CSCI567 Machine Learning (Spring 2018)

Michael Shindler

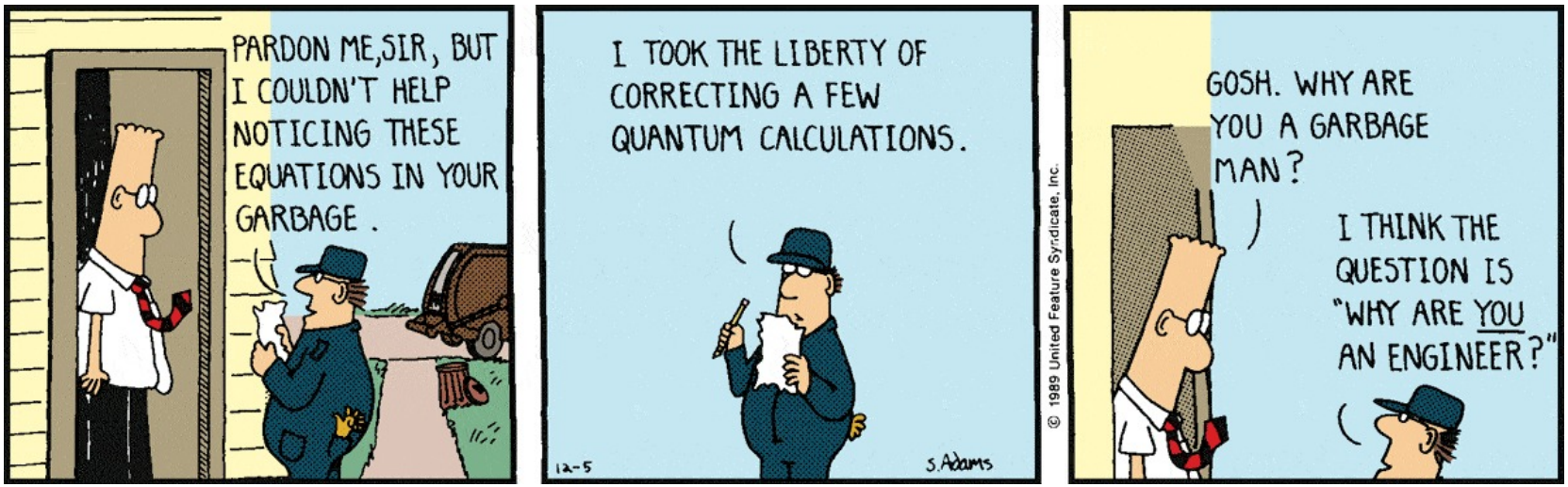
Lecture 22, April 11 2018

Outline

1. Review of last lecture
2. Another example: Latent Dirichlet Allocation
3. Dimensionality reduction

Outline

There are a few fixes from last time



Parameter estimation for HMM: what you need to know

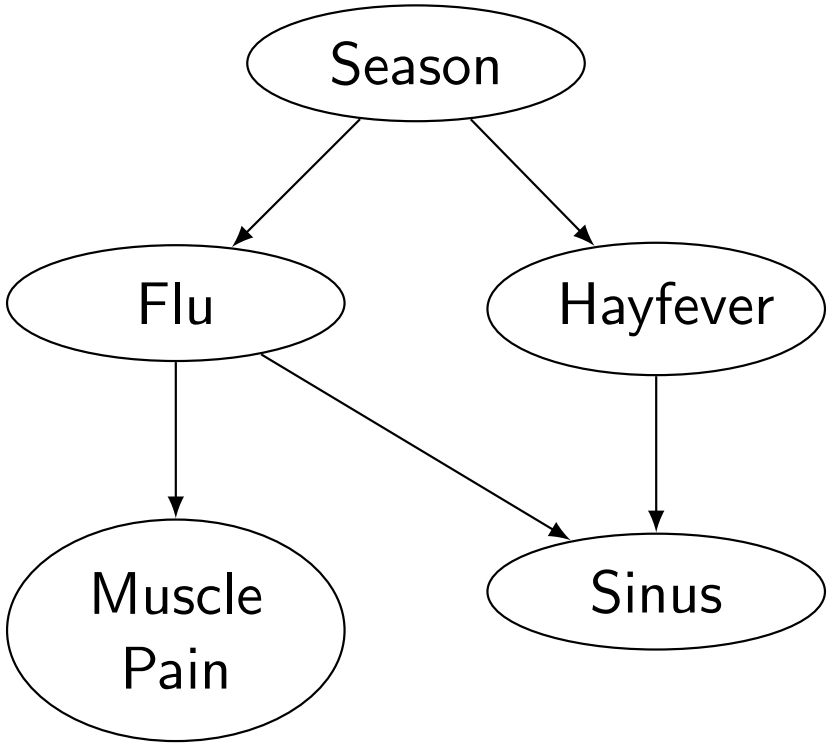
- What are the parameters of HMM?
- How would you estimate them if complete data is given to you? For discrete HMMs, this amounts to counting and calculating the frequencies of event occurring (ie, the fraction/percentage of things happening)
- To estimate with incomplete data, what do you need to do?
  - High-level idea: EM
  - Concretely: compute the posteriors: *what are the posteriors?* How to compute them? What kind of procedures are involved?

Graphical Models

- Bayes Nets
  - Probabilistic distribution represented with directed acyclic graphs (DAGs)
- Markov Networks
  - Probabilistic distribution represented with undirected graphs.

Exploring structures

- Draw links between variables
  - indicate dependencies and encode independence
  - Ex: flu and hayfever are independent in any given season
  - They independently occur *conditioned* on season
- This is example of Bayes Networks
  - Directed acyclic graphs
  - Compact representation of joint distribution



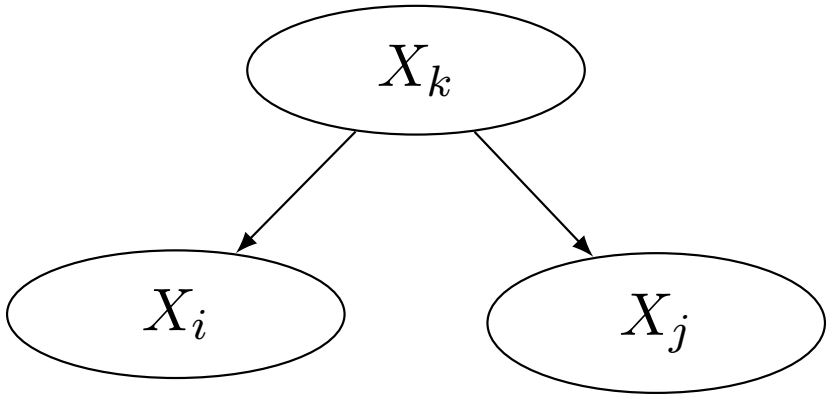
The key concept

- Conditional independence

$$X_i \perp\!\!\!\perp X_j | X_k$$

- This allows us to write:

Graphical representation:



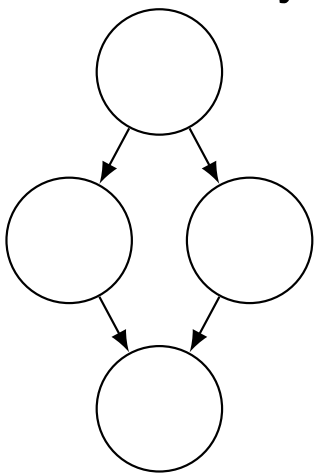
$$\begin{aligned} p(X_i, X_j, X_k) &= p(X_i|X_j, X_k)p(X_j, X_k) \\ &= p(X_i|X_k)p(X_j|X_k)p(X_k) \end{aligned}$$

Formal definition of Bayesian Networks

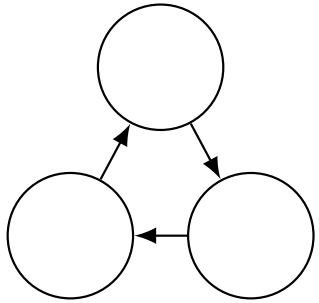
Structure (Graph G):

- Vertex are R.V.
- Edge: child depends on parent
- Is a DAG

This is okay:



This is not:



Conditional probability distributions (CPD): $P(X_i|Pa_{X_i})$ for every vertex.

$$P(X_1, X_2, \ldots, X_N) = \prod_{i=1}^{N} P(X_i|Pa_{X_i})$$

Models we have seen so far

- Naive Bayes
- Gaussian mixture models
- Hidden Markov models

Challenge: can you draw each above model as Bayesian network? (Note that I have never drawn GMM directly – please do this exercise.) Using the structure of the Bayesian network and the definition, can you write down the joint distribution? Are the same as we had been discussion about those models?

Suggested Readings

- Eugene Charniak's famous "Bayesian Networks without Tears" `https://www.aaai.org/ojs/index.php/aimagazine/article/download/918/836`
- Kevin Murphy's tutorial: `https://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html` Please read the "Representation" and "Inference".

Outline

Credits

The following slides are taken from Prof. David Blei's Machine Learning Summer School (2009) tutorial.
The full slides are posted on Piazza; I encourage you to read them.

Topic Models

David M. Blei

Department of Computer Science
Princeton University

September 1, 2009

The problem with information

www.betaversion.org/~stefano/linotype/news/26/

As more information becomes available, it becomes more difficult to access what we are looking for.

