CS6510
Applied Machine Learning

Other Topics:
EM, Time Series Analysis, and HMMs

19 Nov 2016

Vineeth N Balasubramanian

आई आई टी हैदराबाद
IIT Hyderabad

Today

- EM Algorithm and Its Uses in ML
- Time Series Analysis and Hidden Markov Models
- Hackathon/Challenge winners' presentations

Course Evaluation Rubric

- 15%: Quizzes
- 25%: Homework (Theory + Programming)
- 30%: Competitive Coding Assignments (Challenge style)
- 30%: End-semester examination

- 5 grace days to start with – please use them wisely. May not apply to some deadlines, which will be pointed out.
- Best 80% (approx, tentative) of quizzes will be considered
- Not attempting the end-semester exam will result in an FR.

EM Algorithm

Adapted from:

Christopher Morse

Department of Computer Science University of Vermont

Spring 2013

Acknolwedgements

- This presentation is based on the paper:
  - Dempster, A.P. Laird, N.M. Rubin, D.B. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". Journal of the Royal Statistical Society. Series B (Methodological) 39 (1): 1–38. JSTOR 2984875.MR0501537.
- Sections 2, 3, and 6 come from professor Andrew W. Moore's "Clustering with Gaussian Mixtures"
- Please bear with typos – otherwise, well-explained!

Contents

Introduction

- The EM algorithm was explained and given its name in a classic 1977 paper by Arthur Dempster, Nan Laird, and Donald Rubin.
- They pointed out that the method had been "proposed many times in special circumstances" by earlier authors.
- EM is typically used to compute maximum likelihood estimates given incomplete samples.
- The EM algorithm estimates the parameters of a model iteratively.
  - Starting from some initial guess, each iteration consists of
    - an E step (Expectation step)
    - an M step (Maximization step)

Applications

- Filling in missing data in samples
- Discovering the value of latent variables
- Estimating the parameters of HMMs
- Estimating parameters of finite mixtures
- Unsupervised learning of clusters
- Semi-supervised classification and clustering.

Contents

EM Algorithm

Silly Example

Silly Example

Let events be "grades in a class"

w_1 = Gets an A P(A) = ½

w_2 = Gets a B P(B) = μ

w_3 = Gets a C P(C) = 2μ

w_4 = Gets a D P(D) = ½-3μ

(Note $0 \leq \mu \leq 1/6$)

Assume we want to estimate μ from data. In a given class there were

a A's
b B's
c C's
d D's

What's the maximum likelihood estimate of μ given a,b,c,d ?

Trivial Statistics

P(A) = ½ P(B) = μ P(C) = 2μ P(D) = ½-3μ

P(*a,b,c,d* | μ) = $(½)^a(μ)^b(2μ)^c(½-3μ)^d$

log P(*a,b,c,d* | μ) = *a*log ½ + *b*log μ + *c*log 2μ + *d*log (½-3μ)

FOR MAX LIKE μ, SET $\frac{\partial \mathrm{LogP}}{\partial \mu} = 0$

$$\frac{\partial \mathrm{LogP}}{\partial \mu} = \frac{b}{\mu} + \frac{2c}{2\mu} - \frac{3d}{1/2 - 3\mu} = 0$$

$$\text{Gives max like } \mu = \frac{b+c}{6(b+c+d)}$$

So if class got

A	B	C	D
14	6	9	10

$$\text{Max like } \mu = \frac{1}{10}$$

Boring, but true!

Contents

Same Problem with Hidden Information

Someone tells us that

Number of High grades (A's + B's) = h

Number of C's = c

Number of D's = d

What is the max. like estimate of μ now?

REMEMBER

P(A) = ½

P(B) = μ

P(C) = 2μ

P(D) = ½-3μ

Same Problem with Hidden Information

Someone tells us that
Number of High grades (A's + B's) = h
Number of C's = c
Number of D's = d

REMEMBER
P(A) = ½
P(B) = μ
P(C) = 2μ
P(D) = ½-3μ

What is the max. like estimate of μ now?

We can answer this question circularly:

EXPECTATION

If we know the value of μ we could compute the expected value of a and b

Since the ratio a:b should be the same as the ratio ½ : μ

$$a = \frac{1/2}{1/2 + \mu} h \qquad b = \frac{\mu}{1/2 + \mu} h$$

MAXIMIZATION

If we know the expected values of a and b we could compute the maximum likelihood value of μ

$$\mu = \frac{b + c}{6(b + c + d)}$$

E.M. for our Trivial Problem

REMEMBER
P(A) = ½
P(B) = μ
P(C) = 2μ
P(D) = ½-3μ

We begin with a guess for μ
We iterate between EXPECTATION and MAXIMALIZATION to improve our estimates of μ and *a* and *b*.

Define μ(t) the estimate of μ on the t'th iteration
b(t) the estimate of *b* on t'th iteration

$$\mu(0) = \text{initial guess}$$

$$b(t) = \frac{\mu(t)h}{1/2 + \mu(t)} = \mathrm{E}[b \mid \mu(t)]$$

E-step

$$\mu(t+1) = \frac{b(t)+c}{6(b(t)+c+d)}$$

$$= \max \text{ like est of } \mu \text{ given } b(t)$$

M-step

Continue iterating until converged.
Good news: Converging to local optimum is assured.
Bad news: I said "local" optimum.

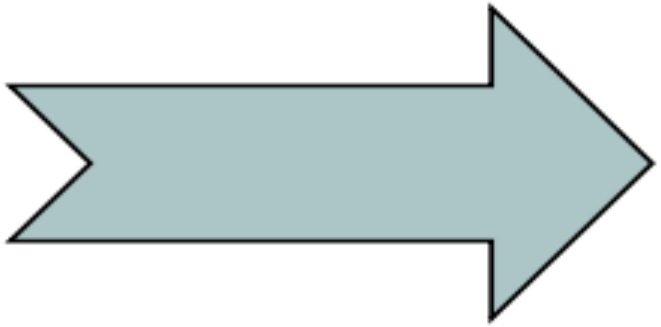
E.M. Convergence

- Convergence proof based on fact that Prob(data | μ) must increase or remain same between each iteration [NOT OBVIOUS]
- But it can never exceed 1 [OBVIOUS]

So it must therefore converge [OBVIOUS]

In our example, suppose we had

h = 20
c = 10
d = 10
μ(0) = 0



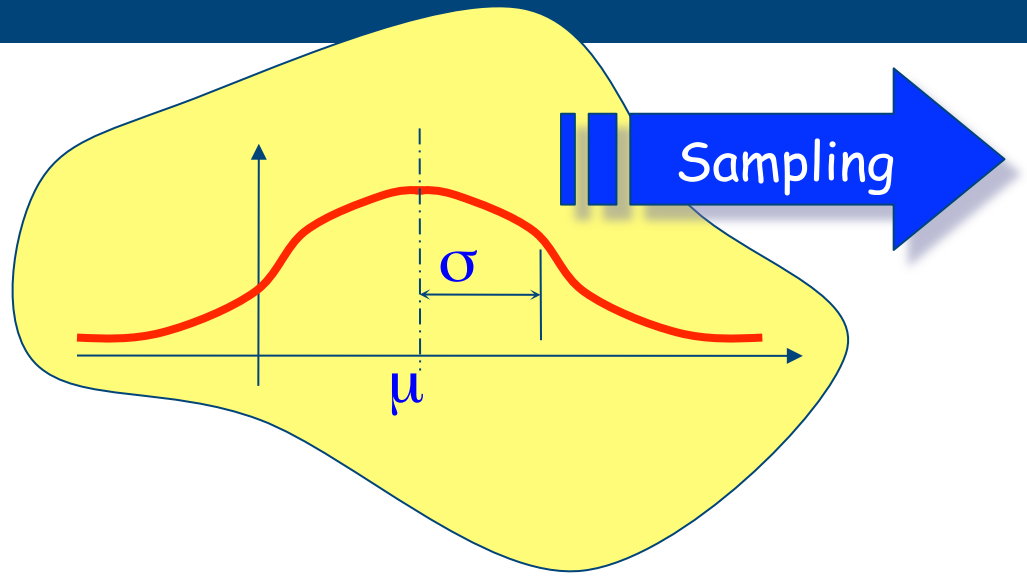
Convergence is generally linear: error decreases by a constant factor each time step.

t	μ(t)	b(t)
0	0	0
1	0.0833	2.857
2	0.0937	3.158
3	0.0947	3.185
4	0.0948	3.187
5	0.0948	3.187
6	0.0948	3.187

Contents

$$X \sim N(\mu, \sigma^2)$$

Normal Sample



$$\mathbf{x} = (x_1, x_2, \mathrm{K}\ , x_n)^T$$

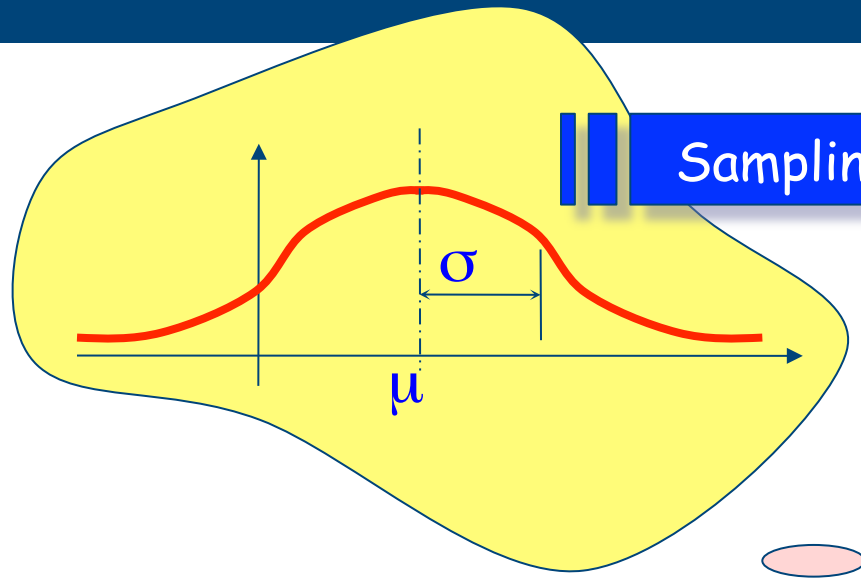
$$\hat{\mu} = ?$$

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)}{2\sigma^2}\right]$$

$$\hat{\sigma}^2 = ?$$

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)}{2\sigma^2}\right]$$

Maximum Likelihood



Sampling

$$\mathbf{x} = (x_1, x_2, \mathrm{K}\ , x_n)^T$$

We want to maximize it.

$$L(\mu, \sigma^2 \mid \mathbf{x}) = f(\mathbf{x} \mid \mu, \sigma^2) = f(x_1 \mid \mu, \sigma^2) \mathrm{L}\ f(x_n \mid \mu, \sigma^2)$$

Given x, it is a function of μ and σ^2

$$= \left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \exp\left[-\sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2}\right]$$

$$L(\mu,\sigma^2 \mid \mathbf{x}) = \left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \exp\left[-\sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right]$$

Log-Likelihood Function

$$l(\mu,\sigma^2 \mid \mathbf{x}) = \log L(\mu,\sigma^2 \mid \mathbf{x})$$

Maximize this instead

$$= \frac{n}{2}\log\frac{1}{2\pi\sigma^2} - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}$$

$$= -\frac{n}{2}\log\sigma^2 - \frac{n}{2}\log 2\pi - \frac{1}{2\sigma^2}\sum_{i=1}^{n}x_i^2 + \frac{\mu}{\sigma^2}\sum_{i=1}^{n}x_i - \frac{n\mu^2}{2\sigma^2}$$

By setting

$$\frac{\partial}{\partial\mu}l(\mu,\sigma^2 \mid \mathbf{x}) = 0 \quad \text{and} \quad \frac{\partial}{\partial\sigma^2}l(\mu,\sigma^2 \mid \mathbf{x}) = 0$$

Max. the Log-Likelihood Function

$$l(\mu,\sigma^2 \mid \mathbf{x}) = -\frac{n}{2}\log\sigma^2 - \frac{n}{2}\log 2\pi - \frac{1}{2\sigma^2}\sum_{i=1}^{n} x_i^2 + \frac{\mu}{\sigma^2}\sum_{i=1}^{n} x_i - \frac{n\mu^2}{2\sigma^2}$$

$$\frac{\partial}{\partial\mu} l(\mu,\sigma^2 \mid \mathbf{x}) = \frac{1}{\sigma^2}\sum_{i=1}^{n} x_i - \frac{n\mu}{\sigma^2} = 0$$

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \hat{\mu}^2$$

Max. the Log-Likelihood Function

$$l(\mu, \sigma^2 \mid \mathbf{x}) = -\frac{n}{2}\log\sigma^2 - \frac{n}{2}\log 2\pi - \frac{1}{2\sigma^2}\sum_{i=1}^{n} x_i^2 + \frac{\mu}{\sigma^2}\sum_{i=1}^{n} x_i - \frac{n\mu^2}{2\sigma^2}$$

$$\frac{\partial}{\partial\sigma^2} l(\mu, \sigma^2 \mid \mathbf{x}) = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{i=1}^{n} x_i^2 - \frac{\mu}{\sigma^4}\sum_{i=1}^{n} x_i + \frac{n\mu^2}{2\sigma^4} = 0$$

$$n\sigma^2 = \sum_{i=1}^{n} x_i^2 - 2\mu\sum_{i=1}^{n} x_i + n\mu^2$$

$$= \sum_{i=1}^{n} x_i^2 - \frac{2}{n}\left(\sum_{i=1}^{n} x_i\right)^2 + \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2$$

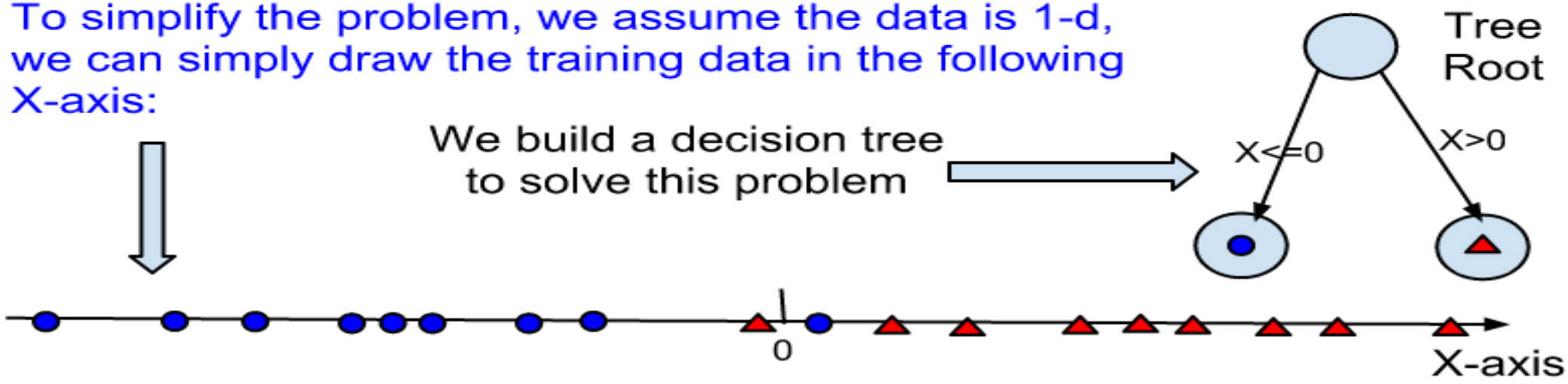
Contents

Begin with Classification

Given a training data set: X={x(1),x(2),…,x(n)}
Z={z(1),z(2),…,z(n)}
z(i) is the calss label of sample x(i).
As we are in classfication setting,
both X and Z are Given.

To simplify the problem, we assume the data is 1-d, we can simply draw the training data in the following X-axis:

We build a decision tree to solve this problem



Solve the problem using another method– parametric method

Now, we model the data by specifying a joint distribution p(x(i), z(i))=p(x(i)|z(i))p(z(i))

$$z^{(i)} \sim \text{Multinomial}(\phi)$$
$$\phi_j \geq 0, \sum_{j=1}^{k} \phi_j = 1$$
$$k = \#\ \text{of}\ z^{(i)}\text{'s values}$$
$$\phi_j = p(z^{(i)} = j)$$
$$x^{(i)} | z^{(i)} = j \sim \mathcal{N}(\mu_j, \Sigma_j)$$

each $x^{(i)}$ was generated by randomly choosing $z^{(i)}$ from $\{1, \ldots, k\}$, and then $x^{(i)}$ was drawn from one of k Gaussians .

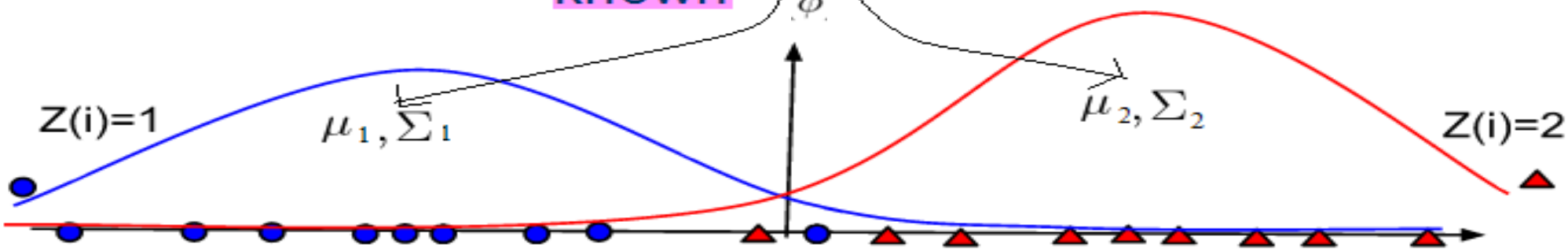
As we are in classfication setting, both X and Z are Given.

known

MLE

unknown

The parameters of our model are ϕ, μ and Σ.



Use our model for classification

Using our model for calssfication

Testing sample x(i) is given

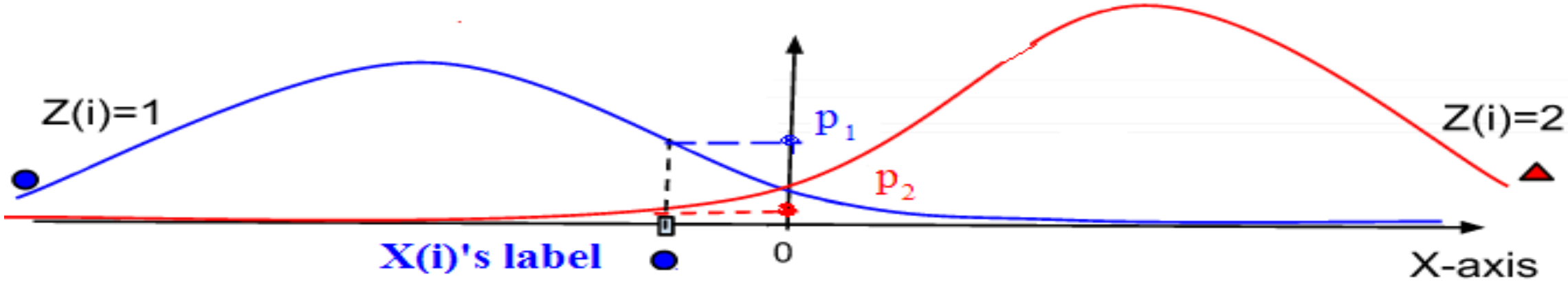
model (μ and Σ) are known by MLE for the training data

calssfication ⟹ z(i) is unknown.

