Unsupervised Learning

Lecture 6: Hierarchical and Nonlinear Models

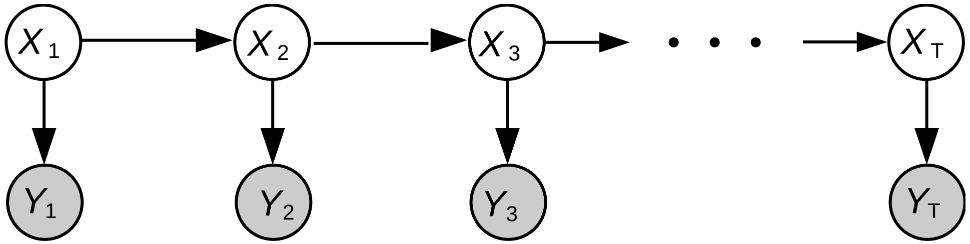
Zoubin Ghahramani
zoubin@gatsby.ucl.ac.uk

Gatsby Computational Neuroscience Unit, and
MSc in Intelligent Systems, Dept Computer Science
University College London

Autumn 2003

Why we need nonlinearities

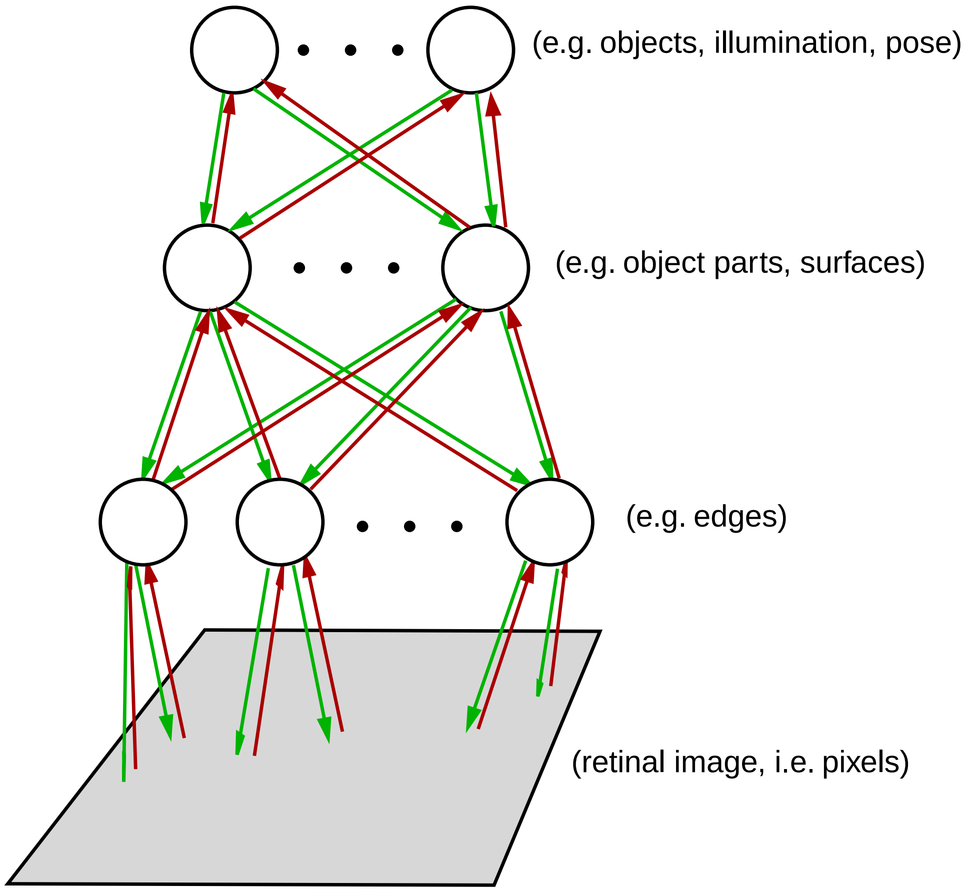
Linear systems have limited modelling capability.



Consider linear-Gaussian state-space models.
Only certain dynamics can be modelled.

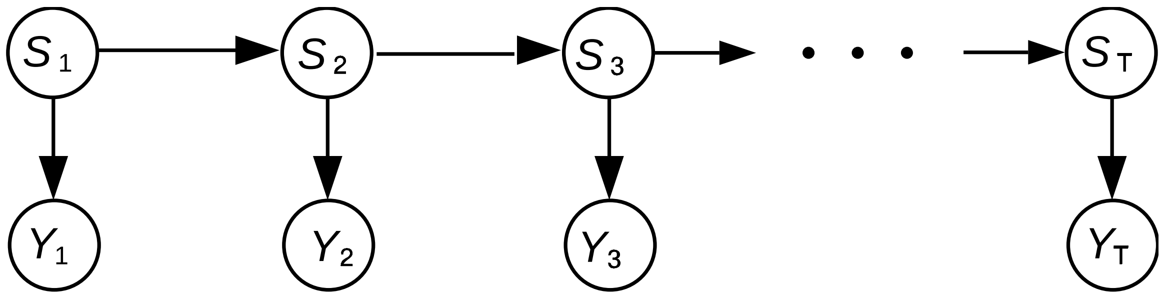
Why we need hierarchical models

Many generative processes can be naturally described at different levels of detail.



Biology seems to have developed hierarchical representations.

