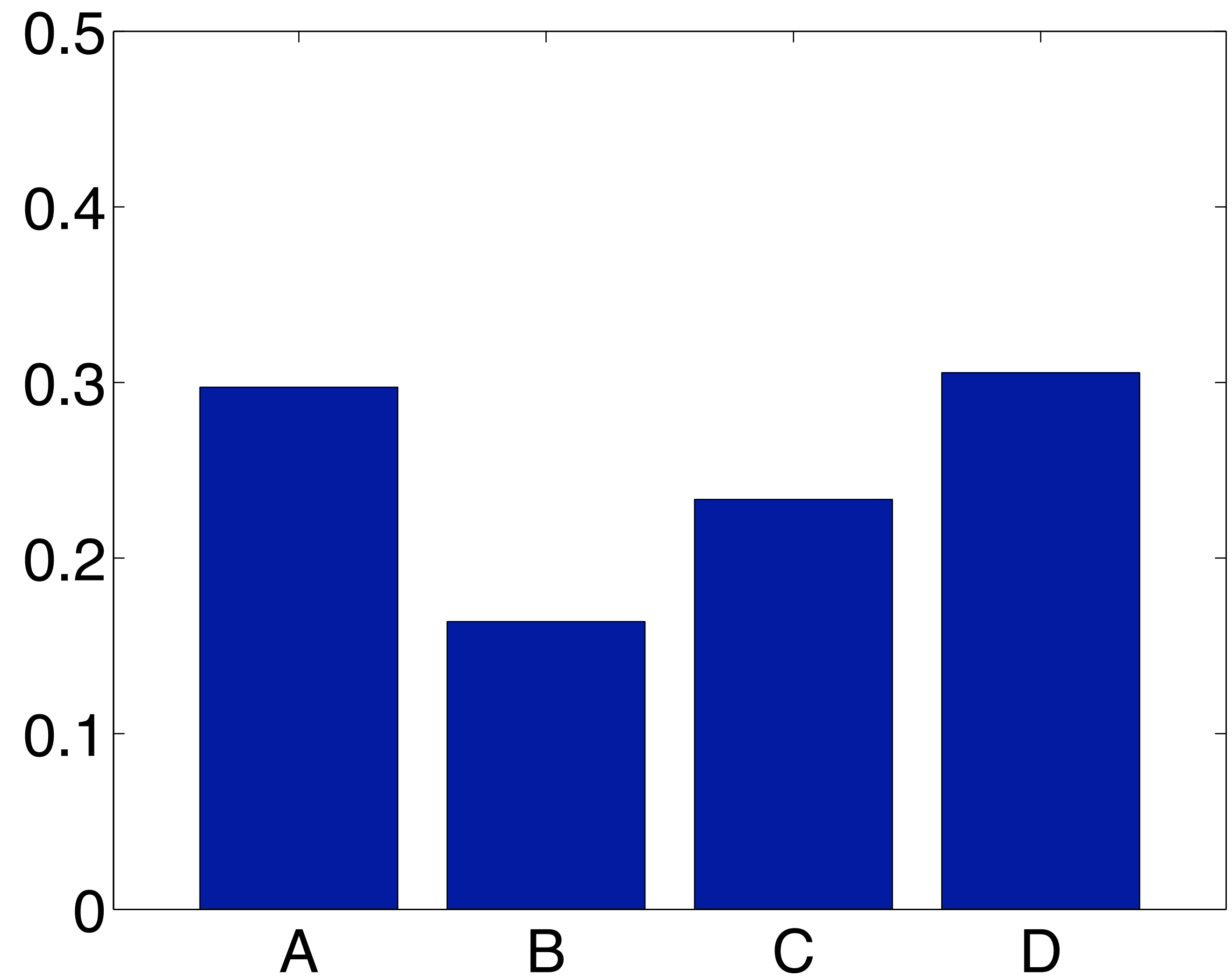
Random surfer model

- W. p. α:
- W. p. (1–α):

- Intuition: page is important if a random surfer is likely to land there



Stationary distribution



Thought experiment

- What if A is symmetric?
  - note: we're going to stop distinguishing A, A'

- So, stationary dist'n for symmetric A is:
- What do people do instead?