$p(x(i), z(i)) = p(x(i)|z(i))p(z(i))$

Solved

$P(x(i),z(i)=1) = p_1$ > $P(x(i),z(i)=2) = p_2$ ⟹ z(i)=1



EM Clustering Algorithm

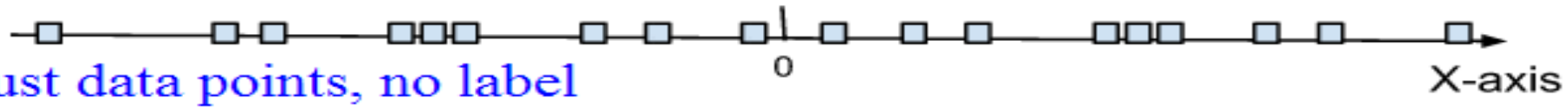
Given a training data set: X={x(1),x(2),...,x(n)}
Z={z(1),z(2),...,z(n)}
z(i) is the calss/group label of sample x(i).
As we are in Clustering setting,
X is Given and Z is unknown

Now, we model the data by specifying a joint distribution p(x(i), z(i))=p(x(i)|z(i))p(z(i))

$$z^{(i)} \sim \text{Multinomial}(\phi)$$
$$\phi_j \geq 0, \sum_{j=1}^{k} \phi_j = 1$$
$$k = \#\text{ of } z^{(i)}\text{'s values}$$
$$\phi_j = p(z^{(i)} = j)$$
$$x^{(i)}|z^{(i)} = j \sim \mathcal{N}(\mu_j, \Sigma_j)$$

⇨ each $x^{(i)}$ was generated by randomly choosing $z^{(i)}$ from $\{1, \ldots, k\}$, and then $x^{(i)}$ was drawn from one of k Gaussians .

The parameters of our model are thus ϕ, μ and Σ.



E-M

Incomplete Data

X={x(1),x(2),…,x(n)} Given
Z={z(1),z(2),…,z(n)} unknown

unknown

The parameters of our model ϕ, μ, Σ

What is the value of z(i)?

We can answer this question circularly:

EXPECTATION

If we know the expected values of Z we could compute the maximum likelihood value of ϕ, μ, Σ

MAXIMIZATION

If we know the values of ϕ, μ, Σ we could compute the expected values of Z

We begin with a guess for ϕ, μ, Σ, and then iterate between EXPECTATION and MAXIMALIZATION to improve our estimates of ϕ, μ, Σ and Z

Continue iterating until converged.

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{m} \log p(x^{(i)}|z^{(i)}; \mu, \Sigma) + \log p(z^{(i)}; \phi)$$

Maximizing this with respect to ϕ, μ and Σ gives the parameters:

$$\begin{aligned}
\phi_j &= \frac{1}{m} \sum_{i=1}^{m} 1\{z^{(i)} = j\}, \\
\mu_j &= \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}, \\
\Sigma_j &= \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}.
\end{aligned}$$

Repeat until convergence: {

(E-step) For each i, j, set

$$w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

(M-step) Update the parameters:

$$\begin{aligned}
\phi_j &:= \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)}, \\
\mu_j &:= \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}}, \\
\Sigma_j &:= \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}
\end{aligned}$$

}

Comparison to K-means

Given a training data set: X={x(1),x(2),…,x(n)}
Z={z(1),z(2),…,z(n)}
z(i) is the calss/group label of sample x(i).
As we are in Clustering setting,
X is Given and Z is unknown

Model of EM

EM model the data by specifying a joint distribution p(x(i), z(i))=p(x(i)|z(i))p(z(i))

$$z^{(i)} \sim \text{Multinomial}(\phi)$$
$$\phi_j \geq 0,\ \sum_{i=1}^{k} \phi_j = 1$$
$$k = \#\ \text{of}\ z^{(i)}\text{'s values}$$
$$\phi_j = p(z^{(i)} = j)$$
$$x^{(i)} | z^{(i)} = j \sim \mathcal{N}(\mu_j, \Sigma_j)$$

⇨ each $x^{(i)}$ was generated by randomly choosing $z^{(i)}$ from $\{1, \ldots, k\}$, and then $x^{(i)}$ was drawn from one of k Gaussians.

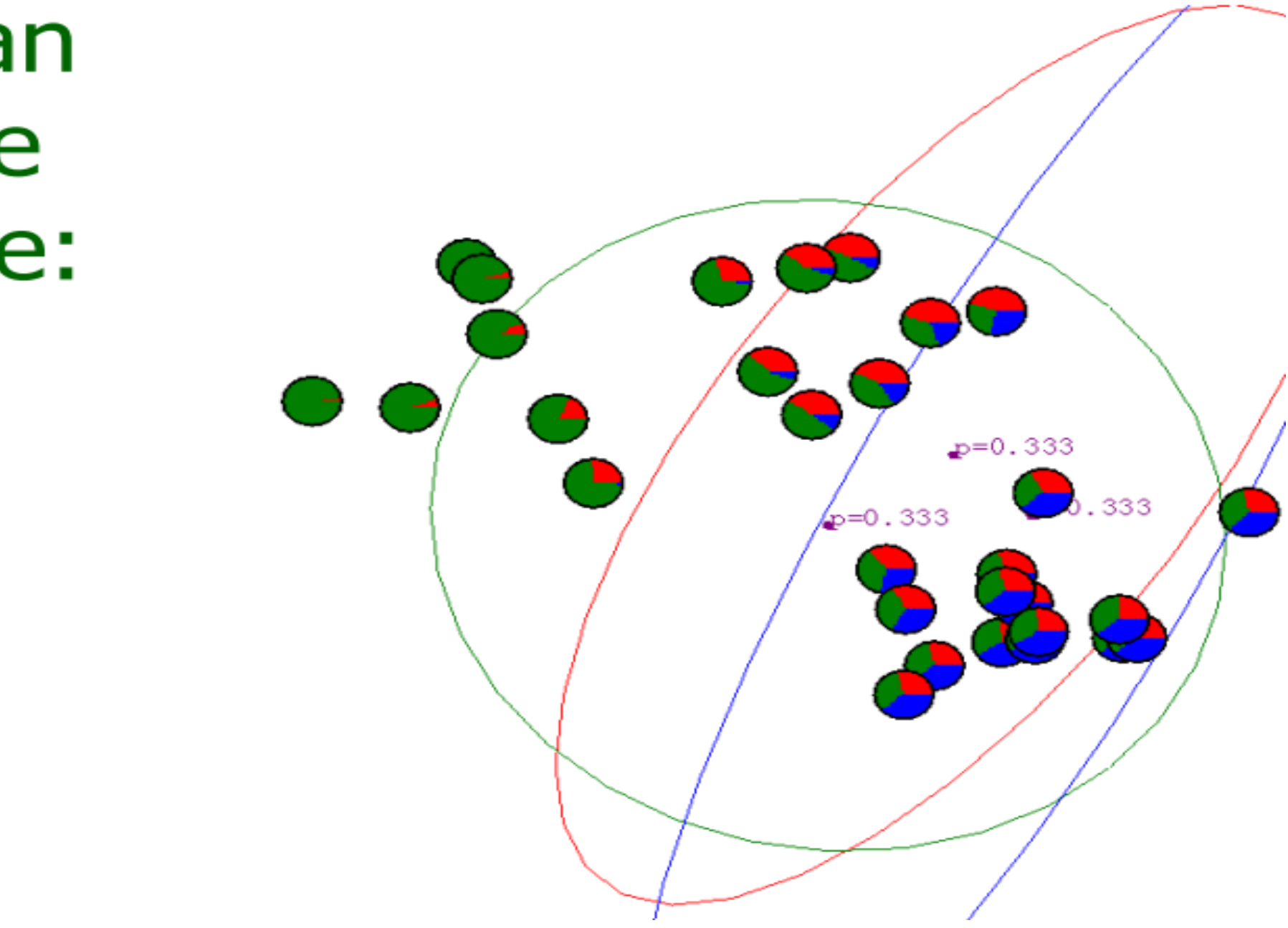
K-mans is a simplified EM, it assumes that

$\phi_j = \phi_i = 1/k$, and $\Sigma_j = \Sigma_i$ for i, j=1,2,...k
k is given by user

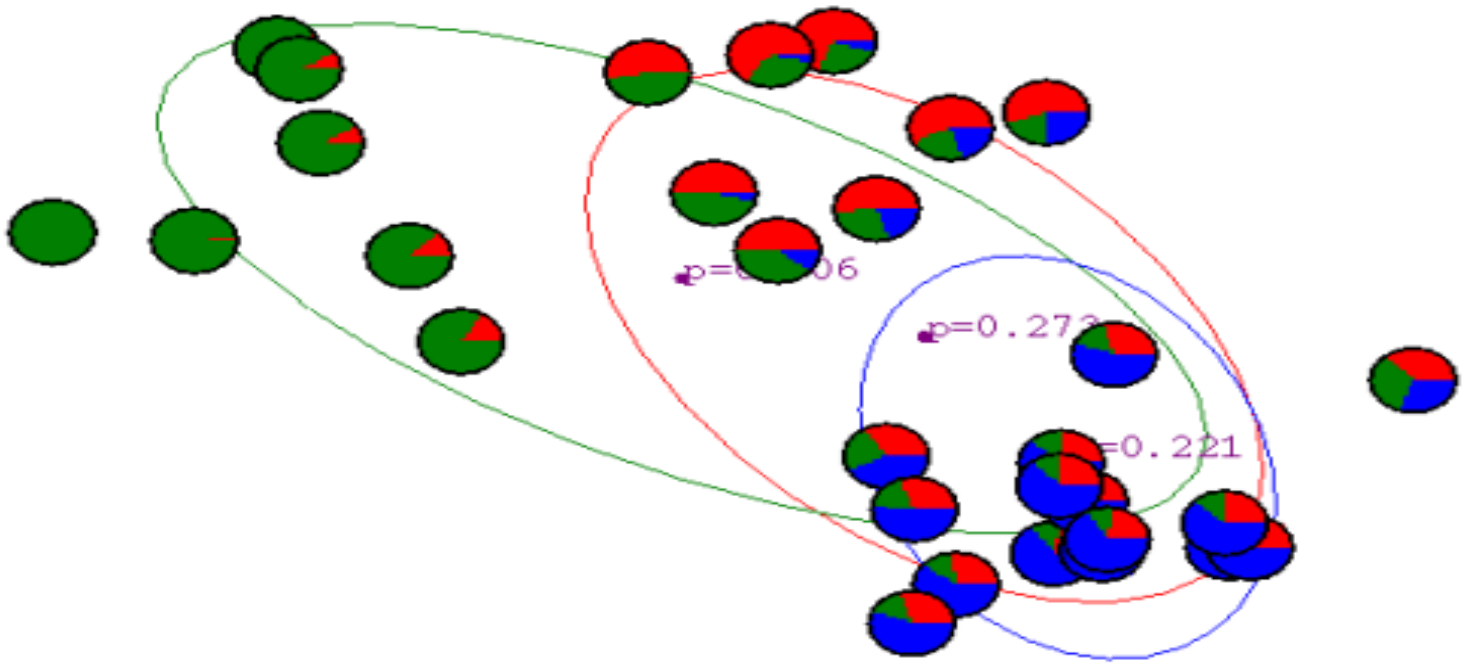
⇨ μ_1, μ_2 ..... μ_k are the only unknown parameters of the model (the means of clusters)