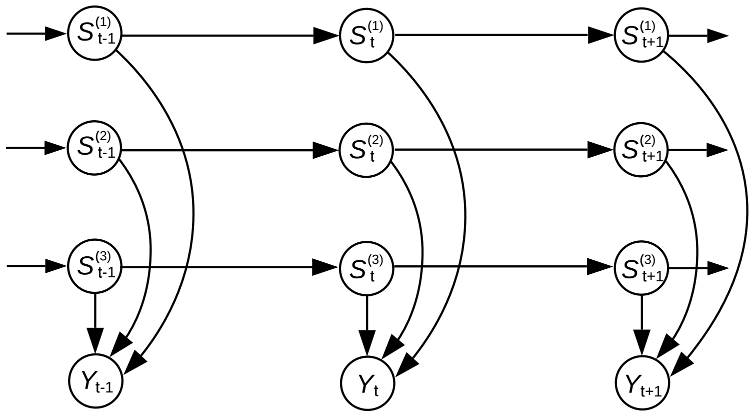
Why we need distributed representations

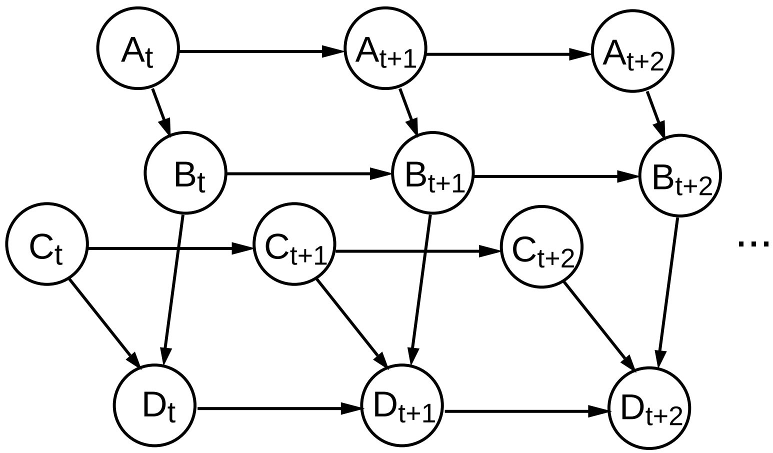


Consider a hidden Markov model.

To capture N bits of information about the history of the sequence, a HMM requires $K = 2^N$ states!

Factorial Hidden Markov Models and Dynamic Bayesian Networks

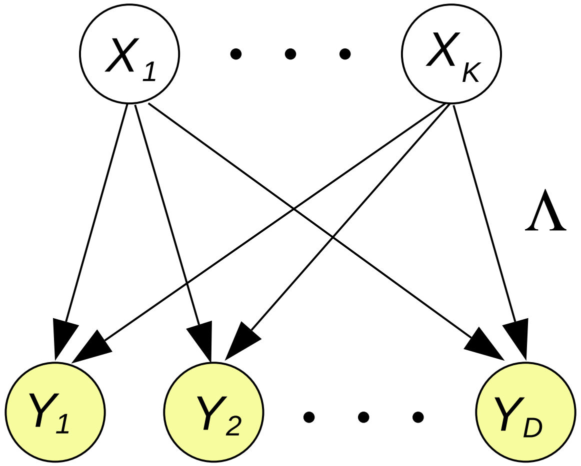




These are hidden Markov models with many state variables (i.e. a distributed representation of the state).

Blind Source Separation

Independent Components Analysis



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

- Equivalently $P(x_k)$ is Gaussian, with a nonlinearity $g(\cdot)$:

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

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

ICA Nonlinearity

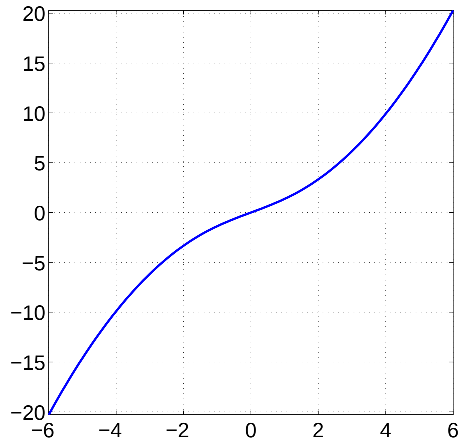
Generative model: $\mathbf{x} = g(\mathbf{w}) \qquad \mathbf{y} = \Lambda\mathbf{x} + \mathbf{v}$
where $\mathbf{w}$ and $\mathbf{v}$ are zero-mean Gaussian noises with covariances I and R respectively.
The density of x can be written in terms of $g(\cdot)$,