We need new tools to help us organize, search, and understand these vast amounts of information.

Topic modeling

Candida Hofer

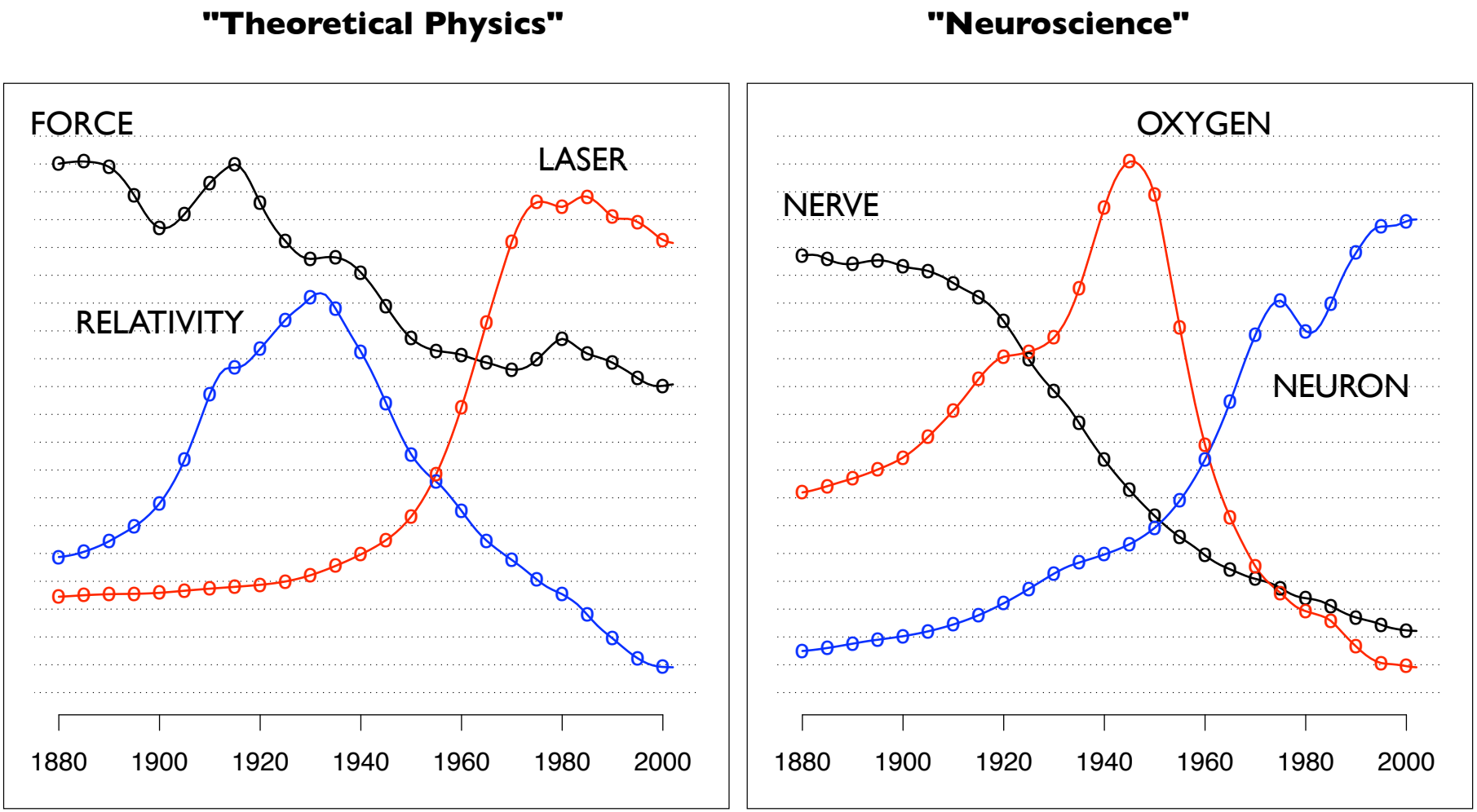
Topic modeling provides methods for automatically organizing, understanding, searching, and summarizing large electronic archives.

1. Uncover the hidden topical patterns that pervade the collection.
2. Annotate the documents according to those topics.
3. Use the annotations to organize, summarize, and search the texts.

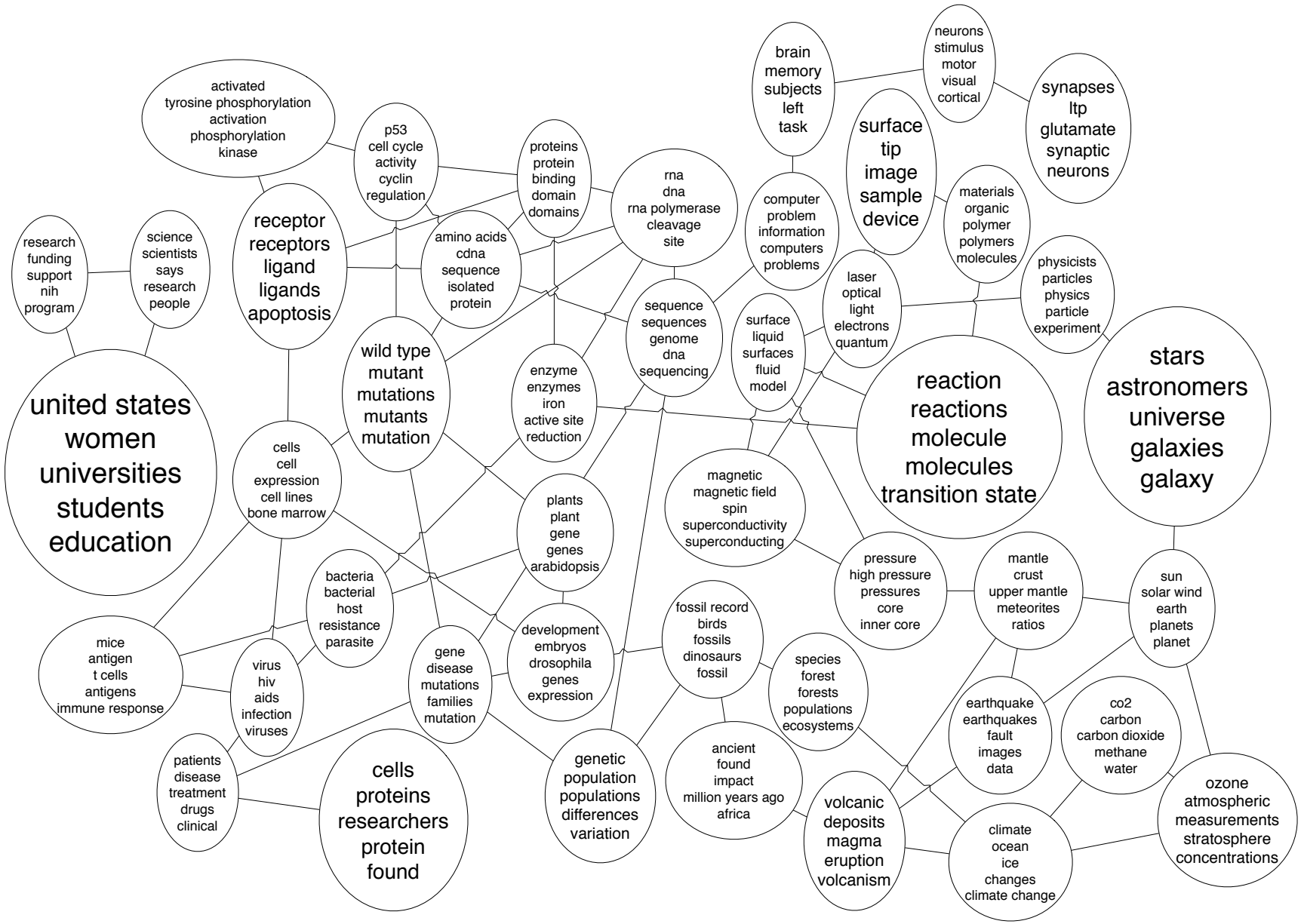
Discover topics from a corpus

human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

Model the evolution of topics over time



Model connections between topics



Annotate images

SKY WATER TREE
MOUNTAIN PEOPLE

SCOTLAND WATER
FLOWER HILLS TREE

SKY WATER BUILDING
PEOPLE WATER

FISH WATER OCEAN
TREE CORAL

PEOPLE MARKET PATTERN
TEXTILE DISPLAY

BIRDS NEST TREE
BRANCH LEAVES

Topic modeling topics

From a machine learning perspective, topic modeling is a case study in applying hierarchical Bayesian models to grouped data, like documents or images. Topic modeling research touches on

- Directed graphical models
- Conjugate priors and nonconjugate priors
- Time series modeling
- Modeling with graphs
- Hierarchical Bayesian methods
- Fast approximate posterior inference (MCMC, variational methods)
- Exploratory data analysis
- Model selection and nonparametric Bayesian methods
- Mixed membership models

Latent Dirichlet allocation (LDA)

1. Introduction to LDA
2. The posterior distribution for LDA

Approximate posterior inference

1. Gibbs sampling
2. Variational inference
3. Comparison/Theory/Advice

Other topic models

1. Topic models for prediction: Relational and supervised topic models
2. The logistic normal: Dynamic and correlated topic models
3. "Infinite" topic models, i.e., the hierarchical Dirichlet process

