

Probabilistic Models for Unsupervised Learning

**Zoubin Ghahramani
Sam Roweis**

**Gatsby Computational Neuroscience Unit
University College London**

<http://www.gatsby.ucl.ac.uk/>

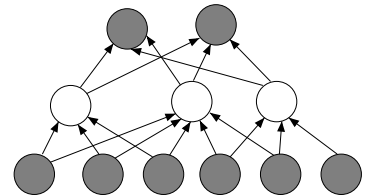
**NIPS Tutorial
December 1999**

Learning

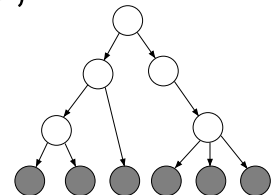
Imagine a machine or organism that experiences over its lifetime a series of sensory inputs:

$$x_1, x_2, x_3, x_4, \dots$$

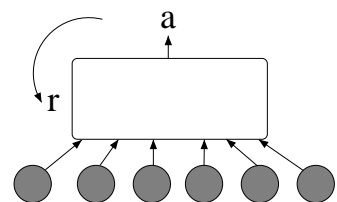
Supervised learning: The machine is also given desired outputs y_1, y_2, \dots , and its goal is to learn to produce the correct output given a new input.



Unsupervised learning: The goal of the machine is to build representations from x that can be used for reasoning, decision making, predicting things, communicating etc.



Reinforcement learning: The machine can also produce actions a_1, a_2, \dots which affect the state of the world, and receives rewards (or punishments) r_1, r_2, \dots . Its goal is to learn to act in a way that maximises rewards in the long term.



Goals of Unsupervised Learning

To find useful representations of the data, for example:

- finding clusters, e.g. k-means, ART
- dimensionality reduction, e.g. PCA, Hebbian learning, multidimensional scaling (MDS)
- building topographic maps, e.g. elastic networks, Kohonen maps
- finding the hidden causes or sources of the data
- modeling the data density

We can quantify what we mean by “useful” later.

Uses of Unsupervised Learning

- data compression
- outlier detection
- classification
- make other learning tasks easier
- a theory of human learning and perception

Probabilistic Models

- A probabilistic model of sensory inputs can:
 - make optimal decisions under a given loss function
 - make inferences about missing inputs
 - generate predictions/fantasies/imagery
 - communicate the data in an efficient way
- Probabilistic modeling is equivalent to other views of learning:
 - information theoretic:
finding compact representations of the data
 - physical analogies: minimising free energy of a corresponding statistical mechanical system

Bayes rule

\mathcal{D} — data set

\mathcal{M} — models (or parameters)

The probability of a model \mathcal{M} given data set \mathcal{D} is:

$$P(\mathcal{M}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{M})P(\mathcal{M})}{P(\mathcal{D})}$$

$P(\mathcal{D}|\mathcal{M})$ is the *evidence* (or *likelihood*)

$P(\mathcal{M})$ is the *prior* probability of \mathcal{M}

$P(\mathcal{M}|\mathcal{D})$ is the *posterior probability* of \mathcal{M}

$$P(\mathcal{D}) = \int P(\mathcal{D}|\mathcal{M})P(\mathcal{M}) d\mathcal{M}$$

Under very weak and reasonable assumptions, Bayes rule is the only rational and consistent way to manipulate uncertainties/beliefs (Pólya, Cox axioms, etc).

Bayes, MAP and ML

Bayesian Learning:



A diagram illustrating Bayesian learning. It shows a broad, low-amplitude curve on the left, followed by a multiplication sign, a narrow, high-amplitude curve, an equals sign, and a narrow, high-amplitude curve on the right. This represents the multiplication of a prior distribution and a likelihood function to produce a posterior distribution.

Assumes a prior over the model parameters. Computes the posterior distribution of the parameters: $P(\theta|\mathcal{D})$.

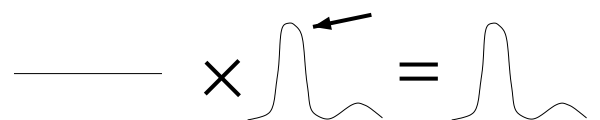
Maximum a Posteriori (MAP) Learning:



A diagram illustrating Maximum a Posteriori (MAP) learning. It shows a broad, low-amplitude curve on the left, followed by a multiplication sign, a narrow, high-amplitude curve, an equals sign, and a narrow, high-amplitude curve on the right. An arrow points to the peak of the posterior distribution on the right.

Assumes a prior over the model parameters $P(\theta)$. Finds a parameter setting that maximises the posterior: $P(\theta|\mathcal{D}) \propto P(\theta) P(\mathcal{D}|\theta)$.

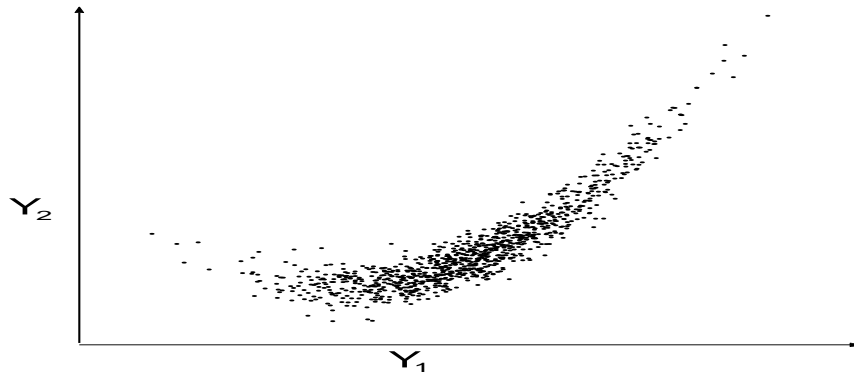
Maximum Likelihood (ML) Learning:



A diagram illustrating Maximum Likelihood (ML) learning. It shows a flat, horizontal line on the left, followed by a multiplication sign, a narrow, high-amplitude curve, an equals sign, and a narrow, high-amplitude curve on the right. An arrow points to the peak of the likelihood function on the right.

Does not assume a prior over the model parameters. Finds a parameter setting that maximises the likelihood of the data: $P(\mathcal{D}|\theta)$.

Modeling Correlations

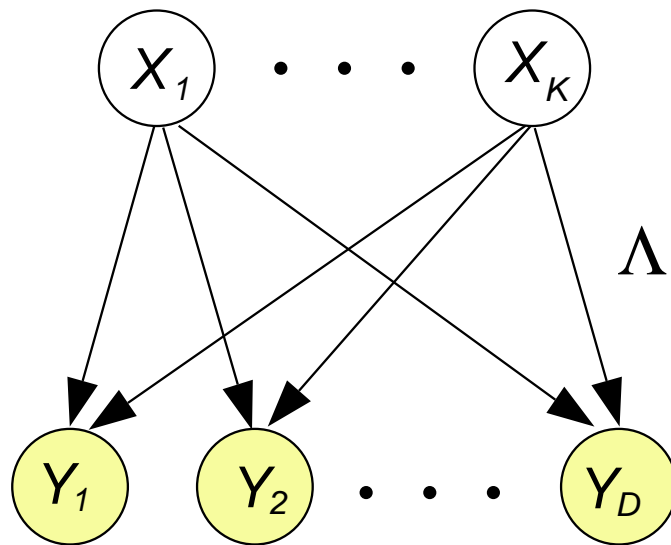


- Consider a set of variables y_1, \dots, y_D .
- A *very* simple model:
means $\mu_i = \langle y_i \rangle$ and
correlations $\Sigma_{ij} = \langle y_i y_j \rangle - \langle y_i \rangle \langle y_j \rangle$
- This corresponds to fitting a **Gaussian** to the data

$$P(\mathbf{y}) = |2\pi\Sigma|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{y} - \boldsymbol{\mu}) \right\}$$

- There are $D(D + 1)/2$ parameters in this model
- What if D is large?

Factor Analysis



Linear generative model:

$$y_d = \sum_{k=1}^K \Lambda_{dk} x_k + \epsilon_d$$

- x_k are independent $\mathcal{N}(0, 1)$ Gaussian **factors**
- ϵ_d are independent $\mathcal{N}(0, \Psi_{dd})$ Gaussian **noise**
- $K < D$

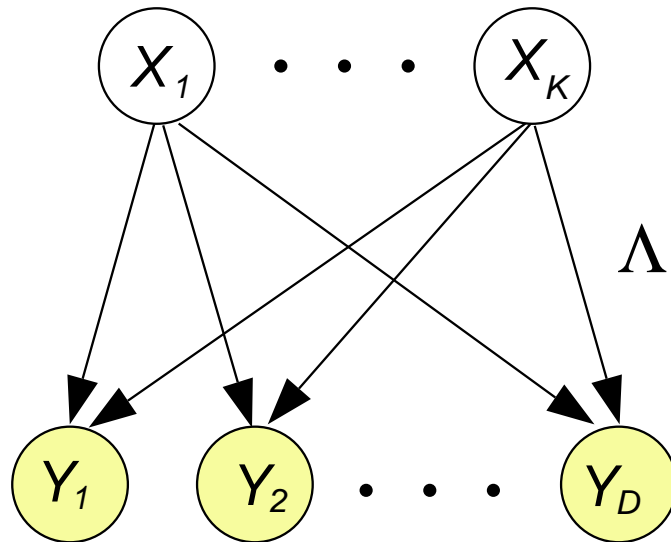
So, Y is Gaussian with:

$$P(\mathbf{y}) = \int P(\mathbf{x})P(\mathbf{y}|\mathbf{x})d\mathbf{x} = \mathcal{N}(0, \Lambda\Lambda^\top + \Psi)$$

where Λ is a $D \times K$ matrix, and Ψ is diagonal.

Dimensionality Reduction: Finds a low-dimensional projection of high dimensional data that captures most of the **correlation structure** of the data.

Factor Analysis: Notes

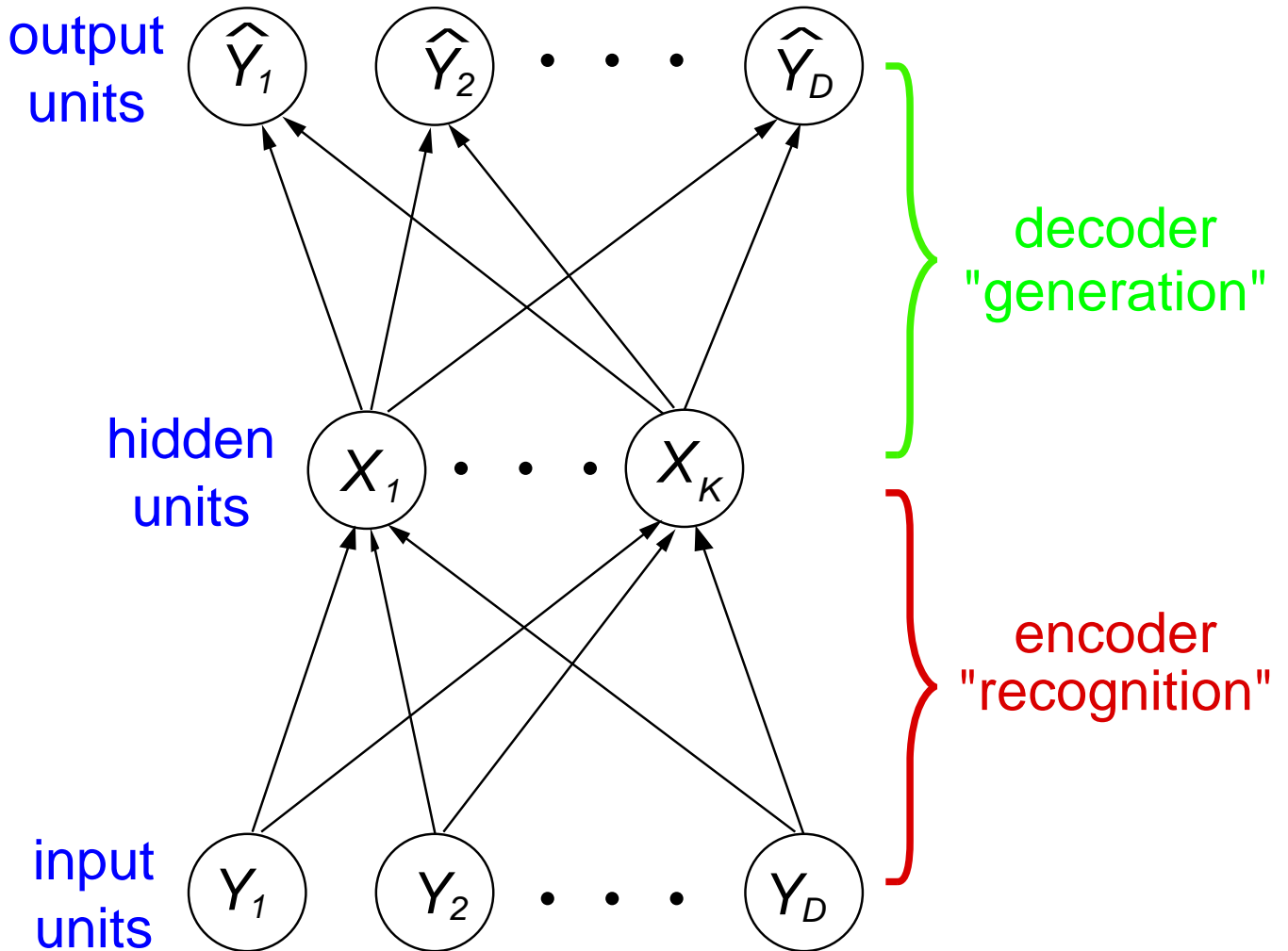


- ML learning finds Λ and Ψ given data
- parameters (with correction from symmetries):

$$DK + D - \frac{K(K - 1)}{2} < \frac{D(D + 1)}{2}$$

- no closed form solution for ML params
- Bayesian treatment would integrate over all Λ and Ψ and would find posterior on number of factors; however it is intractable.