Spectral embedding

- Another famous model: spectral embedding (and its cousin, spectral clustering)
- Embedding: assign low-D coordinates to vertices (e.g., web pages) so that similar nodes in graph $\Rightarrow$ nearby coordinates
  - A, B similar = random surfer tends to reach the same places when starting from A or B

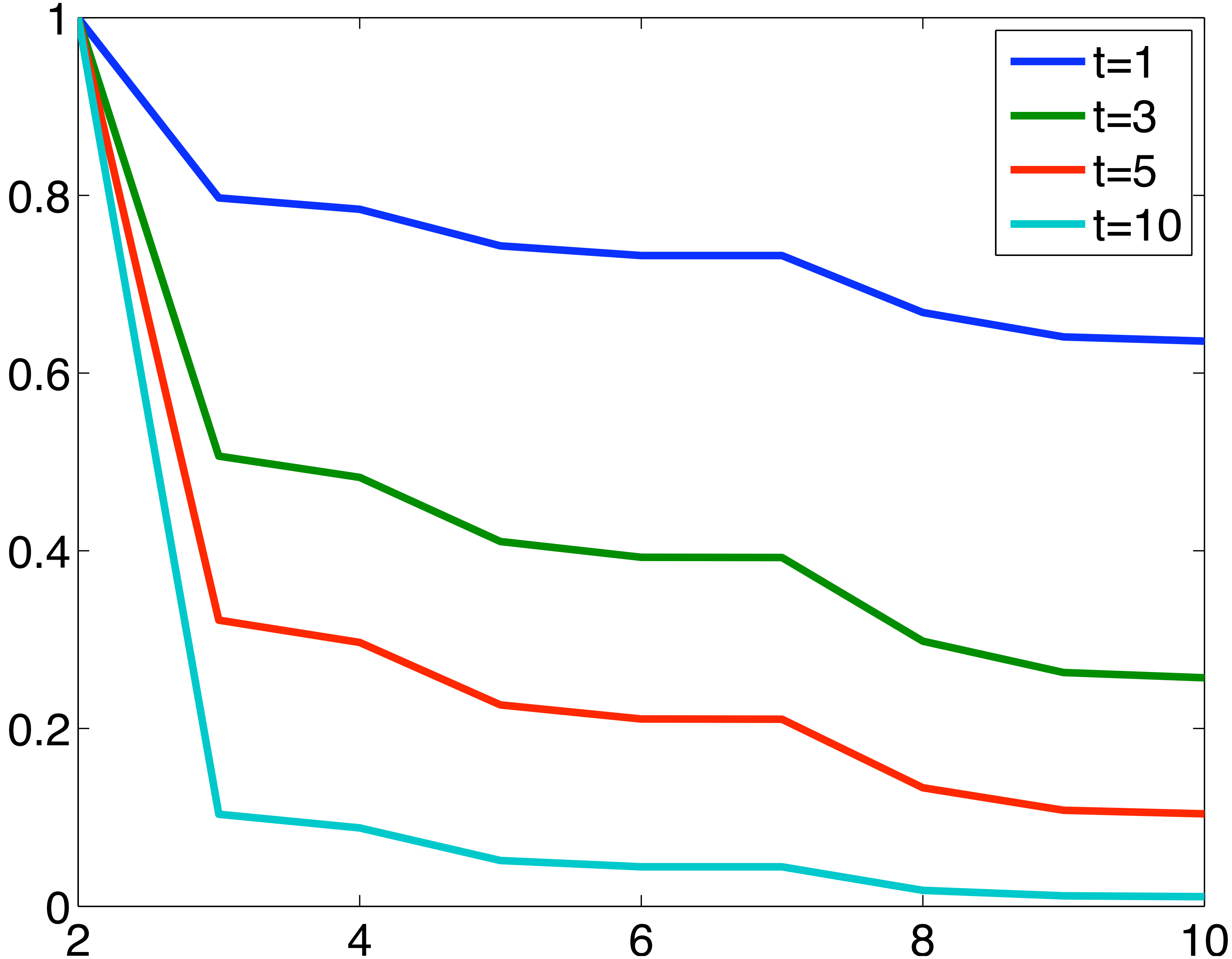
Where does random surfer reach?