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

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

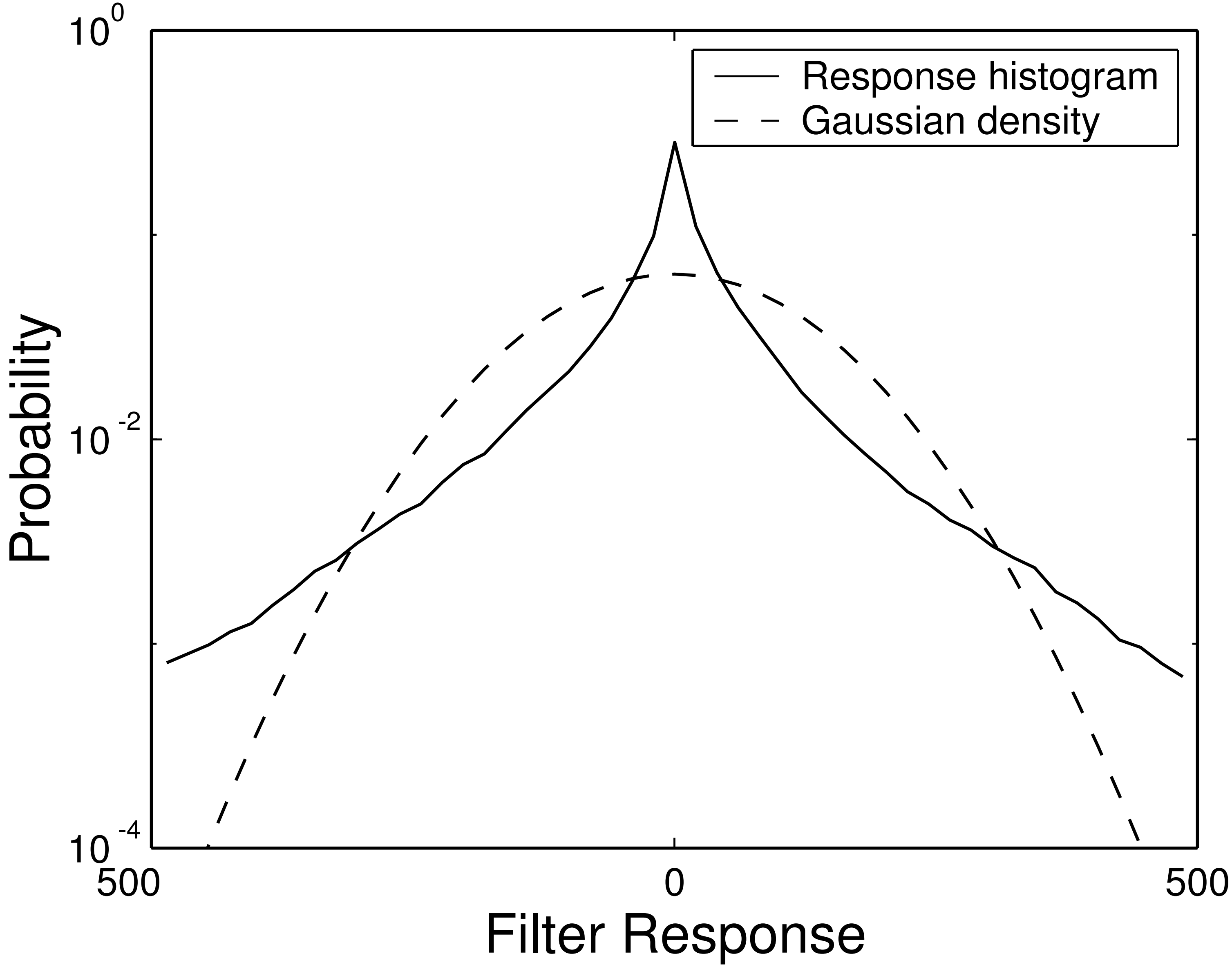
$$g(w) = \ln\left(\tan\left(\frac{\pi}{4}\left(1 + \mathrm{erf}(w/\sqrt{2})\right)\right)\right)$$

generates vectors $\mathbf{x}$ in which each component is distributed 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.