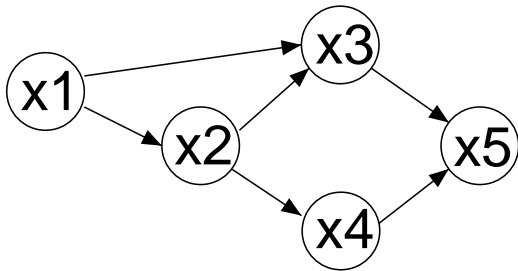
Network Interpretations



- autoencoder neural network
- if trained to minimise MSE, then we get PCA
- if MSE + output noise, we get PPCA
- if MSE + output noises + reg. penalty, we get FA

Graphical Models

A directed acyclic graph (DAG) in which each node corresponds to a random variable.



$$P(\mathbf{X}) = P(\mathbf{x}_1)P(\mathbf{x}_2|\mathbf{x}_1) \\ P(\mathbf{x}_3|\mathbf{x}_1, \mathbf{x}_2) \\ P(\mathbf{x}_4|\mathbf{x}_2)P(\mathbf{x}_5|\mathbf{x}_3, \mathbf{x}_4)$$

Definitions: *children, parents, descendants, ancestors*

Key quantity: joint probability distribution over nodes.

$$P(\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}) = P(\mathbf{X})$$

(1) The graph specifies a factorization of this joint pdf:

$$P(\mathbf{X}) = \prod_i P(\mathbf{x}_i | \text{pa}(\mathbf{x}_i))$$

(2) Each node stores a conditional distribution over its own value given the values of its parents.

(1) & (2) completely specify the joint pdf numerically.

Semantics: Given its parents, each node is *conditionally independent* from its *non-descendants*

(Also known as Bayesian Networks, Belief Networks, Probabilistic Independence Networks.)

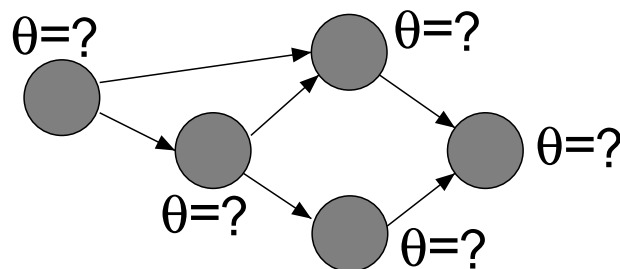
Two Unknown Quantities

In general, two quantities in the graph may be unknown:

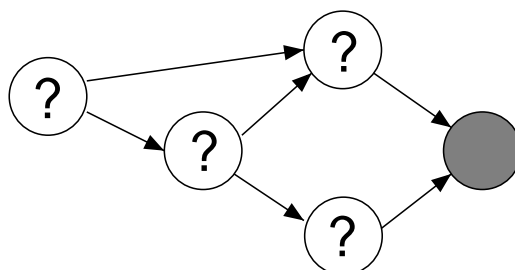
- parameter values in the distributions $P(\mathbf{x}_i | \text{pa}(\mathbf{x}_i))$
- hidden (unobserved) variables not present in the data

Assume you knew one of these:

- Known hidden variables, unknown parameters
⇒ this is **complete data learning** (decoupled problems)

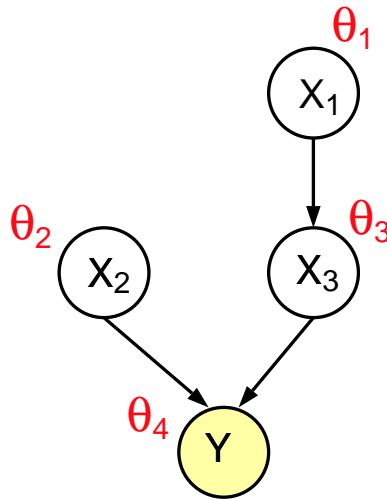


- Known parameters, unknown hidden variables
⇒ this is called **inference** (often the crux)



But what if both were unknown simultaneously...

Learning with Hidden Variables: The EM Algorithm



Assume a model parameterised by θ with observable variables Y and hidden variables X

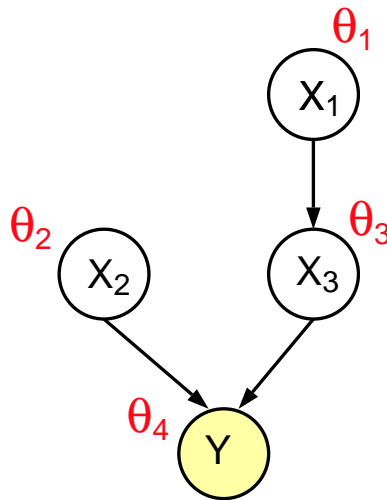
Goal: maximise log likelihood of observables.

$$\mathcal{L}(\theta) = \ln P(Y|\theta) = \ln \sum_X P(Y, X|\theta)$$

- **E-step:** first infer $P(X|Y, \theta_{old})$, then
- **M-step:** find θ_{new} using complete data learning

The E-step requires solving the *inference* problem: finding explanations, X , for the data, Y given the current model θ .

EM algorithm & \mathcal{F} -function



Any distribution $Q(X)$ over the hidden variables defines a **lower bound** on $\ln P(Y|\theta)$ called $\mathcal{F}(Q, \theta)$:

$$\begin{aligned} \ln P(Y|\theta) &= \ln \sum_X P(X, Y|\theta) = \ln \sum_X Q(X) \frac{P(X, Y|\theta)}{Q(X)} \\ &\geq \sum_X Q(X) \ln \frac{P(X, Y|\theta)}{Q(X)} = \mathcal{F}(Q, \theta) \end{aligned}$$

E-step: Maximise \mathcal{F} w.r.t. Q with θ fixed

$$Q^*(X) = P(X|Y, \theta)$$

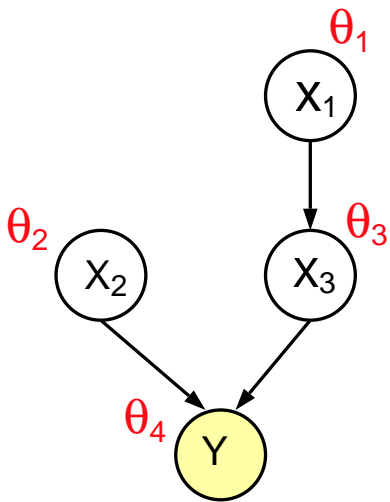
M-step: Maximise \mathcal{F} w.r.t. θ with Q fixed

$$\theta^* = \max_{\theta} \sum_X Q^*(X) \ln P(X, Y|\theta)$$

NB: max of $\mathcal{F}(Q, \theta)$ is max of $\ln P(Y|\theta)$

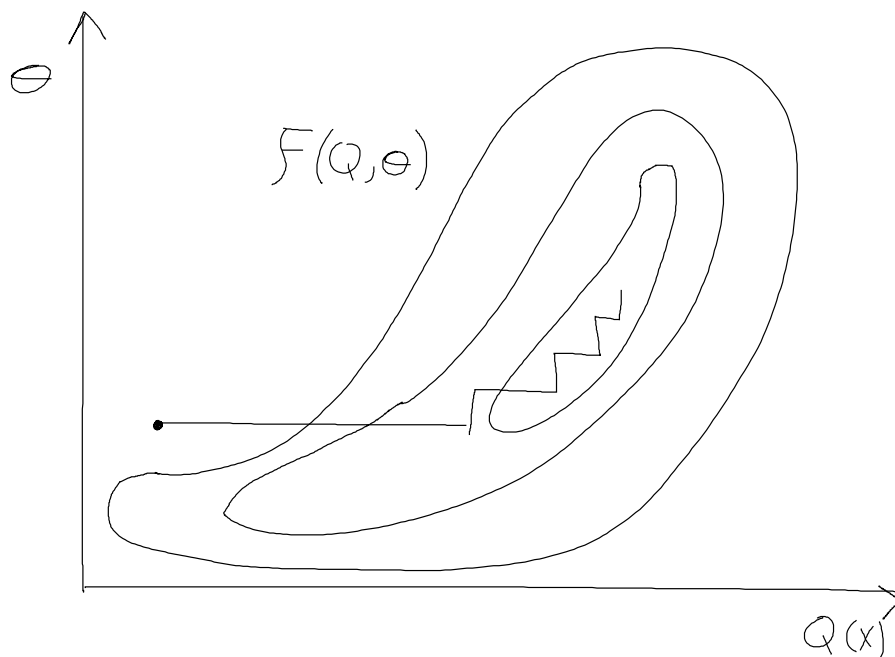
Two Intuitions about EM

I. EM decouples the parameters

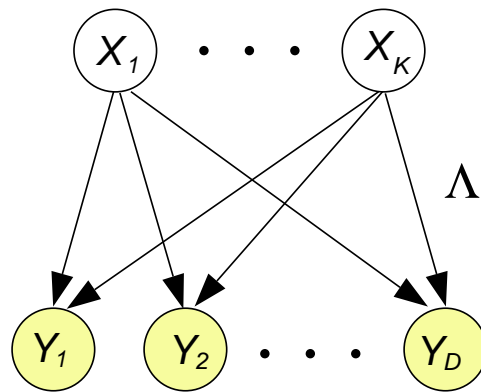


The **E-step** “fills in” values for the hidden variables. With no hidden variables, the likelihood is a simpler function of the parameters. The **M-step** for the parameters at each node can be computed independently, and depends only on the values of the variables at that node and its parents.

II. EM is coordinate ascent in \mathcal{F}



EM for Factor Analysis



$$\mathcal{F}(Q, \theta) = \int Q(\mathbf{x}) \ln P(\mathbf{x}, \mathbf{y} | \theta) d\mathbf{x} - \int Q(\mathbf{x}) \ln Q(\mathbf{x}) d\mathbf{x}$$

E-step: Maximise \mathcal{F} w.r.t. Q with θ fixed

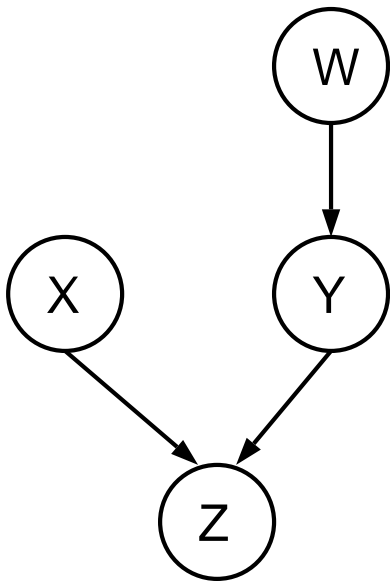
$$Q^*(\mathbf{x}) = P(\mathbf{x} | \mathbf{y}, \theta) = \mathcal{N}(\beta \mathbf{y}, I - \beta \Lambda)$$
$$\beta = \Lambda^\top (\Lambda \Lambda^\top + \Psi)^{-1}$$

M-step: Maximise \mathcal{F} w.r.t. θ with Q fixed:

$$\ln P(\mathbf{x}, \mathbf{y} | \theta) = -\frac{1}{2} (\mathbf{x}^\top \mathbf{x} + (\mathbf{y} - \Lambda \mathbf{x})^\top \Psi^{-1} (\mathbf{y} - \Lambda \mathbf{x}) + \ln |\Psi|) + c$$

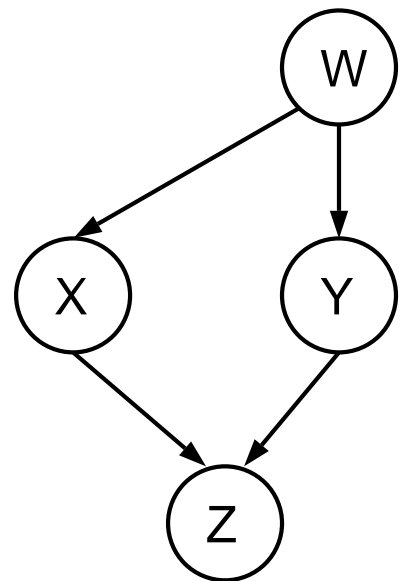
- The E-step reduces to computing the Gaussian posterior distribution over the hidden variables.
- The M-step reduces to solving a weighted linear regression problem.

Inference in Graphical Models



Singly connected nets

The *belief propagation* algorithm.