- Given graph:
- Start from distribution π
  - after 1 step: P(j | π, 1-step) =
  - after 2 steps: P(j | π, 2-step) =
  - after t steps:

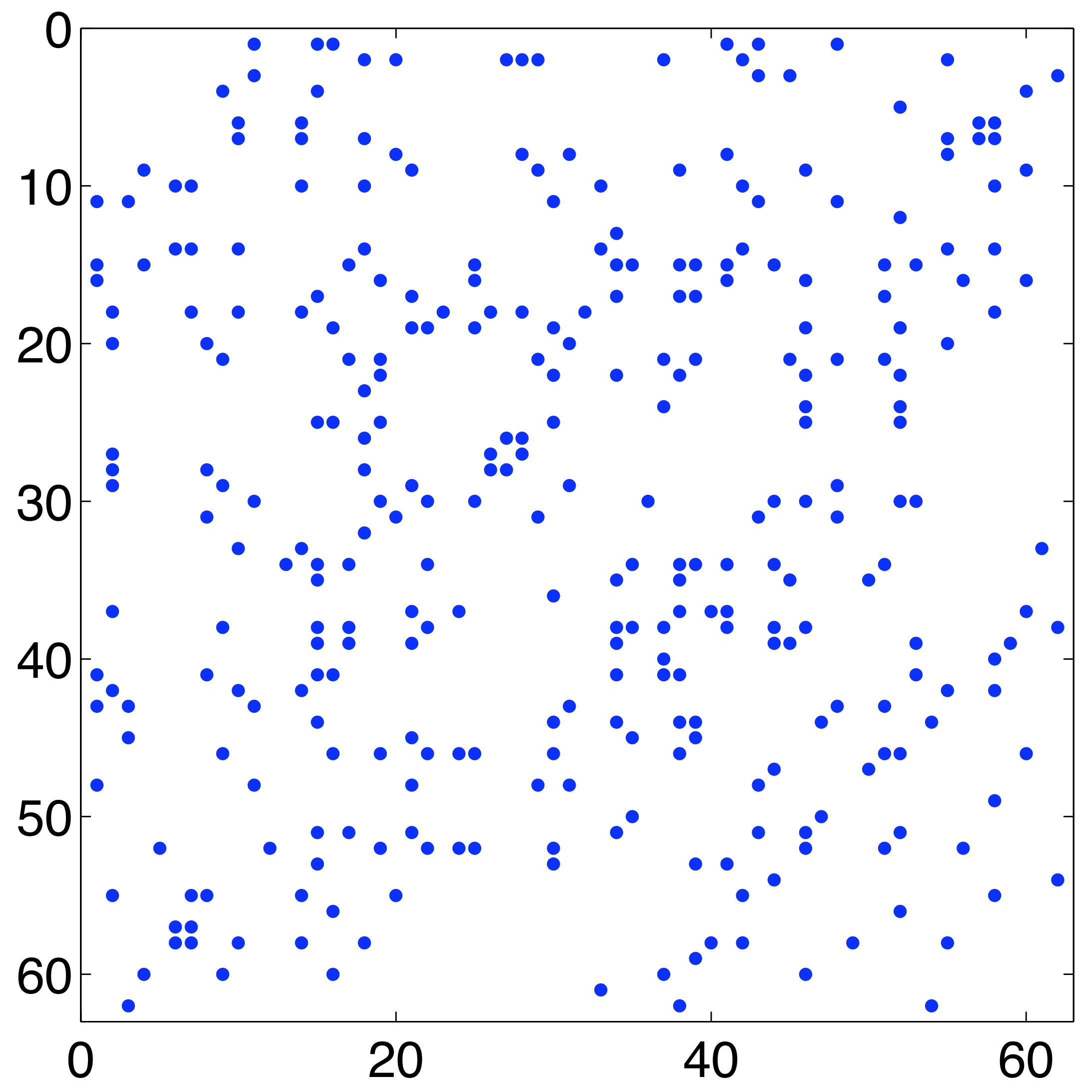
Similarity

- A, B similar = random surfer tends to reach the same places when starting from A or B
- P(j | π, t-step) =
  - If π has all mass on i:
  - Compare i & j:
  - Role of Σ^t:

Role of Σ^t (real data)



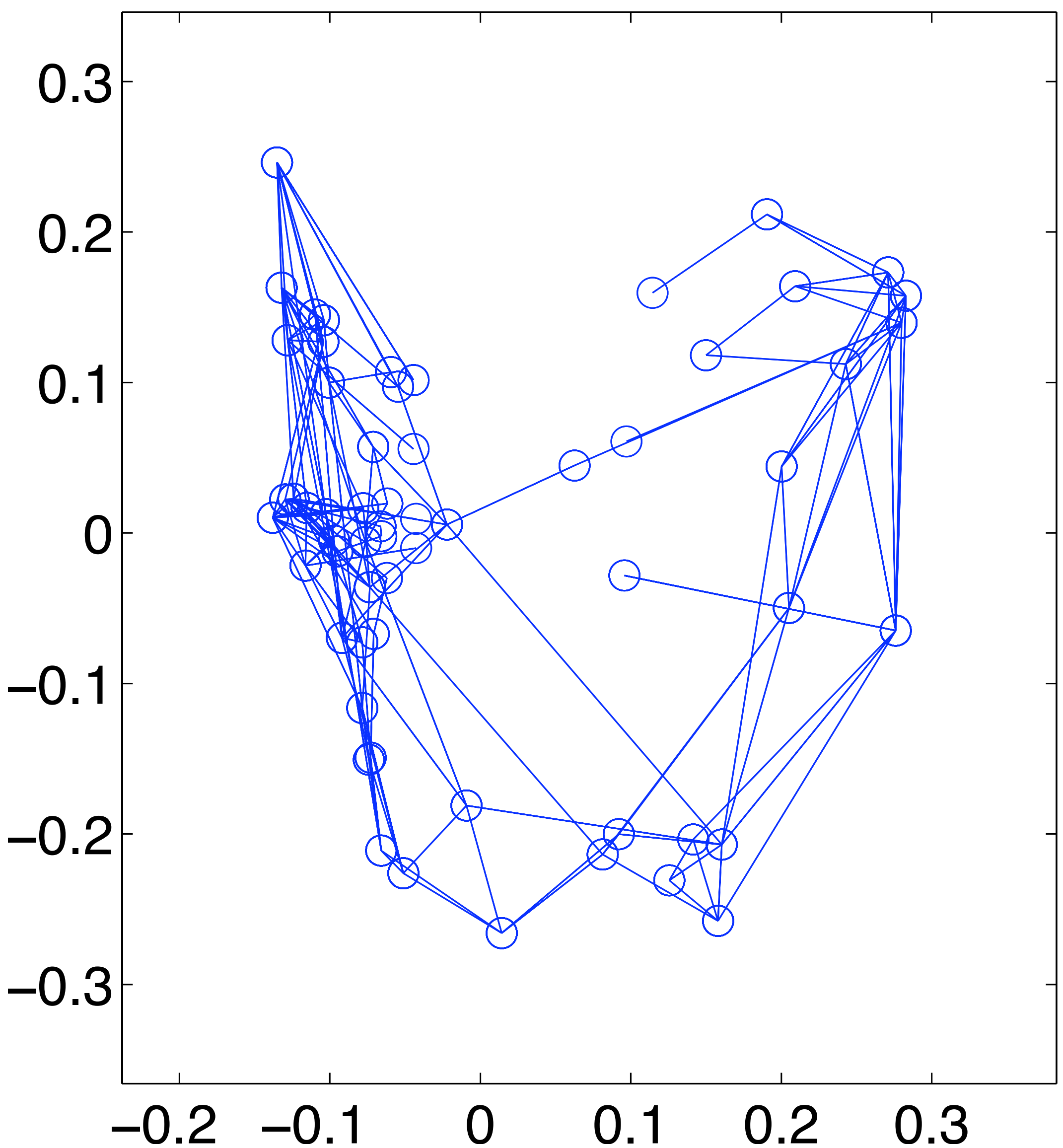
Example: dolphins



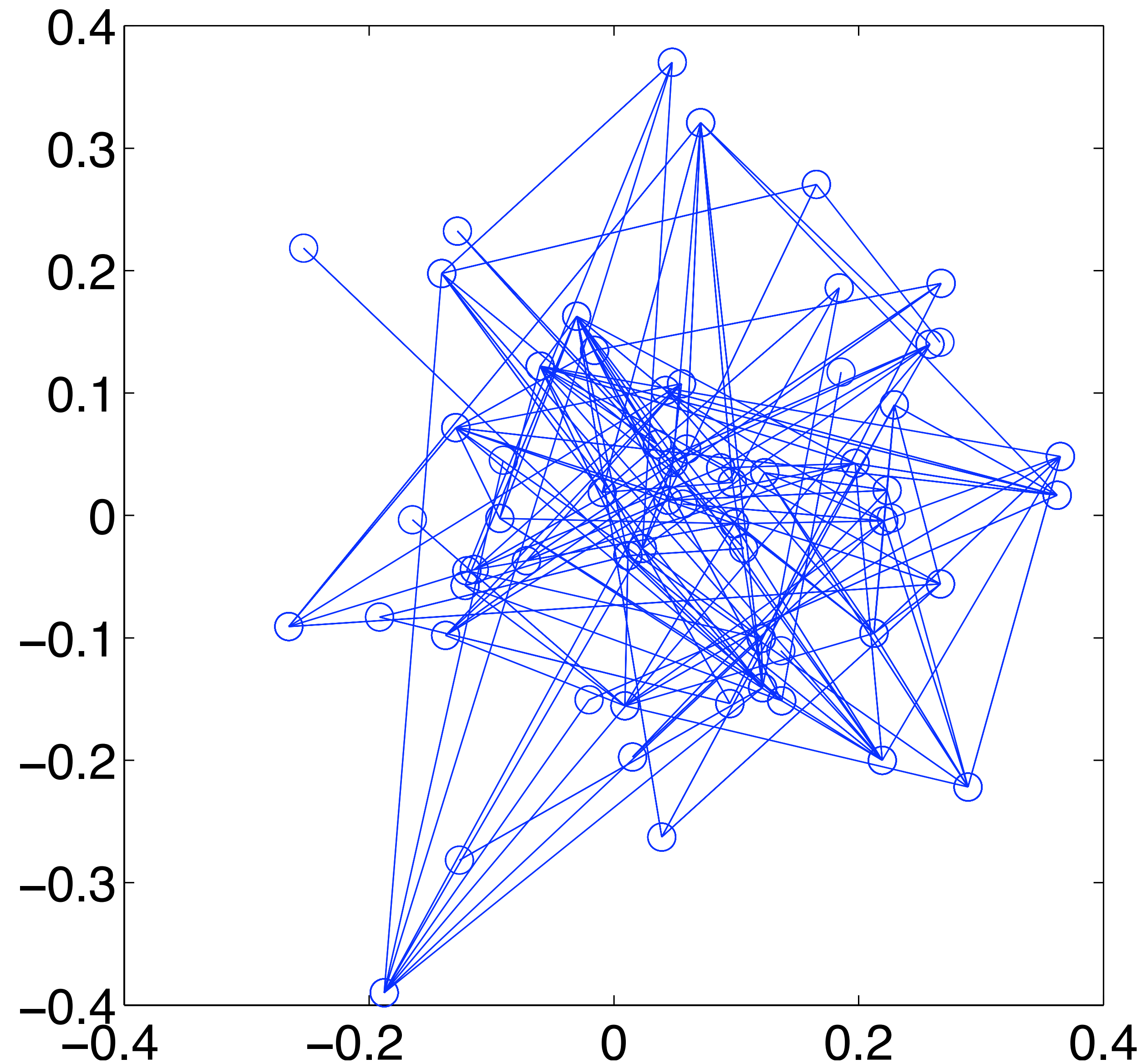
- 62-dolphin social network near Doubtful Sound, New Zealand
  - A_{ij} = 1 if dolphin i friends dolphin j