Natural Scenes and Sounds



Natural Scenes

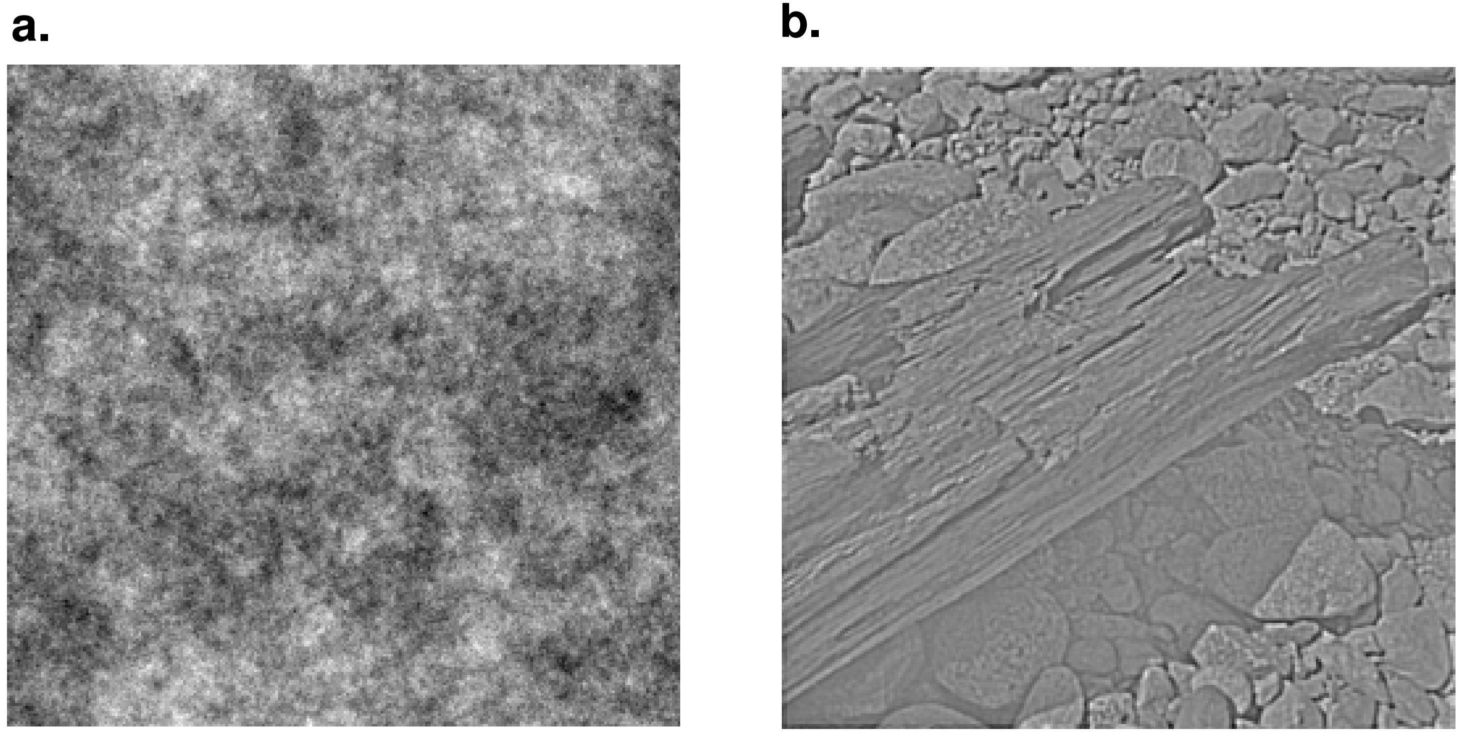


Figure 5: **(a)** Sample of $1/f$ Gaussian noise; **(b)** whitened natural image.

Natural Scenes

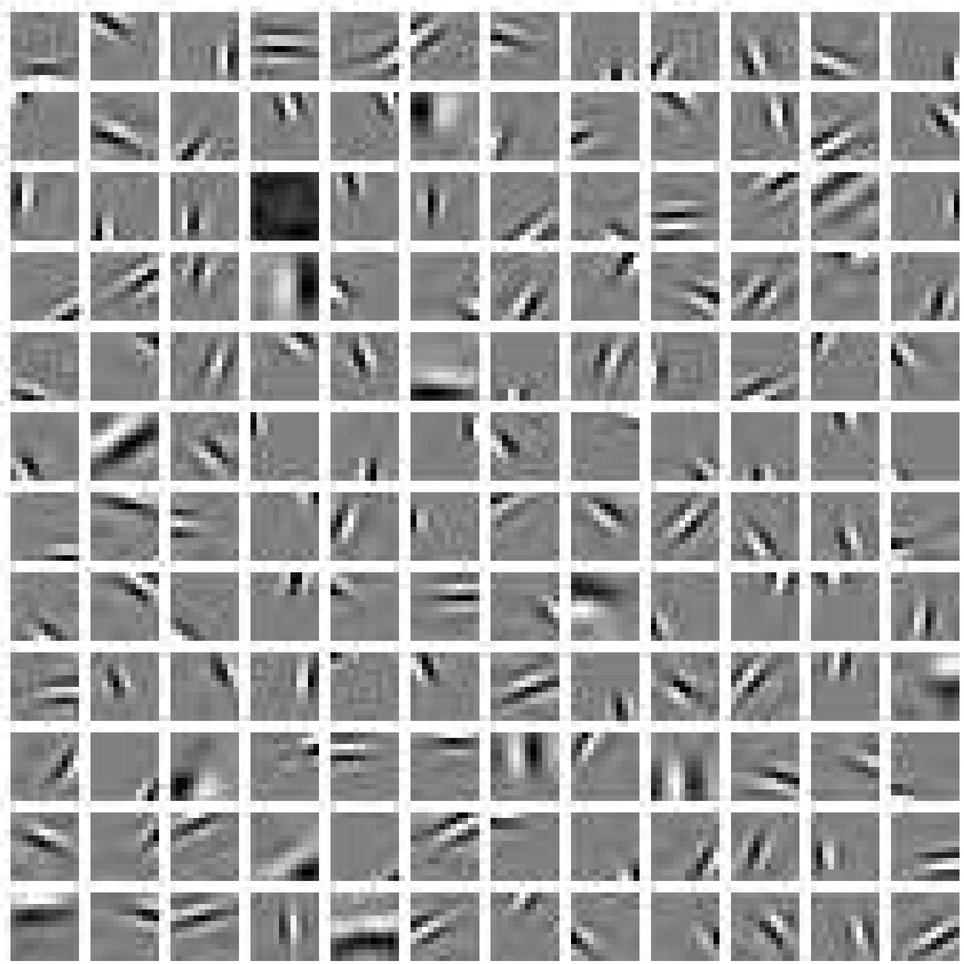


Figure 7: Example basis functions derived using sparseness criterion see (Olshausen & Field 1996).