Multiply connected nets

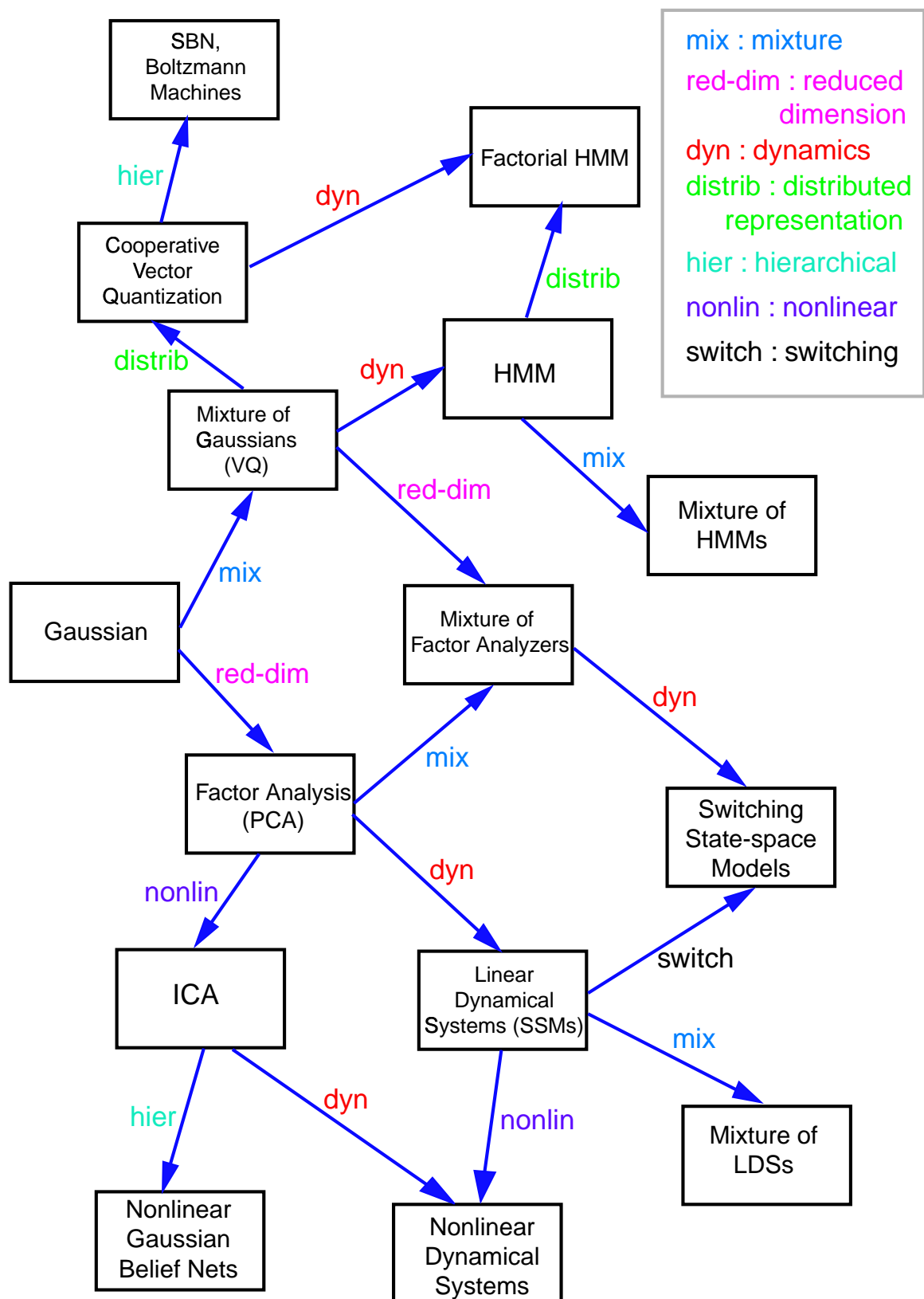
The *junction tree* algorithm.

These are efficient ways of applying Bayes rule using the conditional independence relationships implied by the graphical model.

How Factor Analysis is Related to Other Models

- **Principal Components Analysis (PCA):** Assume no noise on the observations: $\Psi = \lim_{\epsilon \rightarrow 0} \epsilon I$
- **Independent Components Analysis (ICA):** Assume the factors are non-Gaussian (and no noise).
- **Mixture of Gaussians:** A single discrete-valued factor: $x_k = 1$ and $x_j = 0$ for all $j \neq k$.
- **Mixture of Factor Analysers:** Assume the data has several clusters, each of which is modeled by a single factor analyser.
- **Linear Dynamical Systems:** Time series model in which the factor at time t depends linearly on the factor at time $t - 1$, with Gaussian noise.

A Generative Model for Generative Models



Mixture of Gaussians and K-Means

Goal: finding **clusters** in data.

To generate data from this model, assuming K clusters:

- Pick cluster $k \in \{1, \dots, K\}$ with probability π_k
- Generate data according to a Gaussian with mean $\boldsymbol{\mu}_k$ and covariance Σ_k

$$\begin{aligned} P(\mathbf{y}) &= \sum_{k=1}^K P(x = k)P(\mathbf{y}|x = k) \\ &= \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_k, \Sigma_k) \end{aligned}$$

E-step: Compute **responsibilities** for each data vec. $\mathbf{y}^{(i)}$

$$r_{ki} \equiv P(x = k|\mathbf{y}^{(i)}) = \frac{\pi_k \mathcal{N}(\mathbf{y}^{(i)}|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{\ell=1}^K \pi_{\ell} \mathcal{N}(\mathbf{y}^{(i)}|\boldsymbol{\mu}_{\ell}, \Sigma_{\ell})}$$

M-step: Estimate π_k , $\boldsymbol{\mu}_k$ and Σ_k using data **weighted by the responsibilities**.

The **k-means algorithm** for clustering is a special case of EM for mixture of Gaussians where $\Sigma_k = \lim_{\epsilon \rightarrow 0} \epsilon I$

Mixture of Factor Analysers

Assumes the model has several clusters (indexed by a discrete hidden variable x). Each cluster is modeled by a factor analyser:

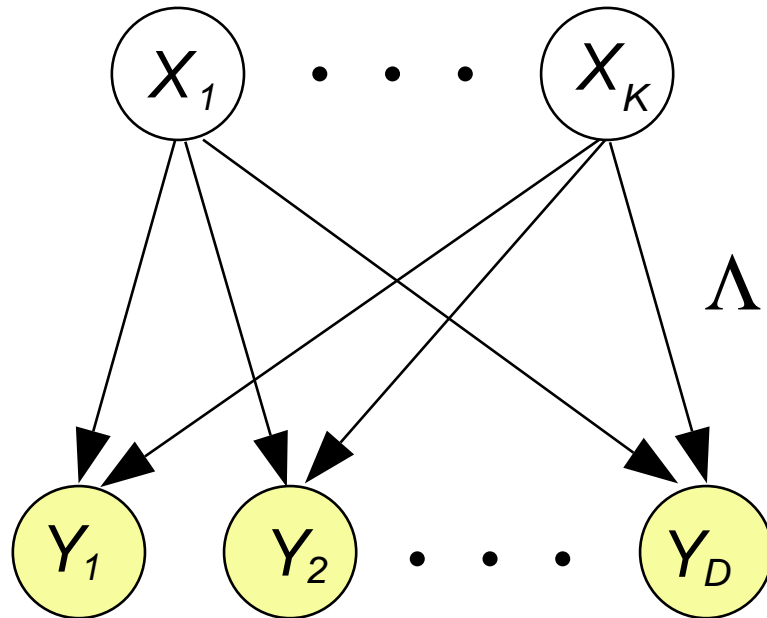
$$P(\mathbf{y}) = \sum_{m=1}^M P(x = m)P(\mathbf{y}|x = m)$$

where

$$P(\mathbf{y}|x = m) = \mathcal{N}(\boldsymbol{\mu}_m, \Lambda_m \Lambda_m^\top + \Psi)$$

- it's a way of fitting a mixture of Gaussians to high-dimensional data
- **clustering and dimensionality reduction**
- Bayesian learning can infer a posterior over the number of clusters and their intrinsic dimensionalities.

Independent Components Analysis



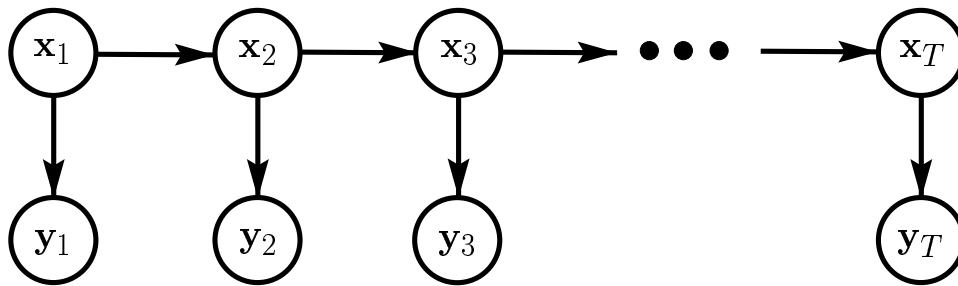
- $P(x_k)$ is non-Gaussian.
- Equivalently $P(x_k)$ is Gaussian and

$$y_d = \sum_{k=1}^K \Lambda_{dk} g(x_k) + \epsilon_d$$

where $g(\cdot)$ is a nonlinearity.

- For $K = D$, and observation noise assumed to be zero, inference and learning are easy (standard ICA). Many extensions possible (e.g. with noise \Rightarrow IFA).

Hidden Markov Models/Linear Dynamical Systems

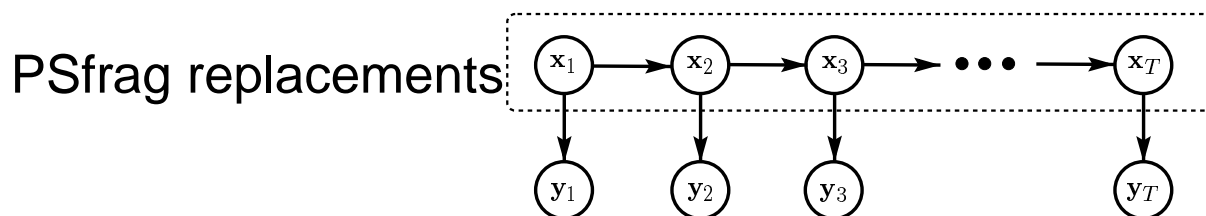


- Hidden states $\{x_t\}$, outputs $\{y_t\}$
Joint probability factorises:

PSfrag replacements

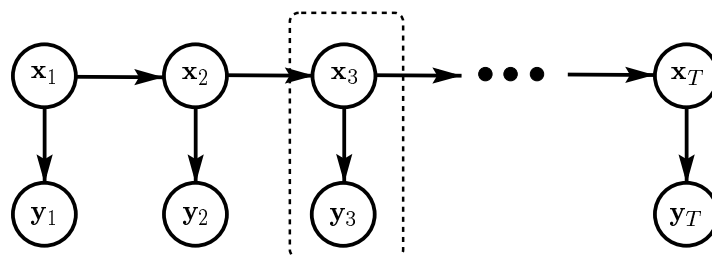
$$P(\{x\}, \{y\}) = \prod_{t=1}^T P(x_t | x_{t-1}) P(y_t | x_t)$$

- you can think of this as:
Markov chain with stochastic measurements.
Gauss-Markov process in a pancake.

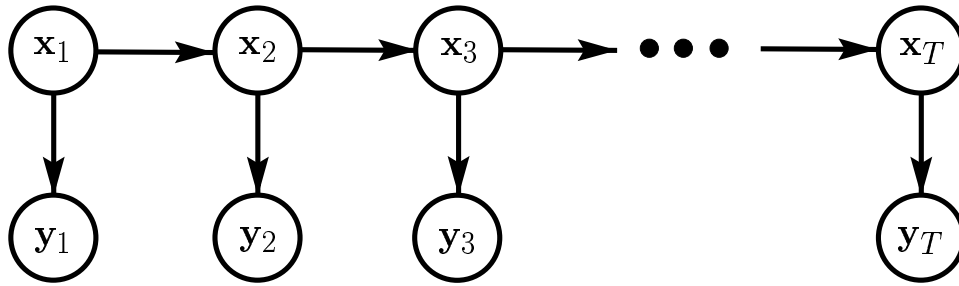


or

Mixture model with states coupled across time.
Factor analysis through time.