Contents

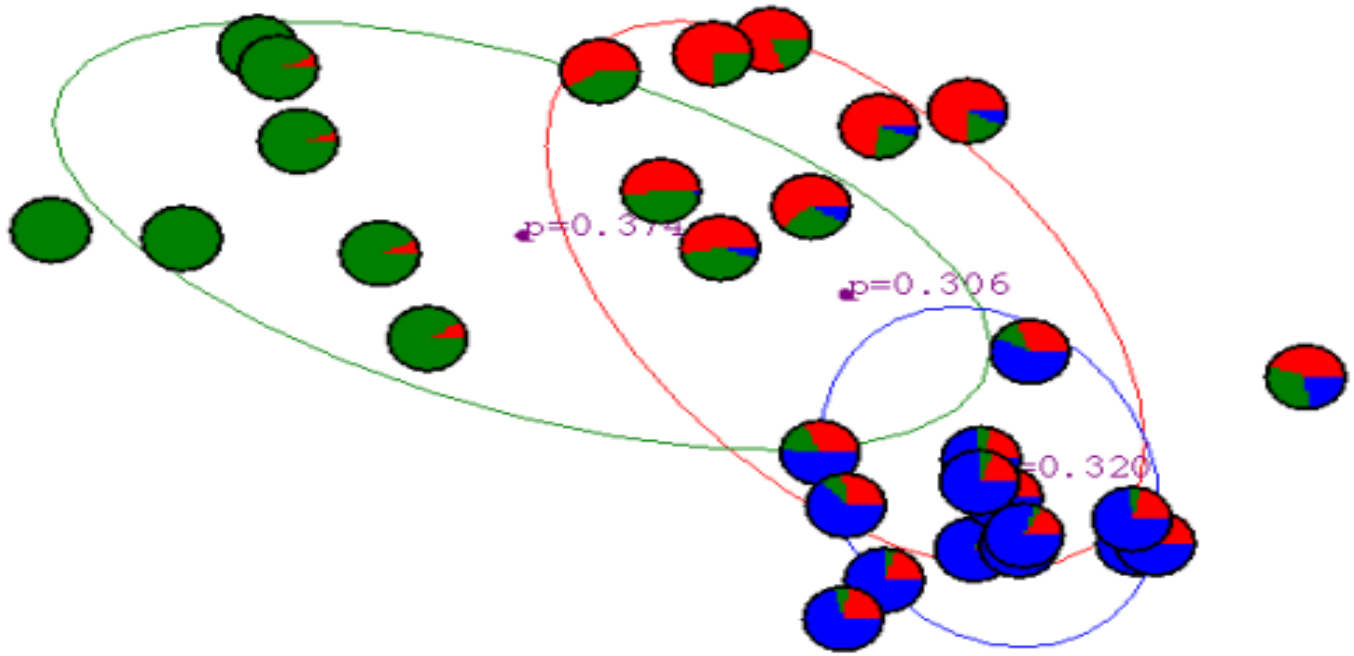
Gaussian Mixture Example: Start



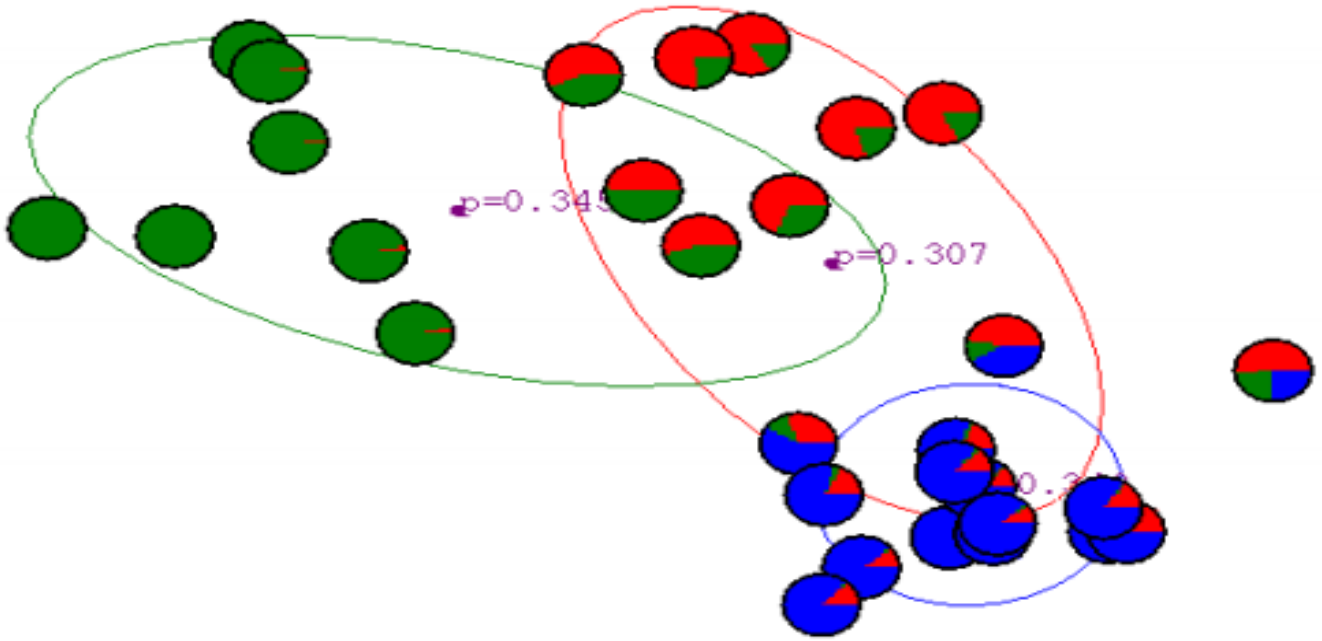
After first iteration



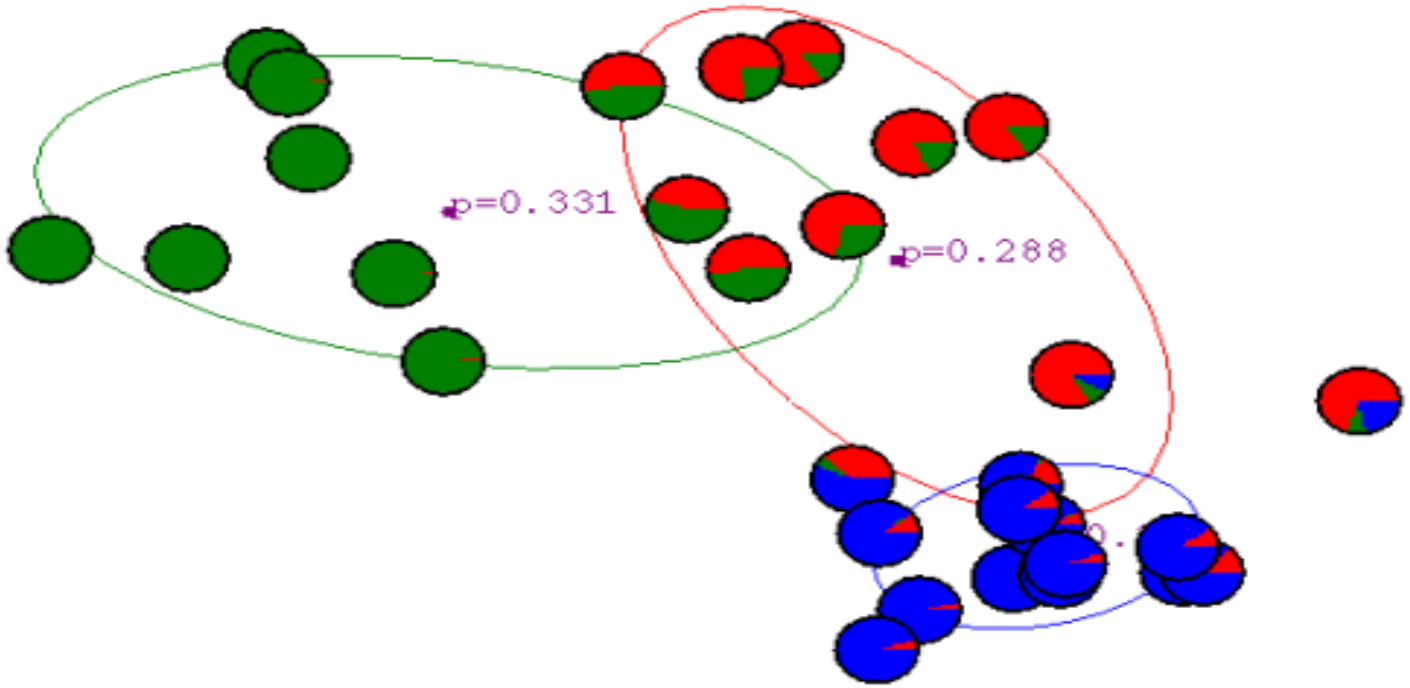
After 2nd iteration



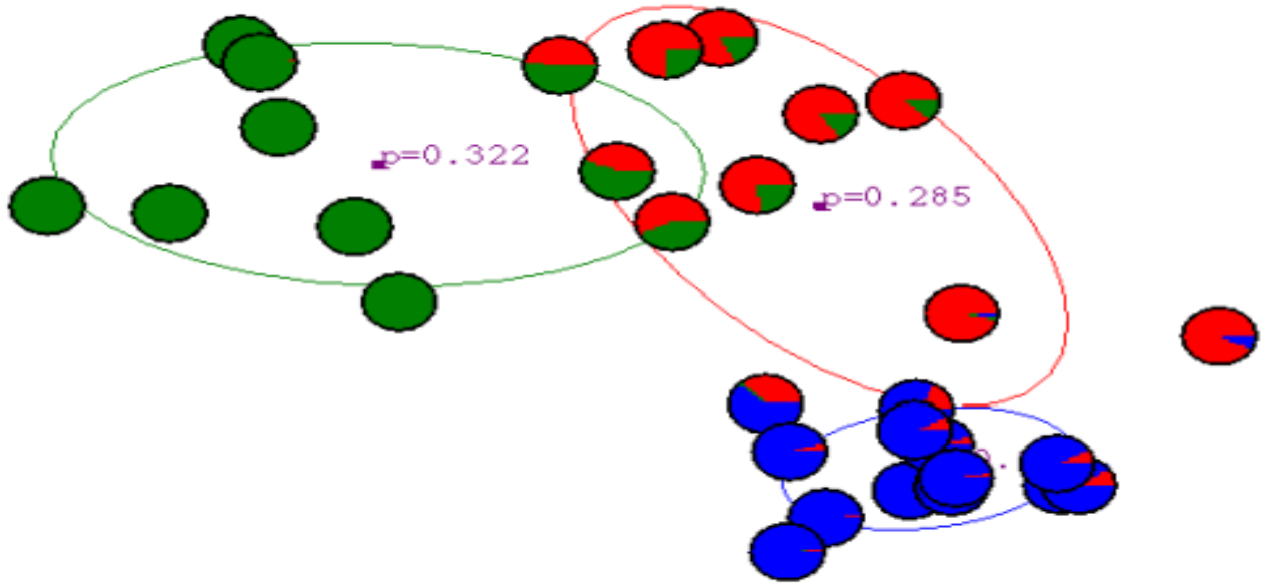
After 3rd iteration



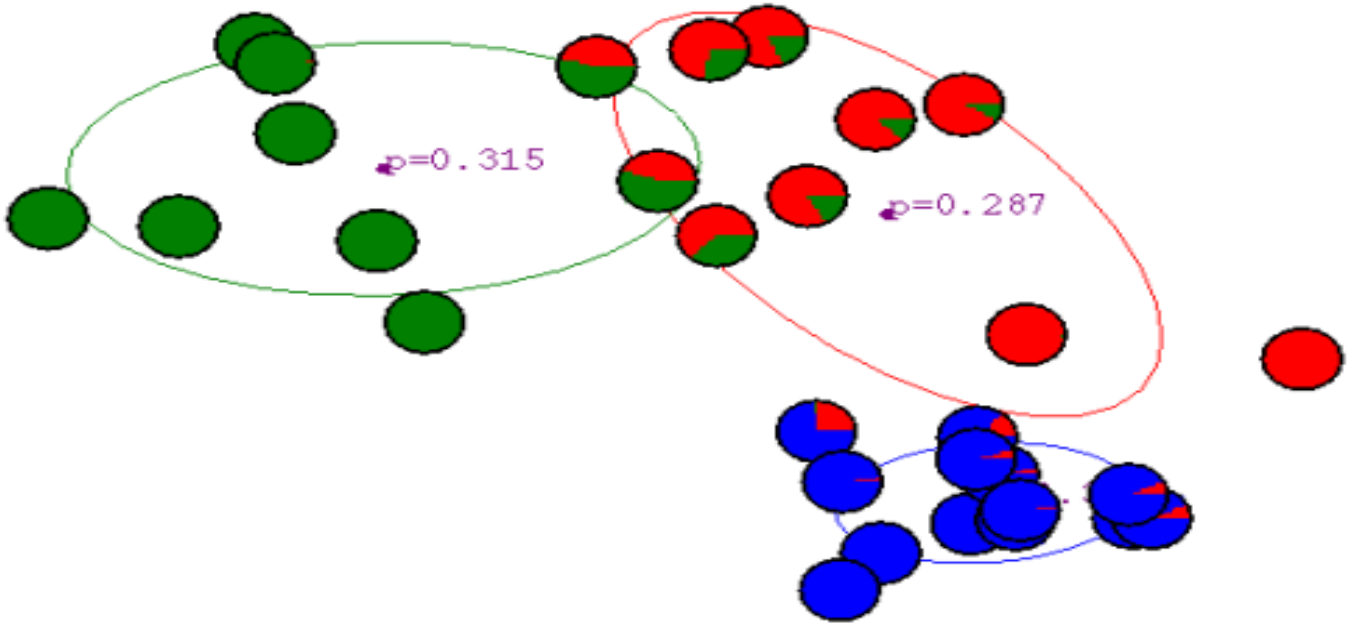
After 4th iteration



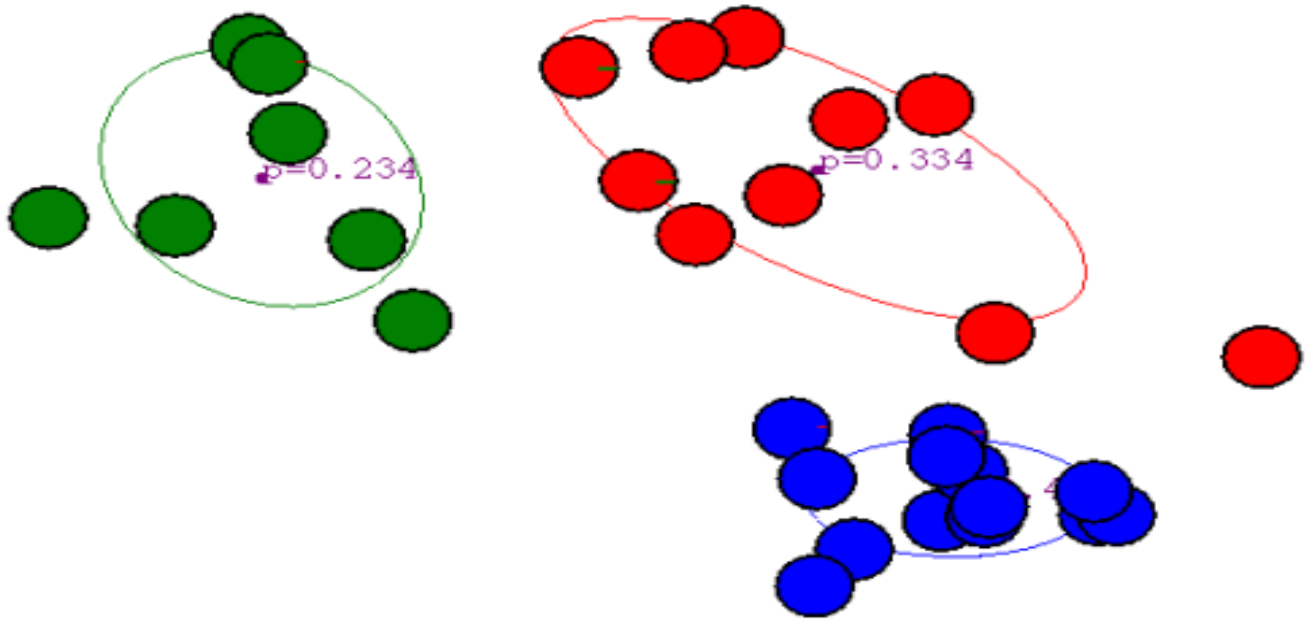
After 5th iteration



After 6th iteration



After 20th iteration



More on EM Algorithm

- What are the EM algorithm initialization methods?
  - Random guess.
  - Any general classifier that builds a parameterized probability distribution model (i.e. naive Bayes).
  - Initialized by k-means. After a few iterations of k-means, using the parameters to initialize EM
- What are the main advantages of parametric methods?
  - You can easily change the model to adapt to different distribution of data sets.
  - Knowledge representation is very compact. Once the model is selected, the model is represented by a specific number of parameters.
  - The number of parameters does not increase with the increasing of training data .

EM Algorithm: Uses in ML

- Clustering and related Unsupervised Learning applications
- Semi-supervised Learning
  - Using EM to improve a classifier by augmenting labeled training data with unlabeled data
  - E.g. Medical Image Reconstruction
- Many more…

Today

- EM Algorithm and Its Uses in ML
- Time Series Analysis and Hidden Markov Models
- Hackathon/Challenge winners' presentations

Acknowledgements

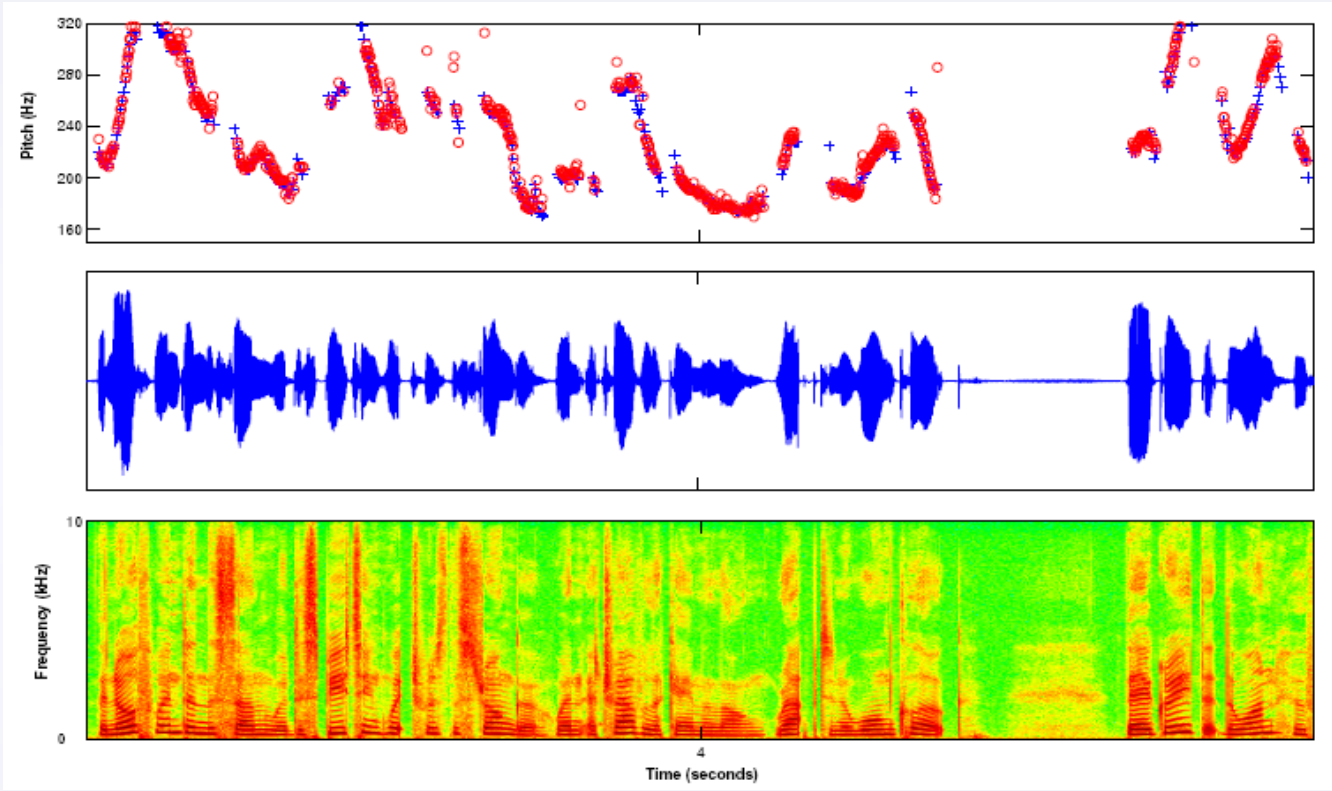
- The following slides on HMMs adapted from Erik Suddherth, Brown University

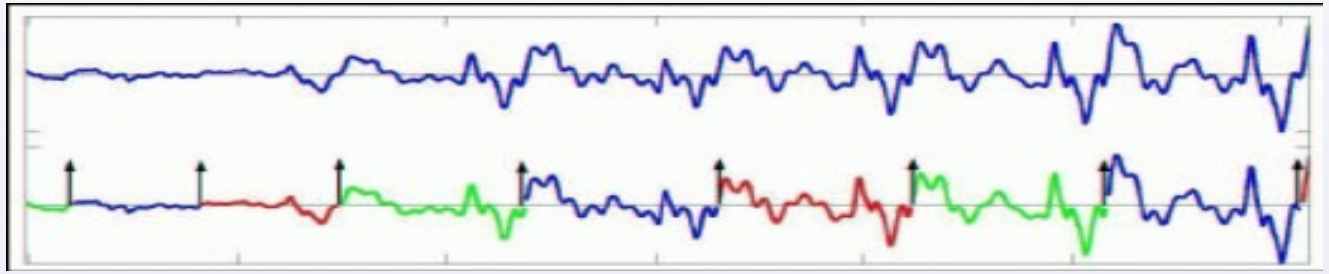
Time Series Analysis

- Time series: a collection of observations made sequentially in time
- Many application fields:
  - Economic time series
  - Speech data
  - Marketing time series
  - Process control
- Characteristics:
  - successive observations are NOT independent (i.i.d. assumption does not hold)
  - The order of observation is crucial

Speech Recognition

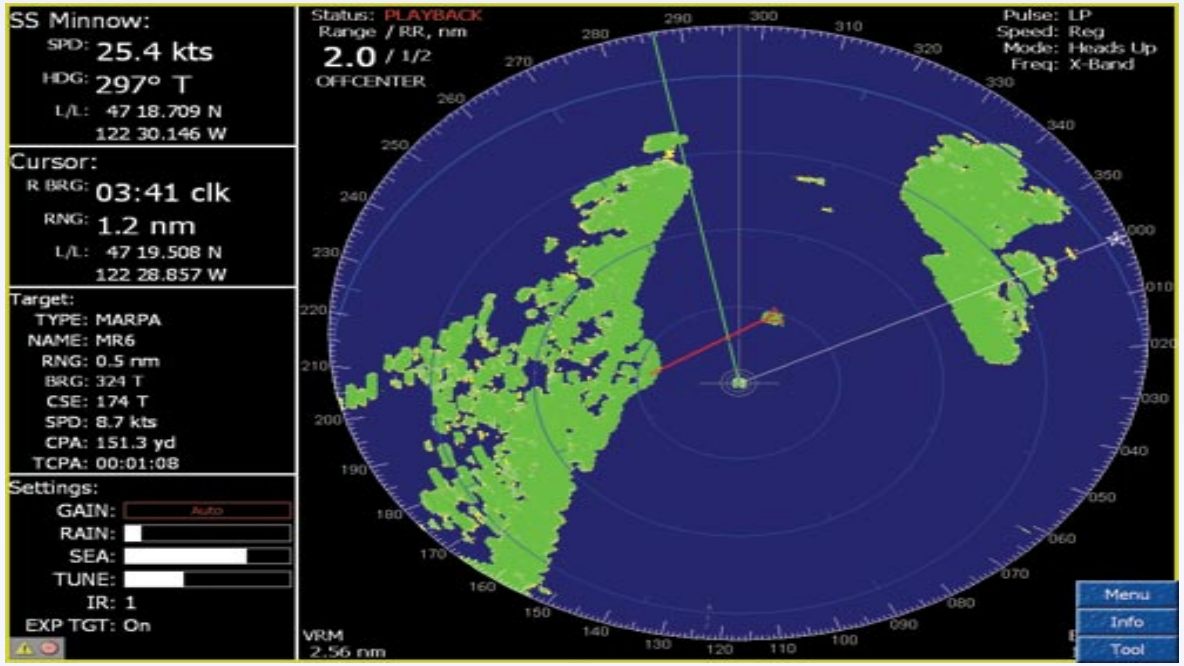
- Given an audio waveform, would like to robustly extract & recognize any spoken words
- Statistical models can be used to
  - ➢ Provide greater robustness to noise
  - ➢ Adapt to accent of different speakers
  - ➢ Learn from training



