# Probabilistic & Unsupervised Learning

## Beyond linear-Gaussian models and Mixtures

**Maneesh Sahani**
maneesh@gatsby.ucl.ac.uk

**Gatsby Computational Neuroscience Unit, and**
**MSc ML/CSML, Dept Computer Science**
**University College London**

**Term 1, Autumn 2017**

# Tractable Models

# Tractable Models

- Factor analysis, principle components analysis, probabilistic PCA.

# Tractable Models

- Factor analysis, principle components analysis, probabilistic PCA.
- Linear regression, Gaussian processes.

# Tractable Models

- Factor analysis, principle components analysis, probabilistic PCA.
- Linear regression, Gaussian processes.
- Mixture of Gaussians, mixture of experts.

# Tractable Models

- Factor analysis, principle components analysis, probabilistic PCA.
- Linear regression, Gaussian processes.
- Mixture of Gaussians, mixture of experts.
- Hidden Markov models, linear-Gaussian state space models.

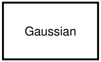
# Tractable Models

- Factor analysis, principle components analysis, probabilistic PCA.
- Linear regression, Gaussian processes.
- Mixture of Gaussians, mixture of experts.
- Hidden Markov models, linear-Gaussian state space models.

Models consisting of various combinations of:

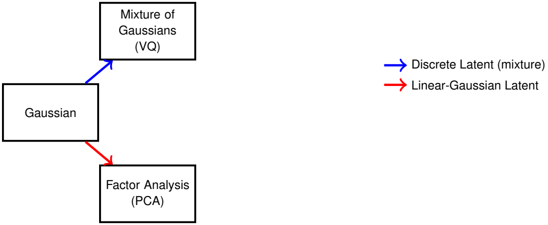
- Linear Gaussian,
- Discrete variables,
- Chains and trees (or junction trees),

# A Generative Model for Generative Models



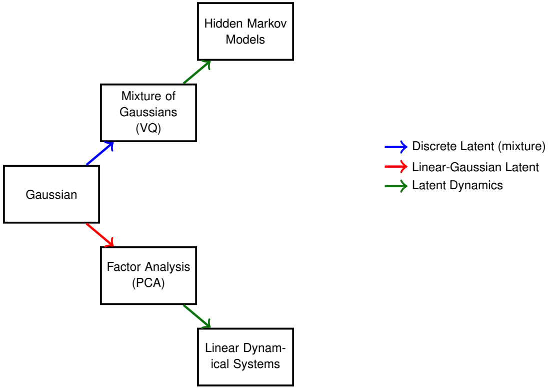

Adapted from Roweis & Ghahramani (1999). A Unifying Review of Linear Gaussian Models. *Neural Comput.* **11**(2).

# A Generative Model for Generative Models



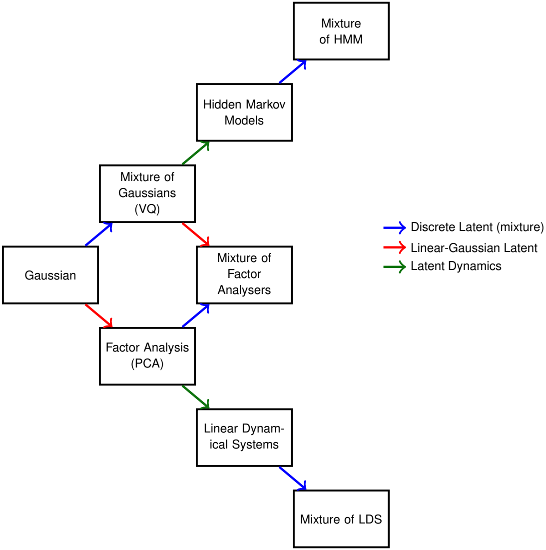

Adapted from Roweis & Ghahramani (1999). A Unifying Review of Linear Gaussian Models. *Neural Comput.* **11**(2).

# A Generative Model for Generative Models




Adapted from Roweis & Ghahramani (1999). A Unifying Review of Linear Gaussian Models. *Neural Comput.* **11**(2).

# A Generative Model for Generative Models




Adapted from Roweis & Ghahramani (1999). A Unifying Review of Linear Gaussian Models. *Neural Comput.* **11**(2).

# Expanding Our Horizons

Although these models can be powerful, they are undoubtedly still restrictive. There is a need to go beyond the confines of these structures

In this half of the course (and today) we will study:

# Expanding Our Horizons

Although these models can be powerful, they are undoubtedly still restrictive. There is a need to go beyond the confines of these structures

In this half of the course (and today) we will study:

- hierarchical models,

# Expanding Our Horizons

Although these models can be powerful, they are undoubtedly still restrictive. There is a need to go beyond the confines of these structures

In this half of the course (and today) we will study:

- hierarchical models,
- distributed models,

# Expanding Our Horizons

Although these models can be powerful, they are undoubtedly still restrictive. There is a need to go beyond the confines of these structures

In this half of the course (and today) we will study:

- hierarchical models,
- distributed models,
- nonlinear models,

# Expanding Our Horizons

Although these models can be powerful, they are undoubtedly still restrictive. There is a need to go beyond the confines of these structures

In this half of the course (and today) we will study:

- hierarchical models,
- distributed models,
- nonlinear models,
- non-Gaussian models.

# Expanding Our Horizons

Although these models can be powerful, they are undoubtedly still restrictive. There is a need to go beyond the confines of these structures

In this half of the course (and today) we will study:

- hierarchical models,
- distributed models,
- nonlinear models,
- non-Gaussian models.

and various combinations of these.

# Expanding Our Horizons

Although these models can be powerful, they are undoubtedly still restrictive. There is a need to go beyond the confines of these structures

In this half of the course (and today) we will study:

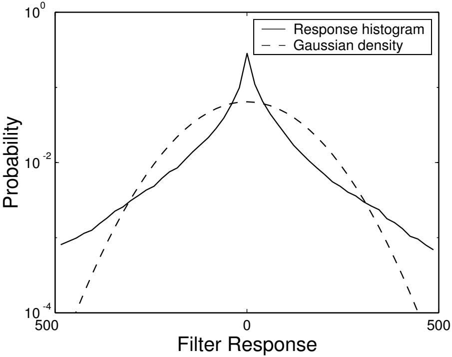
- hierarchical models,
- distributed models,
- nonlinear models,
- non-Gaussian models.

and various combinations of these.

Whilst sometimes tractable (particularly in corner cases), these models will most often require approximate inference.

# Why We Need . . . Nonlinear/Non-Gaussian Models

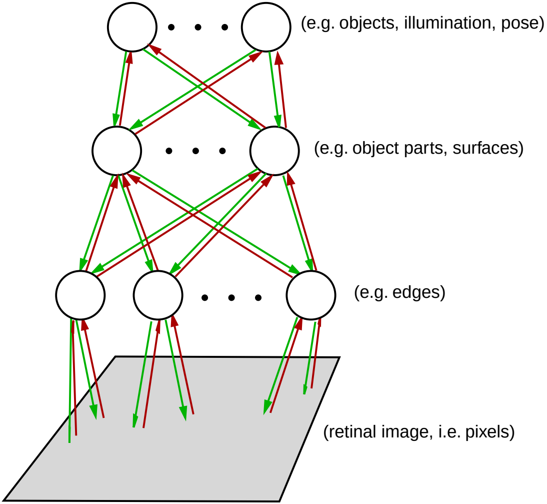
Much of the world is neither linear nor Gaussian




. . . and most interesting structure we would like to learn about is not either.

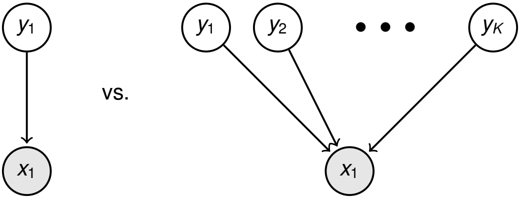
## Why We Need . . . Hierarchical (Deep) Models

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




Biology seems to have developed hierarchical representations.

# Why We Need . . . Distributed Models

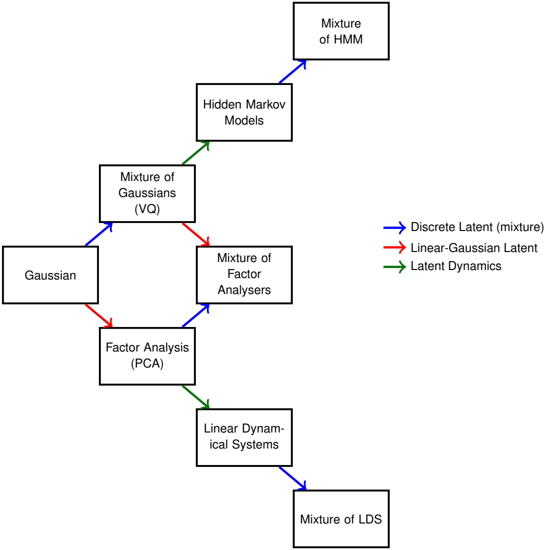



In a distributed representation each observation is characterised by a vector of (discrete or continous) attibutes. Some of these attributes might be latent.

- Unitary representation: categorise voters into small groups who (may) vote similarly e.g.: `London-based university professors of Asian descent`.
- Distributed respresentation: consider contributions from a group of attributes, e.g.: (`Single`, `Black`, `Female`, `34 yrs`, `Urban`, `Liberal`, `£35k p.a.`).
- Attributes resemble factors, but may be discrete or non-Gaussian, and may outnumber observations.

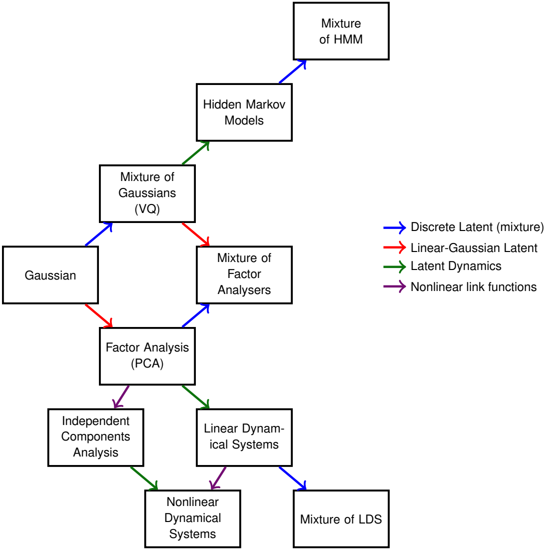
Distributed representations can be exponentially efficient: $K$ binary factors $\Rightarrow 2^K$ bits of info. ($K$ parallel binary state variables in an HMM can replace one variable with $2^K$ states.)

# A Generative Model for Generative Models



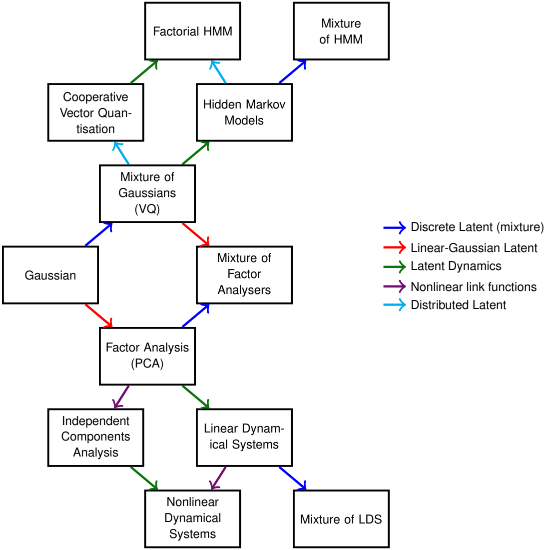

Adapted from Roweis & Ghahramani (1999). A Unifying Review of Linear Gaussian Models. *Neural Comput.* **11**(2).