HMM Generative Model



- plain-vanilla HMM \equiv

“probabilistic function of a Markov chain”:

1. Use a 1st-order Markov chain to generate a hidden state sequence (path):

$$\begin{aligned} P(x_1 = j) &= \pi_j \\ P(x_{t+1} = j | x_t = i) &= T_{ij} \end{aligned}$$

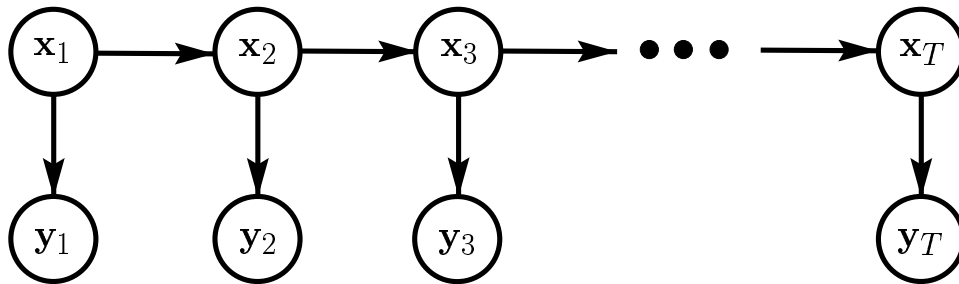
2. Use a set of output prob. distributions $A_j(\cdot)$ (one per state) to convert this state path into a sequence of observable symbols or vectors

$$P(\mathbf{y}_t = y | x_t = j) = A_j(y)$$

- Notes:

- Even though hidden state seq. is 1st-order Markov, the output process is not Markov of *any* order [ex. 11111211111311121111131 ...]
- Discrete state, discrete output models can approximate any continuous dynamics and observation mapping even if nonlinear; however lose ability to interpolate

LDS Generative Model



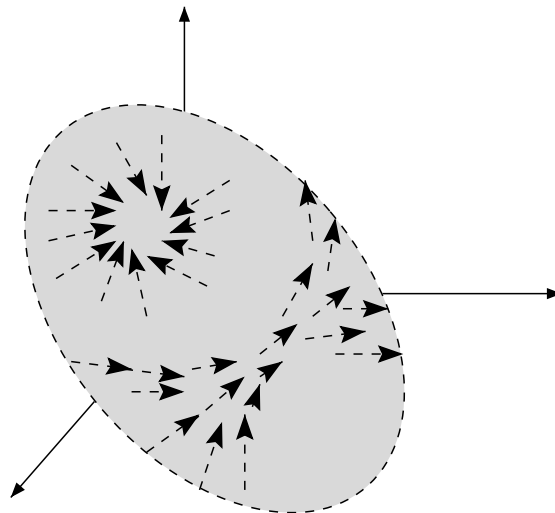
- Gauss-Markov continuous state process:

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{w}_t$$

observed through the “lens” of a noisy linear embedding:

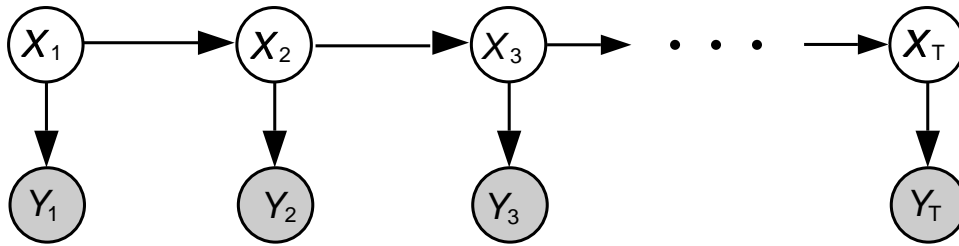
$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_t$$

- Noises \mathbf{w}_\bullet and \mathbf{v}_\bullet are temporally white and uncorrelated with everything else
- Think of this as “*matrix flow in a pancake*”



(Also called *state-space models*, *Kalman filter models*.)

EM applied to HMMs and LDSs



Given a sequence of T observations $\{Y_1, \dots, Y_T\}$

E-step. Compute the posterior probabilities:

- HMM: Forward-backward algorithm: $P(\{x\}|\{y\})$
- LDS: Kalman smoothing recursions: $P(\{\mathbf{x}\}|\{y\})$

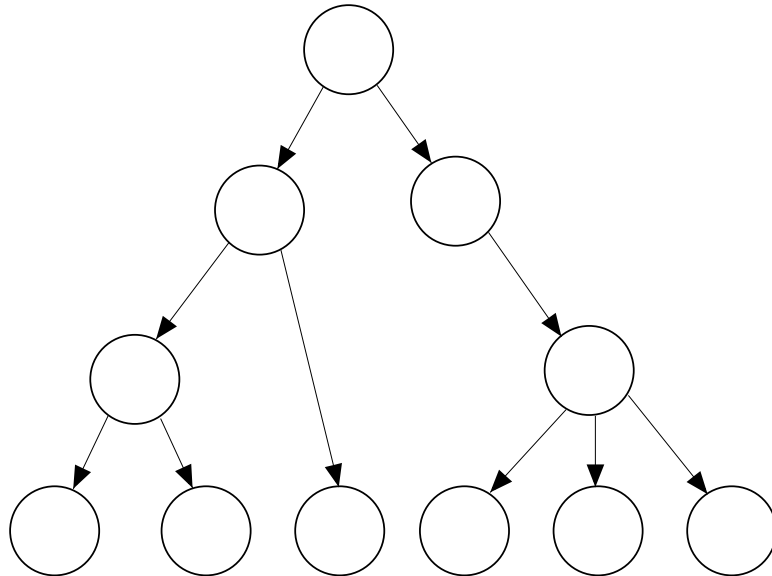
M-step. Re-estimate parameters:

- HMM: Count expected frequencies.
- LDS: Weighted linear regression.

Notes:

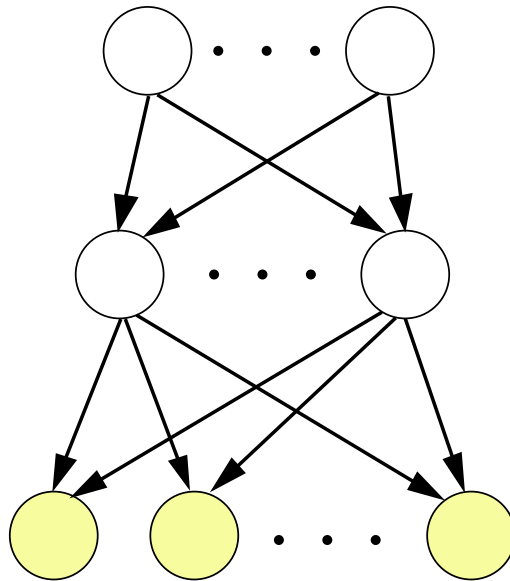
1. forward-backward and Kalman smoothing recursions are special cases of belief propagation.
2. online (causal) inference $P(x_t|\{Y_1, \dots, Y_t\})$ is done by the forward algorithm or the Kalman filter.
3. what sets the (arbitrary) scale of the hidden state?
Scale of Q (usually fixed at I).

Trees/Chains



- Tree-structured \equiv each node has exactly one parent.
- Discrete nodes or linear-Gaussian.
- *Hybrid systems* are possible: mixed discrete & continuous nodes. But, to remain tractable, discrete nodes must have discrete parents.
- Exact & efficient inference is done by belief propagation (generalised Kalman Smoothing).
- Can capture *multiscale structure* (e.g. images)

Polytrees/Layered Networks



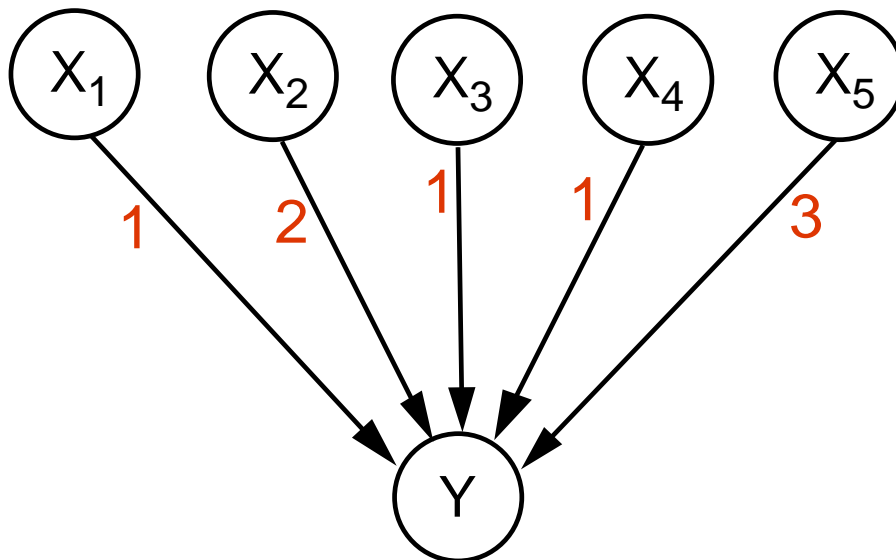
- more complex models for which junction-tree algorithm would be needed to do exact inference
- discrete/linear-Gaussian nodes are possible
- case of binary units is widely studied:
Sigmoid Belief Networks
- but usually intractable

Intractability

For many probabilistic models of interest, exact inference is not computationally feasible.

This occurs for two (main) reasons:

- distributions may have complicated forms (non-linearities in generative model)
- “explaining away” causes coupling from observations observing the value of a child induces dependencies amongst its parents (high order interactions)

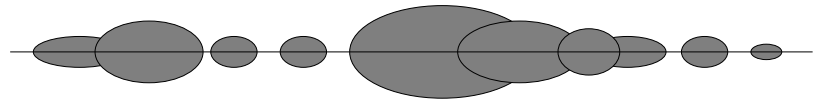


$$Y = X_1 + 2 X_2 + X_3 + X_4 + 3 X_5$$

We can still work with such models by using *approximate inference* techniques to estimate the latent variables.

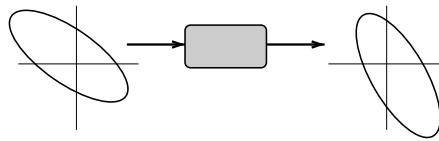
Approximate Inference

- **Sampling:**



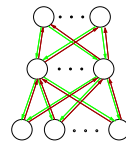
approximate true distribution over hidden variables with a few well chosen samples at certain values

- **Linearization:**



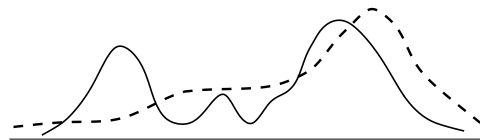
approximate the transformation on the hidden variables by one which keeps the form of the distribution closed (e.g. Gaussians and linear)

- **Recognition Models:**



approximate the true distribution with an approximation that can be computed easily/quickly by an explicit *bottom-up* inference model/network

- **Variational Methods:**



approximate the true distribution with an approximate form that is tractable; maximise a lower bound on the likelihood with respect to free parameters in this form

Sampling

Gibbs Sampling

To sample from a joint distribution $P(x_1, x_2, \dots, x_N)$:

Start from some **initial state** $\mathbf{x}^0 = (x_1^0, x_2^0, \dots, x_N^0)$:

Then **iterate** the following procedure:

- Pick x_1^{k+1} from $P(x_1 | x_2^k, x_3^k, x_4^k, \dots, x_N^k)$
- Pick x_2^{k+1} from $P(x_2 | x_1^{k+1}, x_3^k, x_4^k, \dots, x_N^k)$
- \vdots
- Pick x_N^{k+1} from $P(x_N | x_1^{k+1}, x_2^{k+1}, x_3^{k+1}, \dots, x_{N-1}^{k+1})$

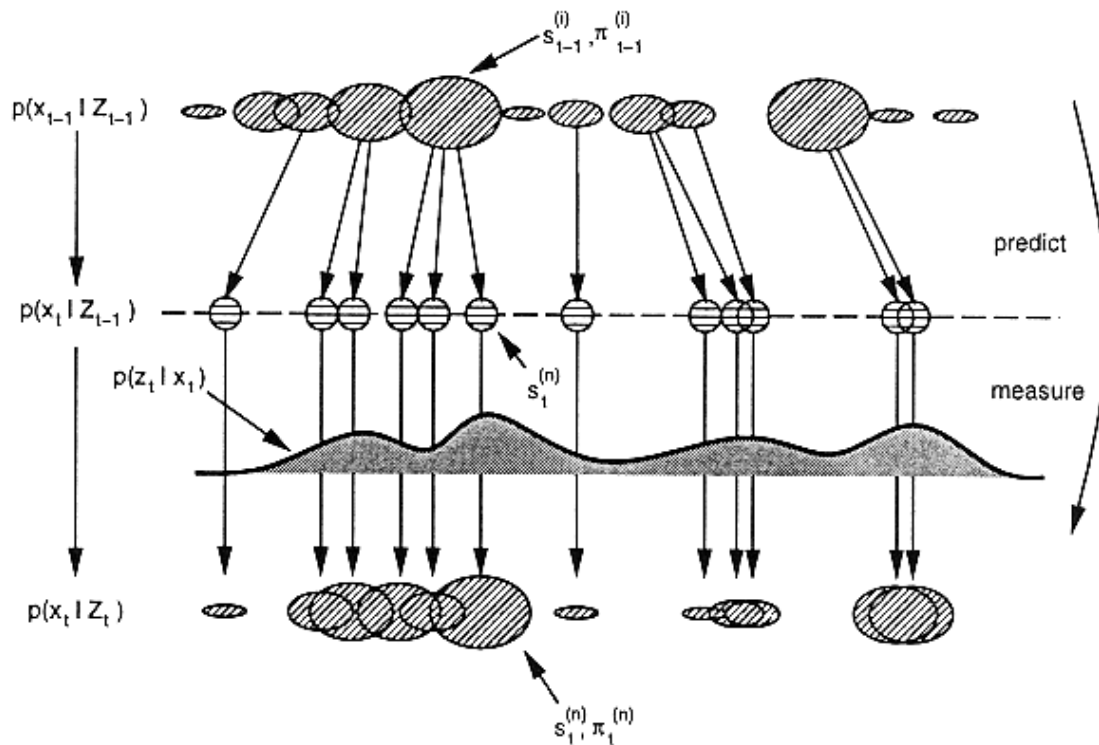
This procedure goes from $\mathbf{x}^k \rightarrow \mathbf{x}^{k+1}$, creating a *Markov chain* which converges to $P(\mathbf{x})$

Gibbs sampling can be used to estimate the expectations under the posterior distribution needed for E-step of EM.