Interpreting and evaluating topic models

Latent Dirichlet Allocation

Probabilistic modeling

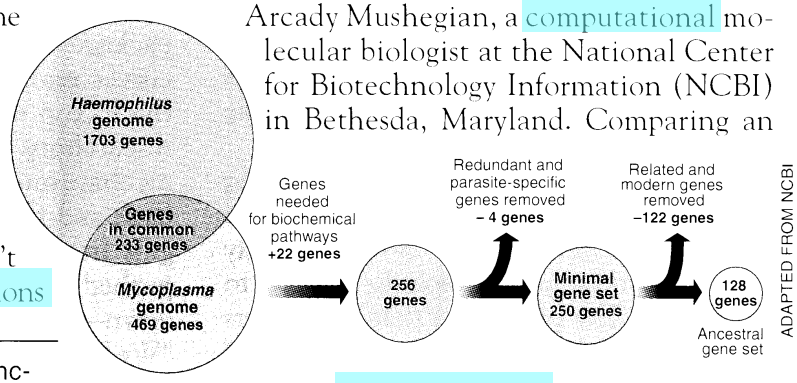
1. Treat data as observations that arise from a generative probabilistic process that includes hidden variables
   - For documents, the hidden variables reflect the thematic structure of the collection.
2. Infer the hidden structure using *posterior inference*
   - What are the topics that describe this collection?
3. Situate new data into the estimated model.
   - How does this query or new document fit into the estimated topic structure?

Intuition behind LDA

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an



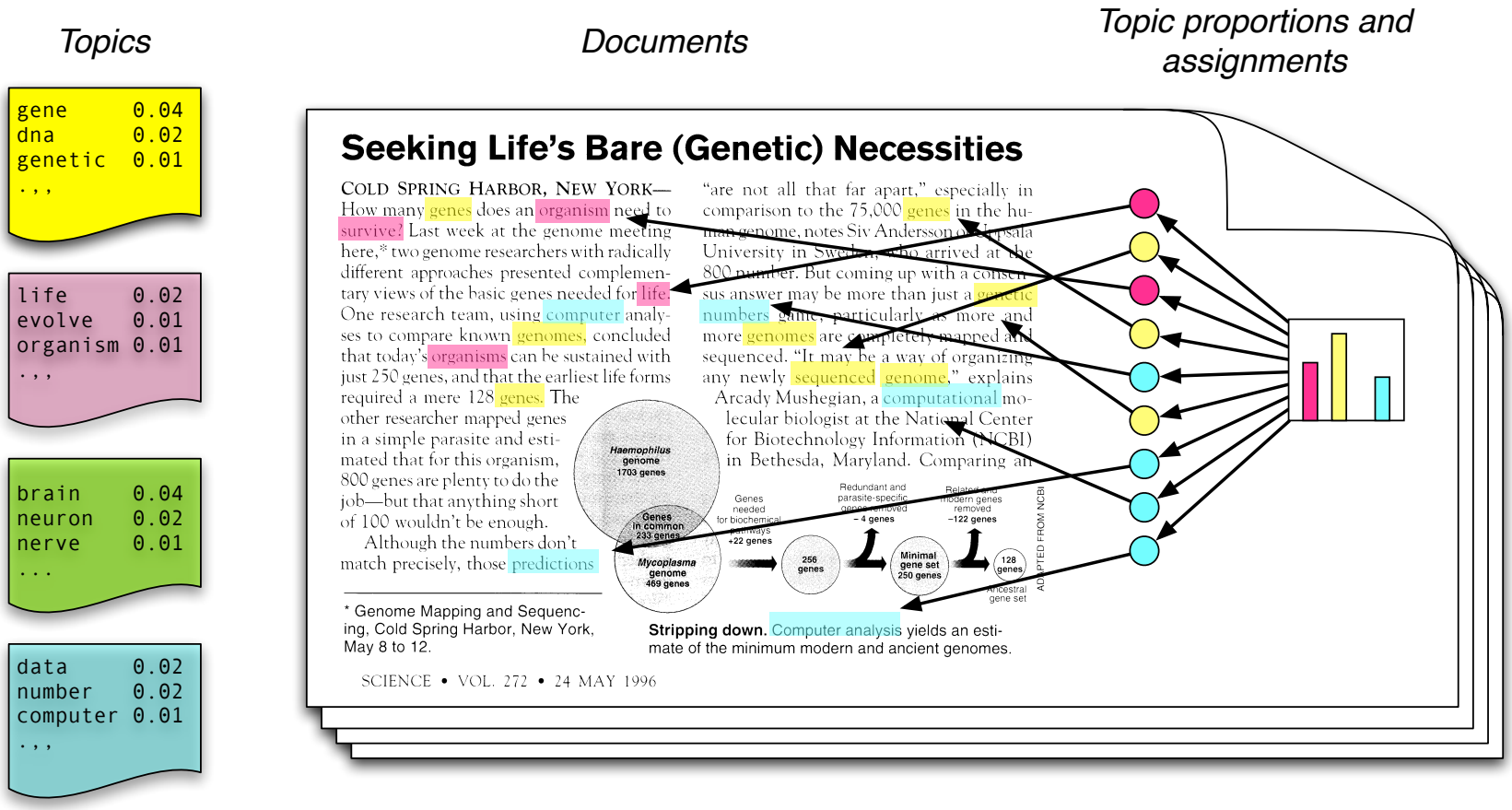
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996

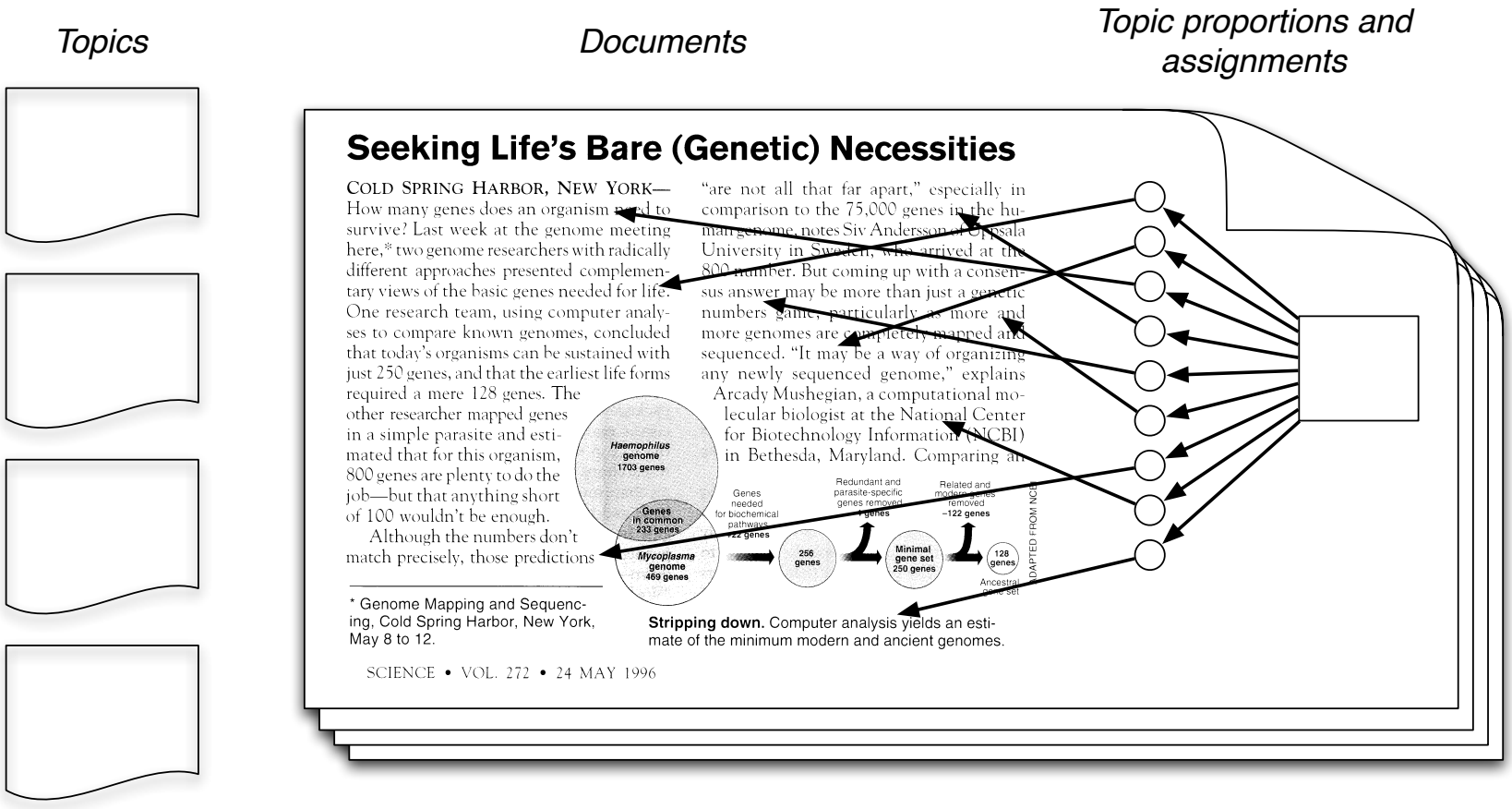
Simple intuition: Documents exhibit multiple topics.