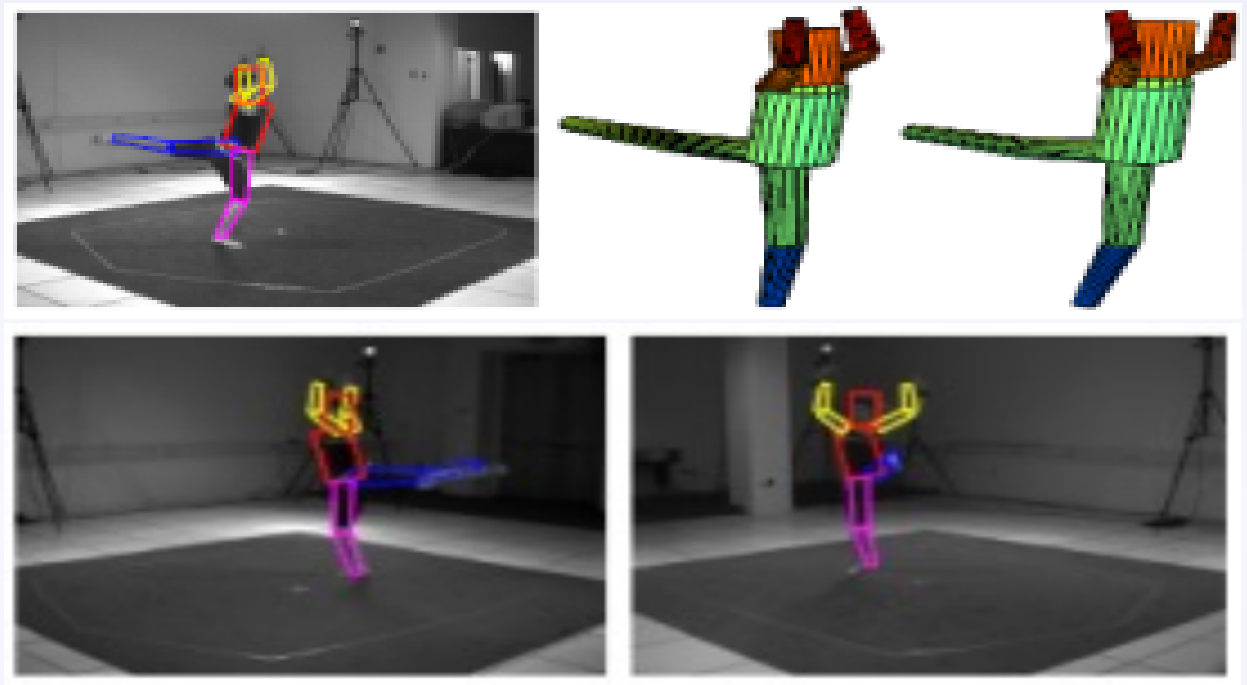


S. Roweis, 2004

Target Tracking



Radar-based tracking of multiple targets



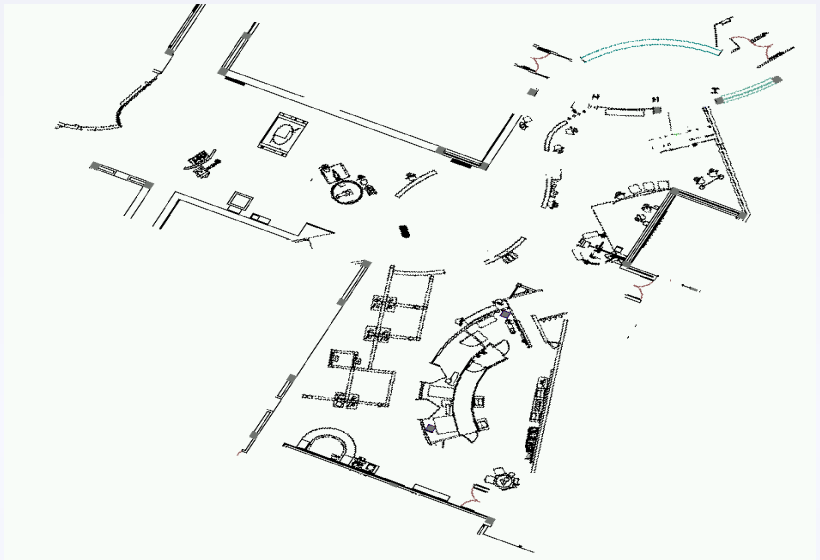
Visual tracking of articulated objects

(L. Sigal et. al., 2006)

- Estimate motion of targets in 3D world from indirect, potentially noisy measurements

Robot Navigation: *SLAM*

Simultaneous Localization and Mapping



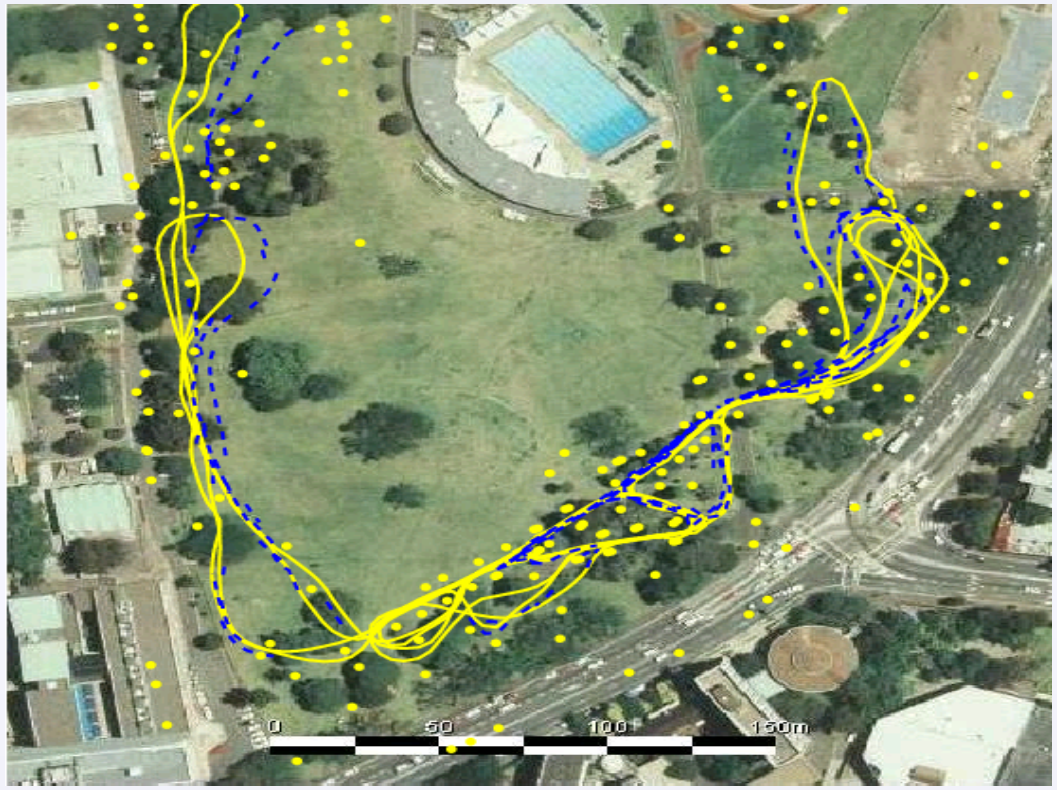
CAD Map

(S. Thrun, San Jose Tech Museum)

Estimated Map

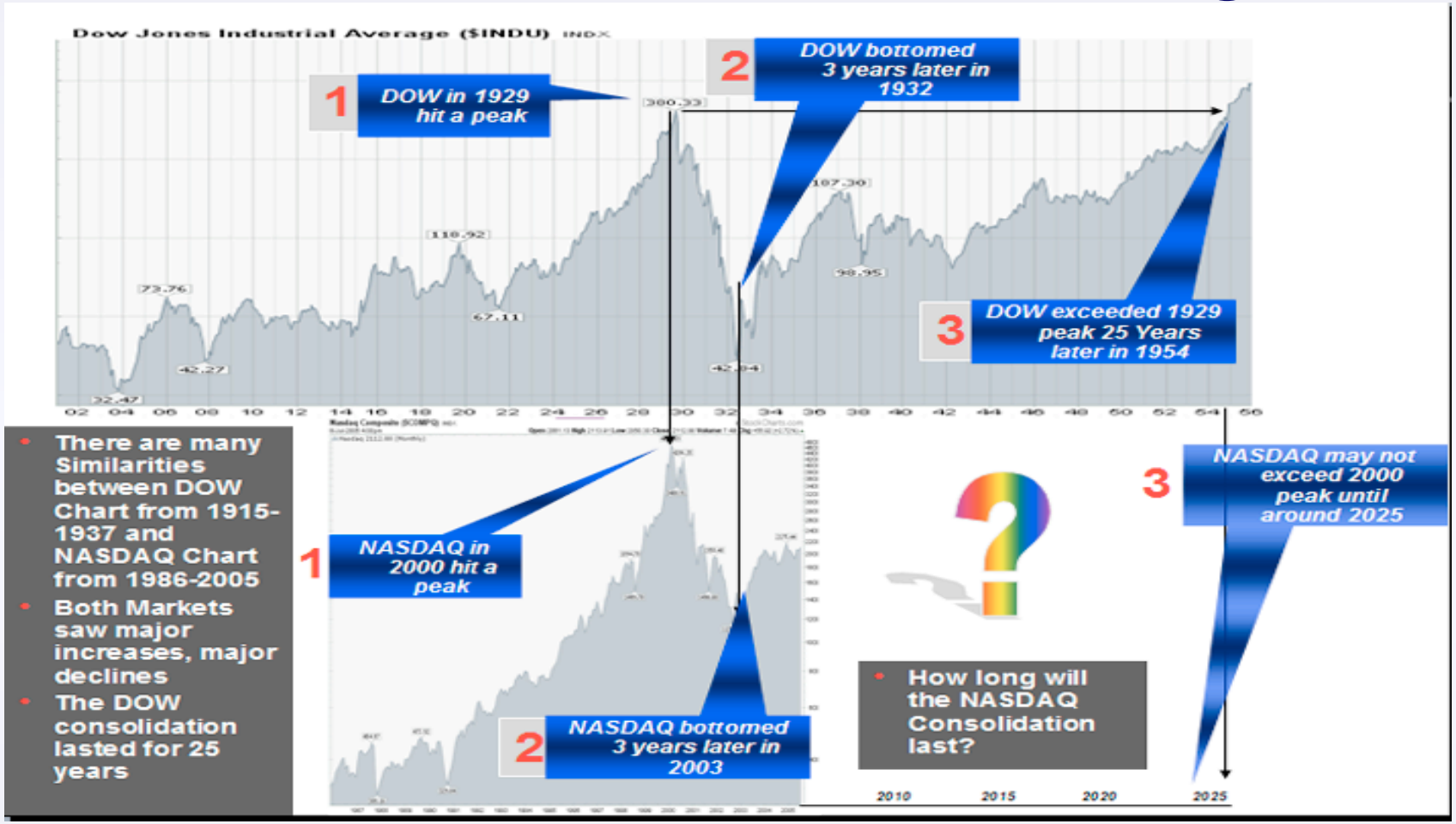
Landmark SLAM

(E. Nebot, Victoria Park)



- As robot moves, estimate its pose & world geometry

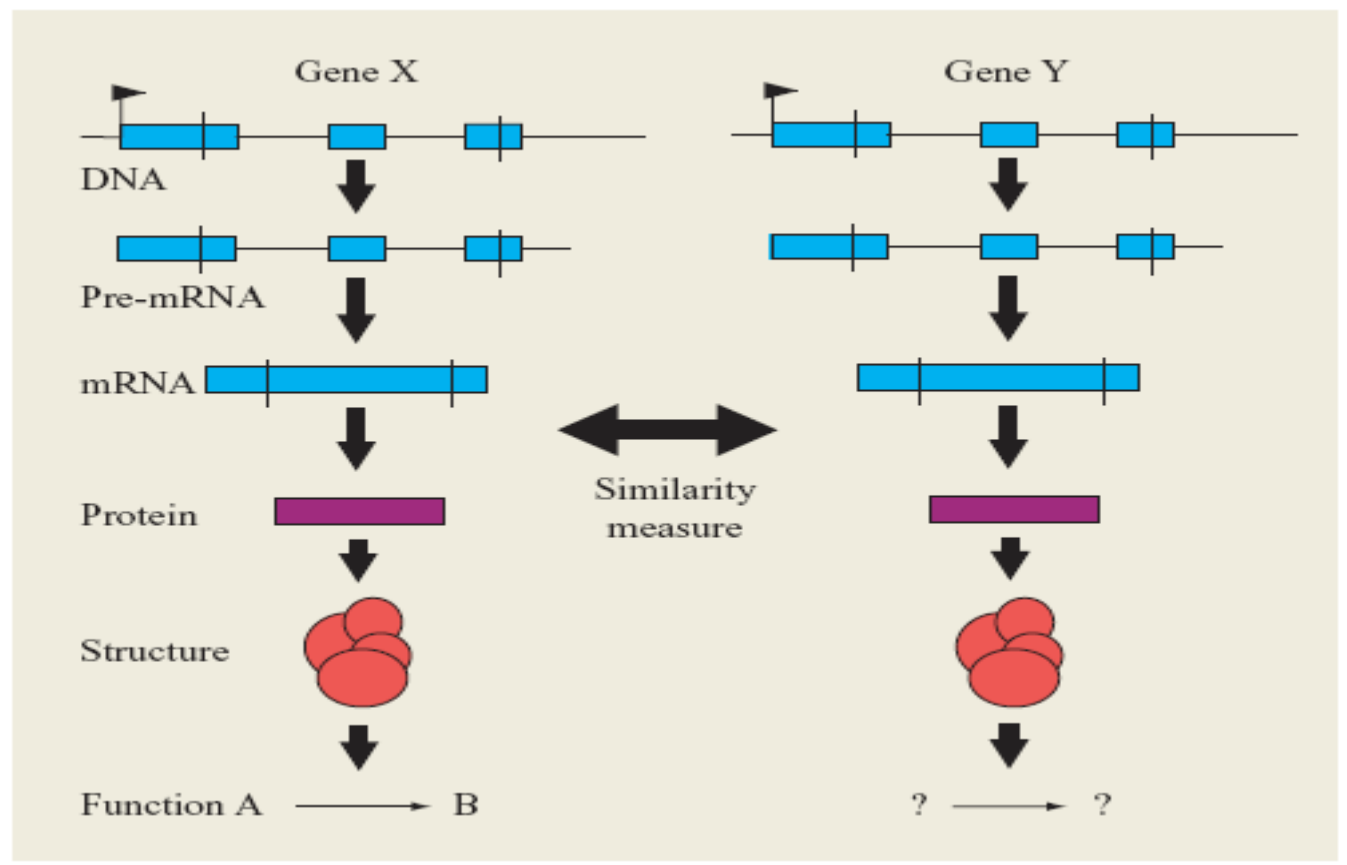
Financial Forecasting



http://www.steadfastinvestor.com/

- Predict future market behavior from historical data, news reports, expert opinions, …

Biological Sequence Analysis



(E. Birney, 2001)

- Temporal models can be adapted to exploit more general forms of *sequential* structure, like those arising in DNA sequences

Modeling Time Series

- Time series: data are correlated; data are realizations of stochastic processes
- Standard machine learning methods are often difficult to directly apply
  - Do not exploit temporal correlations
  - Computation & storage requirements typically scale poorly to realistic applications
- Stochastic linear discrete input-output models
- Two approaches:
  - Model the data as a function of time (a regression over time)
  - Model the data as a function of its past values: ARMA models
- Often, assumption of stationarity (the mean and variance of the process generating the data do not change over time)

Autoregressive (AR) Models

- AR(h) is a regression model that regresses each point on the previous h time points. Example is AR(1)
- Each value is affected by random noise with zero mean and variance s^2
- Can be learned with linear estimation algorithm

$$y_{k+1} = a_1 y_k + e_k$$

Moving Average Model

- A different kind of model is the Moving Average model (MA(h))
- It propagates over time the effect of the random fluctuations
- The autocorrelation function may help in choosing proper models
- An iterative estimation process is needed

$$y_k = b_1 e_{k-1} + e_k$$

ARMA

- AutoRegressive Moving Average model (ARMA(h))
- It can be used to obtain a more parsimonious model, with "difficult" autocorrelation functions

$$y_k = a_1 y_{k-1} + b_1 e_{k-1} + e_k$$

Non-Linear Models

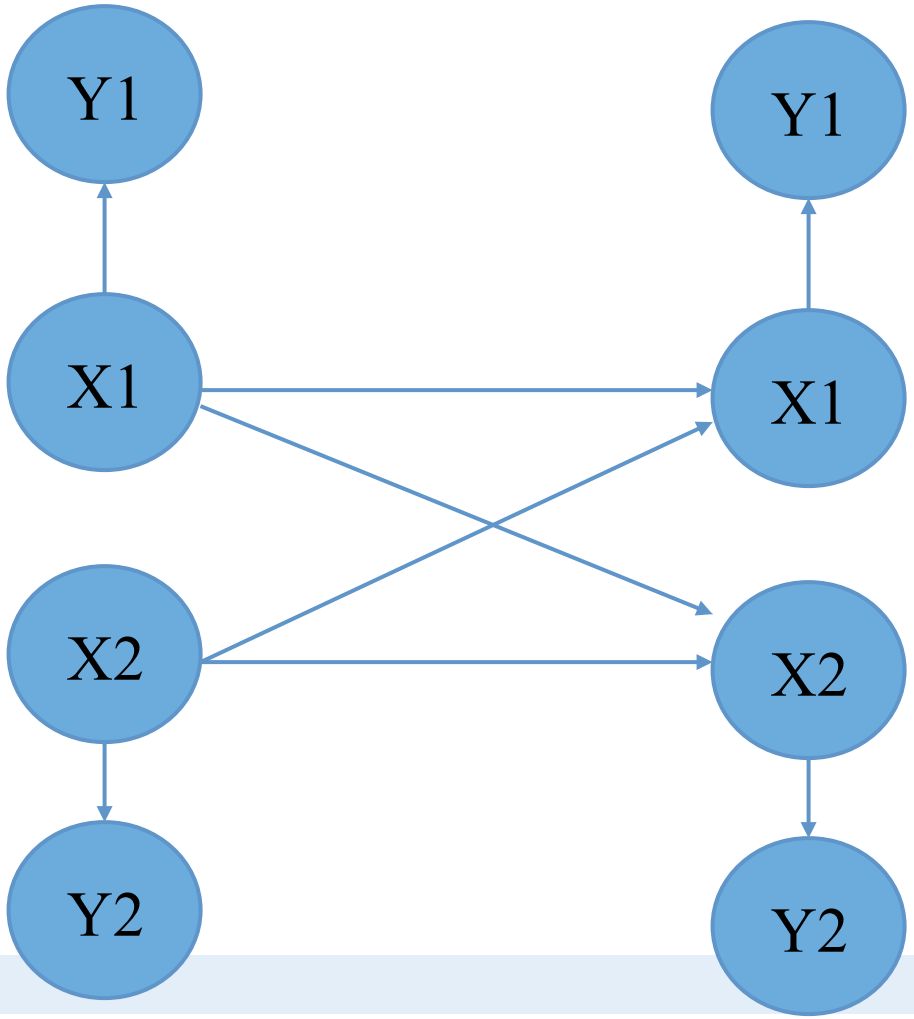
- Also non-linear stochastic models have been proposed in the literature
- Examples are NARX models

$$y_k = \sum d_k \varphi_k (y_{k-1}) + e_k$$

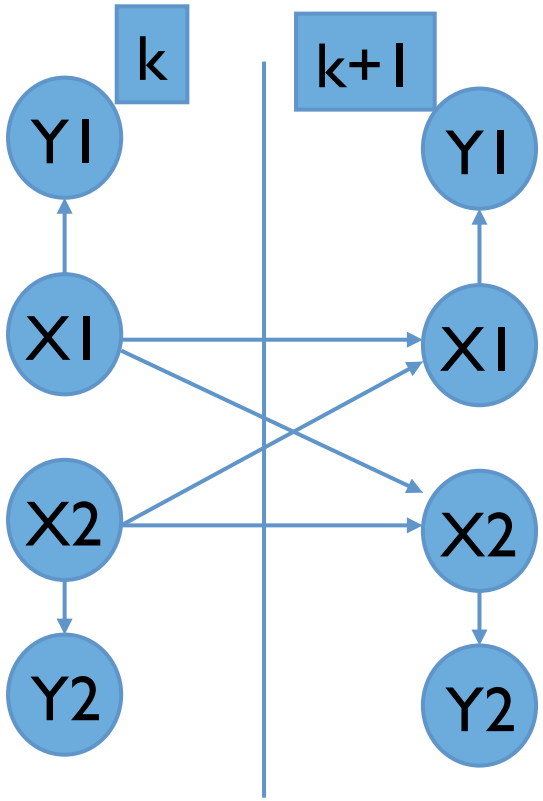
- NARX models can be easily learned from data with Neural Nets