It is just one of many Markov chain Monte Carlo (MCMC) methods. Easy to use if you can easily update subsets of latent variables at a time.

Key questions: how many iterations per sample?
how many samples?

Particle Filters



Assume you have n weighted samples $S_{t-1} = \{s_{t-1}^{(1)}, \dots, s_{t-1}^{(n)}\}$ from $P(\mathbf{x}_{t-1} | Z_{t-1})$, with normalised weights $\pi_{t-1}^{(i)}$

1. generate a new sample set S'_{t-1} by sampling with replacement from S_{t-1} with probabilities proportional to $\pi_{t-1}^{(i)}$
2. for each element of S' **predict**, using the stochastic dynamics, by sampling $s_t^{(i)}$ from $P(\mathbf{x}_t | \mathbf{x}_{t-1} = s_{t-1}^{(i)'})$
3. Using the measurement model, **weight** each new sample by

$$\pi_t^{(i)} = P(\mathbf{z}_t | \mathbf{x}_t = s_t^{(i)})$$

(the likelihoods) and normalise so that $\sum_i \pi_t^{(i)} = 1$.

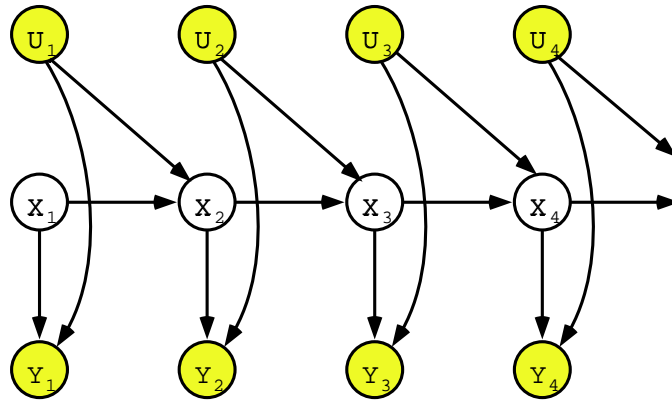
Samples need to be weighted by the ratio of the distribution we draw them from to the true posterior (this is *importance sampling*).

An easy way to do that is draw from prior and weight by likelihood.

(Also known as CONDENSATION algorithm.)

Linearization

Extended Kalman Filtering and Smoothing



$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{w}_t$$

$$\mathbf{y}_t = g(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{v}_t$$

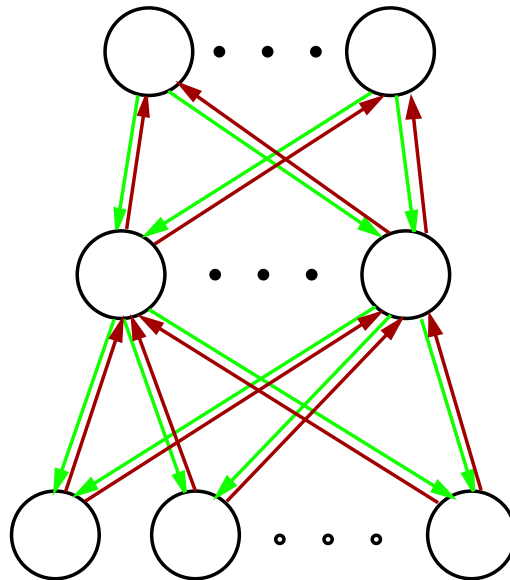
Linearise about the current estimate, i.e. given $\hat{\mathbf{x}}_t, \mathbf{u}_t$:

$$\mathbf{x}_{t+1} \approx f(\hat{\mathbf{x}}_t, \mathbf{u}_t) + \left. \frac{\partial f}{\partial \mathbf{x}_t} \right|_{\hat{\mathbf{x}}_t} (\mathbf{x}_t - \hat{\mathbf{x}}_t) + \mathbf{w}_t$$

$$\mathbf{y}_t \approx g(\hat{\mathbf{x}}_t, \mathbf{u}_t) + \left. \frac{\partial g}{\partial \mathbf{x}_t} \right|_{\hat{\mathbf{x}}_t} (\mathbf{x}_t - \hat{\mathbf{x}}_t) + \mathbf{v}_t$$

Run the **Kalman smoother** (belief propagation for linear-Gaussian systems) on the linearised system. This approximates non-Gaussian posterior by a Gaussian.

Recognition Models



- a function approximator is trained in a supervised way to recover the hidden causes (latent variables) from the observations
- this may take the form of explicit recognition network (e.g. Helmholtz machine) which mirrors the generative network (tractability at the cost of restricted approximating distribution)
- inference is done in a single *bottom-up* pass (no iteration required)

Variational Inference

Goal: maximise $\ln P(Y|\theta)$.

Any distribution $Q(X)$ over the hidden variables defines a **lower bound** on $\ln P(Y|\theta)$:

$$\ln P(Y|\theta) \geq \sum_X Q(X) \ln \frac{P(X, Y|\theta)}{Q(X)} = \mathcal{F}(Q, \theta)$$

Constrain $Q(X)$ to be of a particular **tractable** form (e.g. factorised) and maximise \mathcal{F} subject to this constraint

- **E-step:** Maximise \mathcal{F} w.r.t. Q with θ fixed, subject to the constraint on Q , equivalently minimise:

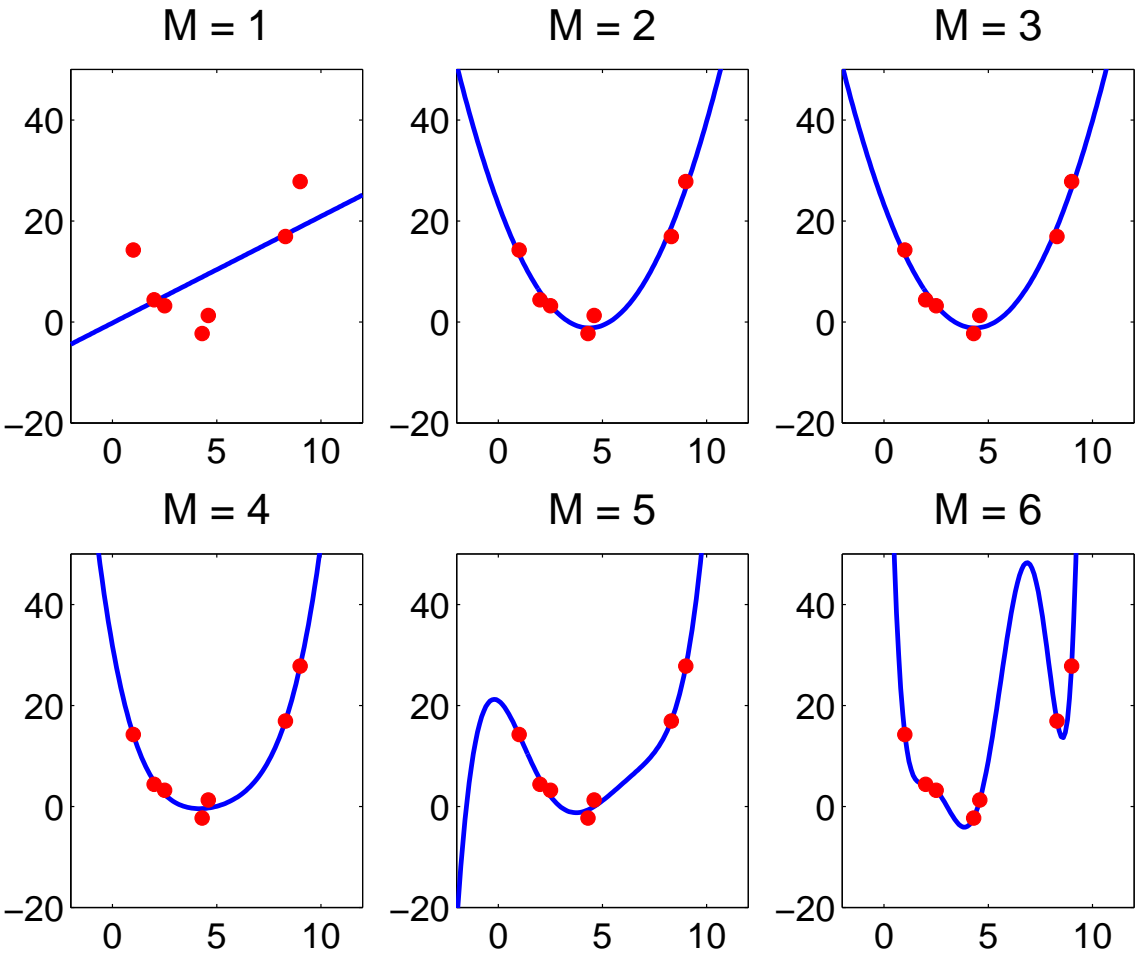
$$\begin{aligned} \ln P(Y|\theta) - \mathcal{F}(Q, \theta) &= \sum_X Q(X) \ln \frac{Q(X)}{P(X|Y)} \\ &= \text{KL}(Q||P) \end{aligned}$$

The inference step therefore tries to find Q closest to the exact posterior distribution.

- **M-step:** Maximise \mathcal{F} w.r.t. θ with Q fixed

(related to *mean-field approximations*)

Beyond Maximum Likelihood: Finding Model Structure and Avoiding Overfitting



Model Selection Questions

How many clusters in this data set?

What is the intrinsic dimensionality of the data?

What is the order of my autoregressive process?

How many sources in my ICA model?

How many states in my HMM?

Is this input relevant to predicting that output?

Is this relationship linear or nonlinear?

Bayesian Learning and Ockham's Razor

data Y , models $\mathcal{M}_1 \dots, \mathcal{M}_n$, parameter sets $\theta_1 \dots, \theta_n$
(let's ignore hidden variables X for the moment; they will just introduce another level of averaging/integration)

Total Evidence:

$$P(Y) = \sum_j P(Y|\mathcal{M}_j)P(\mathcal{M}_j)$$

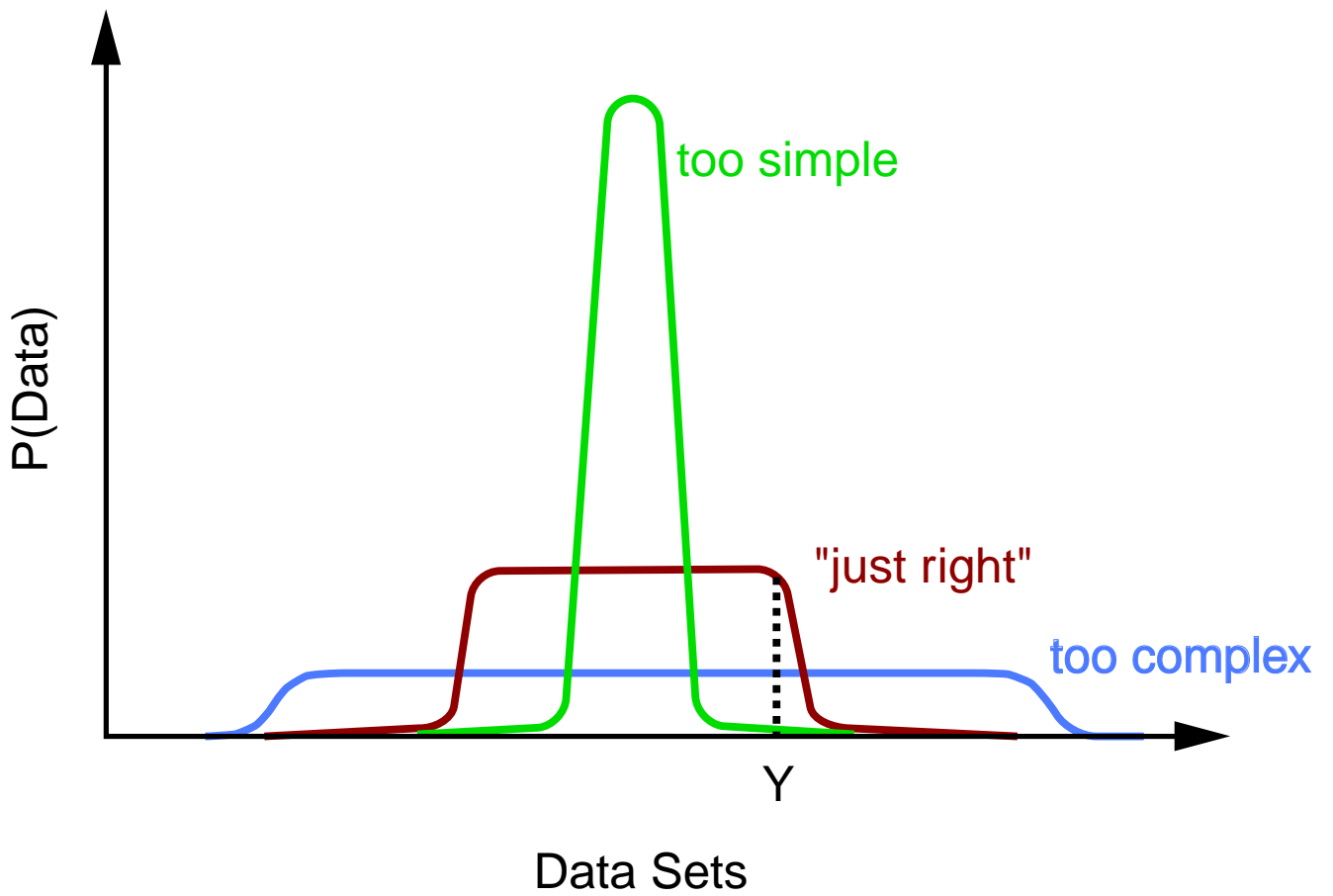
Model Selection:

$$P(\mathcal{M}_i|Y) = \frac{P(Y|\mathcal{M}_i)P(\mathcal{M}_i)}{P(Y)}$$

$$P(Y|\mathcal{M}_i) = \int_{\theta_i} P(Y|\theta_i, \mathcal{M}_i)P(\theta_i|\mathcal{M}_i)$$