Natural Movies

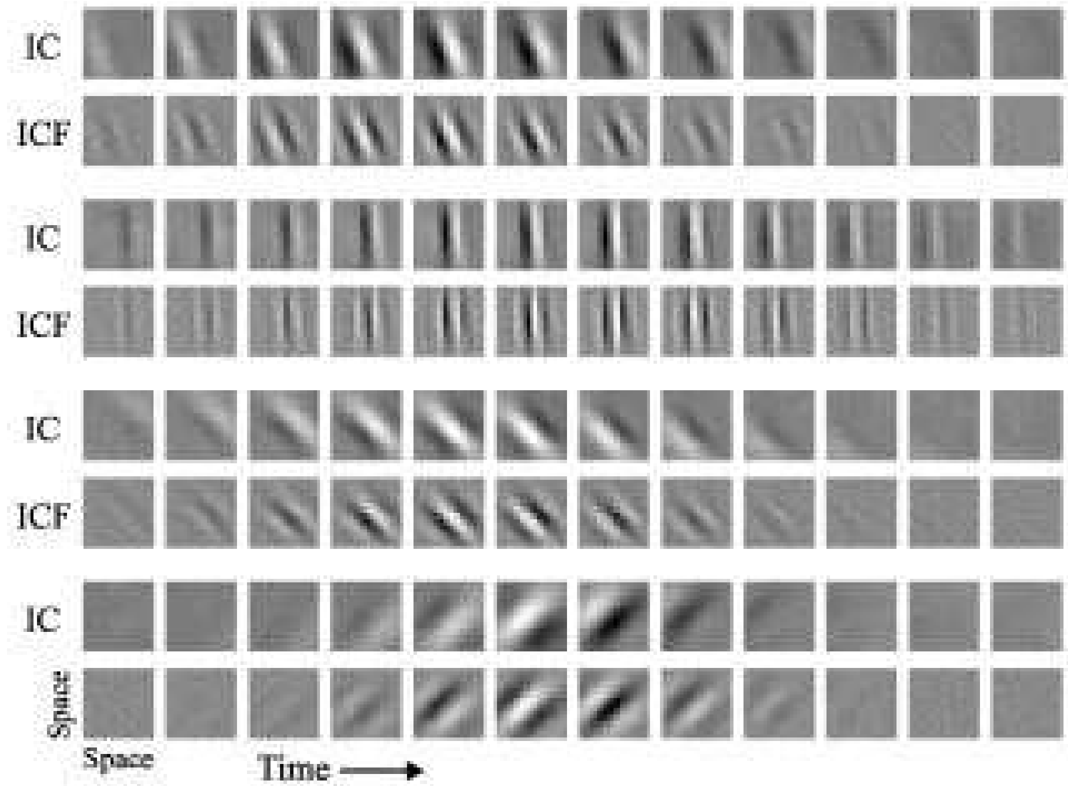


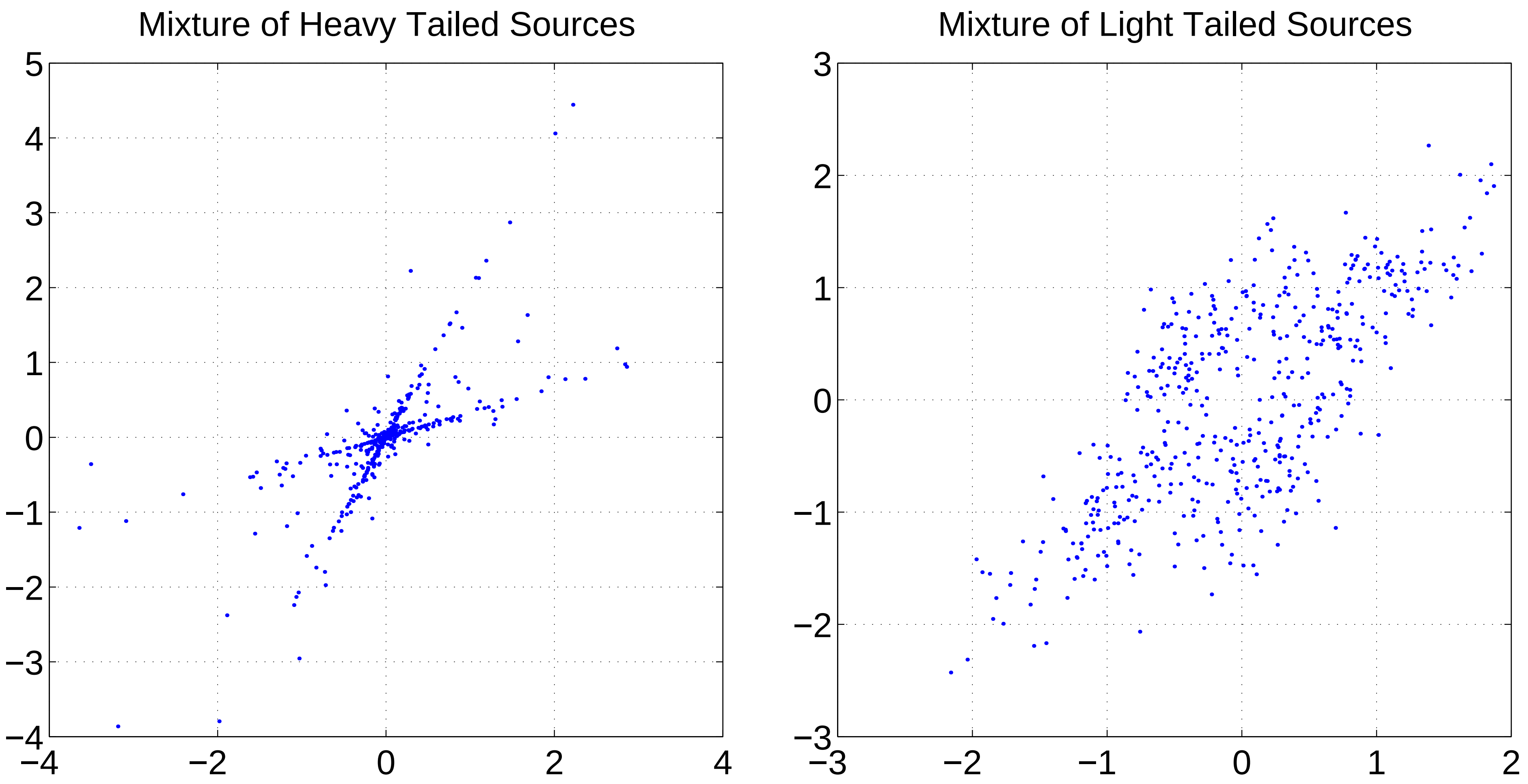
Figure 10: Independent components of natural movies. Shown are four space-time basis functions (rows labeled 'IC') with the corresponding analysis functions (rows labeled 'ICF') which would be convolved with a movie to compute a neuron's output (from van Hateren & Ruderman 1998).