(Lusseau et al., 2003)

Dolphin network

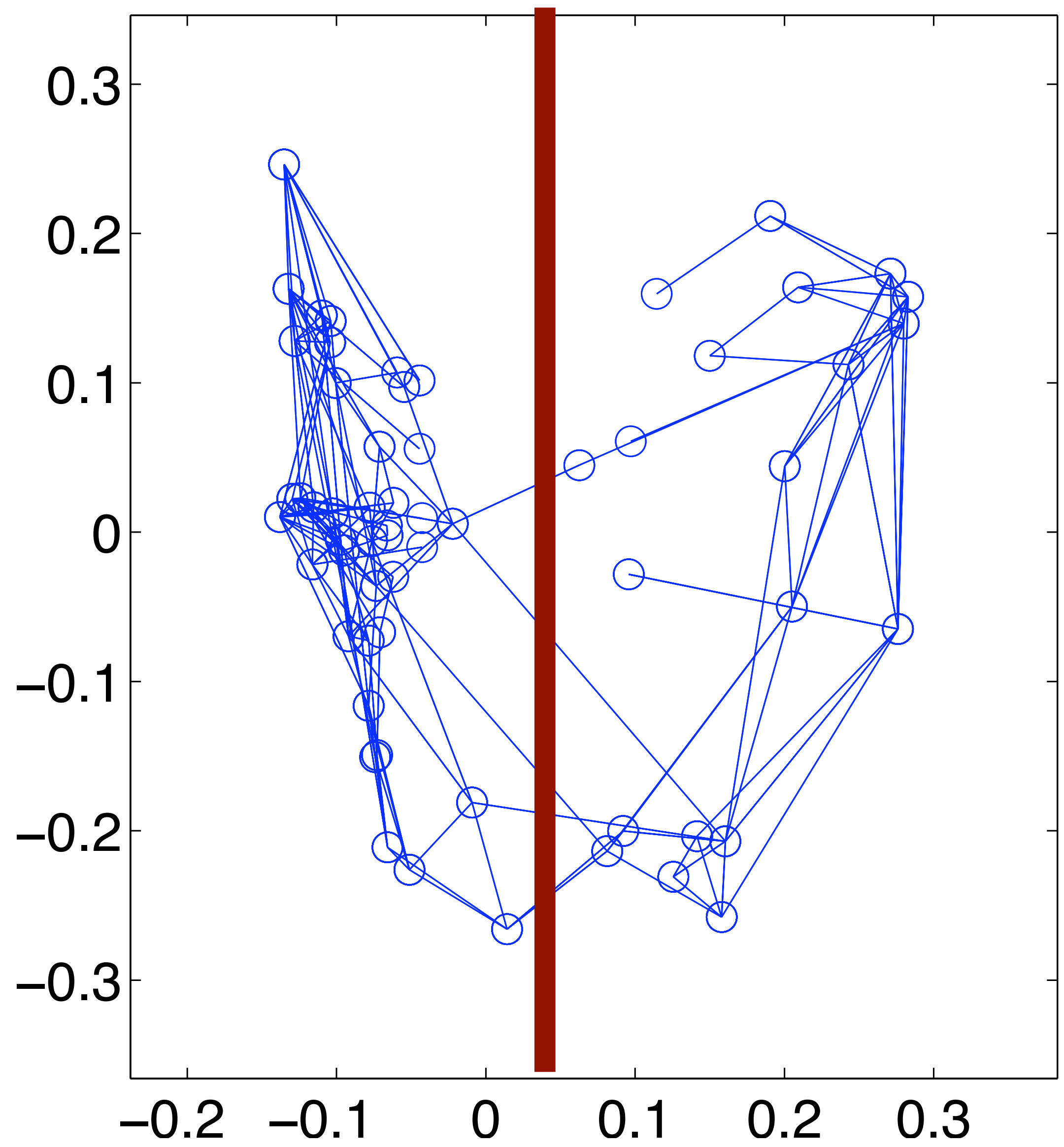


spectral embedding



random embedding

Spectral clustering



- Use your favorite clustering algorithm on coordinates from spectral embedding