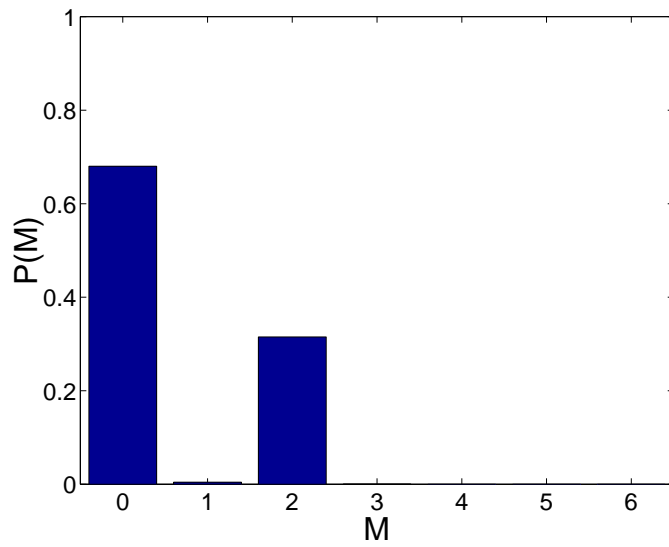
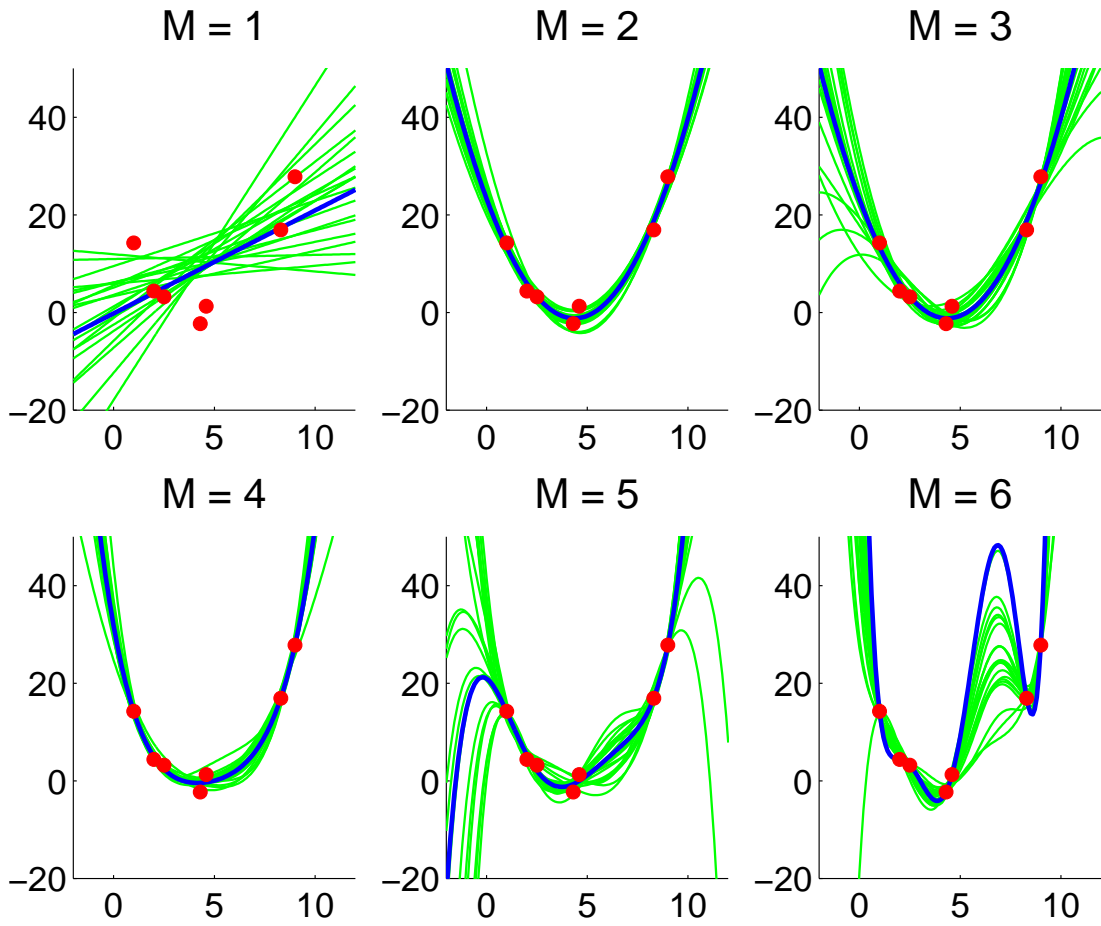
- $P(Y|\mathcal{M}_i)$ is the probability that randomly selected parameter values from model class \mathcal{M}_i would generate data set Y .
- Model classes that are **too simple** will be very unlikely to generate that particular data set.
- Model classes that are **too complex** can generate many possible data sets, so again, they are unlikely to generate that particular data set at random.

Ockham's Razor



(adapted from D.J.C. MacKay)

Overfitting



Practical Bayesian Approaches

- Laplace approximations
- Large sample approximations (e.g. BIC)
- Markov chain Monte Carlo methods
- Variational approximations

Laplace Approximation

data set Y , models $\mathcal{M}_1 \dots, \mathcal{M}_n$, parameter sets $\theta_1 \dots, \theta_n$

Model Selection:

$$P(\mathcal{M}_i|Y) \propto P(\mathcal{M}_i)P(Y|\mathcal{M}_i)$$

For large amounts of data (relative to number of parameters, d) the parameter posterior is approximately Gaussian around the MAP estimate $\hat{\theta}_i$:

$$P(\theta_i|Y, \mathcal{M}_i) \approx (2\pi)^{-\frac{d}{2}} |A|^{\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\theta_i - \hat{\theta}_i)^\top A (\theta_i - \hat{\theta}_i) \right\}$$

$$P(Y|\mathcal{M}_i) = \frac{P(\theta_i, Y|\mathcal{M}_i)}{P(\theta_i|Y, \mathcal{M}_i)}$$

Evaluating the above expression for $\ln P(Y|\mathcal{M}_i)$ at $\hat{\theta}_i$:

$$\ln P(Y|\mathcal{M}_i) \approx \ln P(\hat{\theta}_i|\mathcal{M}_i) + \ln P(Y|\hat{\theta}_i, \mathcal{M}_i) + \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |A|$$

where A is the negative Hessian of the log posterior.

This can be used for model selection.

(Note: A is size $d \times d$.)

BIC

The Bayesian Information Criterion (BIC) can be obtained from the Laplace approximation

$$\ln P(Y|\mathcal{M}_i) \approx \ln P(\hat{\theta}_i|\mathcal{M}_i) + \ln P(Y|\hat{\theta}_i, \mathcal{M}_i) + \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |A|$$

by taking the large sample limit:

$$\ln P(Y|\mathcal{M}_i) \approx \ln P(Y|\hat{\theta}_i, \mathcal{M}_i) + \frac{d}{2} \ln N$$

where N is the number of data points.

Properties:

- Quick and easy to compute
- It does not depend on the prior
- We can use the ML estimate of θ instead of the MAP estimate
- It assumes that in the large sample limit, all the parameters are well-determined (i.e. the model is **identifiable**; otherwise, d should be the number of well-determined parameters)
- It is equivalent to the MDL criterion

MCMC

Assume a model with parameters θ , hidden variables X and observable variables Y

Goal: to obtain samples from the (intractable) posterior distribution over the parameters, $P(\theta|Y)$

Approach: to sample from a Markov chain whose equilibrium distribution is $P(\theta|Y)$.

One such simple Markov chain can be obtained by Gibbs sampling, which alternates between:

- **Step A:** Sample from parameters given hidden variables and observables: $\theta \sim P(\theta|X, Y)$
- **Step B:** Sample from hidden variables given parameters and observables: $X \sim P(X|\theta, Y)$

Note the similarity to the EM algorithm!

Variational Bayesian Learning

Lower bound the evidence:

$$\mathcal{L} \equiv \ln P(Y)$$

$$= \ln \int P(Y, \theta) d\theta$$

$$\geq \int Q(\theta) \ln \frac{P(Y, \theta)}{Q(\theta)} d\theta$$

$$= \int Q(\theta) \left[\ln P(Y|\theta) + \ln \frac{P(\theta)}{Q(\theta)} \right] d\theta$$

$$\geq \int Q(\theta) \left[\left[\int Q(X) \ln \frac{P(X, Y|\theta)}{Q(X)} dX \right] + \ln \frac{P(\theta)}{Q(\theta)} \right] d\theta$$

$$\equiv \mathcal{F}(Q(\theta), Q(X))$$

Assumes the factorisation:

$$P(\theta, X|Y) \approx Q(\theta)Q(X)$$

(also known as “ensemble learning”)

Variational Bayesian Learning

EM-like optimisation:

“**E-step**”: Maximise \mathcal{F} w.r.t. $Q(X)$ with $Q(\theta)$ fixed

“**M-step**”: Maximise \mathcal{F} w.r.t. $Q(\theta)$ with $Q(X)$ fixed

Finds an approximation to the posterior over parameters $Q(\theta) \approx P(\theta|Y)$ and hidden variables $Q(X) \approx P(X|Y)$

- Maximises a lower bound on the log evidence
- Convergence can be assessed by monitoring \mathcal{F}
- Global approximation
- \mathcal{F} transparently incorporates model complexity penalty (i.e. coding cost for all the parameters of the model) so it can be compared across models
- Optimal form of $Q(\theta)$ falls out of free-form variational optimisation (i.e. not assumed to be Gaussian)
- Often simple modification of the EM algorithm

Summary

- Why probabilistic models?
- Factor analysis and beyond
- Inference and the EM algorithm
- Generative Model for Generative Models
- A few models in detail
- Approximate inference
- Practical Bayesian approaches

Appendix

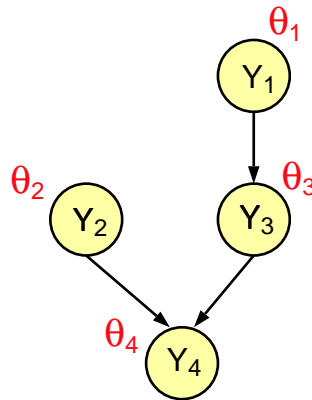
Desiderata (or Axioms) for Computing Plausibilities

Paraphrased from E.T. Jaynes, using the notation $p(A|B)$ is the plausibility of statement A given that you know that statement B is true.

- Degrees of plausibility are represented by real numbers
- Qualitative correspondence with common sense, e.g.
 - If $p(A|C') > p(A|C)$ but $p(B|A\&C') = p(B|A\&C)$ then $p(A\&B|C') \geq p(A\&B|C)$
- Consistency:
 - If a conclusion can be reasoned in more than one way, then every possible way must lead to the same result.
 - All available evidence should be taken into account when inferring a plausibility.
 - Equivalent states of knowledge should be represented with equivalent plausibility statements.

Accepting these desiderata leads to **Bayes Rule** being the only way to manipulate plausibilities.

Learning with Complete Data



Assume a data set of i.i.d. observations $\mathcal{D} = \{Y^{(1)}, \dots, Y^{(n)}\}$ and a parameter vector θ .