Simoncelli EP, Olshausen BA (2001) Natural image statistics and neural representation. Annual Review of Neuroscience, 24.

Applications of ICA and Related Methods

- Separating auditory sources
- Analysis of EEG data
- Analysis of functional MRI data
- Natural scene analysis
- ...

ICA: The magic of unmixing



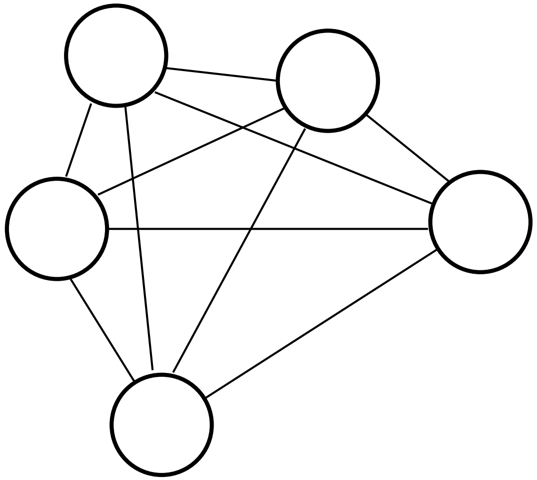
How ICA Relates to Factor Analysis and Other Models

- **Factor Analysis (FA)**: Assumes the factors are Gaussian.
- **Principal Components Analysis (PCA)**: Assumes no noise on the observations: $\Psi = \lim_{\epsilon \to 0} \epsilon I$
- **Independent Components Analysis (ICA)**: Assumes 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**: Assumes 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.

Extensions of ICA

- Fewer or more sources than “microphones” ($K \neq D$) – e.g. Lewicki and Sejnowski (1998).
- Allows noise on microphones
- Time series versions with convolution by linear filter
- Time-varying mixing matrix
- Discovering number of sources

Boltzmann Machines



Undirected graphical model (i.e. a Markov network) over a vector of binary variables $s_i \in \{0,1\}$. Some variables may be hidden, some may be visible (observed).

$$P(\mathbf{s}|W,\mathbf{b}) = \frac{1}{Z}\exp\left\{\sum_{ij} W_{ij}s_i s_j - \sum_i b_i s_i\right\}$$

where Z is the normalization constant (partition function).

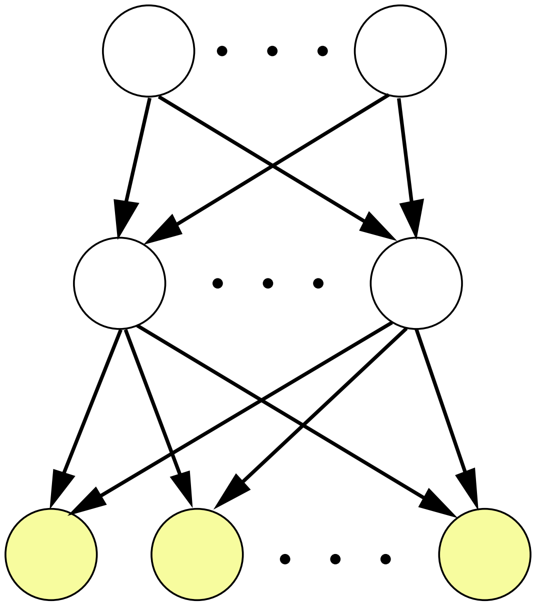
Learning algorithm: a gradient version of EM

- E step involves computing averages w.r.t. $P(\mathbf{s}_H|\mathbf{s}_V,W,\mathbf{b})$ ("clamped phase"). This could be done via a propagation algorithm or (more usually) an approximate method such as Gibbs sampling.