# A Generative Model for Generative Models



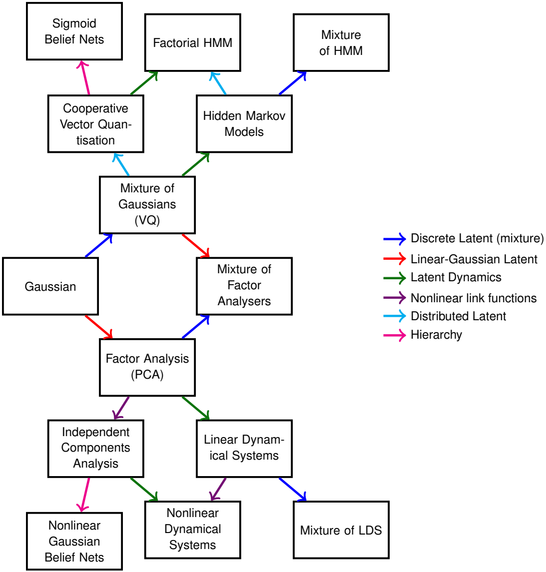

Adapted from Roweis & Ghahramani (1999). A Unifying Review of Linear Gaussian Models. *Neural Comput.* **11**(2).

# A Generative Model for Generative Models




Adapted from Roweis & Ghahramani (1999). A Unifying Review of Linear Gaussian Models. *Neural Comput.* **11**(2).

# A Generative Model for Generative Models

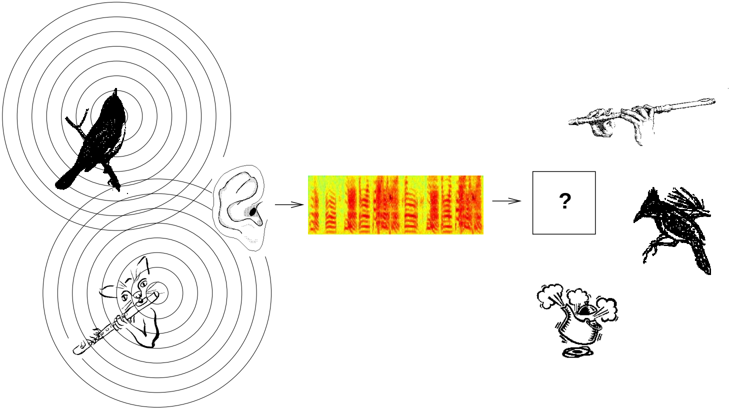



Adapted from Roweis & Ghahramani (1999). A Unifying Review of Linear Gaussian Models. *Neural Comput.* **11**(2).

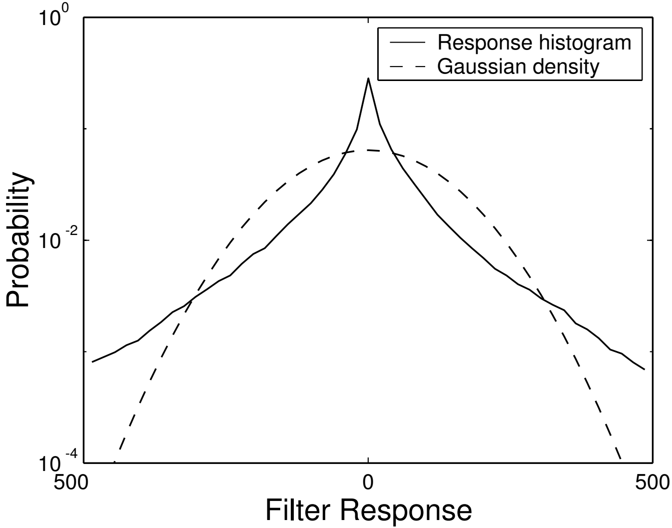
# Independent Components Analysis

# Blind Source Separation

Sometimes called the cocktail party problem.

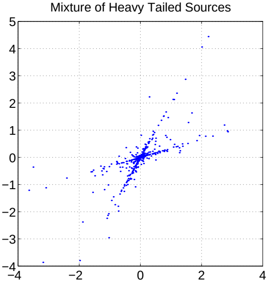


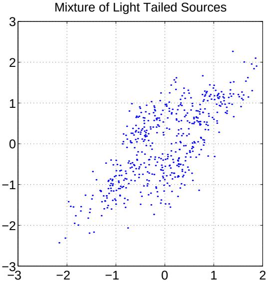

- Given signals from one or more receivers that mix signals from one or more sources, recover the timeseries of the source signals.
- Independent components analysis: assumes that sources are independent and non-Gaussian.



Response histogram
Gaussian density
Probability
$10^0$
$10^{-2}$
$10^{-4}$
500
0
500
Filter Response

# Independent Components Analysis



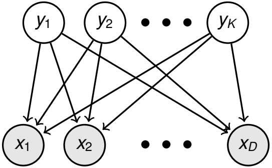




These distributions are generated by linearly combining (or mixing) two *non-Gaussian* sources.

- The ICA graphical model is identical to factor analysis:

$$x_d = \sum_{k=1}^{K} \Lambda_{dk}\ y_k + \epsilon_d$$

but with $y_k \sim P_y$ non-Gaussian.




Differences:

- Well-posed even with $K \geq D$ (e.g., $K = D = 2$ above).
- With non-zero noise, MAP inference is non-linear, and the full posterior is non-Gaussian.
- This makes making exact inference and learning difficult for most $P_y$.

# Square, Noiseless Causal ICA

- The special case of $K = D$, and zero observation noise has been studied extensively (infomax ICA, c.f. information view of PCA):

$$\mathbf{x} = \Lambda\mathbf{y} \quad \text{which implies} \quad \mathbf{y} = W\mathbf{x} \quad \text{with} \quad W = \Lambda^{-1}$$

  where $\mathbf{y}$ are the independent components (factors), $\mathbf{x}$ are the observations, and $W$ is the unmixing matrix.

# Square, Noiseless Causal ICA

- The special case of $K = D$, and zero observation noise has been studied extensively (infomax ICA, c.f. information view of PCA):

$$\mathbf{x} = \Lambda \mathbf{y} \quad \text{which implies} \quad \mathbf{y} = W\mathbf{x} \quad \text{with} \quad W = \Lambda^{-1}$$

  where $\mathbf{y}$ are the independent components (factors), $\mathbf{x}$ are the observations, and $W$ is the unmixing matrix.

- The likelihood can be obtained by transforming the density of $\mathbf{y}$ to that of $\mathbf{x}$. If $F : \mathbf{y} \mapsto \mathbf{x}$ is a differentiable bijection, and if $d\mathbf{y}$ is a small neighbourhood around $\mathbf{y}$, then

$$P_x(\mathbf{x})d\mathbf{x} = P_y(\mathbf{y})d\mathbf{y} = P_y(F^{-1}(\mathbf{x}))\left|\frac{d\mathbf{y}}{d\mathbf{x}}\right| d\mathbf{x} = P_y(F^{-1}(\mathbf{x}))\left|\nabla F^{-1}\right| d\mathbf{x}$$

# Square, Noiseless Causal ICA

- The special case of $K = D$, and zero observation noise has been studied extensively (infomax ICA, c.f. information view of PCA):

$$\mathbf{x} = \Lambda \mathbf{y} \quad \text{which implies} \quad \mathbf{y} = W\mathbf{x} \quad \text{with} \quad W = \Lambda^{-1}$$

  where $\mathbf{y}$ are the independent components (factors), $\mathbf{x}$ are the observations, and $W$ is the unmixing matrix.

- The likelihood can be obtained by transforming the density of $\mathbf{y}$ to that of $\mathbf{x}$. If $F : \mathbf{y} \mapsto \mathbf{x}$ is a differentiable bijection, and if $d\mathbf{y}$ is a small neighbourhood around $\mathbf{y}$, then

$$P_x(\mathbf{x})d\mathbf{x} = P_y(\mathbf{y})d\mathbf{y} = P_y(F^{-1}(\mathbf{x}))\left|\frac{d\mathbf{y}}{d\mathbf{x}}\right| d\mathbf{x} = P_y(F^{-1}(\mathbf{x}))\left|\nabla F^{-1}\right| d\mathbf{x}$$

- This gives (for parameter $W$):

$$P(\mathbf{x}|W) = |W| \prod_k P_y(\underbrace{[W\mathbf{x}]_k}_{y_k})$$

  where $P_y$ is marginal probability distribution of factors.

# Learning in ICA

- Log likelihood of data:

$$\log P(\mathbf{x}) = \log |W| + \sum_i \log P_y(W_i \mathbf{x})$$

# Learning in ICA

- Log likelihood of data:

$$\log P(\mathbf{x}) = \log |W| + \sum_i \log P_y(W_i \mathbf{x})$$

- Learning by gradient ascent:

$$\Delta W \propto \nabla_W \log P(\mathbf{x}) = W^{-T} + g(\mathbf{y})\mathbf{x}^{\mathsf{T}} \qquad g(y) = \frac{\partial \log P_y(y)}{\partial y}$$

# Learning in ICA

- Log likelihood of data:

$$\log P(\mathbf{x}) = \log |W| + \sum_i \log P_y(W_i \mathbf{x})$$

- Learning by gradient ascent:

$$\Delta W \propto \nabla_W \log P(\mathbf{x}) = W^{-T} + g(\mathbf{y})\mathbf{x}^{\mathsf{T}} \qquad g(y) = \frac{\partial \log P_y(y)}{\partial y}$$

- Better approach: "natural" or covariant gradient

$$\Delta W \propto \nabla_W \log P(\mathbf{x}) \cdot \underbrace{(W^{\mathsf{T}} W)}_{\approx \langle -\nabla\nabla \log P \rangle^{-1}} = W + g(\mathbf{y})\mathbf{y}^{\mathsf{T}} W$$

(see MacKay 1996).

# Learning in ICA

- Log likelihood of data:

$$\log P(\mathbf{x}) = \log |W| + \sum_i \log P_y(W_i \mathbf{x})$$

- Learning by gradient ascent:

$$\Delta W \propto \nabla_W \log P(\mathbf{x}) = W^{-T} + g(\mathbf{y})\mathbf{x}^{\mathsf{T}} \qquad g(y) = \frac{\partial \log P_y(y)}{\partial y}$$

- Better approach: "natural" or covariant gradient

$$\Delta W \propto \nabla_W \log P(\mathbf{x}) \cdot \underbrace{(W^{\mathsf{T}} W)}_{\approx \langle -\nabla\nabla \log P \rangle^{-1}} = W + g(\mathbf{y})\mathbf{y}^{\mathsf{T}} W$$

  (see MacKay 1996).

- Note: we can't use EM in the square noiseless causal ICA model. Why?

# Infomax ICA

- Consider a feedforward model:

$$y_i = W_i \mathbf{x}; \qquad z_i = f_i(y_i)$$

with a monotonic squashing function $f_i(-\infty) = 0$, $f_i(+\infty) = 1$.