From Black-box to Structural Stochastic Models

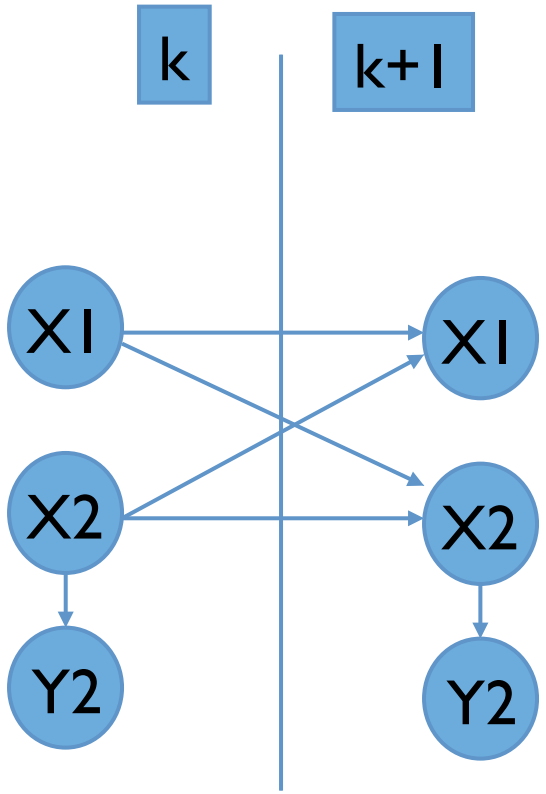
- Examples:
  - Kalman filters
  - Dynamic BNs
  - Hidden Markov Models



Observable and partially observable models



Fully observable



Partially observable

Sequential Processes

- Consider a system which can occupy one of *N* discrete *states* or *categories*

$$x_t \in \{1, 2, \ldots, N\} \longrightarrow \text{state at time } t$$

- We are interested in *stochastic* systems, in which state evolution is random
- Any *joint* distribution can be factored into a series of *conditional* distributions:

$$p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t \mid x_0, \ldots, x_{t-1})$$

Markov Processes

- For a *Markov* process, the next state depends only on the current state:

$$p(x_{t+1} \mid x_0, \ldots, x_t) = p(x_{t+1} \mid x_t)$$

- This property in turn implies that

$$p(x_0, \ldots, x_{t-1}, x_{t+1}, \ldots, x_T \mid x_t)$$
$$= p(x_0, \ldots, x_{t-1} \mid x_t) p(x_{t+1}, \ldots, x_T \mid x_t)$$

"Conditioned on the present,
the past & future are independent"

State Transition Matrices

- A *stationary* Markov chain with *N* states is described by an *NxN transition matrix:*

$$Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix}$$

$$q_{ij} \triangleq p(x_{t+1} = i \mid x_t = j)$$

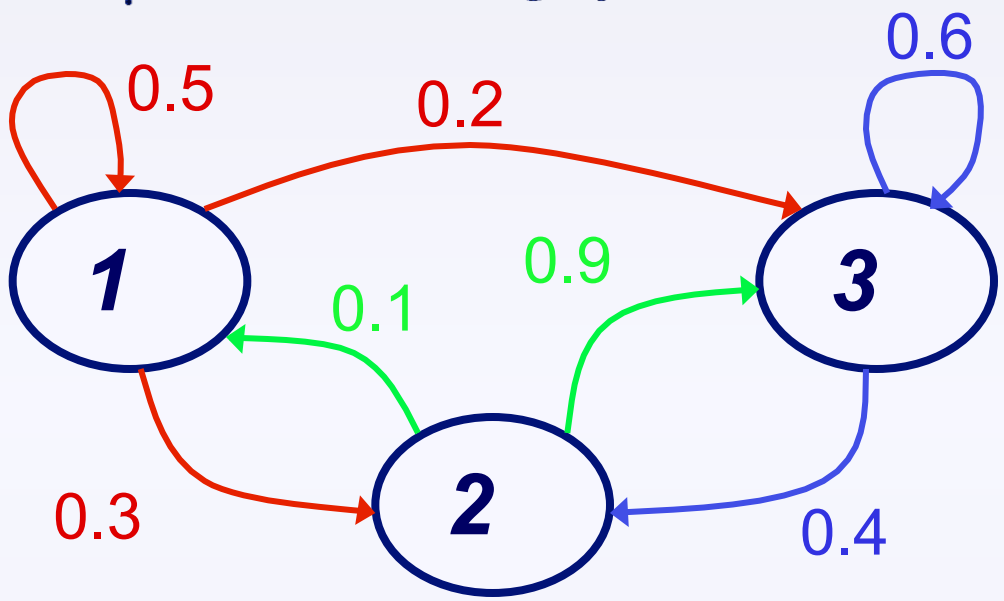
- Constraints on valid transition matrices:

$$q_{ij} \geq 0 \qquad \sum_{i=1}^{N} q_{ij} = 1 \quad \text{for all } j$$

State Transition Diagrams

$$q_{ij} \triangleq p(x_{t+1} = i \mid x_t = j)$$

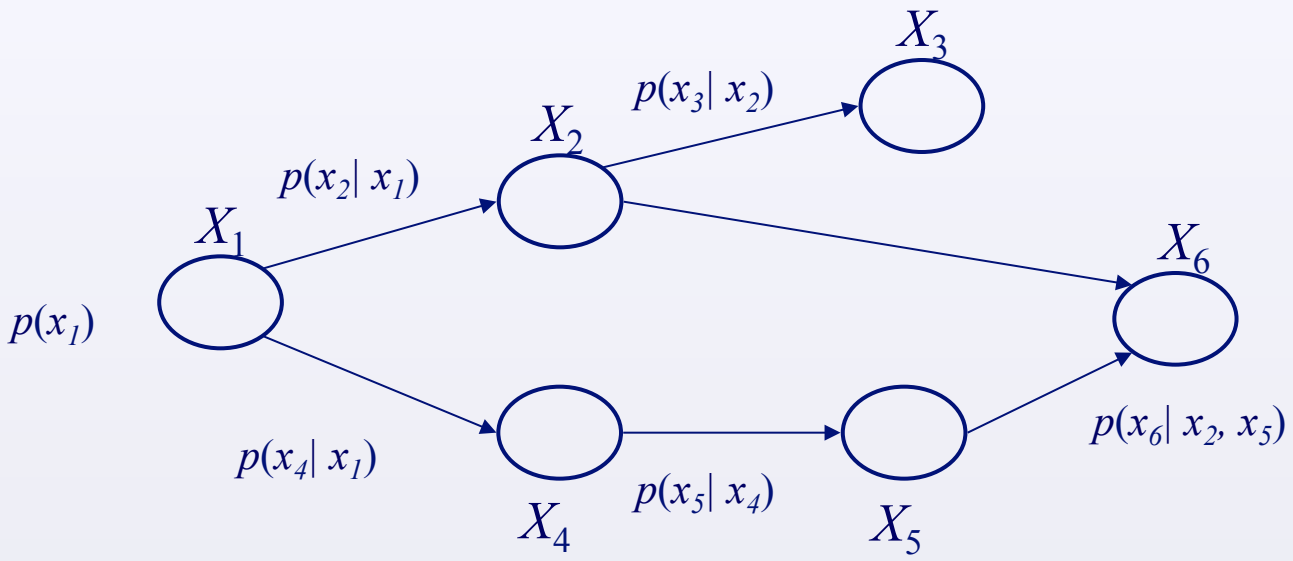
$$Q = \begin{bmatrix} 0.5 & 0.1 & 0.0 \\ 0.3 & 0.0 & 0.4 \\ 0.2 & 0.9 & 0.6 \end{bmatrix}$$



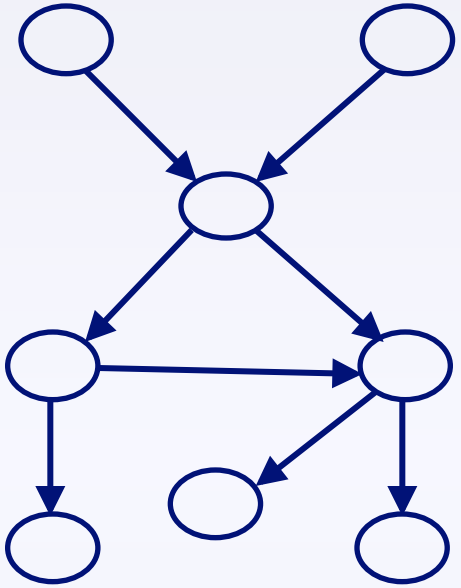
- Think of a particle randomly following an arrow at each discrete time step
- Most useful when *N* small, and *Q* *sparse*

Graphical Models – A Quick Intro

- A way of specifying conditional independences.
- Directed Graphical Modes: a DAG
- Nodes are random variables.
- A node's distribution depends on its parents.
- Joint distribution: $p(x) = \Pi_i p(x_i|\mathrm{Parents}_i)$
- A node's value conditional on its parents is independent of other ancestors.

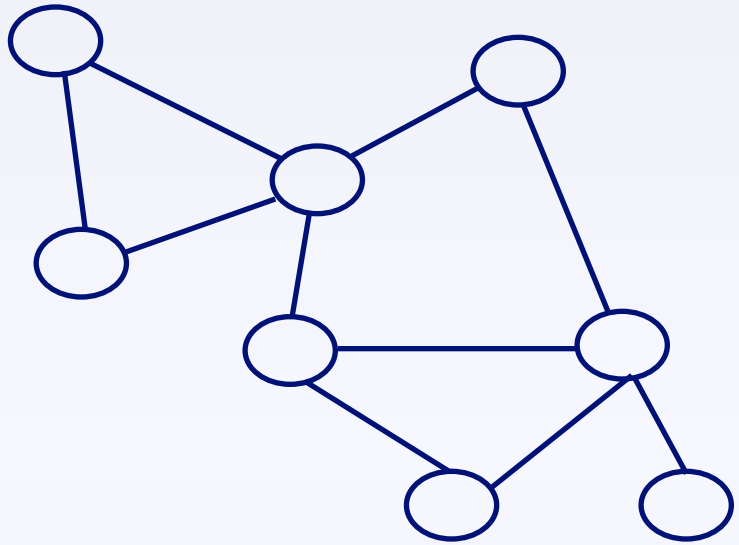


Types of Graphical Models



Nodes ⟷ Random Variables

Edges ⟷ Probabilistic (Markov) Relationships



Directed Graphs

Specify a hierarchical, causal generative process (child nodes depend on parents)

$$p(x) = \Pi_i p(x_i|\text{Parents}_i)$$

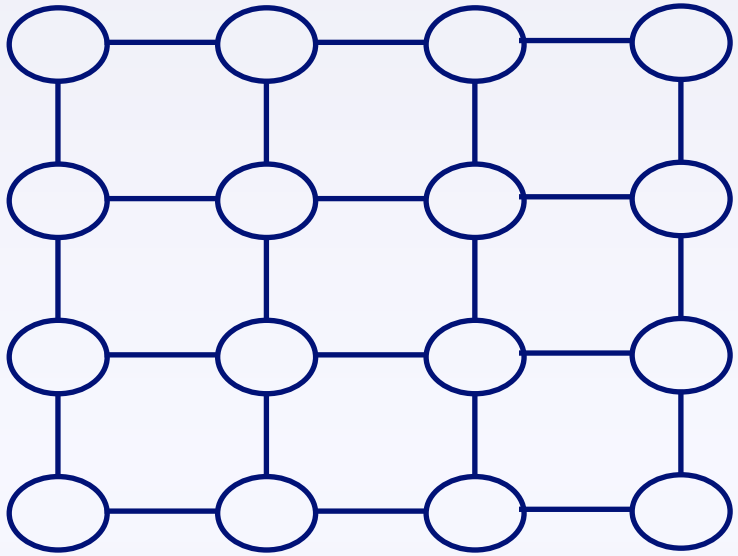
Undirected Graphs

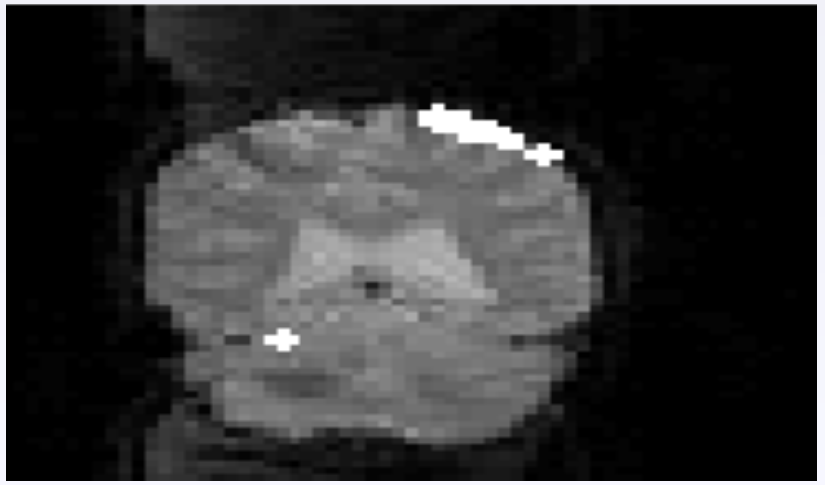
Specific symmetric, non-causal dependencies (soft or probabilistic constraints)

$$p(x) = \prod_{\text{cliques}} \Psi(x_{\text{clique}})$$

Markov Random Fields in Vision

Idea: Nearby pixels are similar.





fMRI Analysis *(Kim et. al. 2000)*

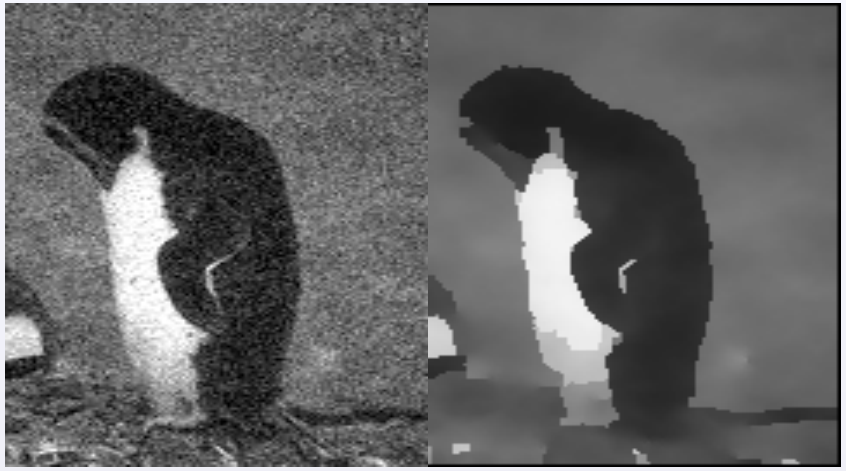
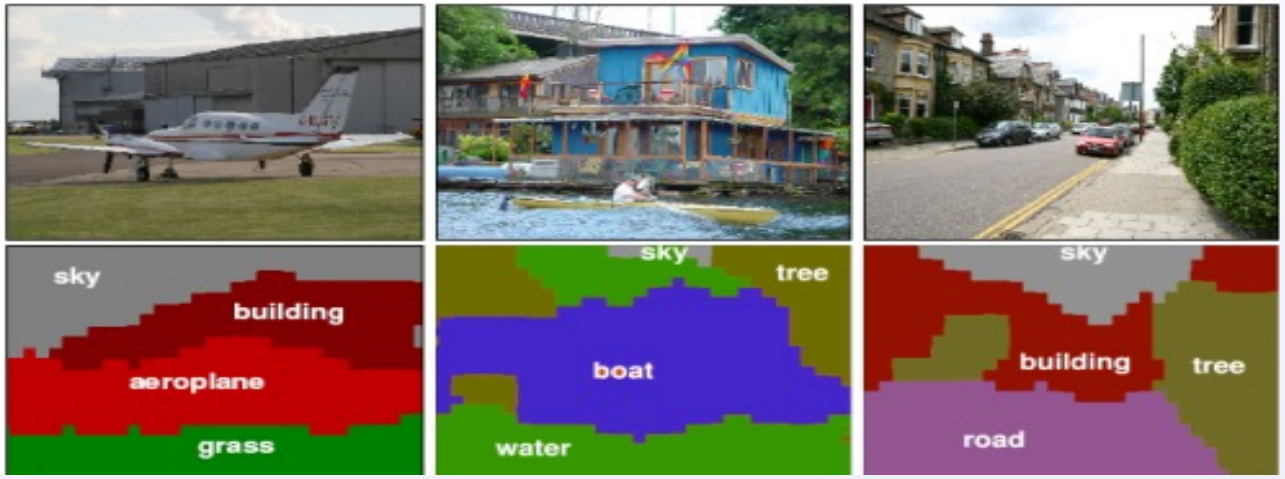


Image Denoising

(Felzenszwalb & Huttenlocher 2004)