- The M step requires gradients w.r.t. Z, which can be computed by averages w.r.t. $P(\mathbf{s}|W,\mathbf{b})$ ("unclamped phase").

$$\Delta W_{ij} = \eta[\langle s_i s_j\rangle_c - \langle s_i s_j\rangle_u]$$

Sigmoid Belief Networks



Directed graphical model (i.e. a Bayesian network) over a vector of binary variables $s_i \in \{0, 1\}$.

$$P(\mathbf{s}|W, \mathbf{b}) = \prod_i P(s_i|\{s_j\}_{j<i}, W, \mathbf{b})$$

$$P(s_i = 1|\{s_j\}_{j<i}, W, \mathbf{b}) = \frac{1}{1 + \exp\{-\sum_{j<i} W_{ij} s_j - b_i\}}$$

A probabilistic version of sigmoid multilayer perceptrons ("neural networks").

Learning algorithm: a gradient version of EM

- E step involves computing averages w.r.t. $P(\mathbf{s}_H|\mathbf{s}_V, W, \mathbf{b})$. This could be done via the Belief Propagation algorithm (if singly connected) or (more usually) an approximate method such as Gibbs sampling or mean field (see later lectures).

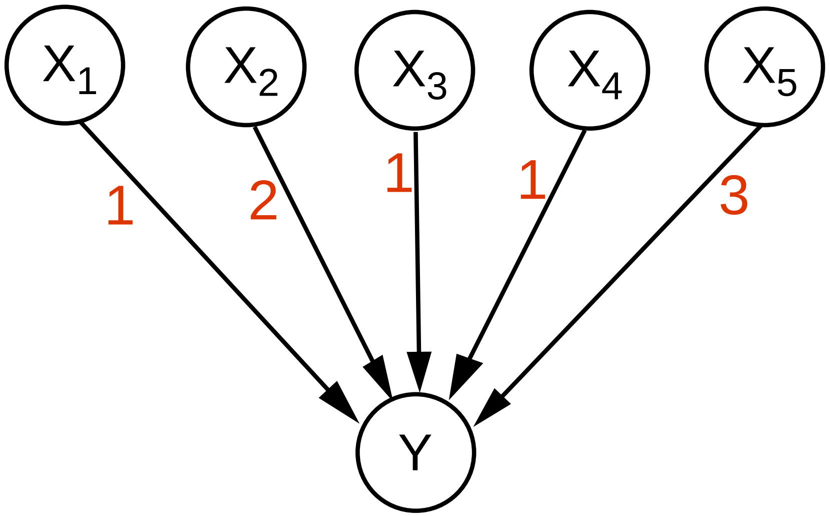
- Unlike Boltzmann machines, there is no partition function, so no need for an unclamped phase in the M step.

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

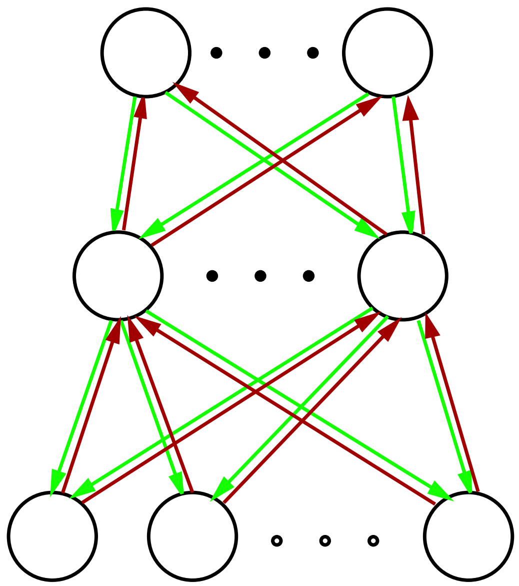
- **Sampling**: Approximate posterior distribution over hidden variables by a set of random samples. We often need **Markov chain Monte carlo** methods to sample from difficult distributions.

- **Linearization**: Approximate nonlinearities by Taylor series expansion about a point (e.g. the approximate mean of the hidden variable distribution). Linear approximations are particularly useful since Gaussian distributions are closed under linear transformations.

- **Recognition Models**: Approximate the hidden variable posterior distribution using an explicit *bottom-up* recognition model/network.

- **Variational Methods**: Approximate the hidden variable posterior $p(H)$ with a tractable form $q(H)$. This gives a lower bound on the likelihood that can be maximised with respect to the parameters of $q(H)$.