Generative model



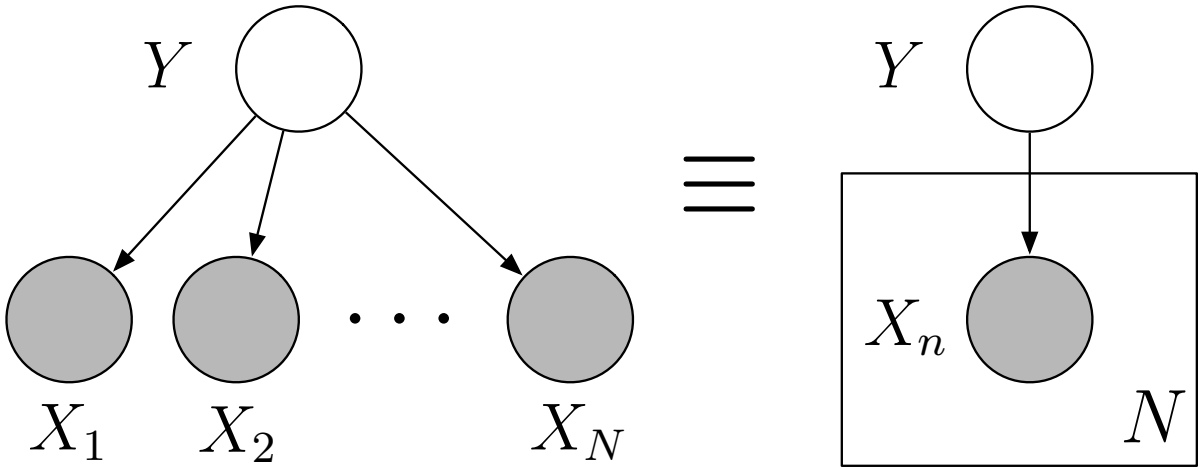
- Each document is a random mixture of corpus-wide topics
- Each word is drawn from one of those topics

The posterior distribution



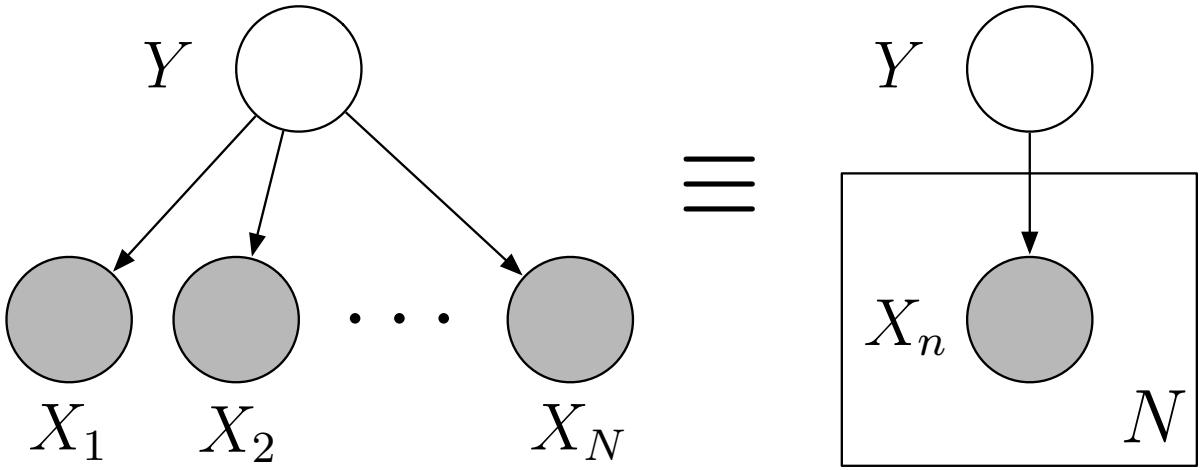
- In reality, we only observe the documents
- Our goal is to **infer** the underlying topic structure

Graphical models (Aside)



- Nodes are random variables
- Edges denote possible dependence
- Observed variables are shaded
- Plates denote replicated structure

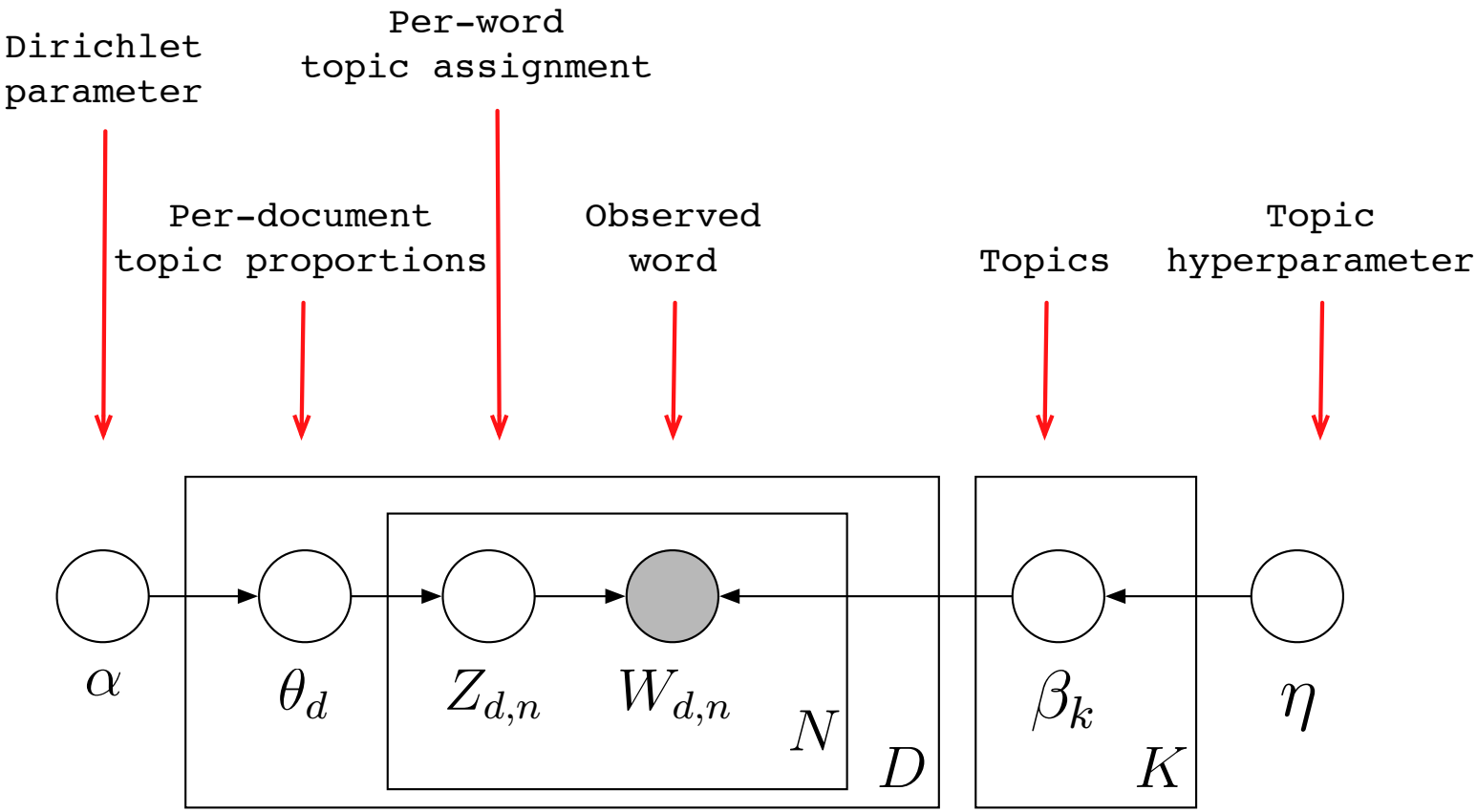
Graphical models (Aside)



- Structure of the graph defines the pattern of conditional dependence between the ensemble of random variables
- E.g., this graph corresponds to

$$p(y, x_1, \ldots, x_N) = p(y) \prod_{n=1}^{N} p(x_n \mid y)$$

Latent Dirichlet allocation



Each piece of the structure is a random variable.