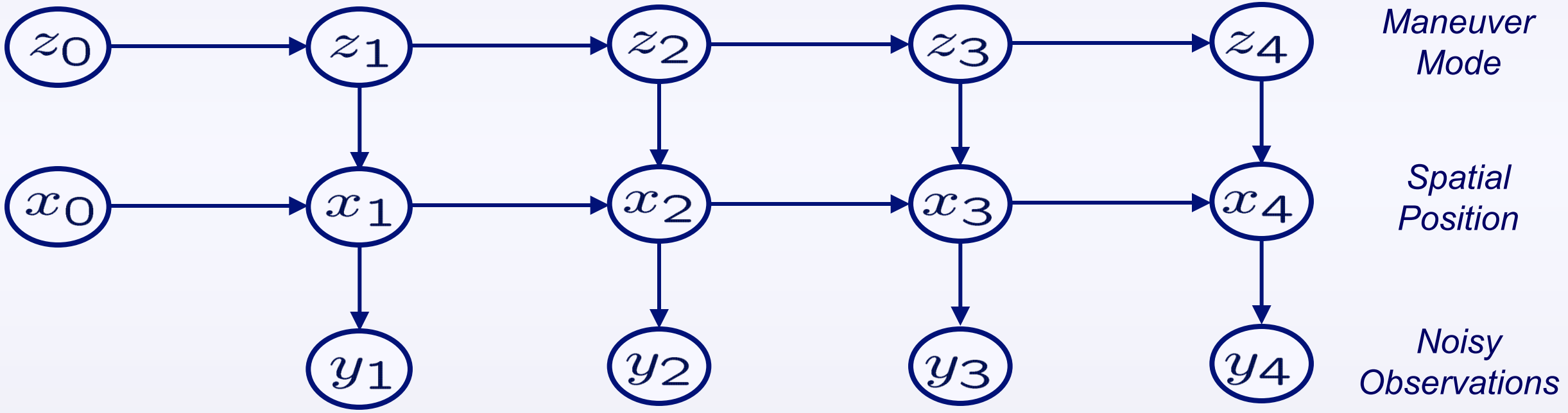


Segmentation & Object Recognition

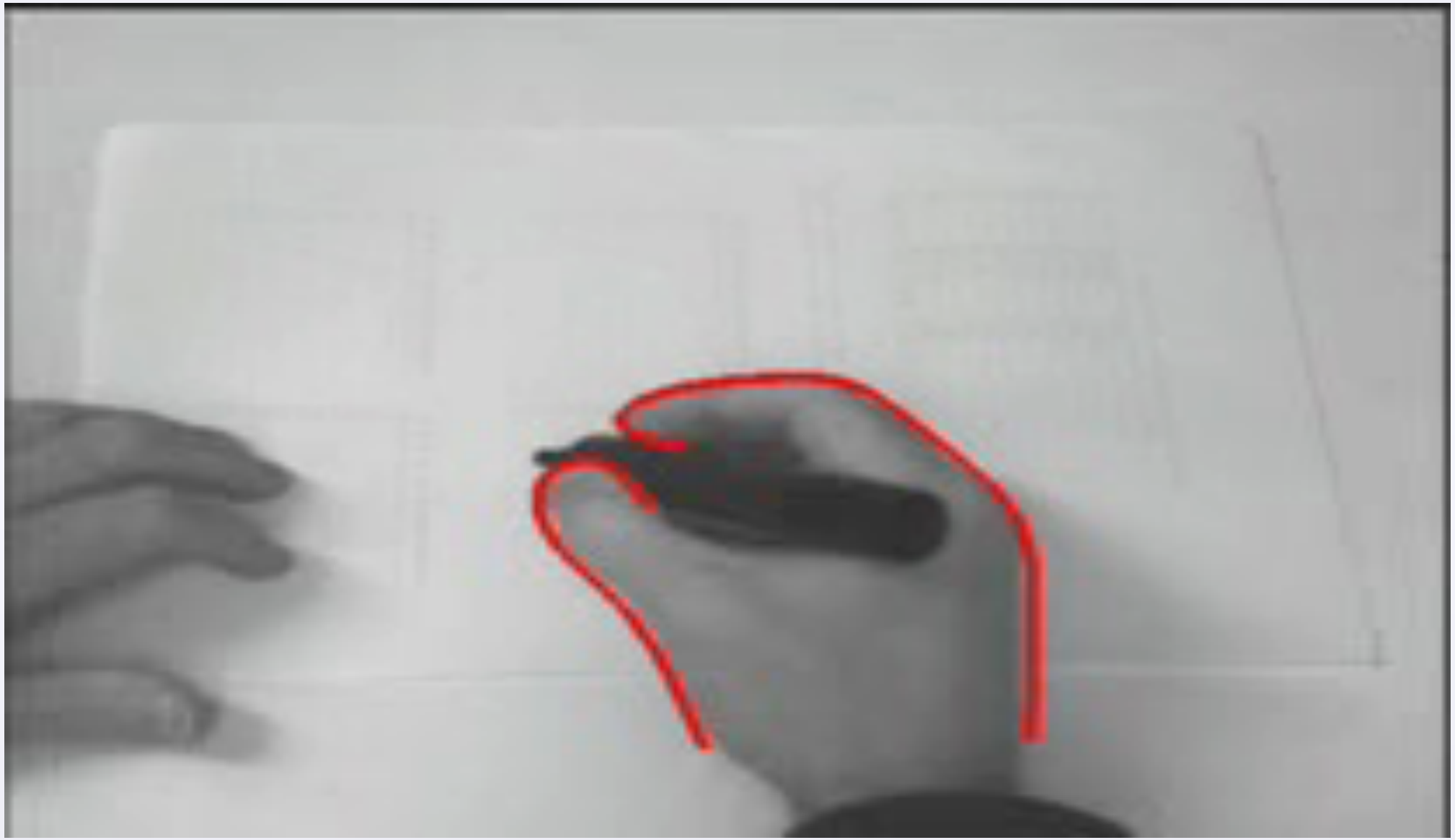
(Verbeek & Triggs 2007)

Dynamic Bayesian Networks

Specify and exploit internal structure in the hidden states underlying a time series. Generalizes HMMs

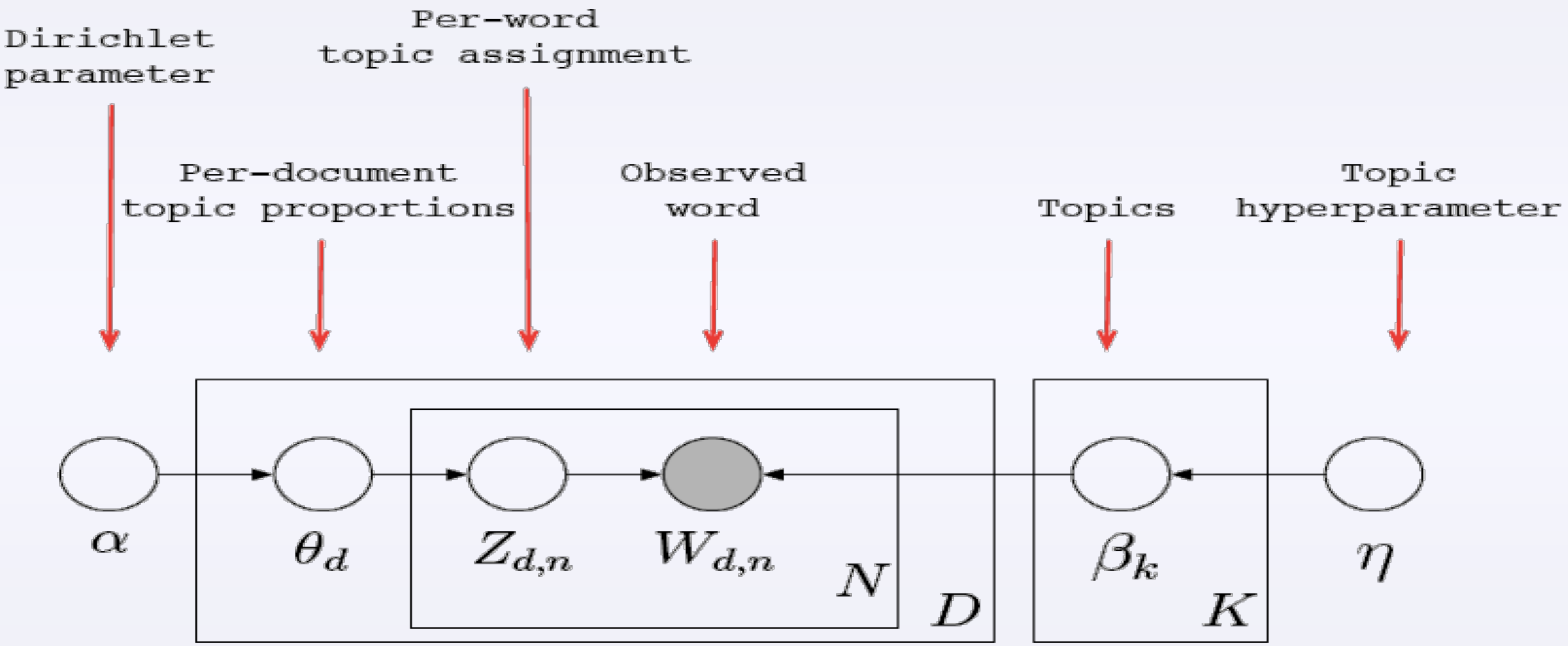


DBN Hand Tracking Video



Isard et. al., 1998

Topic Models for Documents



D. Blei, 2007

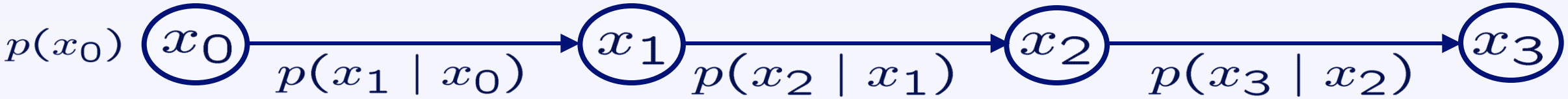
Topics Learned from *Science*

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

D. Blei, 2007

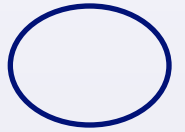
Markov Chains: Graphical Models

$$p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t \mid x_{t-1})$$



$$Q = \begin{bmatrix} 0.5 & 0.1 & 0.0 \\ 0.3 & 0.0 & 0.4 \\ 0.2 & 0.9 & 0.6 \end{bmatrix}$$

- Graph interpretation differs from state transition diagrams:



state values at particular times

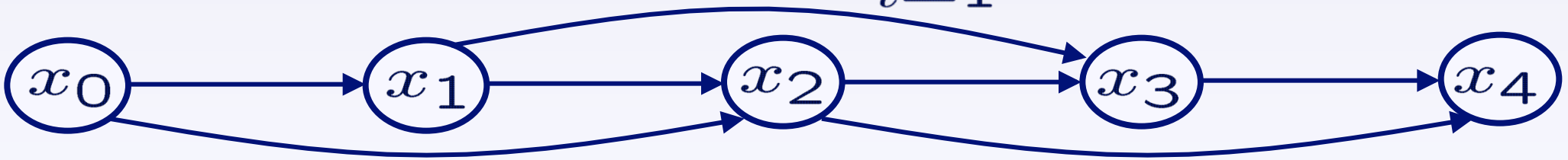
nodes

Markov properties

edges

Embedding Higher-Order Chains

$$p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t \mid x_{t-1}, x_{t-2})$$



- Each new state depends on fixed-length *window* of preceding state values
- We can represent this as a first-order model via *state augmentation*:

$$\bar{x}_t \triangleq \{x_t, x_{t-1}\}$$

(N^2 augmented states)

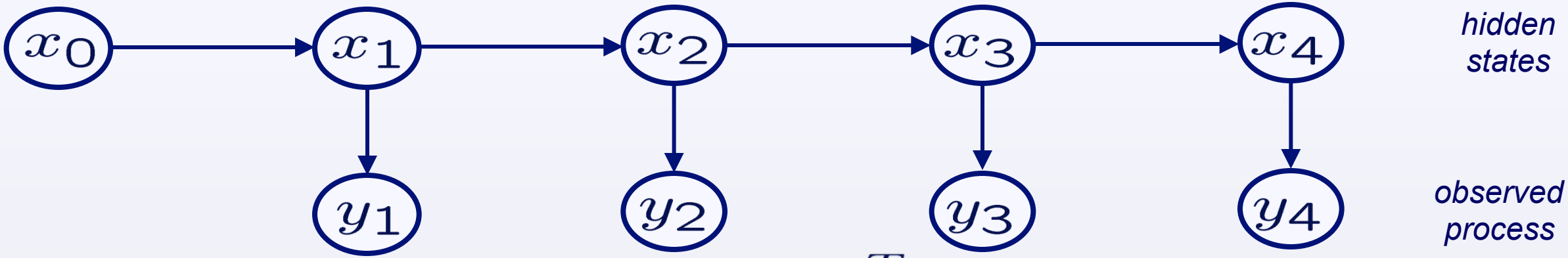
$$p(\bar{x}) = p(\bar{x}_1) \prod_{t=2}^{T} p(\bar{x}_t \mid \bar{x}_{t-1})$$

Hidden Markov Models

- Few realistic time series directly satisfy the assumptions of Markov processes:

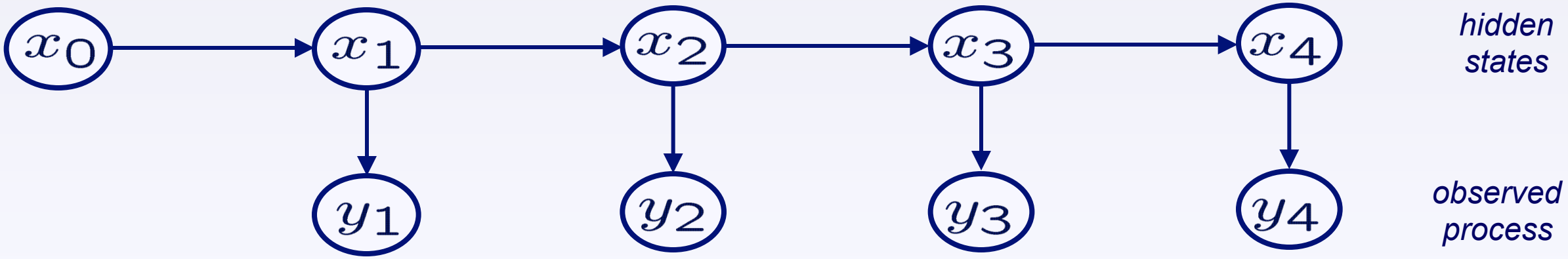
"Conditioned on the present,
the past & future are independent"

- Motivates *hidden Markov models (HMM):*



$$p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t \mid x_{t-1}) p(y_t \mid x_t)$$

Hidden states

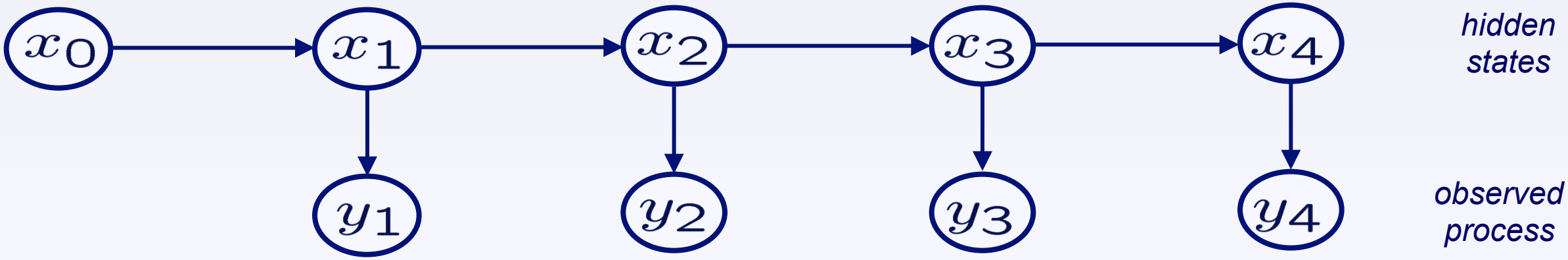


- Given x_t, earlier observations provide no *additional information* about the future:

$$p(y_t, y_{t+1}, \ldots \mid x_t, y_{t-1}, y_{t-2}, \ldots) = p(y_t, y_{t+1}, \ldots \mid x_t)$$

- Transformation of process under which dynamics take a *simple*, first-order form

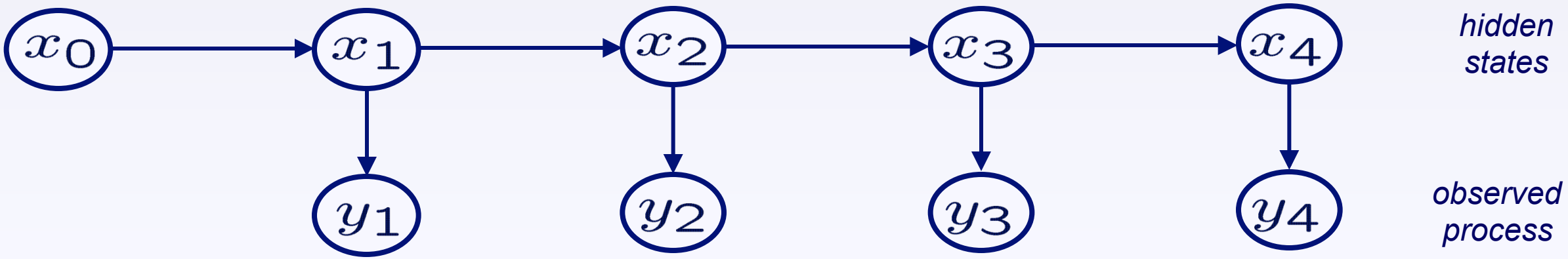
Where do states come from?