# Infomax ICA

- Consider a feedforward model:

$$y_i = W_i \mathbf{x}; \qquad z_i = f_i(y_i)$$

  with a monotonic squashing function $f_i(-\infty) = 0$, $f_i(+\infty) = 1$.

- Infomax finds filtering weights $W$ maximizing the information carried by $\mathbf{z}$ about $\mathbf{x}$:

$$\operatorname*{argmax}_W I(\mathbf{x}; \mathbf{z}) = \operatorname*{argmax}_W H(\mathbf{z}) - H(\mathbf{z}|\mathbf{x}) = \operatorname*{argmax}_W H(\mathbf{z})$$

  Thus we just have to maximize entropy of $\mathbf{z}$: make it as uniform as possible on $[0, 1]$ (note squashing function).

# Infomax ICA

- Consider a feedforward model:

$$y_i = W_i \mathbf{x}; \qquad z_i = f_i(y_i)$$

  with a monotonic squashing function $f_i(-\infty) = 0$, $f_i(+\infty) = 1$.
- Infomax finds filtering weights $W$ maximizing the information carried by $\mathbf{z}$ about $\mathbf{x}$:

$$\underset{W}{\operatorname{argmax}}\, I(\mathbf{x}; \mathbf{z}) = \underset{W}{\operatorname{argmax}}\, H(\mathbf{z}) - H(\mathbf{z}|\mathbf{x}) = \underset{W}{\operatorname{argmax}}\, H(\mathbf{z})$$

  Thus we just have to maximize entropy of $\mathbf{z}$: make it as uniform as possible on $[0, 1]$ (note squashing function).
- But if data were generated from a square noiseless causal ICA then best we can do is if

$$z_i = f_i(y_i) = \mathrm{cdf}_i(y_i) \quad \text{and} \quad W = \Lambda^{-1}$$

  **Infomax ICA $\Leftrightarrow$ square noiseless causal ICA**.

# Infomax ICA

- Consider a feedforward model:

$$y_i = W_i \mathbf{x}; \qquad z_i = f_i(y_i)$$

  with a monotonic squashing function $f_i(-\infty) = 0$, $f_i(+\infty) = 1$.
- Infomax finds filtering weights $W$ maximizing the information carried by $\mathbf{z}$ about $\mathbf{x}$:

$$\underset{W}{\text{argmax}}\, I(\mathbf{x}; \mathbf{z}) = \underset{W}{\text{argmax}}\, H(\mathbf{z}) - H(\mathbf{z}|\mathbf{x}) = \underset{W}{\text{argmax}}\, H(\mathbf{z})$$

  Thus we just have to maximize entropy of $\mathbf{z}$: make it as uniform as possible on $[0, 1]$ (note squashing function).
- But if data were generated from a square noiseless causal ICA then best we can do is if

$$z_i = f_i(y_i) = \text{cdf}_i(y_i) \quad \text{and} \quad W = \Lambda^{-1}$$

  **Infomax ICA $\Leftrightarrow$ square noiseless causal ICA**.
- Another view: redundancy reduction in the representation $\mathbf{z}$ of the data $\mathbf{x}$.

$$\underset{W}{\text{argmax}}\, H(\mathbf{z}) = \underset{W}{\text{argmax}} \sum_i H(z_i) - I(z_1, \ldots, z_D)$$

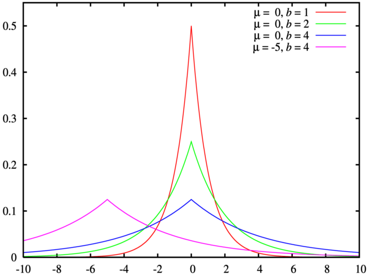
See: MacKay (1996), Pearlmutter and Parra (1996), Cardoso (1997) for equivalence, Teh et al (2003) for an energy-based view.

# Kurtosis

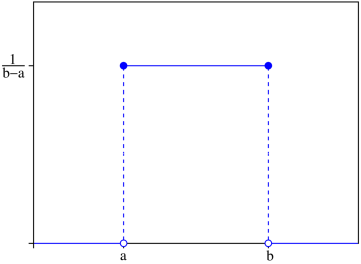
The kurtosis (or excess kurtosis) measures how "peaky" or "heavy-tailed" a distribution is:

$$K = \frac{E((x-\mu)^4)}{E((x-\mu)^2)^2} - 3, \text{ where } \mu = E(x) \text{ is the mean of } x.$$

Gaussian distributions have zero kurtosis.




Heavy tailed: positive kurtosis (leptokurtic).




Light tailed: negative kurtosis (platykurtic).

Some ICA algorithms are essentially kurtosis pursuit approaches. Possibly fewer assumptions about generating distributions.

# ICA and BSS

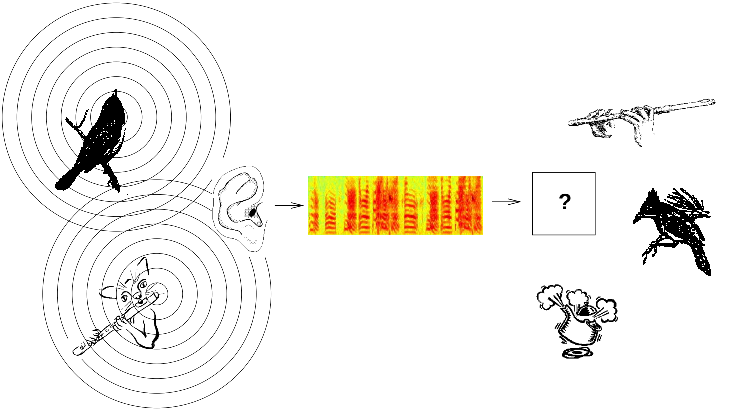
**Applications:**

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

**Extensions:**

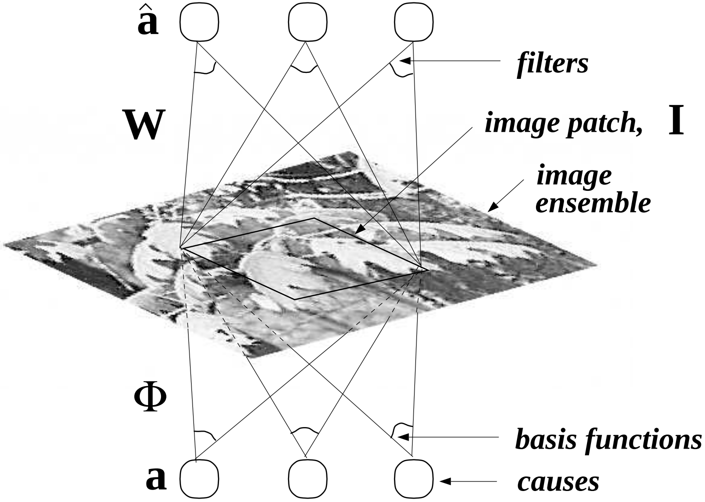
- Non-zero output noise – approximate posteriors and learning.
- Undercomplete ($K < D$) or overcomplete ($K > D$).
- Learning prior distributions (on $\mathbf{y}$).
- Dynamical hidden models (on $\mathbf{y}$).
- Learning number of sources.
- Time-varying mixing matrix.
- Nonparametric, kernel ICA.
- ...

# Blind Source Separation



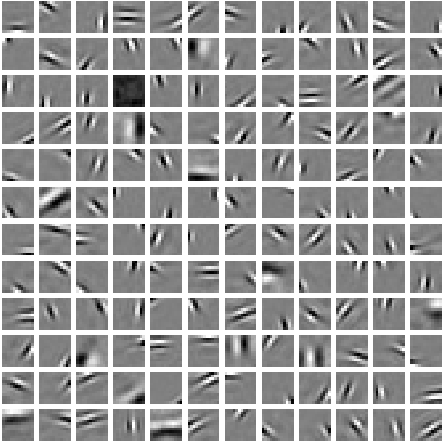

- ICA solution to blind source separation assumes no dependence across time; still works fine much of the time.
- Many other algorithms: DCA, SOBI, JADE, ...

# Images



â
W
Φ
a
filters
image patch, I
image ensemble
basis functions
causes

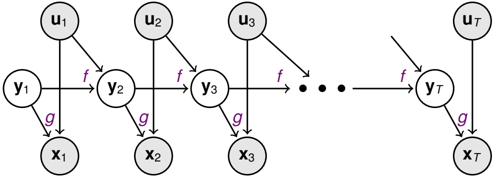
# Natural Scenes



Olshausen & Field (1996)

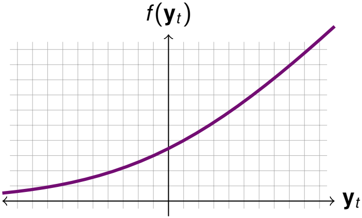
## Nonlinear state-space models

# Nonlinear state-space model (NLSSM)

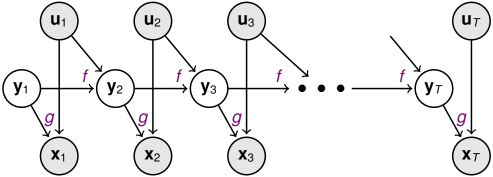



$$\mathbf{y}_{t+1} = f(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{w}_t$$
$$\mathbf{x}_t = g(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{v}_t$$

$\mathbf{w}_t, \mathbf{v}_t$ usually still Gaussian.



# Nonlinear state-space model (NLSSM)



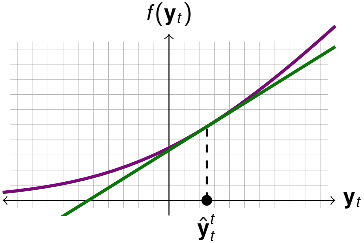

$$\mathbf{y}_{t+1} = f(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{w}_t$$
$$\mathbf{x}_t = g(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{v}_t$$

$\mathbf{w}_t, \mathbf{v}_t$ usually still Gaussian.

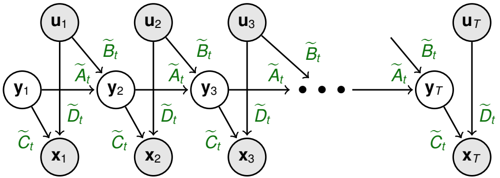
**Extended Kalman Filter (EKF)**: linearise nonlinear functions about current estimate, $\hat{\mathbf{y}}_t^t$:

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

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



# Nonlinear state-space model (NLSSM)




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

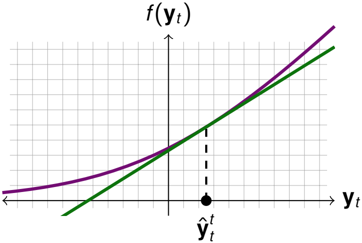
$$\mathbf{x}_t = g(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{v}_t$$

$\mathbf{w}_t, \mathbf{v}_t$ usually still Gaussian.