Outline

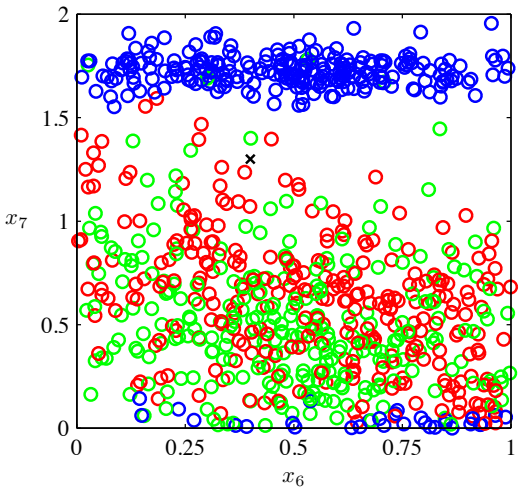
Dimensionality reduction

Motivation Given data that are high-dimensional $\boldsymbol{x} \in \mathbb{R}^D$, we want to find a low-dimensional representation $\boldsymbol{y} \in \mathbb{R}^M$ such that $M < D$:

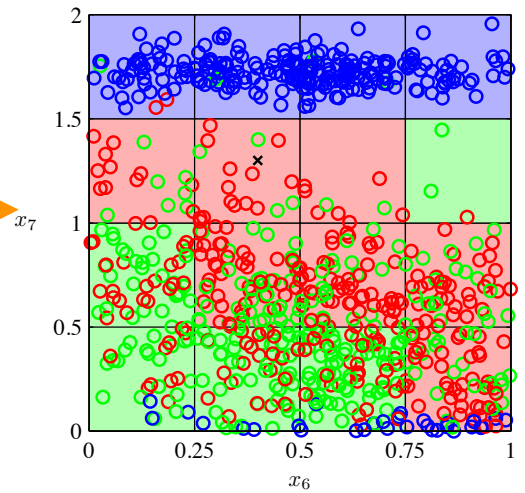
- Visualize data and discover intrinsic structures
- Save computational and storage cost
- Robust statistical modeling: curse of dimensionality

What is curse of dimensionality?

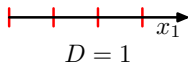
Ex: a simple classification scheme (related to decision tree)

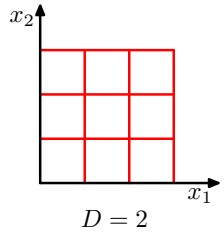


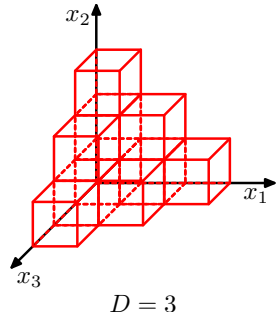
divide up
into small cells



of cells grows exponentially







# of cells r^D

r: number of divisions in each dimension

That is a lot even if D is just moderately large!

So to cover the whole space reasonably well,
you need exponentially number of training data points!

Another example

Nearest neighbor classification

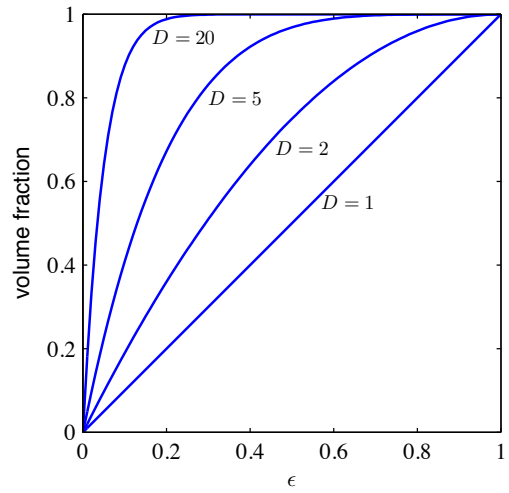
Say we want points of different classes are at least ε away from each other

To put things in scale (otherwise, you can scale ε arbitrarily), assume all points are at most 1 away from origin.

How many points between 1 and 1-ε from the origin?

$$1 - (1 - \epsilon)^D$$

Namely, when D is high, you cannot figure out data points that are ε away even ε is pretty big



Curse of dimensionality: summary

Intuition The higher the dimensionality, the more data points we need to train a model.

- To fill a unit-cube in $\mathbb{R}^D$ uniformly with data points, we need r^D where r is the edge length of small cells (i.e., dividing each axis in equal size of r.)

  Thus, if data is distributed that way, models such as decision trees need r^D training samples in order to make sure every cell is covered – in case a test sample falls into one of those cells.

- For a unit-ball $\boldsymbol{x} \leq 1$, a large percentage of data live in the shell — between the surface$\boldsymbol{x} = 1$ and the surface $\boldsymbol{x} = 1 - \epsilon$. The percentage is

$$1 - (1 - \epsilon)^D$$

  approaches 1. Thus, most data points in the high-dimensional space are crowded in the shell and are about the same distant from each other. To have data points that "look" substantially different from others, we need a exponentially large (in D) number of samples to fill the void $\boldsymbol{x} < 1 - \epsilon$.

An overview of dimensionality reduction

Parametric approaches

Linear: PCA, Fisher LDA, NMF, random projection, CCA, etc

Nonlinear: Neural networks, generative topological mapping, ICA

Nonparametric approaches

kernelized version of parametric methods: k-PCA, kICA, kCCA

manifold learning

Gaussian processes

Dimensionality reduction

- Linear: the low-dimensional coordinates $\boldsymbol{y}$ is parameterized as

$$\boldsymbol{y} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{x},$$

  where $\boldsymbol{U} \in \mathbb{R}^{M \times D}$
- Nonlinear: the relationship is through a nonlinear mapping

$$\boldsymbol{y} = f(\boldsymbol{x})$$

In both categories, there are many methods. We will focus on Principal Component Analysis (PCA) — a linear method for dimensionality reduction.

Linear dimensionality reductions

Many examples

Principal component analysis (PCA)

Fisher discriminant analysis (FDA)

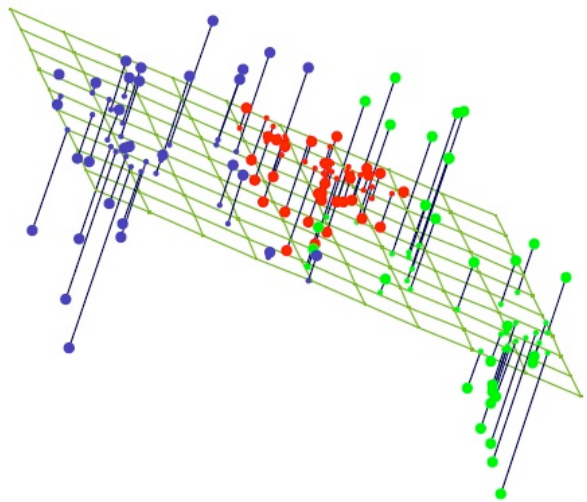
Nonnegative matrix factorization (NMF)

Framework

$$\boldsymbol{x} \in \Re^D \rightarrow \boldsymbol{y} \in \Re^M$$

$$D \gg M$$

$$\boldsymbol{y} = \boldsymbol{U}^\top \boldsymbol{x}$$



linear transformation of original space

Linear dimensionality reduction: PCA

Intuition Consider the special case $M = 1$, namely, we are transforming $\boldsymbol{x}$ into a scalar via

$$y = \boldsymbol{u}^{\mathrm{T}} \boldsymbol{x}$$

which $\boldsymbol{u}$ is sensible?

PCA: First principal component

Objective: Maximize variance of points projected onto a line.
Data: $\{x_i\}_{1\ldots N}$, all D-dimensional with mean $\overline{x}$.
Assume: All x shifted: $x_i \leftarrow x_i - \overline{x}$
Define:

- Covariance Matrix

$$S = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T \in \Re^{D\times D}$$

- Design matrix $\begin{pmatrix} x_1^T \\ x_2^T \\ \ldots \\ x_N^T \end{pmatrix}$

$$S = \frac{1}{N} X^T X$$

Derivation of PCA

We want to find a projection u_1 such that the projected variance is maximized.
Projection of x_i is $u_1^T x_i$
The mean of the projection is

$$\frac{1}{N}\sum_i (u_1^T x_i)^2 = \frac{1}{N}\sum_i u_1^T x_i x_i^T u_1$$
$$= u_1^T \frac{1}{N}\sum_i x_i x_i^T u_1 = u_1^T S u_1$$

Derivation of PCA

We want to $\max_{u_1} u_1^T S u_1$
This is not well defined

Derivation of PCA

We want to $\max_{u_1} u_1^T S u_1$
This is not well defined as

$$u_1^T S u_1 \geq 0 \text{ Since S is positive semi-definite}$$

So u_1 can grow without bound, as can $u_1^T S u_1$
So restrict u_1 such that $||u_1||_2^2 = 1$

Derivation of PCA

$$
\begin{aligned}
L &= u_1^T S u_1 + \lambda(1 - ||u_1||_2^2) \\
&= u_1^T S u_1 + \lambda(1 - u_1^T u_1) \\
\frac{\partial L}{\partial u_1} &= 2Su_1 - 2\lambda u_1 = 0
\end{aligned}
$$

So $Su_1 = \lambda u_1$.
And what does that make (λ, u_1)?

But...

But $S \in \Re^{D \times D}$ so it has D (eigenvalue, eigenvector) pairs.
Which to choose?
The objective function is

$$\max u_1^T S u_1 = u_1^T(\lambda u_1) = \lambda(u_1^T u_1) = \lambda$$

Other principal components

Find u_2 such that $u_2^T u_1 = 0$ and $u_2^T S u_2$ is maximized.
This results in (λ, u_2) being another (eigenvalue, eigenvector) pair.