- Analysis of a *physical phenomenon*:
  - Dynamical models of an aircraft or robot
  - Geophysical models of climate evolution
- Discovered from *training data*:
  - Recorded examples of spoken English
  - Historic behavior of stock prices

Discrete State HMMs

$$x_t \in \{1, 2, \ldots, N\}$$



- Associate each of the *N* hidden states with a different observation distribution:

$$p(y_t \mid x_t = 1) \qquad p(y_t \mid x_t = 2) \qquad \cdots$$

- Observation densities are typically chosen to encode domain knowledge

Discrete HMMs: Observations

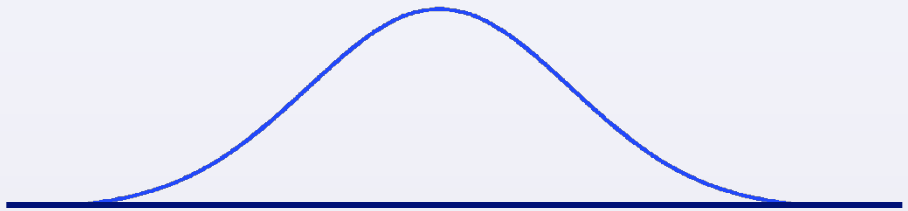
Discrete Observations

$$y_t \in \{1, 2, \ldots, M\}$$

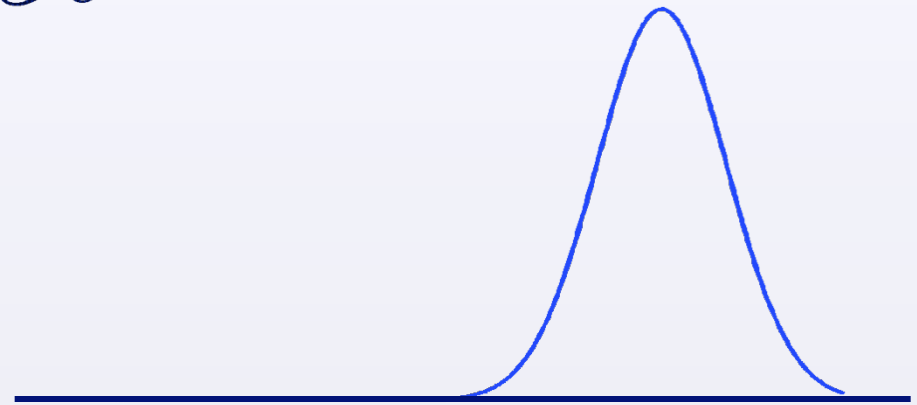
$$p(y_t \mid x_t = 1) = \begin{bmatrix} 0.3 \\ 0.1 \\ 0.5 \\ 0.1 \end{bmatrix} \qquad p(y_t \mid x_t = 2) = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.1 \\ 0.5 \end{bmatrix}$$

Continuous Observations

$$y_t \in \mathbb{R}^k$$

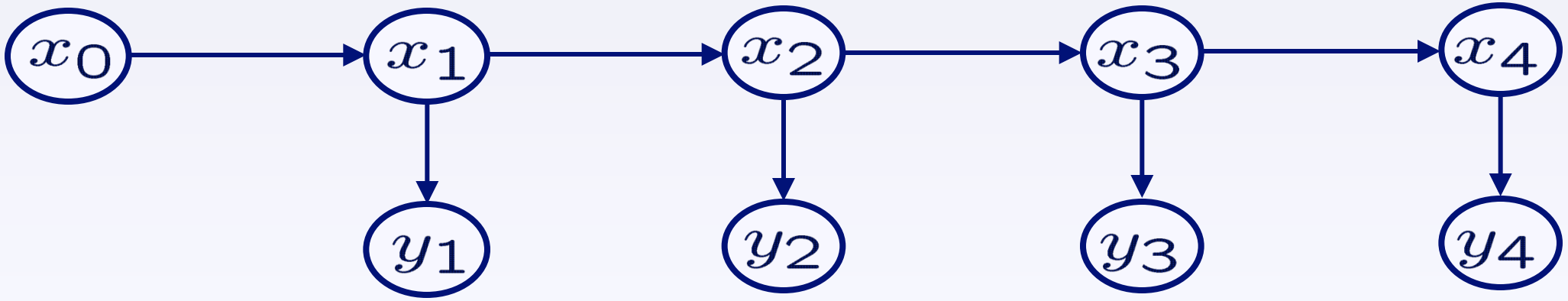


$p(y_t \mid x_t = 1)$



$p(y_t \mid x_t = 2)$

Specifying an HMM



- Observation model: $P(y_i|x_i)$
- Transition model: $P(x_i|x_{i-1})$
- Initial state distribution: $P(x_0)$

Filtering & Smoothing

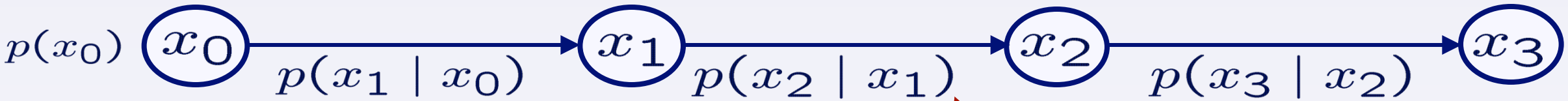
- For online data analysis, we seek *filtered* state estimates given earlier observations:

$$p(x_t \mid y_1, y_2, \ldots, y_t) \qquad t = 1, 2, \ldots$$

- In other cases, find *smoothed* estimates given earlier and later of observations:

$$p(x_t \mid y_1, y_2, \ldots, y_T) \qquad t = 1, 2, \ldots, T$$

Markov Chain Statistics



$$\alpha_t \triangleq \begin{bmatrix} p(x_t = 1) \\ p(x_t = 2) \\ p(x_t = 3) \end{bmatrix}$$

$$Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix}$$

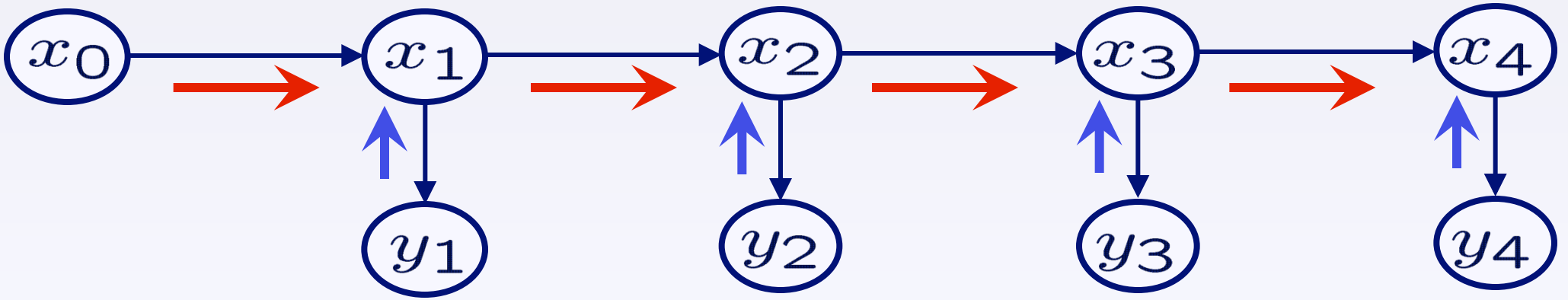
$$q_{ij} \triangleq p(x_{t+1} = i \mid x_t = j)$$

- By definition of conditional probabilities,

$$\alpha_1(i) = \sum_{j=1}^{N} q_{ij}\alpha_0(j)$$

$$\begin{cases} \alpha_1 = Q\alpha_0 \\ \alpha_t = Q^t\alpha_0 \\ \alpha_\infty = ??? \end{cases}$$

Discrete HMMs: Filtering



$$\alpha_t(x_t) \triangleq p(x_t \mid y_1, \ldots, y_t)$$

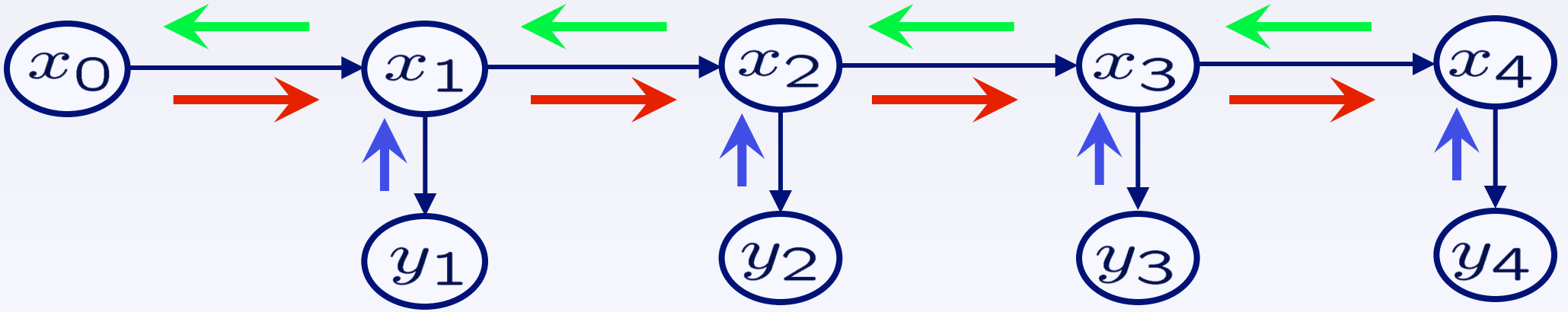
$$= \frac{1}{Z_t} p(y_t \mid x_t) \sum_{x_{t-1}} p(x_t \mid x_{t-1}) \alpha_{t-1}(x_{t-1})$$

Normalization constant

Prediction: $p(x_t \mid y_1, \ldots, y_{t-1})$

Update: $p(x_t \mid y_1, \ldots, y_t)$

Discrete HMMs: Smoothing



$$p(x_t \mid y) \propto \underbrace{p(x_t \mid y_1, \ldots, y_t)}_{\alpha_t(x_t)} \underbrace{p(y_{t+1}, \ldots, y_T \mid x_t)}_{\beta_t(x_t)}$$

- The *forward-backward* algorithm updates filtering via a *reverse-time* recursion:

$$\beta_t(x_t) = \frac{1}{Z_t} \sum_{x_{t+1}} p(x_{t+1} \mid x_t) p(y_{t+1} \mid x_{t+1}) \beta_{t+1}(x_{t+1})$$

Optimal State Estimation

$$p(x_t \mid y) = \frac{1}{Z_t}\alpha_t(x_t)\beta_t(x_t)$$

- Probabilities measure the posterior *confidence* in the true hidden states
- The *posterior mode* minimizes the number of incorrectly assigned states:

$$C(x, \hat{x}) = T - \sum_{t=1}^{T} \delta(x_t, \hat{x}_t)$$

Bit or symbol error rate

- What about the state *sequence* with the highest *joint* probability?

Word or sequence error rate

Viterbi Algorithm

$$\hat{x} = \arg\max_{x} \; p(x_0, x_1, \ldots, x_T \mid y_1, \ldots, y_T)$$

- Use *dynamic programming* to recursively find the probability of the most likely state sequence ending with each $x_t \in \{1, \ldots, N\}$

$$\gamma_t(x_t) \triangleq \max_{x_1, \ldots, x_{t-1}} p(x_1, \ldots, x_{t-1}, x_t \mid y_1, \ldots, y_t)$$

$$\propto p(y_t \mid x_t) \cdot \left[\max_{x_{t-1}} \; p(x_t \mid x_{t-1}) \gamma_{t-1}(x_{t-1}) \right]$$

- A reverse-time, *backtracking* procedure then picks the maximizing state sequence

Time Series Classification

- Suppose I'd like to know which of 2 HMMs best explains an observed sequence
- This *classification* is optimally determined by the following log-likelihood ratio:

$$\log \frac{p(y_1, \ldots, y_T \mid \mathcal{M}_1)}{p(y_1, \ldots, y_T \mid \mathcal{M}_0)} = \log \frac{p(y \mid \mathcal{M}_1)}{p(y \mid \mathcal{M}_0)}$$

$$\log p(y \mid \mathcal{M}_i) = \log \sum_x p(y \mid x, \mathcal{M}_i) p(x \mid \mathcal{M}_i)$$

- These log-likelihoods can be computed from filtering *normalization constants*

Discrete HMMs: Learning I

- Suppose first that the latent state sequence is available during training
- The *maximum likelihood* estimate equals

$$(\hat{Q}, \hat{\theta}) = \arg\max_{Q,\theta} \; p(x \mid Q) p(y \mid x, \theta)$$

$$Q = \left[q_{ij}\right] = \left[p(x_{t+1} = i \mid x_t = j)\right]$$

$$\theta = \{\theta_i\}_{i=1}^{N} \qquad \textit{(observation distributions)}$$

- For simplicity, assume observations are *Gaussian* with known variance & mean θ_i

Discrete HMMs: Learning II

- The ML estimate of the transition matrix is determined by *normalized counts*:

$$n(i,j) \triangleq \begin{cases} \text{number of times} \\ \text{observed before} \end{cases} \quad \begin{array}{l} x_t = j \\ x_{t+1} = i \end{array}$$

$$\widehat{q}_{ij} = \frac{n(i,j)}{\sum_k n(k,j)}$$

- Given x, *independently* estimate the output distribution for each state:

$$\widehat{\theta}_i = \frac{1}{|\tau_i|} \sum_{t \in \tau_i} y_t \qquad \tau_i \triangleq \{t \mid x_t = i\}$$

Discrete HMMs: EM Algorithm

- In practice, we typically don't know the hidden states for our training sequences
- The *EM algorithm* iteratively maximizes a *lower bound* on the true data likelihood:

E-Step: Use current parameters to *estimate* state

$$\hat{p}^{(s)}(x) = p(x \mid y, \hat{Q}^{(s-1)}, \hat{\theta}^{(s-1)})$$

M-Step: Use *soft* state estimates to update parameters

$$(\hat{Q}^{(s)}, \hat{\theta}^{(s)}) = \arg\max_{Q,\theta} \sum_{x} \hat{p}^{(s)}(x) \log p(x \mid Q) p(y \mid x, \theta)$$

Applied to HMMs, equivalent to the Baum-Welch algorithm

Discrete HMMs: EM Algorithm

- Due to Markov structure, EM parameter updates use local statistics, computed by the *forward-backward* algorithm *(E-step)*
- The *M-step* then estimates observation distributions via a weighted average:

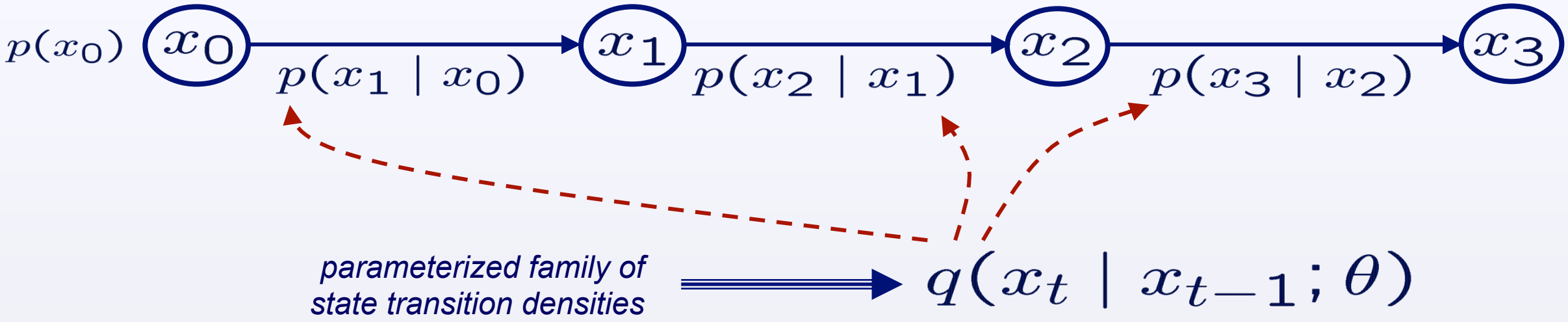
$$\widehat{\theta}_i^{(s)} = \frac{\sum_t \widehat{p}^{(s)}(x_t = i) y_t}{\sum_t \widehat{p}^{(s)}(x_t = i)}$$

- Transition matrices estimated similarly…
- May encounter *local minima*; initialization important.

Continuous State Processes

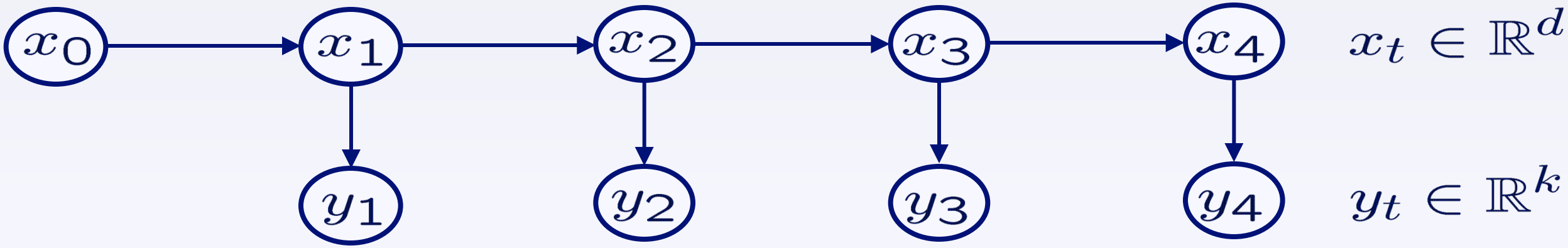
- In many applications, it is more natural to define states in some *continuous*, Euclidean space:

$$x_t \in \mathbb{R}^d$$



- *Examples:* stock price, aircraft position, …

Linear State Space Models

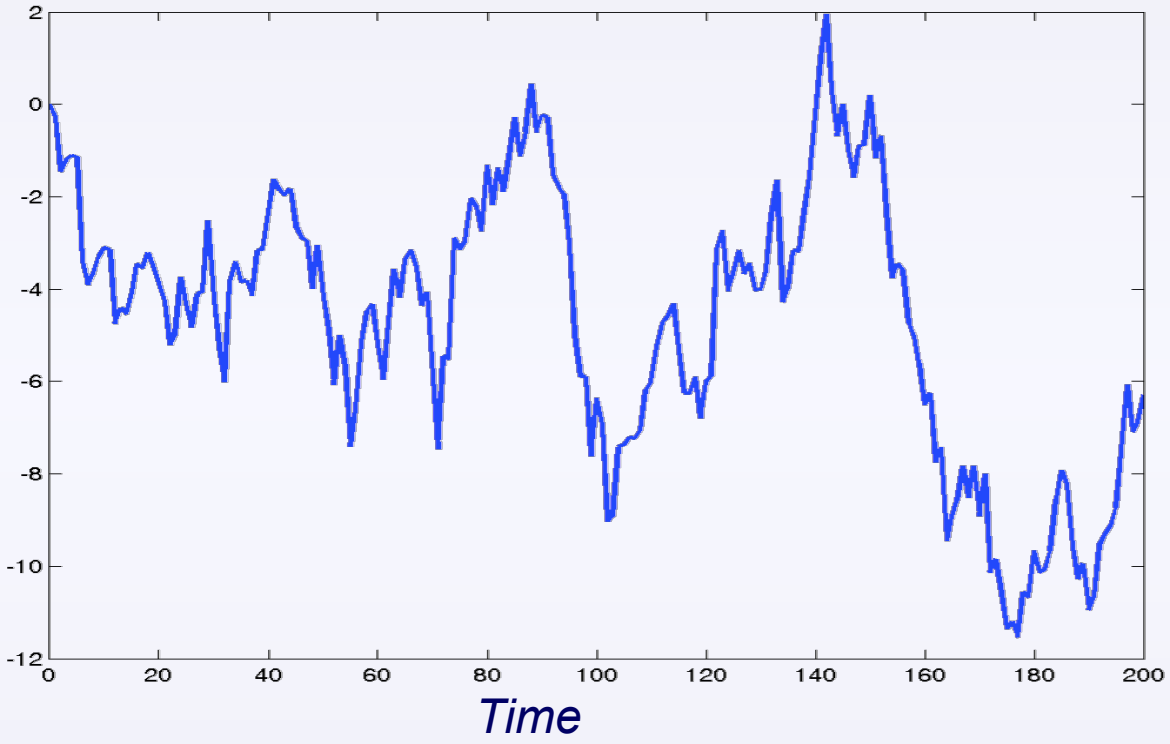


$$x_{t+1} = Ax_t + w_t \qquad w_t \sim \mathcal{N}(0, Q)$$
$$y_t = Cx_t + v_t \qquad v_t \sim \mathcal{N}(0, R)$$