**Extended Kalman Filter (EKF):** linearise nonlinear functions about current estimate, $\hat{\mathbf{y}}_t^t$:

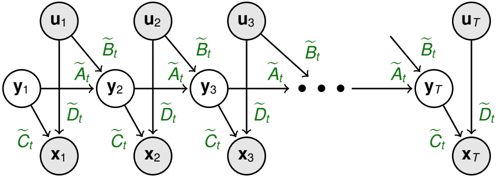
$$\mathbf{y}_{t+1} \approx \underbrace{f(\hat{\mathbf{y}}_t^t, \mathbf{u}_t)}_{\widetilde{B}_t \mathbf{u}_t} + \underbrace{\left.\frac{\partial f}{\partial \mathbf{y}_t}\right|_{\hat{\mathbf{y}}_t^t}}_{\widetilde{A}_t} (\mathbf{y}_t - \hat{\mathbf{y}}_t^t) + \mathbf{w}_t$$

$$\mathbf{x}_t \approx \underbrace{g(\hat{\mathbf{y}}_t^{t-1}, \mathbf{u}_t)}_{\widetilde{D}_t \mathbf{u}_t} + \underbrace{\left.\frac{\partial g}{\partial \mathbf{y}_t}\right|_{\hat{\mathbf{y}}_t^{t-1}}}_{\widetilde{C}_t} (\mathbf{y}_t - \hat{\mathbf{y}}_t^{t-1}) + \mathbf{v}_t$$




Run the Kalman filter (smoother) on non-stationary linearised system $(\widetilde{A}_t, \widetilde{B}_t, \widetilde{C}_t, \widetilde{D}_t)$:

# Nonlinear state-space model (NLSSM)



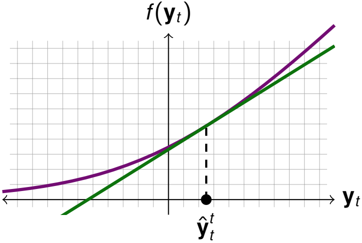

$$\mathbf{y}_{t+1} = f(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{w}_t$$
$$\mathbf{x}_t = g(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{v}_t$$

$\mathbf{w}_t, \mathbf{v}_t$ usually still Gaussian.

**Extended Kalman Filter (EKF)**: linearise nonlinear functions about current estimate, $\hat{\mathbf{y}}_t^t$:

$$\mathbf{y}_{t+1} \approx \underbrace{f(\hat{\mathbf{y}}_t^t, \mathbf{u}_t)}_{\widetilde{B}_t \mathbf{u}_t} + \underbrace{\left.\frac{\partial f}{\partial \mathbf{y}_t}\right|_{\hat{\mathbf{y}}_t^t}}_{\widetilde{A}_t} (\mathbf{y}_t - \hat{\mathbf{y}}_t^t) + \mathbf{w}_t$$

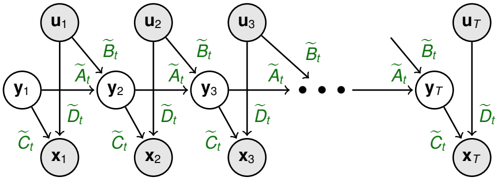
$$\mathbf{x}_t \approx \underbrace{g(\hat{\mathbf{y}}_t^{t-1}, \mathbf{u}_t)}_{\widetilde{D}_t \mathbf{u}_t} + \underbrace{\left.\frac{\partial g}{\partial \mathbf{y}_t}\right|_{\hat{\mathbf{y}}_t^{t-1}}}_{\widetilde{C}_t} (\mathbf{y}_t - \hat{\mathbf{y}}_t^{t-1}) + \mathbf{v}_t$$




Run the Kalman filter (smoother) on non-stationary linearised system ($\widetilde{A}_t, \widetilde{B}_t, \widetilde{C}_t, \widetilde{D}_t$):

- Adaptively approximates non-Gaussian messages by Gaussians.

# Nonlinear state-space model (NLSSM)



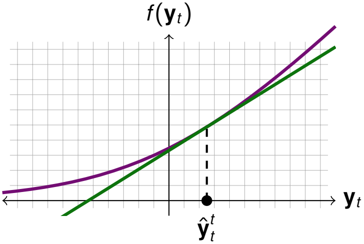

$$\mathbf{y}_{t+1} = f(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{w}_t$$
$$\mathbf{x}_t = g(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{v}_t$$

$\mathbf{w}_t, \mathbf{v}_t$ usually still Gaussian.

**Extended Kalman Filter (EKF)**: linearise nonlinear functions about current estimate, $\hat{\mathbf{y}}_t^t$:

$$\mathbf{y}_{t+1} \approx \underbrace{f(\hat{\mathbf{y}}_t^t, \mathbf{u}_t)}_{\widetilde{B}_t \mathbf{u}_t} + \underbrace{\left.\frac{\partial f}{\partial \mathbf{y}_t}\right|_{\hat{\mathbf{y}}_t^t}}_{\widetilde{A}_t} (\mathbf{y}_t - \hat{\mathbf{y}}_t^t) + \mathbf{w}_t$$

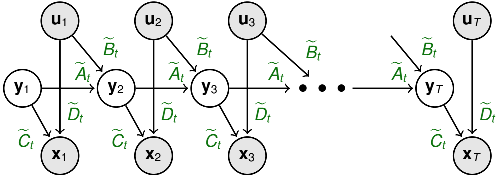
$$\mathbf{x}_t \approx \underbrace{g(\hat{\mathbf{y}}_t^{t-1}, \mathbf{u}_t)}_{\widetilde{D}_t \mathbf{u}_t} + \underbrace{\left.\frac{\partial g}{\partial \mathbf{y}_t}\right|_{\hat{\mathbf{y}}_t^{t-1}}}_{\widetilde{C}_t} (\mathbf{y}_t - \hat{\mathbf{y}}_t^{t-1}) + \mathbf{v}_t$$




Run the Kalman filter (smoother) on non-stationary linearised system $(\widetilde{A}_t, \widetilde{B}_t, \widetilde{C}_t, \widetilde{D}_t)$:

- Adaptively approximates non-Gaussian messages by Gaussians.
- Local linearisation depends on central point of distribution $\Rightarrow$ approximation degrades with increased state uncertainty.

# Nonlinear state-space model (NLSSM)



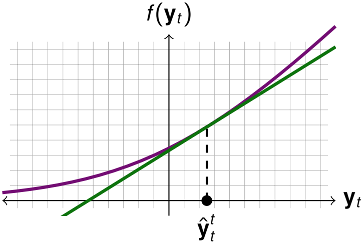

$$\mathbf{y}_{t+1} = f(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{w}_t$$
$$\mathbf{x}_t = g(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{v}_t$$

$\mathbf{w}_t, \mathbf{v}_t$ usually still Gaussian.

**Extended Kalman Filter (EKF)**: linearise nonlinear functions about current estimate, $\hat{\mathbf{y}}_t^t$:

$$\mathbf{y}_{t+1} \approx \underbrace{f(\hat{\mathbf{y}}_t^t, \mathbf{u}_t)}_{\widetilde{B}_t \mathbf{u}_t} + \underbrace{\left.\frac{\partial f}{\partial \mathbf{y}_t}\right|_{\hat{\mathbf{y}}_t^t}}_{\widetilde{A}_t} (\mathbf{y}_t - \hat{\mathbf{y}}_t^t) + \mathbf{w}_t$$

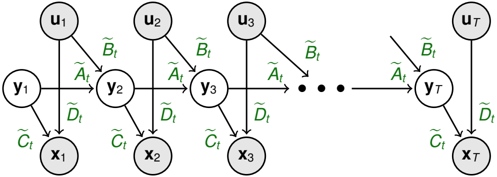
$$\mathbf{x}_t \approx \underbrace{g(\hat{\mathbf{y}}_t^{t-1}, \mathbf{u}_t)}_{\widetilde{D}_t \mathbf{u}_t} + \underbrace{\left.\frac{\partial g}{\partial \mathbf{y}_t}\right|_{\hat{\mathbf{y}}_t^{t-1}}}_{\widetilde{C}_t} (\mathbf{y}_t - \hat{\mathbf{y}}_t^{t-1}) + \mathbf{v}_t$$




Run the Kalman filter (smoother) on non-stationary linearised system $(\widetilde{A}_t, \widetilde{B}_t, \widetilde{C}_t, \widetilde{D}_t)$:

- Adaptively approximates non-Gaussian messages by Gaussians.
- Local linearisation depends on central point of distribution $\Rightarrow$ approximation degrades with increased state uncertainty. May work acceptably for close-to-linear systems.

# Nonlinear state-space model (NLSSM)



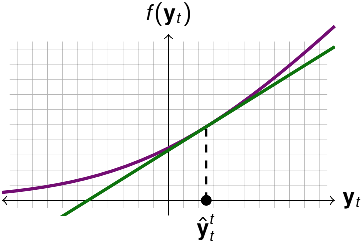

$$\mathbf{y}_{t+1} = f(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{w}_t$$
$$\mathbf{x}_t = g(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{v}_t$$

$\mathbf{w}_t, \mathbf{v}_t$ usually still Gaussian.

**Extended Kalman Filter (EKF)**: linearise nonlinear functions about current estimate, $\hat{\mathbf{y}}_t^t$:

$$\mathbf{y}_{t+1} \approx \underbrace{f(\hat{\mathbf{y}}_t^t, \mathbf{u}_t)}_{\widetilde{B}_t \mathbf{u}_t} + \underbrace{\left.\frac{\partial f}{\partial \mathbf{y}_t}\right|_{\hat{\mathbf{y}}_t^t}}_{\widetilde{A}_t} (\mathbf{y}_t - \hat{\mathbf{y}}_t^t) + \mathbf{w}_t$$

$$\mathbf{x}_t \approx \underbrace{g(\hat{\mathbf{y}}_t^{t-1}, \mathbf{u}_t)}_{\widetilde{D}_t \mathbf{u}_t} + \underbrace{\left.\frac{\partial g}{\partial \mathbf{y}_t}\right|_{\hat{\mathbf{y}}_t^{t-1}}}_{\widetilde{C}_t} (\mathbf{y}_t - \hat{\mathbf{y}}_t^{t-1}) + \mathbf{v}_t$$




Run the Kalman filter (smoother) on non-stationary linearised system $(\widetilde{A}_t, \widetilde{B}_t, \widetilde{C}_t, \widetilde{D}_t)$:

- Adaptively approximates non-Gaussian messages by Gaussians.
- Local linearisation depends on central point of distribution $\Rightarrow$ approximation degrades with increased state uncertainty. May work acceptably for close-to-linear systems.

Can base EM-like algorithm on EKF/EKS (or alternatives).

## Learning (online EKF)

Nonlinear message passing can also be used to implement online parameter learning in (non)linear latent state-space systems:

# Learning (online EKF)

Nonlinear message passing can also be used to implement online parameter learning in (non)linear latent state-space systems:

Eg: for linear model, augment state vector to include the model parameters : $\bar{\bar{\mathbf{y}}}_t = \begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}$, and introduce nonlinear transition $\bar{\bar{f}}$ and output map $\bar{\bar{g}}$:

$$\bar{\bar{\mathbf{y}}}_{t+1} = \bar{\bar{f}}(\bar{\bar{\mathbf{y}}}_t) + \bar{\bar{\mathbf{w}}}_t \qquad \bar{\bar{f}}\left(\begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}\right) = \begin{bmatrix} A\mathbf{y}_t \\ A \\ C \end{bmatrix}; \qquad \bar{\bar{\mathbf{w}}}_t = \begin{bmatrix} \mathbf{w_t} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{x}_t = \bar{\bar{g}}(\bar{\bar{\mathbf{y}}}_t) + \mathbf{v}_t \qquad \bar{\bar{g}}\left(\begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}\right) = C\mathbf{y}_t$$

(where $A$ and $C$ need to be vectorised and de-vectorised as appropriate).