Goal is to **maximise likelihood**: $P(\mathcal{D}|\theta) = \prod_{i=1}^n P(Y^{(i)}|\theta)$

Equivalently, maximise log likelihood:

$$\mathcal{L}(\theta) = \sum_{i=1}^n \ln P(Y^{(i)}|\theta)$$

Using the **graphical model factorisation**:

$$P(Y^{(i)}|\theta) = \prod_j P(Y_j^{(i)}|Y_{\text{pa}(j)}^{(i)}, \theta_j)$$

So:
$$\mathcal{L}(\theta) = \sum_{i=1}^n \sum_j \ln P(Y_j^{(i)}|Y_{\text{pa}(j)}^{(i)}, \theta_j)$$

In other words, the parameter estimation problem breaks into many independent, local problems (uncoupled).

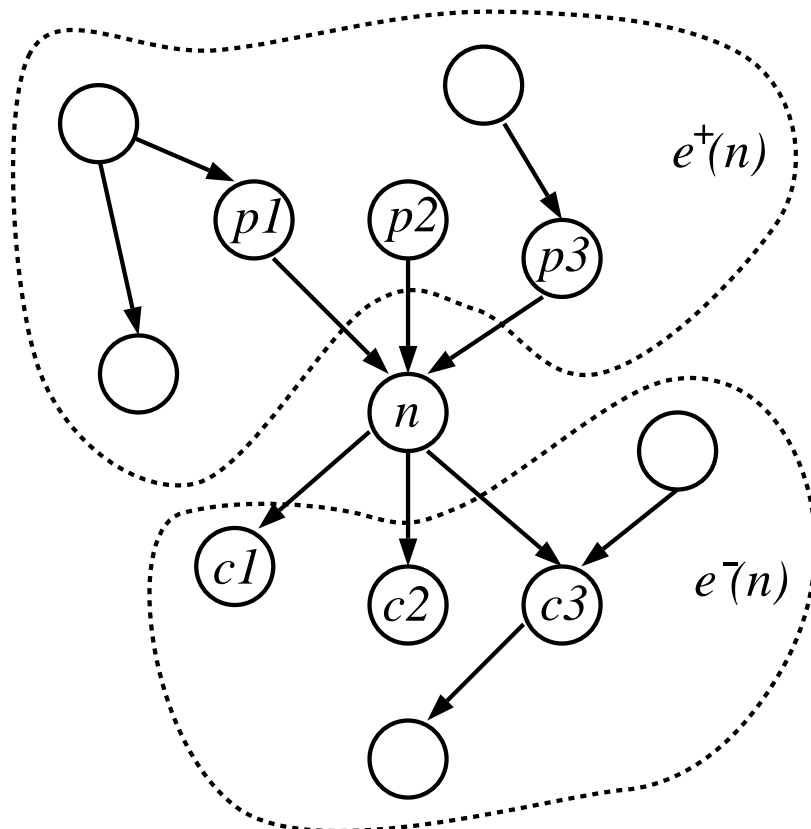
Building a Junction Tree

- Start with the recursive factorization from the DAG:

$$P(\mathbf{X}) = \prod_i P(\mathbf{x}_i | \text{pa}(\mathbf{x}_i))$$

- Convert these local conditional probabilities into *potential functions* over both \mathbf{x}_i and all its parents.
- This is called *moralising* the DAG since the parents get connected. Now the product of the potential functions gives the correct joint
- When evidence is absorbed, potential functions must agree on the prob. of shared variables: consistency.
- This can be achieved by passing messages between potential functions to do local marginalising and rescaling.
- Problem: a variable may appear in two non-neighbouring cliques. To avoid this we need to *triangulate* the original graph to give the potential functions the running intersection property. Now local consistency will imply global consistency.

Bayesian Networks: Belief Propagation



Each node n divides the *evidence*, e , in the graph into two disjoint sets: $e^+(n)$ and $e^-(n)$

Assume a node n with parents $\{p_1, \dots, p_k\}$ and $\{c_1, \dots, c_\ell\}$

$$\begin{aligned}
 P(n|e) &\propto \left[\sum_{\{p_1, \dots, p_k\}} P(n|p_1, \dots, p_k) \prod_{i=1}^k P(p_i|e^+(p_i)) \right] \times \\
 &\quad \prod_{j=1}^{\ell} P(c_j, e^-(c_j)|n) \\
 &\propto P(n|e^+(n))P(e^-(n)|n)
 \end{aligned}$$

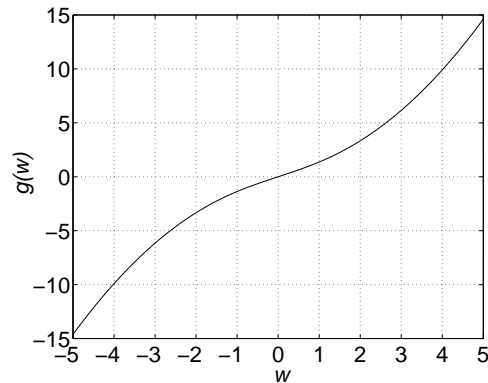
ICA Nonlinearity

Generative model:

$$\mathbf{x} = g(\mathbf{w})$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{v}$$

where \mathbf{w} and \mathbf{v} are zero-mean Gaussian noises with covariances I and \mathbf{R} respectively.



The density of x can be written in terms of $g(\cdot)$,

$$p_x(x) = \frac{\mathcal{N}(0, 1) |g^{-1}(x)|}{|g'(g^{-1}(x))|}$$

For example, if $p_x(x) = \frac{1}{\pi \cosh(x)}$ we find that setting:

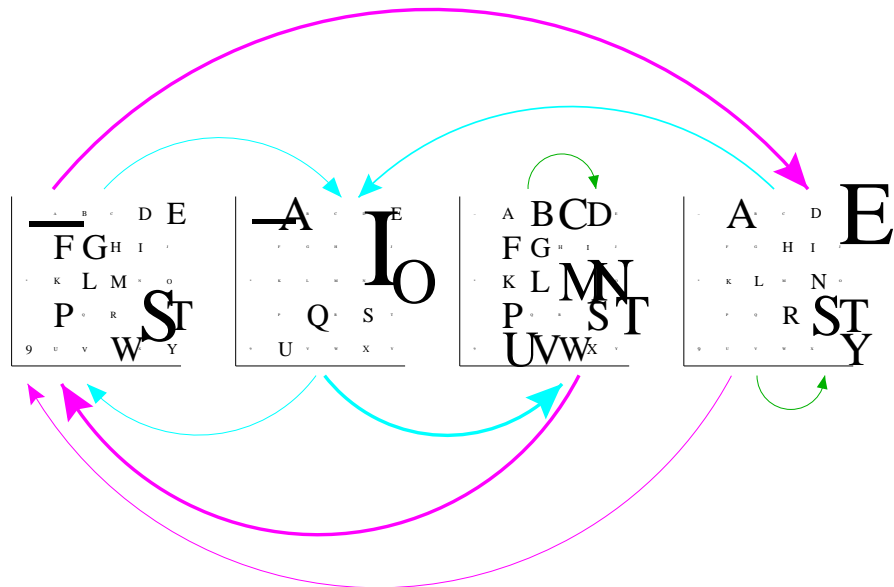
$$g(w) = \ln \left(\tan \left(\frac{\pi}{4} \left(1 + \operatorname{erf}(w/\sqrt{2}) \right) \right) \right)$$

generates vectors \mathbf{x} in which each component is distributed exactly according to $1/(\pi \cosh(x))$.

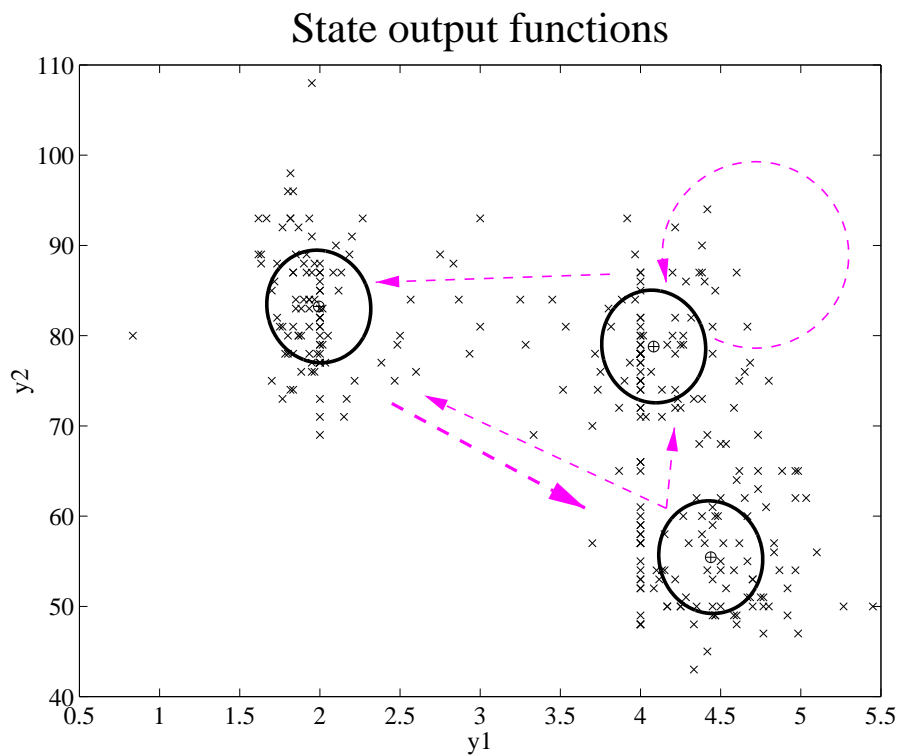
So, ICA can be seen either as a linear generative model with non-Gaussian priors for the hidden variables, or as a nonlinear generative model with Gaussian priors for the hidden variables.

HMM Example

- Character sequences (discrete outputs)

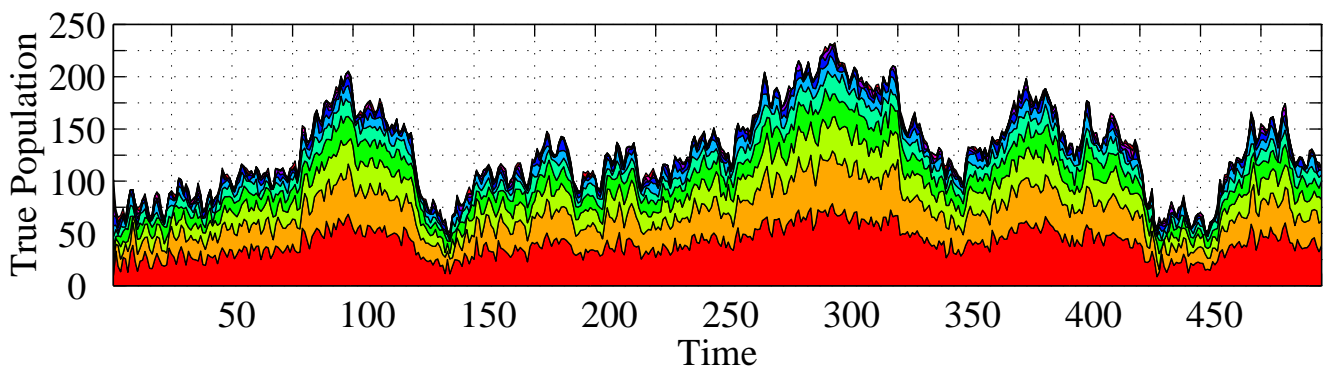
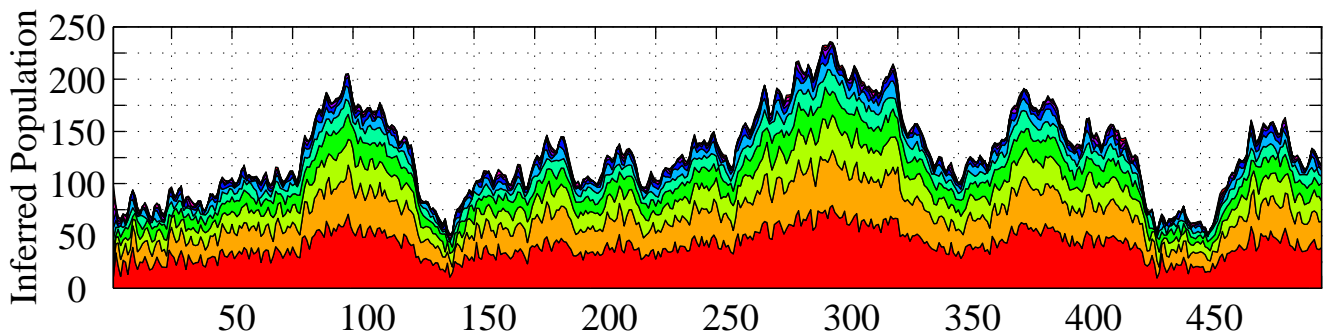
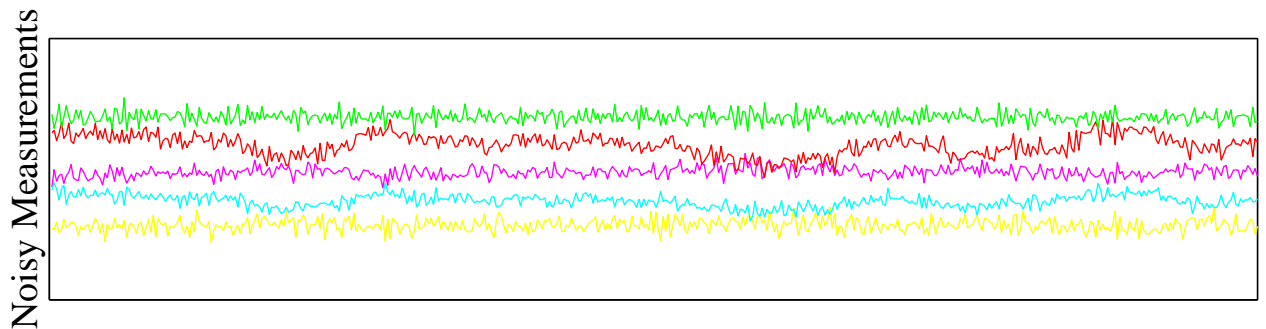


- Geyser data (continuous outputs)



LDS Example

- Population model:
 - state \Leftrightarrow population histogram
 - first row of $A \Leftrightarrow$ birthrates
 - subdiagonal of $A \Leftrightarrow$ 1-deathrates
 - $Q \Leftrightarrow$ immigration/emmigration
 - $C \Leftrightarrow$ noisy indicators



Viterbi Decoding

- The numbers $\gamma_j(t)$ in forward-backward gave the posterior probability distribution over all states at any time.
- By choosing the state $\gamma_*(t)$ with the largest probability at each time, we can make a “best” state path. This is the path with the *maximum expected number of correct states*.
- But it *is not* the single path with the highest likelihood of generating the data.
In fact it may be a path of probability zero!
- To find the *single best path*, we do *Viterbi decoding* which is just Bellman’s dynamic programming algorithm applied to this problem.
- The recursions look the same, except with \max instead of \sum .
- There is also a modified Baum-Welch training based on the Viterbi decode.