The framework of PCA

Assumption:

Centered inputs (if not, subtract mean)

$$\boldsymbol{x} \in \mathbb{R}^D \quad \sum_i \boldsymbol{x}_i = \boldsymbol{0}$$

Projection into subspace

$$\boldsymbol{U} \in \mathbb{R}^{D \times M} \quad \boldsymbol{y}_i = \boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}_i$$

$$\boldsymbol{U}^{\mathrm{T}} \boldsymbol{U} = \boldsymbol{I}$$

Solution

each row is a data sample

Computer covariance matrix

$$\boldsymbol{S} = \frac{1}{N} \boldsymbol{X}^{\mathrm{T}} \boldsymbol{X}$$

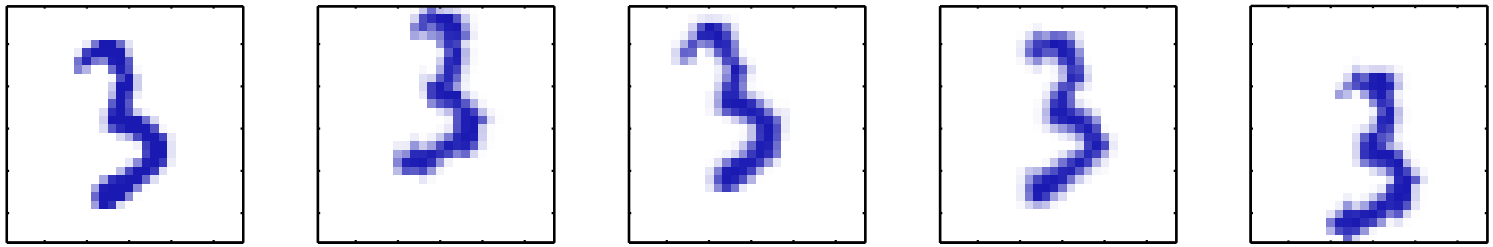
Diagonalize S: $(\boldsymbol{u}_d, \lambda_d)$ $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$

Use top D eigenvectors to form U

$$\boldsymbol{U} = (\boldsymbol{u}_1 \ \boldsymbol{u}_2 \ \cdots \ \boldsymbol{u}_M)$$

Examples of running PCA

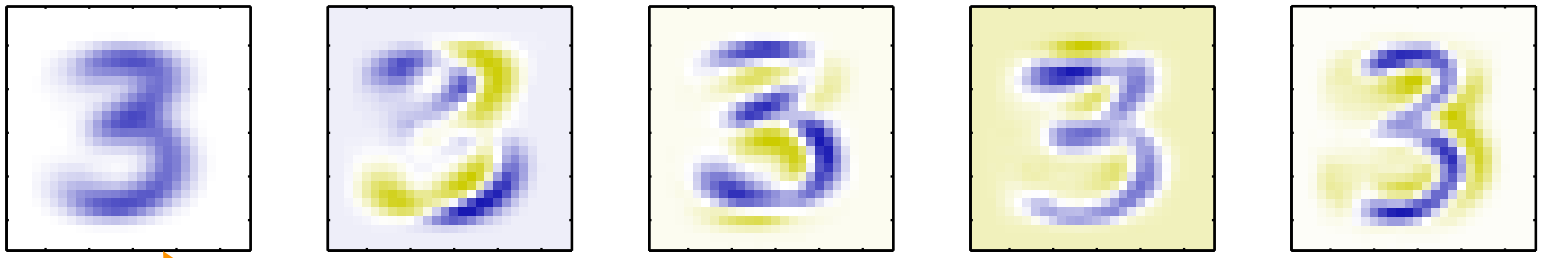
Original Images



Eigenvectors

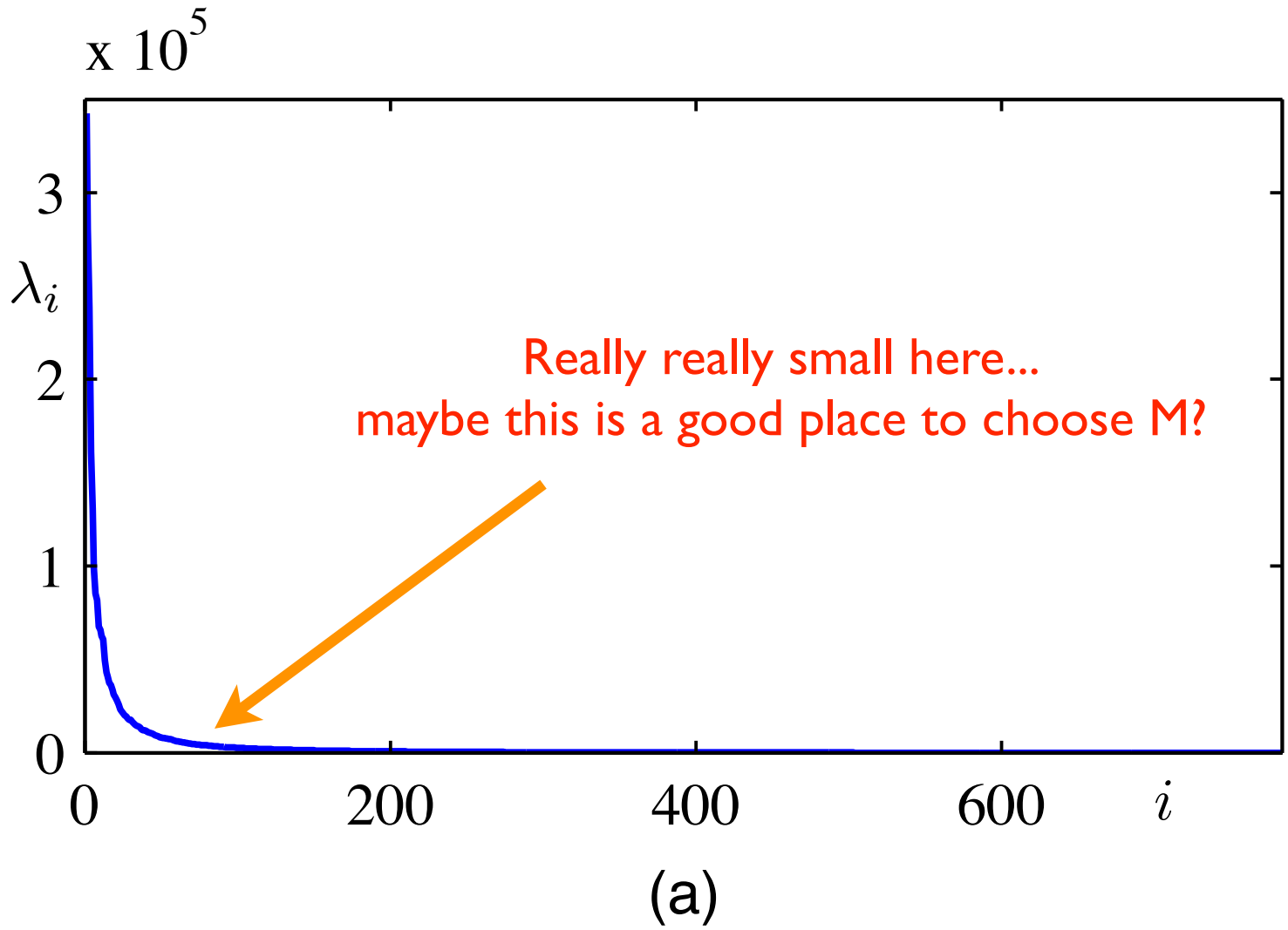
they look like blurred original images

Mean $\lambda_1 = 3.4 \cdot 10^5$ $\lambda_2 = 2.8 \cdot 10^5$ $\lambda_3 = 2.4 \cdot 10^5$ $\lambda_4 = 1.6 \cdot 10^5$



Used to centralize inputs

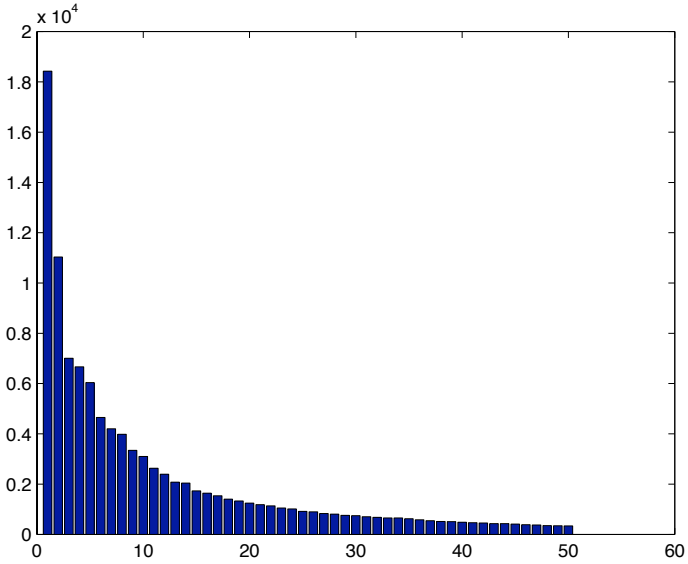
Eigenspectrum of the covariance matrix

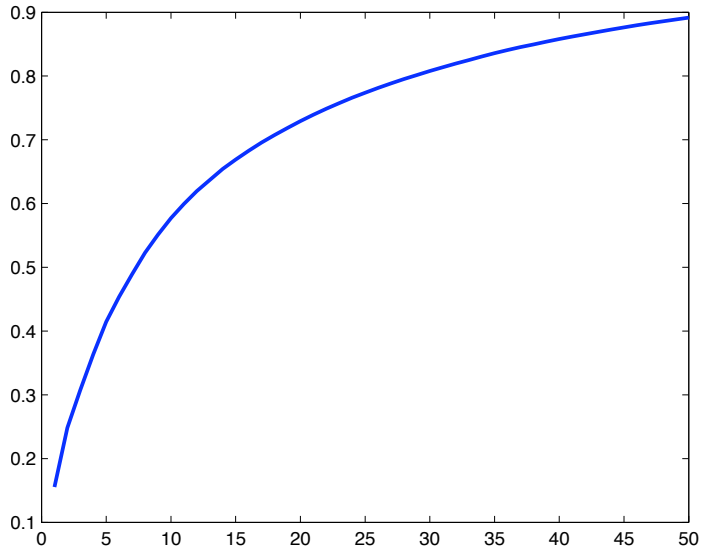


A slightly sensible approach

common choice is 95% or 90%

$$\frac{\sum_{d=1}^{M} \lambda_d}{\sum_{d=1}^{D} \lambda_d} \geq \text{Threshold}$$