## Learning (online EKF)

Nonlinear message passing can also be used to implement online parameter learning in (non)linear latent state-space systems:

Eg: for linear model, augment state vector to include the model parameters : $\bar{\bar{\mathbf{y}}}_t = \begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}$, and introduce nonlinear transition $\bar{\bar{f}}$ and output map $\bar{\bar{g}}$:

$$\bar{\bar{\mathbf{y}}}_{t+1} = \bar{\bar{f}}(\bar{\bar{\mathbf{y}}}_t) + \bar{\bar{\mathbf{w}}}_t \qquad \bar{\bar{f}}\left(\begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}\right) = \begin{bmatrix} A\mathbf{y}_t \\ A \\ C \end{bmatrix}; \qquad \bar{\bar{\mathbf{w}}}_t = \begin{bmatrix} \mathbf{w_t} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{x}_t = \bar{\bar{g}}(\bar{\bar{\mathbf{y}}}_t) + \mathbf{v}_t \qquad \bar{\bar{g}}\left(\begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}\right) = C\mathbf{y}_t$$

(where $A$ and $C$ need to be vectorised and de-vectorised as appropriate).

Use EKF to compute online estimates of $E[\bar{\bar{\mathbf{y}}}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t]$ and $\mathrm{Cov}[\bar{\bar{\mathbf{y}}}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t]$. These now include mean and posterior variance of parameter estimates.

# Learning (online EKF)

Nonlinear message passing can also be used to implement online parameter learning in (non)linear latent state-space systems:

Eg: for linear model, augment state vector to include the model parameters : $\bar{\bar{\mathbf{y}}}_t = \begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}$, and introduce nonlinear transition $\bar{\bar{f}}$ and output map $\bar{\bar{g}}$:

$$\bar{\bar{\mathbf{y}}}_{t+1} = \bar{\bar{f}}(\bar{\bar{\mathbf{y}}}_t) + \bar{\bar{\mathbf{w}}}_t \qquad \bar{\bar{f}}\left(\begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}\right) = \begin{bmatrix} A\mathbf{y}_t \\ A \\ C \end{bmatrix}; \qquad \bar{\bar{\mathbf{w}}}_t = \begin{bmatrix} \mathbf{w_t} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{x}_t = \bar{\bar{g}}(\bar{\bar{\mathbf{y}}}_t) + \mathbf{v}_t \qquad \bar{\bar{g}}\left(\begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}\right) = C\mathbf{y}_t$$

(where $A$ and $C$ need to be vectorised and de-vectorised as appropriate).

Use EKF to compute online estimates of $E[\bar{\bar{\mathbf{y}}}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t]$ and $\mathrm{Cov}[\bar{\bar{\mathbf{y}}}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t]$. These now include mean and posterior variance of parameter estimates.

- Pseudo-Bayesian approach: gives Gaussian distributions over parameters.

## Learning (online EKF)

Nonlinear message passing can also be used to implement online parameter learning in (non)linear latent state-space systems:

Eg: for linear model, augment state vector to include the model parameters : $\bar{\bar{\mathbf{y}}}_t = \begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}$,
and introduce nonlinear transition $\bar{\bar{f}}$ and output map $\bar{\bar{g}}$:

$$\bar{\bar{\mathbf{y}}}_{t+1} = \bar{\bar{f}}(\bar{\bar{\mathbf{y}}}_t) + \bar{\bar{\mathbf{w}}}_t \qquad \bar{\bar{f}}\left(\begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}\right) = \begin{bmatrix} A\mathbf{y}_t \\ A \\ C \end{bmatrix}; \qquad \bar{\bar{\mathbf{w}}}_t = \begin{bmatrix} \mathbf{w_t} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{x}_t = \bar{\bar{g}}(\bar{\bar{\mathbf{y}}}_t) + \mathbf{v}_t \qquad \bar{\bar{g}}\left(\begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}\right) = C\mathbf{y}_t$$

(where $A$ and $C$ need to be vectorised and de-vectorised as appropriate).
Use EKF to compute online estimates of $E[\bar{\bar{\mathbf{y}}}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t]$ and $\mathrm{Cov}[\bar{\bar{\mathbf{y}}}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t]$. These now include mean and posterior variance of parameter estimates.

- Pseudo-Bayesian approach: gives Gaussian distributions over parameters.
- Can model nonstationarity by assuming non-zero innovations noise in $A$, $C$.

# Learning (online EKF)

Nonlinear message passing can also be used to implement online parameter learning in (non)linear latent state-space systems:

Eg: for linear model, augment state vector to include the model parameters :$\bar{\bar{\mathbf{y}}}_t = \begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}$, and introduce nonlinear transition $\bar{\bar{f}}$ and output map $\bar{\bar{g}}$:

$$\bar{\bar{\mathbf{y}}}_{t+1} = \bar{\bar{f}}(\bar{\bar{\mathbf{y}}}_t) + \bar{\bar{\mathbf{w}}}_t \qquad \bar{\bar{f}}\left(\begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}\right) = \begin{bmatrix} A\mathbf{y}_t \\ A \\ C \end{bmatrix}; \qquad \bar{\bar{\mathbf{w}}}_t = \begin{bmatrix} \mathbf{w_t} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{x}_t = \bar{\bar{g}}(\bar{\bar{\mathbf{y}}}_t) + \mathbf{v}_t \qquad \bar{\bar{g}}\left(\begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}\right) = C\mathbf{y}_t$$

(where $A$ and $C$ need to be vectorised and de-vectorised as appropriate).

Use EKF to compute online estimates of $E[\bar{\bar{\mathbf{y}}}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t]$ and $\mathrm{Cov}[\bar{\bar{\mathbf{y}}}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t]$. These now include mean and posterior variance of parameter estimates.

- Pseudo-Bayesian approach: gives Gaussian distributions over parameters.
- Can model nonstationarity by assuming non-zero innovations noise in $A$, $C$.
- Not simple to implement for $Q$ and $R$ (e.g. covariance constraints?).

# Learning (online EKF)

Nonlinear message passing can also be used to implement online parameter learning in (non)linear latent state-space systems:

Eg: for linear model, augment state vector to include the model parameters :$\bar{\bar{\mathbf{y}}}_t = \begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}$, and introduce nonlinear transition $\bar{\bar{f}}$ and output map $\bar{\bar{g}}$:

$$\bar{\bar{\mathbf{y}}}_{t+1} = \bar{\bar{f}}(\bar{\bar{\mathbf{y}}}_t) + \bar{\bar{\mathbf{w}}}_t \qquad \bar{\bar{f}}\left(\begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}\right) = \begin{bmatrix} A\mathbf{y}_t \\ A \\ C \end{bmatrix}; \qquad \bar{\bar{\mathbf{w}}}_t = \begin{bmatrix} \mathbf{w_t} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{x}_t = \bar{\bar{g}}(\bar{\bar{\mathbf{y}}}_t) + \mathbf{v}_t \qquad \bar{\bar{g}}\left(\begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}\right) = C\mathbf{y}_t$$

(where $A$ and $C$ need to be vectorised and de-vectorised as appropriate).

Use EKF to compute online estimates of $E[\bar{\bar{\mathbf{y}}}_t | \mathbf{x}_1, \ldots, \mathbf{x}_t]$ and $\mathrm{Cov}[\bar{\bar{\mathbf{y}}}_t | \mathbf{x}_1, \ldots, \mathbf{x}_t]$. These now include mean and posterior variance of parameter estimates.

- Pseudo-Bayesian approach: gives Gaussian distributions over parameters.
- Can model nonstationarity by assuming non-zero innovations noise in $A$, $C$.
- Not simple to implement for $Q$ and $R$ (e.g. covariance constraints?).
- May be faster than EM/gradient approaches.

## Learning (online EKF)

Nonlinear message passing can also be used to implement online parameter learning in (non)linear latent state-space systems:

Eg: for linear model, augment state vector to include the model parameters :$\bar{\bar{\mathbf{y}}}_t = \begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}$,
and introduce nonlinear transition $\bar{\bar{f}}$ and output map $\bar{\bar{g}}$:

$$\bar{\bar{\mathbf{y}}}_{t+1} = \bar{\bar{f}}(\bar{\bar{\mathbf{y}}}_t) + \bar{\bar{\mathbf{w}}}_t \qquad \bar{\bar{f}}\left(\begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}\right) = \begin{bmatrix} A\mathbf{y}_t \\ A \\ C \end{bmatrix}; \qquad \bar{\bar{\mathbf{w}}}_t = \begin{bmatrix} \mathbf{w_t} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{x}_t = \bar{\bar{g}}(\bar{\bar{\mathbf{y}}}_t) + \mathbf{v}_t \qquad \bar{\bar{g}}\left(\begin{bmatrix} \mathbf{y}_t \\ A \\ C \end{bmatrix}\right) = C\mathbf{y}_t$$

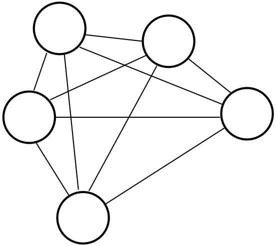
(where $A$ and $C$ need to be vectorised and de-vectorised as appropriate).
Use EKF to compute online estimates of $E[\bar{\bar{\mathbf{y}}}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t]$ and $\mathrm{Cov}[\bar{\bar{\mathbf{y}}}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t]$. These now include mean and posterior variance of parameter estimates.

- Pseudo-Bayesian approach: gives Gaussian distributions over parameters.
- Can model nonstationarity by assuming non-zero innovations noise in $A$, $C$.
- Not simple to implement for $Q$ and $R$ (e.g. covariance constraints?).
- May be faster than EM/gradient approaches.

Sometimes called the joint-EKF approach.

# Binary models: Boltzmann Machines and Sigmoid Belief Nets

## Boltzmann Machines



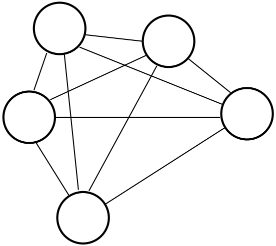
Undirected graphical model (i.e. a Markov network) over a vector of binary variables $s_i \in \{0, 1\}$. Some variables may be hidden, some may be visible (observed).

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

where $Z$ is the normalization constant (partition function).

A jointly exponential-family model, with intractable normaliser.

# 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).

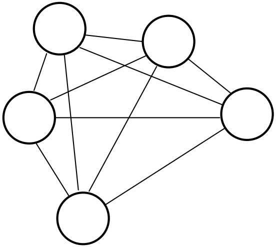
A jointly exponential-family model, with intractable normaliser.