HMM Pseudocode

- Forward-backward including scaling tricks

$$q_j(t) = A_j(\mathbf{y}_t)$$

$$\alpha(1) = \pi \cdot q(1) \quad \rho(1) = \sum \alpha(1) \quad \alpha(1) = \alpha(1)/\rho(1)$$

$$\alpha(t) = (T' * \alpha(t-1)) \cdot q(t) \quad \rho(t) = \sum \alpha(t) \quad \alpha(t) = \alpha(t)/\rho(t) \quad [t = 2 : \tau]$$

$$\beta(\tau) = 1$$

$$\beta(t) = T * (\beta(t+1) \cdot q(t+1)) / \rho(t+1) \quad [t = (\tau - 1) : 1]$$

$$\xi = 0$$

$$\xi = \xi + T * (\alpha(t) * (\beta(t+1) \cdot q(t+1)))' / \rho(t+1) \quad [t = 1 : (\tau - 1)]$$

$$\gamma = (\alpha \cdot \beta)$$

$$\log P(\mathbf{y}_1^\tau) = \sum \log(\rho(t))$$

- Baum-Welch parameter updates

$$\delta_j = 0 \quad \hat{T}_{ij} = 0 \quad \hat{\pi} = 0 \quad \hat{A} = 0$$

for each sequence, run forward backward to get γ and ξ , then

$$\hat{T} = \hat{T} + \xi \quad \hat{\pi} = \hat{\pi} + \gamma(1) \quad \delta = \delta + \sum_t \gamma(t)$$

$$\hat{A}_j(\mathbf{y}) = \sum_{t|\mathbf{y}_t=y} \gamma_j(t) \quad \text{or} \quad \hat{A} = \hat{A} + \sum_t \mathbf{y}_t \gamma(t)$$

$$\hat{T}_{ij} = \hat{T}_{ij} / \sum_k \hat{T}_{ik} \quad \hat{\pi} = \hat{\pi} / \sum \hat{\pi} \quad \hat{A}_j = \hat{A}_j / \delta_j$$

LDS Pseudocode

- Kalman filter/smoothen including scaling tricks

$$\mathbf{x}^+ = \mathbf{x}_0 \qquad \mathbf{V}^+ = \mathbf{V}_0$$

$$\rho_t = \mathcal{N}(\mathbf{C}\mathbf{x}^+, \mathbf{C}\mathbf{V}^+\mathbf{C}' + \mathbf{R}) |_{\mathbf{y}_t} \qquad [t = 1 : \tau]$$

$$\mathbf{K} = \mathbf{V}^+\mathbf{C}'(\mathbf{C}\mathbf{V}^+\mathbf{C}' + \mathbf{R})^{-1}$$

$$\mathbf{x}_t = \mathbf{x}^+ + \mathbf{K}(\mathbf{y}_t - \mathbf{C}\mathbf{x}^+)$$

$$\mathbf{V}_t = (\mathbf{I} - \mathbf{K}\mathbf{C})\mathbf{V}^+$$

$$\mathbf{x}^+ = \mathbf{A}\mathbf{x}_t$$

$$\mathbf{V}^+ = \mathbf{A}\mathbf{V}_t\mathbf{A}' + \mathbf{Q}$$

$$\hat{\mathbf{x}}_\tau = \mathbf{x}_\tau$$

$$\hat{\mathbf{V}}_\tau = \mathbf{V}_\tau$$

$$\mathbf{V}^+ = \mathbf{A}\mathbf{V}_t\mathbf{A}' + \mathbf{Q} \qquad [t = (\tau - 1) : 1]$$

$$\mathbf{J} = \mathbf{V}_t\mathbf{A}'(\mathbf{V}^+)^{-1}$$

$$\hat{\mathbf{x}}_t = \mathbf{x}_t + \mathbf{J}(\hat{\mathbf{x}}_{t+1} - \mathbf{A}\mathbf{x}_t)$$

$$\hat{\mathbf{V}}_t = \mathbf{V}_t + \mathbf{J}(\hat{\mathbf{V}}_{t+1} - \mathbf{V}^+)\mathbf{J}'$$

$$\log P(\mathbf{y}_1^\tau) = \sum \log(\rho(t))$$

- EM parameter updates

$$\delta = 0 \quad \beta = 0 \quad \gamma = 0 \quad \alpha = 0 \quad \mathbf{x}_0 = 0$$

for each sequence, run Kalman smoother to get \mathbf{x}_t , $\hat{\mathbf{x}}_t$, \mathbf{V}_t and $\hat{\mathbf{V}}_t$

$$\mathbf{x}_0 = \mathbf{x}_0 + \hat{\mathbf{x}}_1/N \qquad \hat{\mathbf{V}}_{t-1}^t = \mathbf{V}_{t-1}\mathbf{A}'(\mathbf{A}\mathbf{V}_{t-1}\mathbf{A}' + \mathbf{Q})^{-1}\hat{\mathbf{V}}_t$$

$$\delta = \delta + \sum_t \mathbf{y}_t \hat{\mathbf{x}}_t' \qquad \beta = \beta + \sum_{t=2}^{\tau} \hat{\mathbf{x}}_t \hat{\mathbf{x}}_{t-1}' + \hat{\mathbf{V}}_t^{t-1} \qquad \alpha = \alpha + \sum_t \mathbf{y}_t \mathbf{y}_t'$$

$$\gamma = \gamma + \sum_t \hat{\mathbf{x}}_t \hat{\mathbf{x}}_t' + \hat{\mathbf{V}}_t \qquad \gamma_\tau = \gamma_\tau + \hat{\mathbf{x}}_\tau \hat{\mathbf{x}}_\tau' + \hat{\mathbf{V}}_\tau \qquad \gamma_1 = \gamma_1 + \hat{\mathbf{x}}_1 \hat{\mathbf{x}}_1' + \hat{\mathbf{V}}_1$$

$$\mathbf{C} = \delta \gamma^{-1}$$

$$\mathbf{A} = \beta (\gamma - \gamma_\tau)^{-1}$$

$$\mathbf{R} = (\alpha - \mathbf{C}\delta') / \tau_N$$

$$\mathbf{Q} = (\gamma - \gamma_1 - \mathbf{A}\beta') / (\tau_N - N)$$

Selected References

Graphical Models and the EM algorithm:

- ★ *Learning in Graphical Models* (1998) Edited by M.I. Jordan. Dordrecht: Kluwer Academic Press. Also available from MIT Press (paperback).
- **Motivation for Bayes Rule:** "Probability Theory - the Logic of Science," E.T.Jaynes, <http://www.math.albany.edu:8008/JaynesBook.html>
- **EM:**

Baum, L., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41:164–171;

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society Series B*, 39:1–38;

Neal, R. M. and Hinton, G. E. (1998). A new view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*.
- **Factor Analysis and PCA:**

Mardia, K.V., Kent, J.T., and Bibby J.M. (1979) *Multivariate Analysis* Academic Press, London

Roweis, S. T. (1998). EM algorithms for PCA and SPCA. NIPS98

Ghahramani, Z. and Hinton, G. E. (1996). The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1 [<http://www.gatsby.ucl.ac.uk/~zoubin/papers/tr-96-1.ps.gz>] Department of Computer Science, University of Toronto.

Tipping, M. and Bishop, C. (1999). Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):435–474.

- **Belief propagation:**

Kim, J.H. and Pearl, J. (1983) A computational model for causal and diagnostic reasoning in inference systems.

In *Proc of the Eighth International Joint Conference on AI*: 190-193;

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.

- **Junction tree:** Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statistical Society B*, pages 157–224.

Other graphical models:

- ★ Roweis, S.T and Ghahramani, Z. (1999) A unifying review of linear Gaussian models. *Neural Computation* **11**(2): 305–345.

- **ICA:**

Comon, P. (1994). Independent component analysis: A new concept. *Signal Processing*, 36:287–314;

Baram, Y. and Roth, Z. (1994) Density shaping by neural networks with application to classification, estimation and forecasting. Technical Report TR-CIS-94-20, Center for Intelligent Systems, Technion, Israel Institute for Technology, Haifa, Israel.

Bell, A. J. and Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159.

- **Trees:**

Chou, K.C., Willsky, A.S., Benveniste, A. (1994) Multiscale recursive estimation, data fusion, and regularization. *IEEE Trans. Automat. Control* **39**:464-478;

Bouman, C. and Shapiro, M. (1994). A multiscale random field model for Bayesian segmentation. *IEEE Transactions on Image Processing* 3(2):162–177.

- **Sigmoid Belief Networks:**

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113.

Saul, L.K. and Jordan, M.I. (1999) Attractor dynamics in feedforward neural networks. To appear in *Neural Computation*.

Approximate Inference & Learning

- **MCMC:** Neal, R. M. (1993). Probabilistic inference using Markov chain monte carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto.

- **Particle Filters:**

Gordon, N.J. , Salmond, D.J. and Smith, A.F.M. (1993)
A novel approach to nonlinear/non-Gaussian Bayesian state estimation
IEE Proceedings on Radar, Sonar and Navigation **140**(2):107-113;

Kitagawa, G. (1996) Monte Carlo filter and smoother for non-Gaussian nonlinear state space models,
Journal of Computational and Graphical Statistics **5**(1):1–25.

Isard, M. and Blake, A. (1998) CONDENSATION – conditional density propagation for visual tracking *Int. J. Computer Vision* **29** 1:5–28.

- **Extended Kalman Filtering:** Goodwin, G. and Sin, K. (1984). *Adaptive filtering prediction and control*. Prentice-Hall.

- **Recognition Models:**

Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. (1995). The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1158–1161.

Dayan, P., Hinton, G. E., Neal, R., and Zemel, R. S. (1995)
The Helmholtz Machine . *Neural Computation*, 7, 1022-1037.

- **Ockham's Razor and Laplace Approximation:**

Jefferys, W.H., Berger, J.O. (1992)
Ockham's Razor and Bayesian Analysis. *American Scientist* **80**:64-72;

Mackay, D.J.C. (1995) Probable Networks and Plausible Predictions - A Review of Practical Bayesian Methods for Supervised Neural Networks. *Network: Computation in Neural Systems*. **6** 469-505

- **BIC:** Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics* **6**:461-464.
- **MDL:** Wallace, C.S. and Freeman, P.R. (1987) Estimation and inference by compact coding. *J. of the Royal Stat. Society series B* **49**(3): 240-265; J. Rissanen (1987)

- **Bayesian Gibbs sampling software:**
The BUGS Project – <http://www.iph.cam.ac.uk/bugs/>

- **Variational Bayesian Learning:**

Hinton, G. E. and van Camp, D. (1993) Keeping Neural Networks Simple by Minimizing the Description Length of the Weights.
In *Sixth ACM Conference on Computational Learning Theory*, Santa Cruz.

Waterhouse, S., MacKay, D., and Robinson, T. (1995)
Bayesian methods for Mixtures of Experts. NIPS95.
(See also several unpublished papers by MacKay).

Bishop, C.M. (1999) Variational PCA. In *Proc. Ninth Int. Conf. on Artificial Neural Networks ICANN99* 1:509 - 514.

Attias, H. (1999). Inferring parameters and structure of latent variable models by variational Bayes. In *Proc. 15th Conf. on Uncertainty in Artificial Intelligence*;

Ghahramani, Z. and Beal, M.J. (1999)
Variational Bayesian Inference for Mixture of Factor Analysers. NIPS99;

<http://www.gatsby.ucl.ac.uk/>