Application of PCA

Preprocessing

Diagonalize data

$$\boldsymbol{y}_i = \boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}_i$$

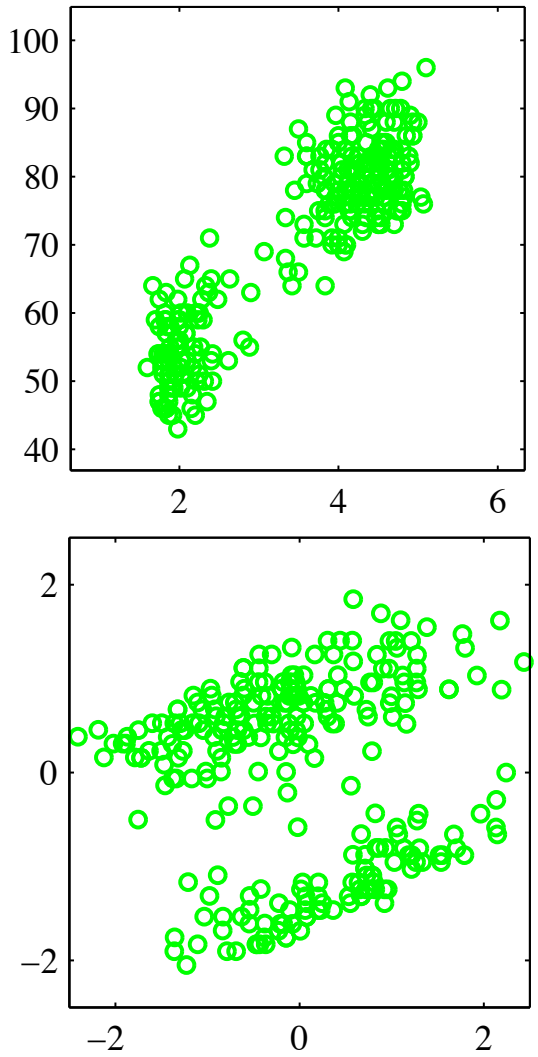
Normalize data (whitening)

$$\boldsymbol{y}_i = \boldsymbol{\lambda}^{-1/2} \boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}_i$$

Benefits:

1) depress noisy features

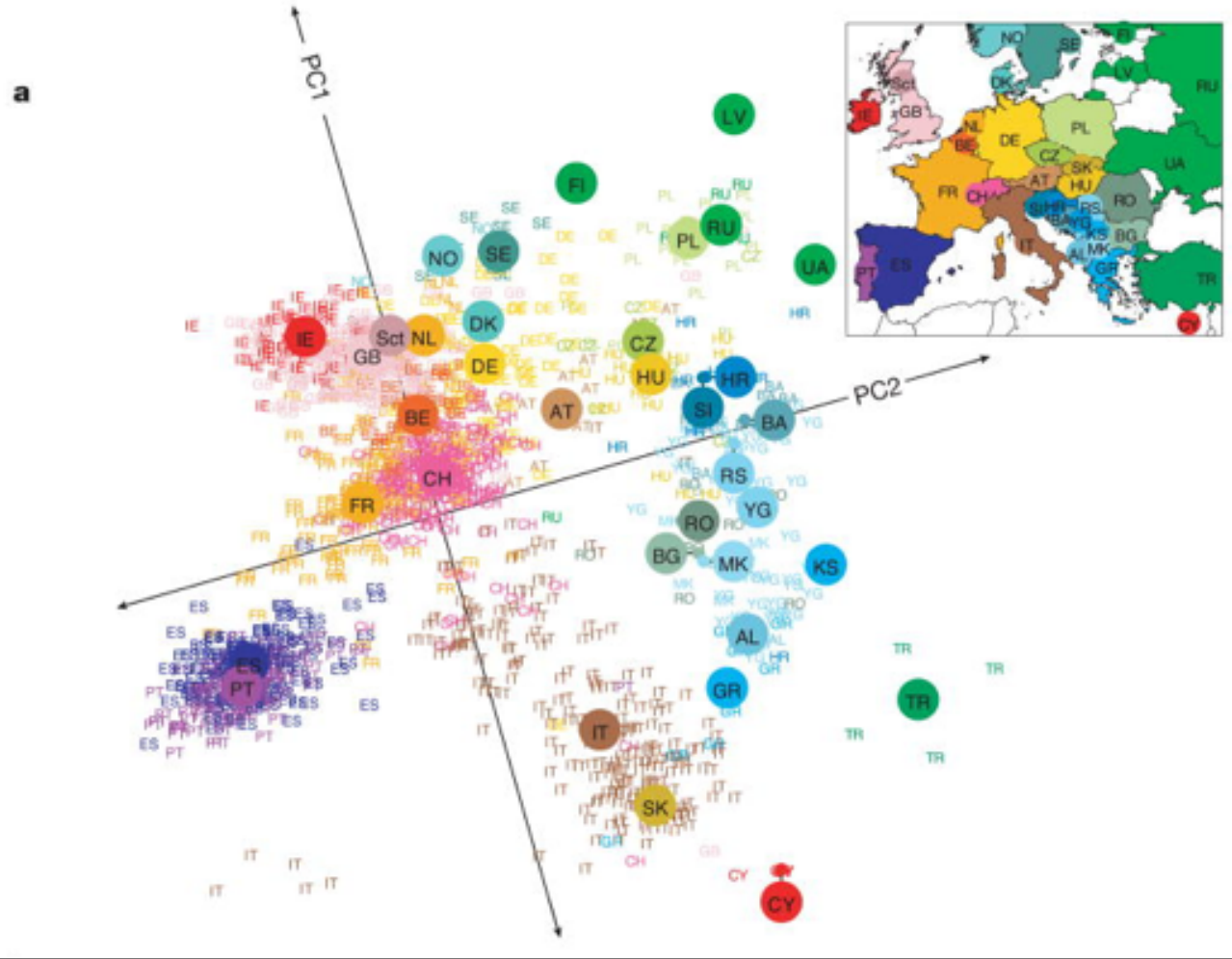
2) couple with other models



A very cool application

From genetic data to discover the origin of your ancestors:
`https://www.ncbi.nlm.nih.gov/pubmed/18758442`

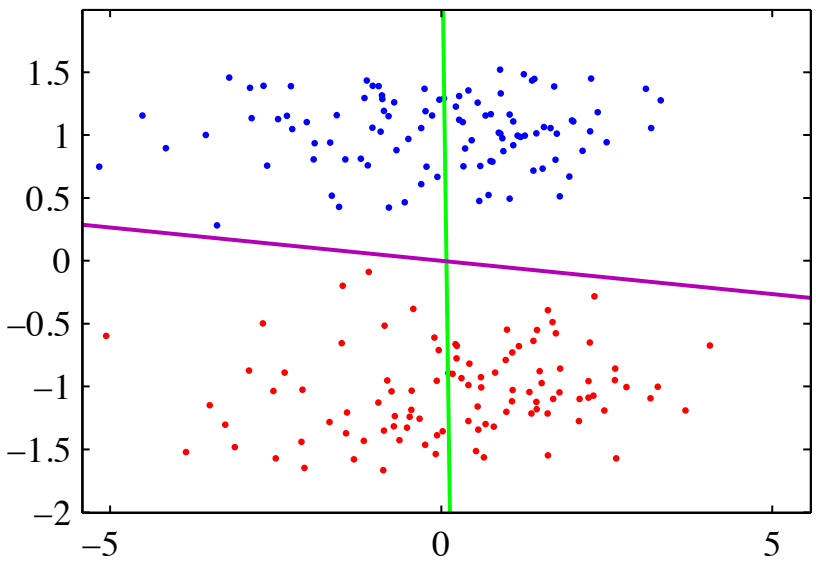
Visualizing data with PCA



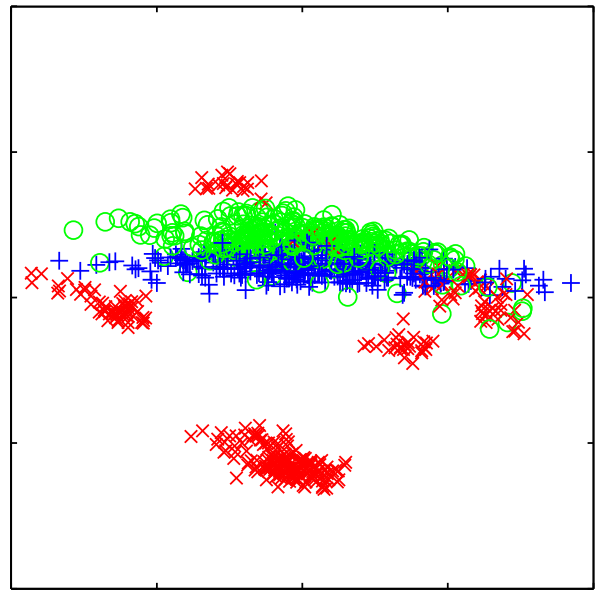
Does PCA help for classification?