**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"). May be exact or (more usually) approximate using Gibbs sampling or loopy BP.
- The M step requires gradients w.r.t. $Z$, which can be computed by averages under $P(\mathbf{s}|W, \mathbf{b})$ ("unclamped phase").

$$\nabla_W \log P(\mathbf{s}^V, \mathbf{s}^H) = \left\langle \mathbf{s}\mathbf{s}^\mathsf{T} \right\rangle_c - \left\langle \mathbf{s}\mathbf{s}^\mathsf{T} \right\rangle_u$$

# Learning in Boltzmann Machines

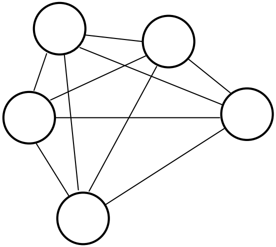


$$\log P(\mathbf{s}^V \mathbf{s}^H | W, \mathbf{b}) = \sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i - \log Z$$

with $Z = \sum_{\mathbf{s}} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i}$
Generalised (gradient M-step) EM requires parameter step

$$\Delta W_{ij} \propto \frac{\partial}{\partial W_{ij}} \Big\langle \log P(\mathbf{s}^V \mathbf{s}^H | W, \mathbf{b}) \Big\rangle_{P(\mathbf{s}^H | \mathbf{s}^V)}$$

# Learning in Boltzmann Machines



$$\log P(\mathbf{s}^V \mathbf{s}^H | W, \mathbf{b}) = \sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i - \log Z$$

with $Z = \sum_{\mathbf{s}} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i}$

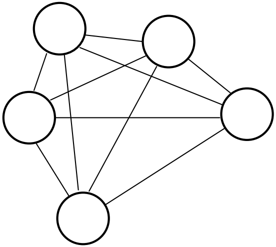
Generalised (gradient M-step) EM requires parameter step

$$\Delta W_{ij} \propto \frac{\partial}{\partial W_{ij}} \left\langle \log P(\mathbf{s}^V \mathbf{s}^H | W, \mathbf{b}) \right\rangle_{P(\mathbf{s}^H|\mathbf{s}^V)}$$

Write $\langle\rangle_c$ (clamped) for expectations under $P(\mathbf{s}|\mathbf{s}^V_{obs})$ (with $P(\mathbf{s}^V|\mathbf{s}^V_{obs}) = \prod \delta_{s^V_i, s^V_{i,obs}}$). Then

$$[\nabla_W \log P(\mathbf{s}^V, \mathbf{s}^H)]_{ij} = \frac{\partial}{\partial W_{ij}} \left[ \sum_{ij} W_{ij} \langle s_i s_j \rangle_c - \sum_i b_i \langle s_i \rangle_c - \log Z \right]$$

# Learning in Boltzmann Machines



$$\log P(\mathbf{s}^V \mathbf{s}^H | W, \mathbf{b}) = \sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i - \log Z$$

with $Z = \sum_{\mathbf{s}} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i}$

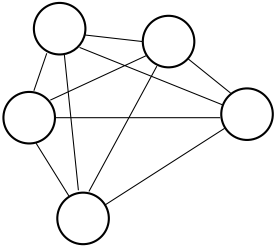
Generalised (gradient M-step) EM requires parameter step

$$\Delta W_{ij} \propto \frac{\partial}{\partial W_{ij}} \left\langle \log P(\mathbf{s}^V \mathbf{s}^H | W, \mathbf{b}) \right\rangle_{P(\mathbf{s}^H | \mathbf{s}^V)}$$

Write $\langle \rangle_c$ (clamped) for expectations under $P(\mathbf{s}|\mathbf{s}^V_{obs})$ (with $P(\mathbf{s}^V|\mathbf{s}^V_{obs}) = \prod \delta_{s^V_i, s^V_{i,obs}}$). Then

$$[\nabla_W \log P(\mathbf{s}^V, \mathbf{s}^H)]_{ij} = \frac{\partial}{\partial W_{ij}} \left[ \sum_{ij} W_{ij} \langle s_i s_j \rangle_c - \sum_i b_i \langle s_i \rangle_c - \log Z \right] = \langle s_i s_j \rangle_c - \frac{\partial}{\partial W_{ij}} \log Z$$

# Learning in Boltzmann Machines



$$\log P(\mathbf{s}^V\mathbf{s}^H|W,\mathbf{b}) = \sum_{ij} W_{ij}s_is_j - \sum_i b_is_i - \log Z$$

with $Z = \sum_{\mathbf{s}} e^{\sum_{ij} W_{ij}s_is_j - \sum_i b_is_i}$

Generalised (gradient M-step) EM requires parameter step

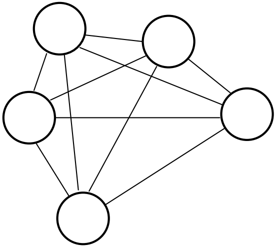
$$\Delta W_{ij} \propto \frac{\partial}{\partial W_{ij}} \Big\langle \log P(\mathbf{s}^V\mathbf{s}^H|W,\mathbf{b}) \Big\rangle_{P(\mathbf{s}^H|\mathbf{s}^V)}$$

Write $\langle\rangle_c$ (clamped) for expectations under $P(\mathbf{s}|\mathbf{s}^V_{obs})$ (with $P(\mathbf{s}^V|\mathbf{s}^V_{obs}) = \prod \delta_{s_i^V, s_{i,obs}^V}$). Then

$$[\nabla_W \log P(\mathbf{s}^V,\mathbf{s}^H)]_{ij} = \frac{\partial}{\partial W_{ij}} \Big[\textstyle\sum_{ij} W_{ij}\langle s_is_j\rangle_c - \sum_i b_i\langle s_i\rangle_c - \log Z\Big] = \langle s_is_j\rangle_c - \frac{\partial}{\partial W_{ij}} \log Z$$

$$= \langle s_is_j\rangle_c - \frac{1}{Z}\frac{\partial}{\partial W_{ij}} \sum_{\mathbf{s}} e^{\sum_{ij} W_{ij}s_is_j - \sum_i b_is_i}$$

# Learning in Boltzmann Machines



$$\log P(\mathbf{s}^V \mathbf{s}^H | W, \mathbf{b}) = \sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i - \log Z$$

with $Z = \sum_{\mathbf{s}} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i}$

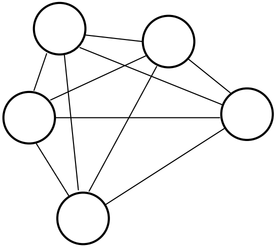
Generalised (gradient M-step) EM requires parameter step

$$\Delta W_{ij} \propto \frac{\partial}{\partial W_{ij}} \Big\langle \log P(\mathbf{s}^V \mathbf{s}^H | W, \mathbf{b}) \Big\rangle_{P(\mathbf{s}^H | \mathbf{s}^V)}$$

Write $\langle \rangle_c$ (clamped) for expectations under $P(\mathbf{s} | \mathbf{s}^V_{obs})$ (with $P(\mathbf{s}^V | \mathbf{s}^V_{obs}) = \prod \delta_{s^V_i, s^V_{i,obs}}$). Then

$$\begin{aligned}
[\nabla_W \log P(\mathbf{s}^V, \mathbf{s}^H)]_{ij} &= \frac{\partial}{\partial W_{ij}} \Big[ \textstyle\sum_{ij} W_{ij} \langle s_i s_j \rangle_c - \sum_i b_i \langle s_i \rangle_c - \log Z \Big] = \langle s_i s_j \rangle_c - \frac{\partial}{\partial W_{ij}} \log Z \\
&= \langle s_i s_j \rangle_c - \frac{1}{Z} \frac{\partial}{\partial W_{ij}} \sum_{\mathbf{s}} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i} \\
&= \langle s_i s_j \rangle_c - \sum_{\mathbf{s}} \frac{1}{Z} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i} s_i s_j
\end{aligned}$$

# Learning in Boltzmann Machines



$$\log P(\mathbf{s}^V \mathbf{s}^H | W, \mathbf{b}) = \sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i - \log Z$$

with $Z = \sum_{\mathbf{s}} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i}$

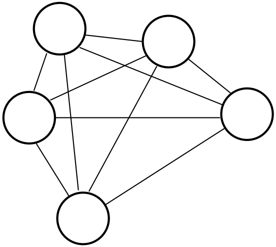
Generalised (gradient M-step) EM requires parameter step

$$\Delta W_{ij} \propto \frac{\partial}{\partial W_{ij}} \Big\langle \log P(\mathbf{s}^V \mathbf{s}^H | W, \mathbf{b}) \Big\rangle_{P(\mathbf{s}^H | \mathbf{s}^V)}$$

Write $\langle\rangle_c$ (clamped) for expectations under $P(\mathbf{s}|\mathbf{s}^V_{obs})$ (with $P(\mathbf{s}^V|\mathbf{s}^V_{obs}) = \prod \delta_{s^V_i, s^V_{i,obs}}$). Then

$$\begin{aligned}
[\nabla_W \log P(\mathbf{s}^V, \mathbf{s}^H)]_{ij} &= \frac{\partial}{\partial W_{ij}} \Big[ \textstyle\sum_{ij} W_{ij} \langle s_i s_j \rangle_c - \sum_i b_i \langle s_i \rangle_c - \log Z \Big] = \langle s_i s_j \rangle_c - \frac{\partial}{\partial W_{ij}} \log Z \\
&= \langle s_i s_j \rangle_c - \frac{1}{Z} \frac{\partial}{\partial W_{ij}} \sum_{\mathbf{s}} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i} \\
&= \langle s_i s_j \rangle_c - \sum_{\mathbf{s}} \frac{1}{Z} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i} s_i s_j \\
&= \langle s_i s_j \rangle_c - \sum_{\mathbf{s}} P(\mathbf{s}|W, \mathbf{b}) s_i s_j
\end{aligned}$$

# Learning in Boltzmann Machines



$$\log P(\mathbf{s}^V\mathbf{s}^H|W,\mathbf{b}) = \sum_{ij} W_{ij}s_is_j - \sum_i b_is_i - \log Z$$

with $Z = \sum_{\mathbf{s}} e^{\sum_{ij} W_{ij}s_is_j - \sum_i b_is_i}$

Generalised (gradient M-step) EM requires parameter step

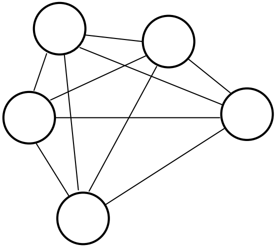
$$\Delta W_{ij} \propto \frac{\partial}{\partial W_{ij}} \left\langle \log P(\mathbf{s}^V\mathbf{s}^H|W,\mathbf{b}) \right\rangle_{P(\mathbf{s}^H|\mathbf{s}^V)}$$

Write $\langle\rangle_c$ (clamped) for expectations under $P(\mathbf{s}|\mathbf{s}^V_{obs})$ (with $P(\mathbf{s}^V|\mathbf{s}^V_{obs}) = \prod \delta_{s_i^V, s_{i,obs}^V}$). Then

$$\begin{aligned}
[\nabla_W \log P(\mathbf{s}^V,\mathbf{s}^H)]_{ij} &= \frac{\partial}{\partial W_{ij}}\left[\sum_{ij} W_{ij}\langle s_is_j\rangle_c - \sum_i b_i\langle s_i\rangle_c - \log Z\right] = \langle s_is_j\rangle_c - \frac{\partial}{\partial W_{ij}}\log Z \\
&= \langle s_is_j\rangle_c - \frac{1}{Z}\frac{\partial}{\partial W_{ij}}\sum_{\mathbf{s}} e^{\sum_{ij} W_{ij}s_is_j - \sum_i b_is_i} \\
&= \langle s_is_j\rangle_c - \sum_{\mathbf{s}}\frac{1}{Z} e^{\sum_{ij} W_{ij}s_is_j - \sum_i b_is_i} s_is_j \\
&= \langle s_is_j\rangle_c - \sum_{\mathbf{s}} P(\mathbf{s}|W,\mathbf{b})s_is_j \quad = \langle s_is_j\rangle_c - \langle s_is_j\rangle_u
\end{aligned}$$

with $\langle\rangle_u$ (unclamped) expectation under the current joint.