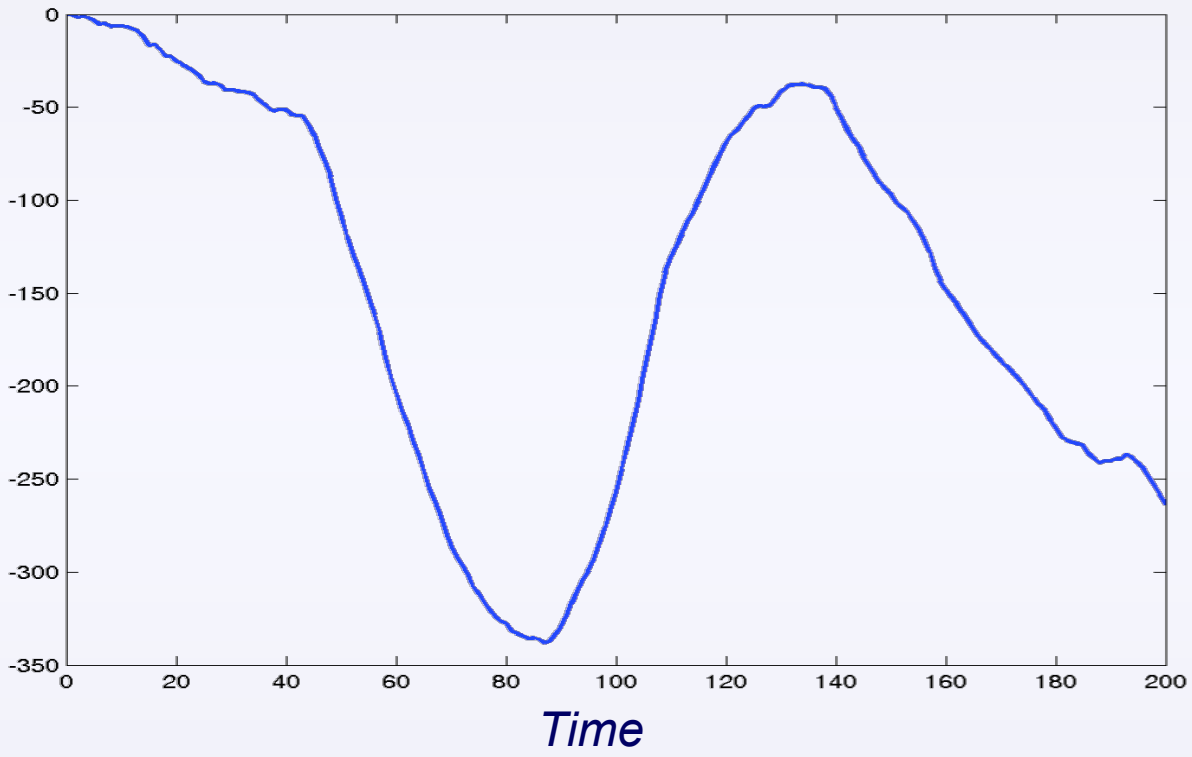
- States & observations jointly Gaussian:
  - All marginals & conditionals Gaussian
  - Linear transformations remain Gaussian

Simple Linear Dynamics

Brownian Motion



Constant Velocity



$$x_{t+1} = x_t + w_t$$

$$\begin{bmatrix} x_{t+1} \\ \delta_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ \delta_t \end{bmatrix} + w_t$$

Kalman Filter

$$x_{t+1} = Ax_t + w_t \qquad w_t \sim \mathcal{N}(0, Q)$$
$$y_t = Cx_t + v_t \qquad v_t \sim \mathcal{N}(0, R)$$

- Represent Gaussians by *mean* & *covariance*:

$$p(x_t \mid y_1, \ldots, y_{t-1}) = \mathcal{N}(x; \tilde{\mu}_t, \tilde{\Lambda}_t)$$
$$p(x_t \mid y_1, \ldots, y_t) = \mathcal{N}(x; \mu_t, \Lambda_t)$$

Prediction:

$$\tilde{\mu}_t = A\mu_{t-1}$$
$$\tilde{\Lambda}_t = A\Lambda_{t-1}A^T + Q$$

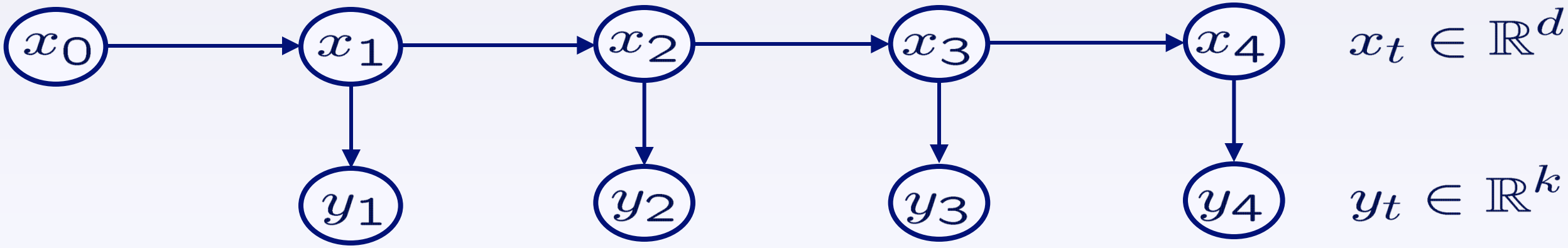
Kalman Gain:

$$K_t = \tilde{\Lambda}_t C^T (C\tilde{\Lambda}_t C^T + R)^{-1}$$

Update:

$$\mu_t = \tilde{\mu}_t + K_t(y_t - C\tilde{\mu}_t)$$
$$\Lambda_t = \tilde{\Lambda}_t - K_t C\tilde{\Lambda}_t$$

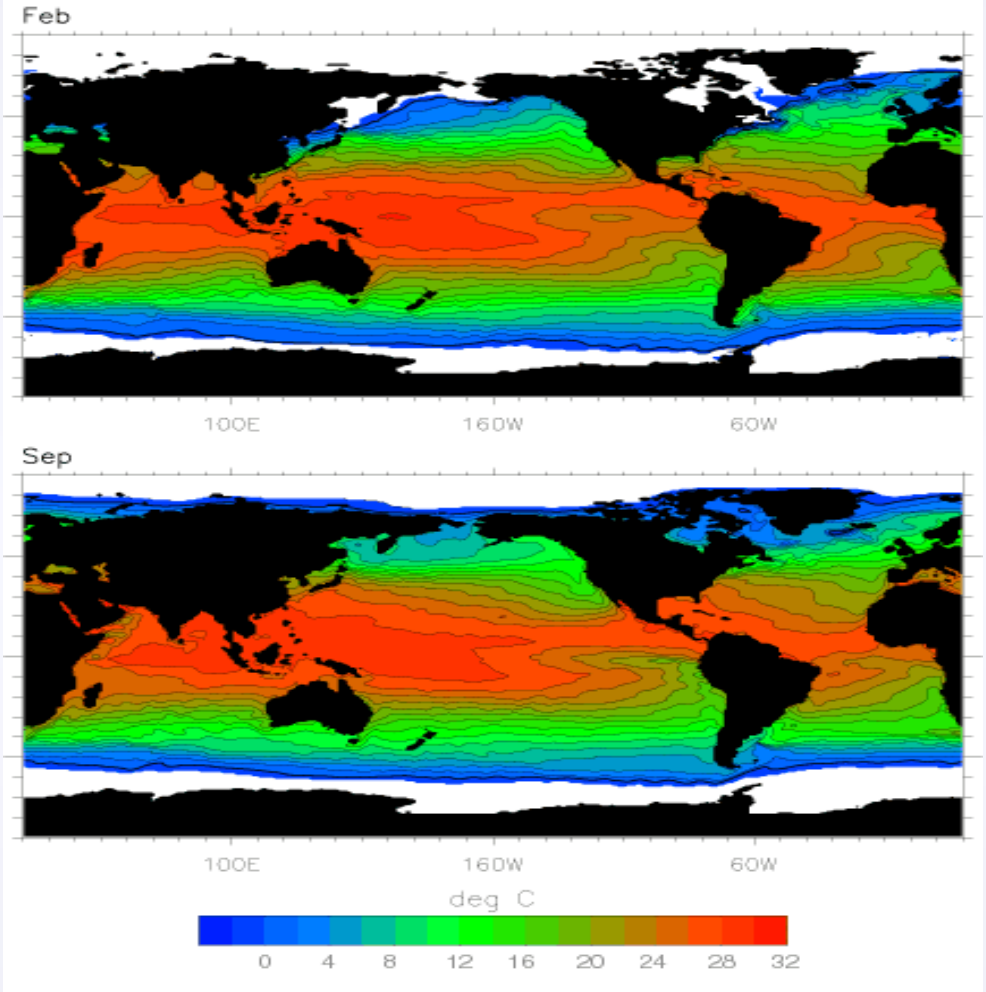
Nonlinear State Space Models



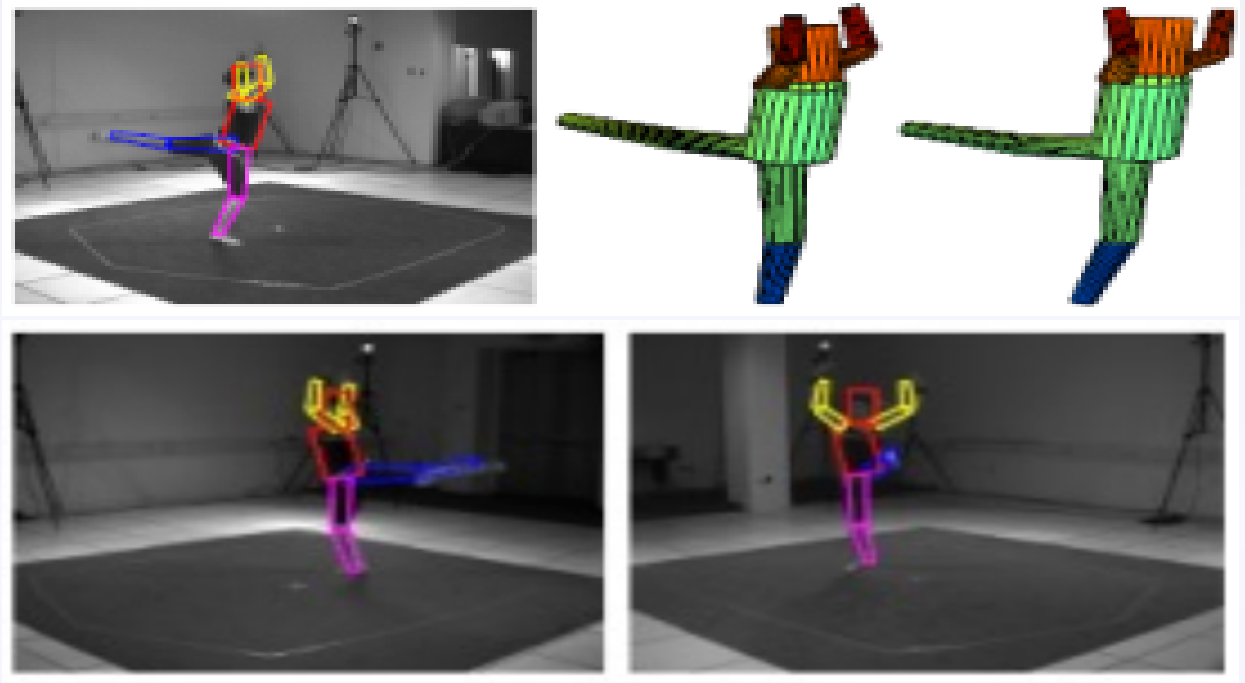
$$x_{t+1} = f(x_t, w_t) \qquad w_t \sim \mathcal{F}$$
$$y_t = g(x_t, v_t) \qquad v_t \sim \mathcal{G}$$

- State dynamics and measurements given by potentially complex *nonlinear functions*
- Noise sampled from *non-Gaussian* distributions

Examples of Nonlinear Models

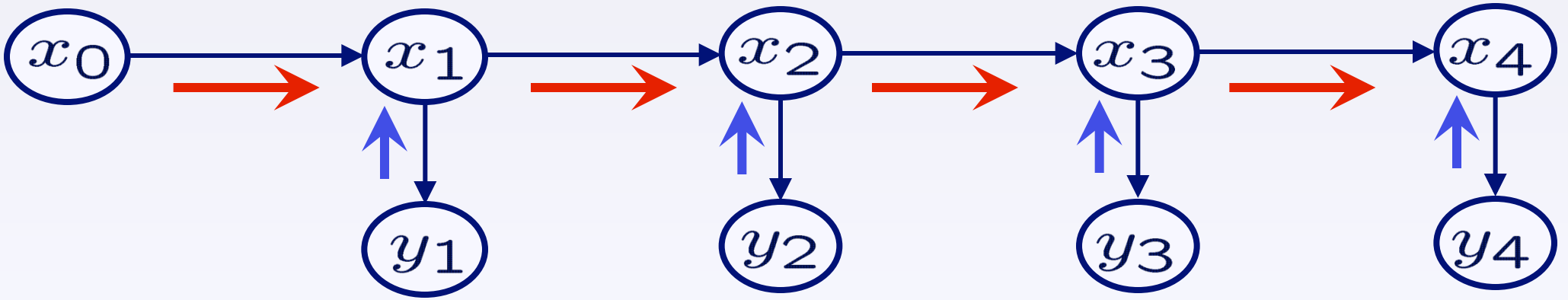


Dynamics implicitly determined by geophysical simulations



Observed image is a complex function of the 3D pose, other nearby objects & clutter, lighting conditions, camera calibration, etc.

Nonlinear Filtering



$$p(x_t \mid y_1, \ldots, y_{t-1}) = \tilde{q}_t(x_t)$$

$$p(x_t \mid y_1, \ldots, y_t) = q_t(x_t)$$

Prediction:

$$\tilde{q}_t(x_t) = \int p(x_t \mid x_{t-1}) q_{t-1}(x_{t-1}) \, dx_{t-1}$$

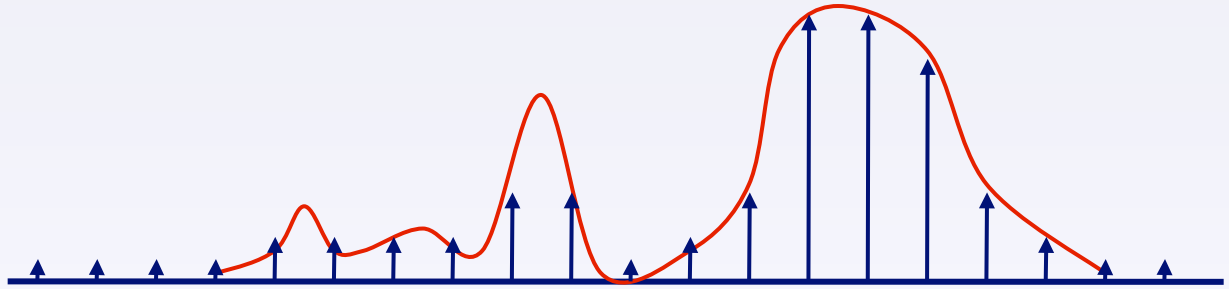
Update:

$$q_t(x_t) = \frac{1}{Z_t} \tilde{q}_t(x_t) p(y_t \mid x_t)$$

Nonlinear Filtering Taxonomy

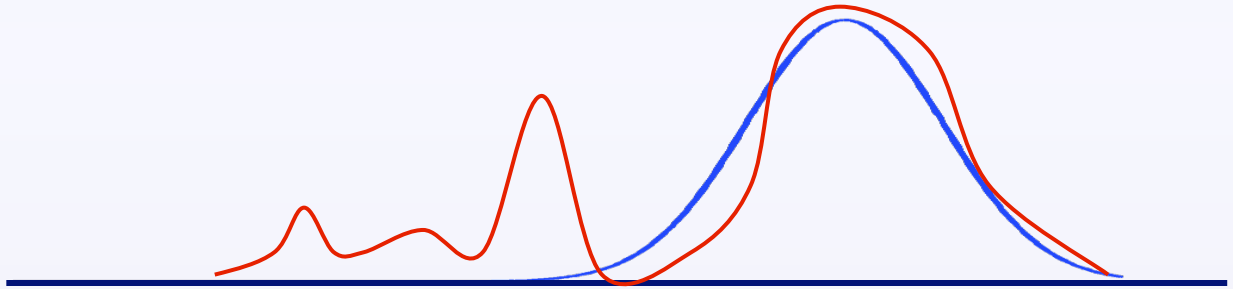
Histogram Filter:

- *Evaluate on fixed discretization grid*
- *Only feasible in low dimensions*
- *Expensive or inaccurate*



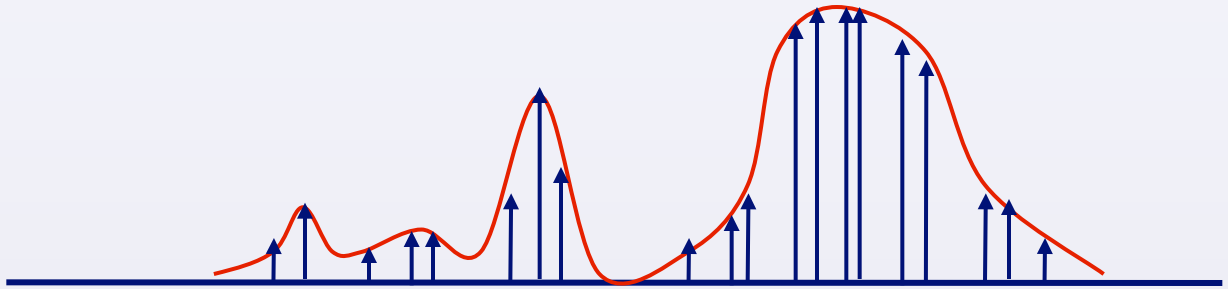
Extended/Unscented Kalman Filter:

- *Approximate posterior as Gaussian via linearization, quadrature, …*
- *Inaccurate for multimodal posterior distributions*



Particle Filter:

- *Dynamically evaluate states with highest probability*
- *Monte Carlo approximation*



Readings

- "Introduction to Machine Learning" by Ethem Alpaydin, Chapters 6, 15
- Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2001). "8.5 The EM algorithm". The Elements of Statistical Learning. New York: Springer. pp. 236–243.
- Rabiner, A Tutorial on HMMs and Selected Applications in Speech Recognition

Today

- EM Algorithm and Its Uses in ML
- Time Series Analysis and Hidden Markov Models
- Hackathon/Challenge winners' presentations