Not always!

no help



help



Visualization

The second interpretation of PCA

minimum reconstruction error

Consider a basis of a subspace with M-dimension where M < D

$$\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_M \in \mathbb{R}^D$$

Consider the approximation error by linearly combining the bassis

$$\sum \|\boldsymbol{x}_n - \sum_m \alpha_{nm} \boldsymbol{u}_m\|_2^2$$

If we minimize the error by optimizing both the basis and the coefficients, we get

$$\alpha_{nm} = \boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{u}_m$$

$\boldsymbol{u}_m$ is the m-th largest eigenvector of S(ample) covariance matrix

Unexplained variance and residual error

Unexplained variance

Each projection direction u_i contributes λ_i variance

With D-dimensional data, M projection directions, the "leftover" is

$$\sum_{d=1}^{D} \lambda_d - \sum_{m=1}^{M} \lambda_m = \sum_{m=M+1}^{D} \lambda_m$$

Reconstruction error

$$\sum \| \boldsymbol{x}_n - \sum_m \alpha_{nm} \boldsymbol{u}_m \|_2^2$$

The two are exactly the same!!!

Issues with PCA

Choose dimensionality

ad hoc approach: we have seen that

more principled approach: bayesian PCA, etc

Missing values

what if data is missing?

Probabilistic PCA: a little bit later

How to get nonlinear dimensionality reduction

- Kernel PCA
- Autoencoder with nonlinear neural networks
- Manifold learning

We need nonlinear projection

Intuition:

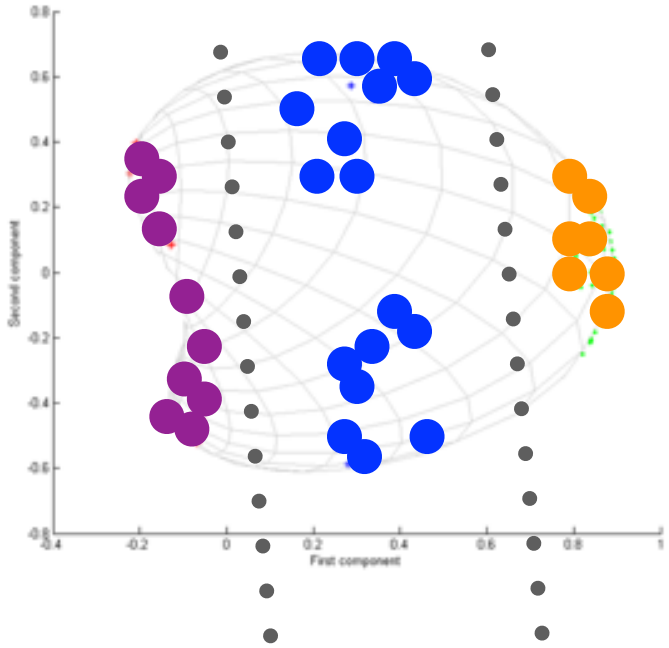
Find a good space through

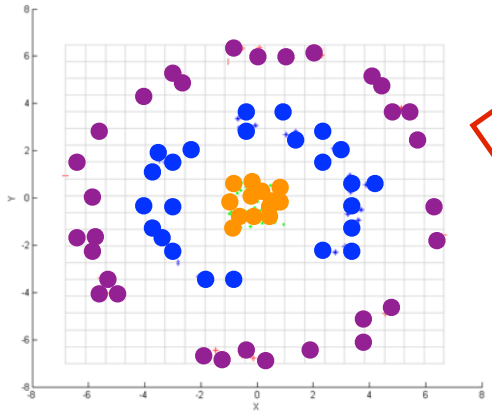
nonlinear mapping;

Then do linear projection

(as in PCA)

✓ Data separated!

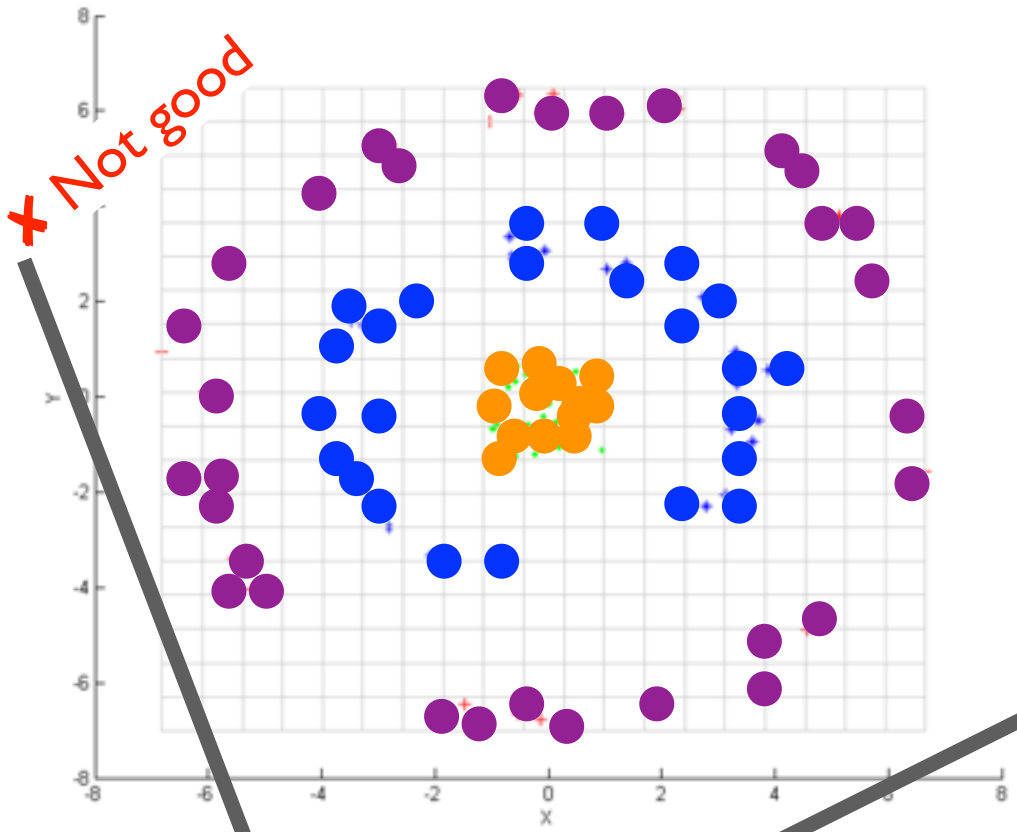




How to find this
nonlinear mapping?

it is called kernel PCA!

Why? ---- Linear is not enough



Issue:
projection on any 1D line will cause mixing of data from different classes.

✘ Not good

Kernel PCA

Consider standard PCA:

$$x \in \Re^{N \times D}$$

$$\frac{1}{N} x^T x u = \lambda u$$

$$\Rightarrow u = \frac{1}{\lambda N} x^T x u = x^T (\frac{1}{\lambda N} x u) = x^T \alpha$$

What is α?

$$\Rightarrow \frac{1}{N} x^T x x^T \alpha = x x^T \alpha$$
$$\Rightarrow \frac{1}{N} x \cdot x^T x x^T \alpha = x x x^T \alpha$$
$$\Rightarrow \frac{1}{N} (x \cdot x^T)(x \cdot x^T)\alpha = \lambda (x x^T)\alpha$$

And what do we do from here?

What is α?

$$\Rightarrow \frac{1}{N} x^T x x^T \alpha = x x^T \alpha$$
$$\Rightarrow \frac{1}{N} x \cdot x^T x x^T \alpha = x x x^T \alpha$$
$$\Rightarrow \frac{1}{N} (x \cdot x^T)(x \cdot x^T) \alpha = \lambda (x x^T) \alpha$$

And what do we do from here?
We replace xx^T with any kernel matrix

$$\Rightarrow \frac{1}{N} k \cdot k\alpha = \lambda k \alpha$$
$$\Rightarrow \frac{1}{N} k\alpha = x\alpha$$
$$\Rightarrow k\alpha = \lambda N \alpha$$

So α is what now?

What is u then?

$$u = x^T \alpha = x^T \alpha$$

- x is still the original features
- α : use the kernel matrix.

What are the projection coordinates?

What is u then?

$u = x^T\alpha = x^T\alpha$

- x is still the original features
- α : use the kernel matrix.

What are the projection coordinates?

$$z = x^T u = x \cdot x^T\alpha$$

This is

$$[k(x_1, x), k(x_2, x) \ldots k(x_N, x)]\alpha$$

So we don't have to figure out what u is to compute z.

Two subtle issues

1. In PCA, we assumed x is centered. Can we assume this in Kernel-PCA?
2. We still need to ensure $||u||_2^2 = 1$.