# Learning in Boltzmann Machines



$$\log P(\mathbf{s}^V \mathbf{s}^H | W, \mathbf{b}) = \sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i - \log Z$$

with $Z = \sum_{\mathbf{s}} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i}$

Generalised (gradient M-step) EM requires parameter step

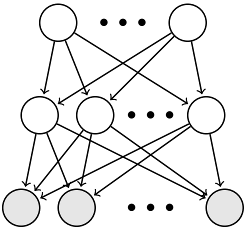
$$\Delta W_{ij} \propto \frac{\partial}{\partial W_{ij}} \Big\langle \log P(\mathbf{s}^V \mathbf{s}^H | W, \mathbf{b}) \Big\rangle_{P(\mathbf{s}^H | \mathbf{s}^V)}$$

Write $\langle\rangle_c$ (clamped) for expectations under $P(\mathbf{s}|\mathbf{s}^V_{obs})$ (with $P(\mathbf{s}^V | \mathbf{s}^V_{obs}) = \prod \delta_{s^V_i, s^V_{i,obs}}$). Then

$$\begin{aligned}
[\nabla_W \log P(\mathbf{s}^V, \mathbf{s}^H)]_{ij} &= \frac{\partial}{\partial W_{ij}} \left[ \textstyle\sum_{ij} W_{ij} \langle s_i s_j \rangle_c - \sum_i b_i \langle s_i \rangle_c - \log Z \right] = \langle s_i s_j \rangle_c - \frac{\partial}{\partial W_{ij}} \log Z \\
&= \langle s_i s_j \rangle_c - \frac{1}{Z} \frac{\partial}{\partial W_{ij}} \sum_{\mathbf{s}} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i} \\
&= \langle s_i s_j \rangle_c - \sum_{\mathbf{s}} \frac{1}{Z} e^{\sum_{ij} W_{ij} s_i s_j - \sum_i b_i s_i} s_i s_j \\
&= \langle s_i s_j \rangle_c - \sum_{\mathbf{s}} P(\mathbf{s} | W, \mathbf{b}) s_i s_j \quad = \langle s_i s_j \rangle_c - \langle s_i s_j \rangle_u
\end{aligned}$$

with $\langle\rangle_u$ (unclamped) expectation under the current joint. $\Rightarrow$ ExpFam moment matching, but requires simulation and gradient ascent.

# Sigmoid Belief Networks



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

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

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

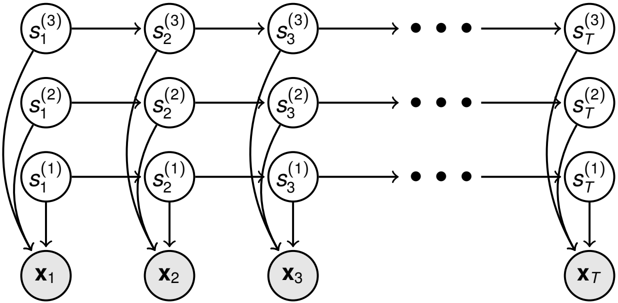
- parents most often grouped into layers
- logistic function of linear combination of parents
- "generative multilayer perceptron" ("neural network")

**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 either exactly or approximately using Gibbs sampling or mean field approximations. Or using a parallel 'recognition network' (the Helmholtz machine).
- Unlike Boltzmann machines, there is no partition function, so no need for an unclamped phase in the M step.

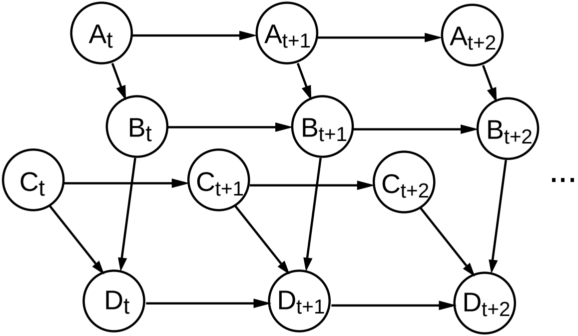
# Distributed state models

# Factorial Hidden Markov Models




- Hidden Markov models with many state variables (i.e. distributed state representation).
- Each state variable evolves independently.
- The state can capture many bits of information about the sequence (linear in the number of state variables).
- E step is typically intractable (due to explaining away in latent states).

# Dynamic Bayesian Networks




- Distributed HMM with structured dependencies amongst latent states.

# Latent Dirichlet Allocation

# Topic Modelling

**Topic modelling**: given a corpus of documents, find the “topics” they discuss.

# Topic Modelling

**Topic modelling**: given a corpus of documents, find the “topics” they discuss.

Example: consider abstracts of papers PNAS.

***Global climate change and mammalian species diversity in U.S. national parks***

*National parks and bioreserves are key conservation tools used to protect species and their habitats within the confines of fixed political boundaries. This inflexibility may be their ”Achilles’ heel” as conservation tools in the face of emerging global-scale environmental problems such as climate change. Global climate change, brought about by rising levels of greenhouse gases, threatens to alter the geographic distribution of many habitats and their component species....*

***The influence of large-scale wind power on global climate***

*Large-scale use of wind power can alter local and global climate by extracting kinetic energy and altering turbulent transport in the atmospheric boundary layer. We report climate-model simulations that address the possible climatic impacts of wind power at regional to global scales by using two general circulation models and several parameterizations of the interaction of wind turbines with the boundary layer....*

***Twentieth century climate change: Evidence from small glaciers***

*The relation between changes in modern glaciers, not including the ice sheets of Greenland and Antarctica, and their climatic environment is investigated to shed light on paleoglacier evidence of past climate change and for projecting the effects of future climate warming on cold regions of the world. Loss of glacier volume has been more or less continuous since the 19th century, but it is not a simple adjustment to the end of an ”anomalous” Little Ice Age....*

# Topic Modelling

Example topics discovered from PNAS abstracts (each topic represented in terms of the top 5 most common words in that topic).

| | | | | | |
|---|---|---|---|---|---|
| 217 | 274 | 126 | 63 | 200 | 209 |
| INSECT | SPECIES | GENE | STRUCTURE | FOLDING | NUCLEAR |
| MYB | PHYLOGENETIC | VECTOR | ANGSTROM | NATIVE | NUCLEUS |
| PHEROMONE | EVOLUTION | VECTORS | CRYSTAL | PROTEIN | LOCALIZATION |
| LENS | EVOLUTIONARY | EXPRESSION | RESIDUES | STATE | CYTOPLASM |
| LARVAE | SEQUENCES | TRANSFER | STRUCTURES | ENERGY | EXPORT |
| 42 | 2 | 280 | 15 | 64 | 102 |
| NEURAL | SPECIES | SPECIES | CHROMOSOME | CELLS | TUMOR |
| DEVELOPMENT | GLOBAL | SELECTION | REGION | CELL | CANCER |
| DORSAL | CLIMATE | EVOLUTION | CHROMOSOMES | ANTIGEN | TUMORS |
| EMBRYOS | CO2 | GENETIC | KB | LYMPHOCYTES | HUMAN |
| VENTRAL | WATER | POPULATIONS | MAP | CD4 | CELLS |
| 112 | 210 | 201 | 165 | 142 | 222 |
| HOST | SYNAPTIC | RESISTANCE | CHANNEL | PLANTS | CORTEX |
| BACTERIAL | NEURONS | RESISTANT | CHANNELS | PLANT | BRAIN |
| BACTERIA | POSTSYNAPTIC | DRUG | VOLTAGE | ARABIDOPSIS | SUBJECTS |
| STRAINS | HIPPOCAMPAL | DRUGS | CURRENT | TOBACCO | TASK |
| SALMONELLA | SYNAPSES | SENSITIVE | CURRENTS | LEAVES | AREAS |
| 39 | 105 | 221 | 270 | 55 | 114 |
| THEORY | HAIR | LARGE | TIME | FORCE | POPULATION |
| TIME | MECHANICAL | SCALE | SPECTROSCOPY | SURFACE | POPULATIONS |
| SPACE | MB | DENSITY | NMR | MOLECULES | GENETIC |
| GIVEN | SENSORY | OBSERVED | SPECTRA | SOLUTION | DIVERSITY |
| PROBLEM | EAR | OBSERVATIONS | TRANSFER | SURFACES | ISOLATES |
| | | 109 | 120 | | |
| | | RESEARCH | AGE | | |
| | | NEW | OLD | | |
| | | INFORMATION | AGING | | |
| | | UNDERSTANDING | LIFE | | |
| | | PAPER | YOUNG | | |

# Recap: Beta Distributions

Recall the Bayesian coin toss example.

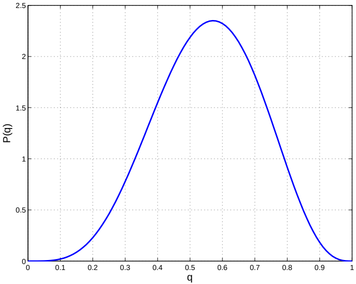
$$P(H|q) = q \qquad\qquad P(T|q) = 1 - q$$

The probability of a sequence of coin tosses is:

$$P(HHTT \cdots HT|q) = q^{\#\text{heads}}(1-q)^{\#\text{tails}}$$

A conjugate prior for $q$ is the Beta distribution:

$$P(q) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} q^{a-1}(1-q)^{b-1} \qquad\qquad a, b \geq 0$$



## Dirichlet Distributions

Imagine a Bayesian dice throwing example.

$$P(1|\mathbf{q}) = q_1 \quad P(2|\mathbf{q}) = q_2 \quad P(3|\mathbf{q}) = q_3 \quad P(4|\mathbf{q}) = q_4 \quad P(5|\mathbf{q}) = q_5 \quad P(6|\mathbf{q}) = q_6$$

with $q_i \geq 0, \sum_i q_i = 1$.

# Dirichlet Distributions

Imagine a Bayesian dice throwing example.

$$P(1|\mathbf{q}) = q_1 \quad P(2|\mathbf{q}) = q_2 \quad P(3|\mathbf{q}) = q_3 \quad P(4|\mathbf{q}) = q_4 \quad P(5|\mathbf{q}) = q_5 \quad P(6|\mathbf{q}) = q_6$$

with $q_i \geq 0, \sum_i q_i = 1$. The probability of a sequence of dice throws is:

$$P(34156 \cdots 12|\boldsymbol{q}) = \prod_{i=1}^{6} q_i^{\#\text{ face } i}$$

# Dirichlet Distributions

Imagine a Bayesian dice throwing example.

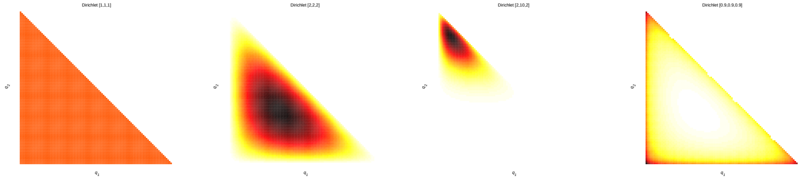
$$P(1|\mathbf{q}) = q_1 \quad P(2|\mathbf{q}) = q_2 \quad P(3|\mathbf{q}) = q_3 \quad P(4|\mathbf{q}) = q_4 \quad P(5|\mathbf{q}) = q_5 \quad P(6|\mathbf{q}) = q_6$$

with $q_i \geq 0, \sum_i q_i = 1$. The probability of a sequence of dice throws is:

$$P(34156 \cdots 12|\boldsymbol{q}) = \prod_{i=1}^{6} q_i^{\#\text{ face } i}$$

A conjugate prior for $\boldsymbol{q}$ is the Dirichlet distribution:

$$P(\boldsymbol{q}) = \frac{\Gamma(\sum_i a_i)}{\prod_i \Gamma(a_i)} \prod_i q_i^{a_i - 1} \qquad q_i \geq 0, \sum_i q_i = 1 \qquad a_i \geq 0$$

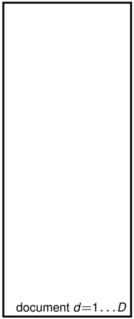


# Latent Dirichlet Allocation

Each document is a sequence of words, we model it using a mixture model by ignoring the sequential nature—“bag-of-words” assumption.