Recognition Models

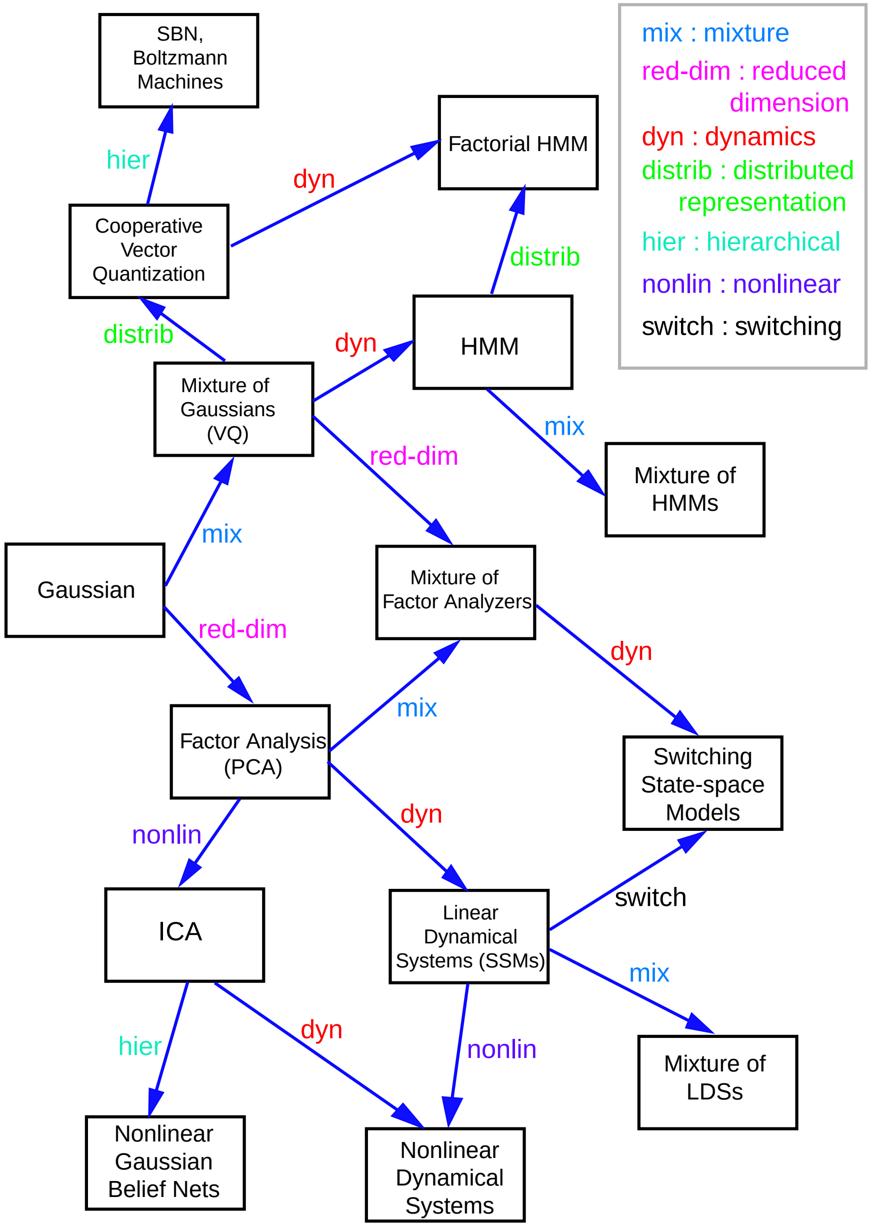


- a model is trained in a supervised way to recover the hidden causes (latent variables) from the observations

- 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)

A Generative Model for Generative Models



Suggested Readings and References

1. Attias, H. (1999) Independent Factor Analysis. Neural Computation 11:803-851.
2. Bell, A. and Sejnowski, T. (1995) An information maximization approach to blind source separation and blind deconvolution. Neural Computation 7:1129–1159.
3. Comon, P. (1994) Independent components analysis, a new concept? Signal Processing. 36:287–314.
4. Ghahramani, Z. and Beal, M.J. (2000) Graphical models and variational methods. In Saad & Opper (eds) Advanced Mean Field Method—Theory and Practice. MIT Press. Also available from my web page.
5. Ghahramani, Z. and Jordan, M.I. (1997) Factorial Hidden Markov Models. Machine Learning 29:245-273. `http://www.gatsby.ucl.ac.uk/~zoubin/papers/fhmmML.ps.gz`
6. David MacKay's Notes on ICA.
7. Lewicki, M. S. and Sejnowski, T. J. (1998) Learning overcomplete representations. Neural Computation.
8. Olshausen and Field (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381:607-609.
9. Barak Pearlmutter and Lucas Parra paper.
10. Roweis, S.T. and Ghahramani, Z. (1999) A Unifying Review of Linear Gaussian Models. Neural Computation 11(2). `http://www.gatsby.ucl.ac.uk/~zoubin/papers/lds.ps.gz` or `lds.pdf`
11. Max Welling's Notes on ICA: `http://www.gatsby.ucl.ac.uk/~zoubin/course01/WellingICA.ps`

Appendix: Matlab Code for ICA

```
% ICA using tanh nonlinearity and batch covariant algorithm
% (c) Zoubin Ghahramani
%
% function [W, Mu, LL]=ica(X,cyc,eta,Winit);
%
% X - data matrix (each row is a data point),   cyc - cycles of learning (default = 200)
% eta - learning rate (default = 0.2),          Winit - initial weight
%
% W - unmixing matrix,  Mu - data mean,         LL - log likelihoods during learning

function [W, Mu, LL]=ica(X,cyc,eta,Winit);

if nargin<2,  cyc=200; end;
if nargin<3,  eta=0.2; end;
[N D]=size(X);                  % size of data
Mu=mean(X); X=X-ones(N,1)*Mu;   % subtract mean
if nargin>3,    W=Winit;        % initialize matrix
else,   W=rand(D,D);  end;
LL=zeros(cyc,1);                % initialize log likelihoods

for i=1:cyc,
  U=X*W';
  logP=N*log(abs(det(W)))-sum(sum(log(cosh(U))))-N*D*log(pi);
  W=W+eta*(W-tanh(U')*U*W/N);                     % covariant algorithm
  % W=W+eta*(inv(W)-X'*tanh(U)/N)';               % standard algorithm
  LL(i)=logP; fprintf('cycle %g log P= %g\n',i,logP);
end;
```