# Latent Dirichlet Allocation

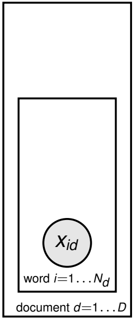
Each document is a sequence of words, we model it using a mixture model by ignoring the sequential nature—"bag-of-words" assumption.




- For each document:

## Latent Dirichlet Allocation

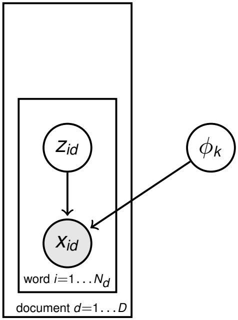
Each document is a sequence of words, we model it using a mixture model by ignoring the sequential nature—"bag-of-words" assumption.




- For each document:
  - generate words iid:

# Latent Dirichlet Allocation

Each document is a sequence of words, we model it using a mixture model by ignoring the sequential nature—"bag-of-words" assumption.

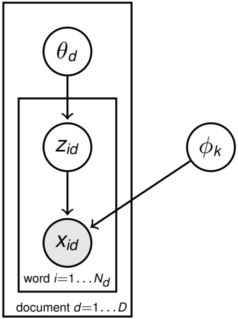



- For each document:
  - generate words iid:
    - draw word from a topic-specific dist:

$$x_{id} \sim \text{Discrete}(\phi_{z_{id}})$$

# Latent Dirichlet Allocation

Each document is a sequence of words, we model it using a mixture model by ignoring the sequential nature—"bag-of-words" assumption.




- For each document:
  - generate words iid:
    - draw topic from a document-specific dist:

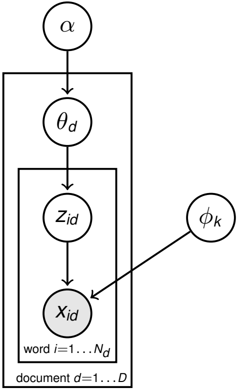
      $$z_{id} \sim \text{Discrete}(\boldsymbol{\theta}_d)$$

    - draw word from a topic-specific dist:

      $$x_{id} \sim \text{Discrete}(\phi_{z_{id}})$$

# Latent Dirichlet Allocation

Each document is a sequence of words, we model it using a mixture model by ignoring the sequential nature—"bag-of-words" assumption.




- For each document:
  - draw a distribution over topics

    $$\boldsymbol{\theta}_d \sim \text{Dir}(\alpha, \ldots, \alpha)$$

  - generate words iid:
    - draw topic from a document-specific dist:

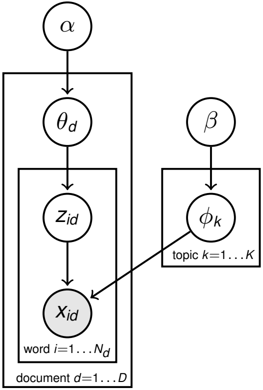
      $$z_{id} \sim \text{Discrete}(\boldsymbol{\theta}_d)$$

    - draw word from a topic-specific dist:

      $$x_{id} \sim \text{Discrete}(\phi_{z_{id}})$$

# Latent Dirichlet Allocation

Each document is a sequence of words, we model it using a mixture model by ignoring the sequential nature—“bag-of-words” assumption.




- Draw topic distributions from a prior

$$\phi_k \sim \text{Dir}(\beta, \ldots, \beta)$$

- For each document:
  - draw a distribution over topics

$$\boldsymbol{\theta}_d \sim \text{Dir}(\alpha, \ldots, \alpha)$$

  - generate words iid:
    - draw topic from a document-specific dist:

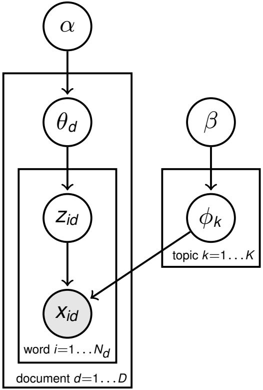
$$z_{id} \sim \text{Discrete}(\boldsymbol{\theta}_d)$$

    - draw word from a topic-specific dist:

$$x_{id} \sim \text{Discrete}(\phi_{z_{id}})$$

# Latent Dirichlet Allocation

Each document is a sequence of words, we model it using a mixture model by ignoring the sequential nature—"bag-of-words" assumption.




- Draw topic distributions from a prior

$$\phi_k \sim \text{Dir}(\beta, \ldots, \beta)$$

- For each document:
  - draw a distribution over topics

$$\boldsymbol{\theta}_d \sim \text{Dir}(\alpha, \ldots, \alpha)$$

  - generate words iid:
    - draw topic from a document-specific dist:

$$z_{id} \sim \text{Discrete}(\boldsymbol{\theta}_d)$$

    - draw word from a topic-specific dist:

$$x_{id} \sim \text{Discrete}(\phi_{z_{id}})$$

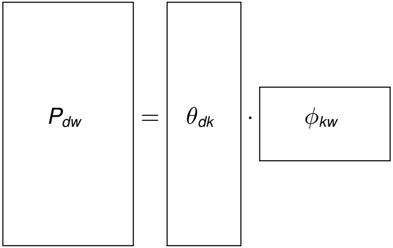
Multiple mixtures of discrete distributions, sharing the same set of components (topics).

# Latent Dirichlet Allocation as Matrix Decomposition

Let $N_{dw}$ be the number of times word $w$ appears in document $d$, and $P_{dw}$ is the probability of word $w$ appearing in document $d$.

$$p(N|P) = \prod_{dw} P_{dw}^{N_{dw}} \quad \text{likelihood term}$$

$$P_{dw} = \sum_{k} p(\text{pick topic } k) p(\text{pick word } w|k) = \sum_{k=1}^{K} \theta_{dk}\phi_{kw}$$




This decomposition is similar to PCA and factor analysis, but not Gaussian. Related to non-negative matrix factorisation (NMF).

# Latent Dirichlet Allocation

- Exact inference in latent Dirichlet allocation is intractable, and typically either variational or Markov chain Monte Carlo approximations are deployed.
- Latent Dirichlet allocation is an example of a mixed membership model from statistics.
- Latent Dirichlet allocation has also been applied to computer vision, social network modelling, natural language processing. . .
- Generalizations:
  - Relax the bag-of-words assumption (e.g. a Markov model).
  - Model changes in topics through time.
  - Model correlations among occurrences of topics.
  - Model authors, recipients, multiple corpora.
  - Cross modal interactions (images and tags).
  - Nonparametric generalisations.

# Nonlinear Dimensionality Reduction / Manifold Recovery

## Nonlinear Dimensionality Reduction

We can see matrix factorisation methods as performing linear dimensionaliy reduction.

There are many ways to generalise PCA and FA to deal with data which lie on a nonlinear manifold:

- Nonlinear autoencoders
- Generative topographic mappings (GTM) and Kohonen self-organising maps (SOM)
- Multi-dimensional scaling (MDS)
- Kernel PCA (based on MDS representation)
- Isomap
- Locally linear embedding (LLE)
- Stochastic Neighbour Embedding
- Gaussian Process Latent Variable Models (GPLVM)

# Another view of PCA: matching inner products

We have viewed PCA as providing a decomposition of the covariance or scatter matrix $S$. We obtain similar results if we approximate the Gram matrix:

$$\text{minimise} \quad \mathcal{E} = \sum_{ij} (G_{ij} - \mathbf{y}_i \cdot \mathbf{y}_j)^2$$

for $\mathbf{y} \in \mathbb{R}^k$.

That is, look for a $k$-dimensional embedding in which dot products (which depend on lengths, and angles) are preserved as well as possible.

We will see that this is also equivalent to preserving distances between points.

# Another view of PCA: matching inner products

Consider the eigendecomposition of $G$:

$$G = U\Lambda U^{\mathsf{T}} \quad \text{arranged so} \quad \lambda_1 \geq \cdots \geq \lambda_m \geq 0$$

The best rank-$k$ approximation $G \approx Y^{\mathsf{T}} Y$ is given by:

$$\begin{aligned} Y^{\mathsf{T}} &= [U]_{1:m,1:k}[\Lambda^{1/2}]_{1:k,1:k}; \\ &= [U\Lambda^{1/2}]_{1:m,1:k} \end{aligned}$$

$$Y = [\Lambda^{1/2}U^{\mathsf{T}}]_{1:k,1:m}$$

$$\begin{bmatrix} \text{———} & \sqrt{\lambda_1}\,\mathbf{u}_1^{\mathsf{T}} & \text{———} \\ \text{———} & \sqrt{\lambda_2}\,\mathbf{u}_2^{\mathsf{T}} & \text{———} \\ & \vdots & \\ \text{———} & \sqrt{\lambda_k}\,\mathbf{u}_k^{\mathsf{T}} & \text{———} \\ & \vdots & \\ \text{———} & \sqrt{\lambda_m}\,\mathbf{u}_m^{\mathsf{T}} & \text{———} \end{bmatrix}$$

# Another view of PCA: matching inner products

Consider the eigendecomposition of $G$:

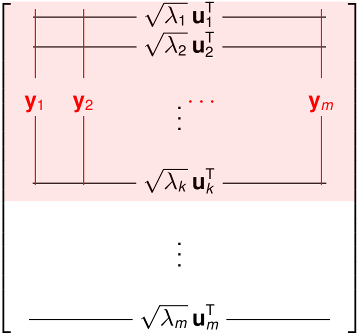
$$G = U \Lambda U^{\top} \quad \text{arranged so} \quad \lambda_1 \geq \cdots \geq \lambda_m \geq 0$$

The best rank-$k$ approximation $G \approx Y^{\top} Y$ is given by:

$$Y^{\top} = [U]_{1:m,1:k} [\Lambda^{1/2}]_{1:k,1:k};$$

$$= [U \Lambda^{1/2}]_{1:m,1:k}$$

$$Y = [\Lambda^{1/2} U^{\top}]_{1:k,1:m}$$



# Another view of PCA: matching inner products

Consider the eigendecomposition of $G$:

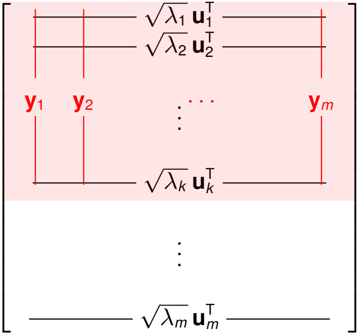
$$G = U \Lambda U^{\mathsf{T}} \quad \text{arranged so} \quad \lambda_1 \geq \cdots \geq \lambda_m \geq 0$$

The best rank-$k$ approximation $G \approx Y^{\mathsf{T}} Y$ is given by:

$$Y^{\mathsf{T}} = [U]_{1:m,1:k} [\Lambda^{1/2}]_{1:k,1:k};$$

$$= [U \Lambda^{1/2}]_{1:m,1:k}$$

$$Y = [\Lambda^{1/2} U^{\mathsf{T}}]_{1:k,1:m}$$




The same operations can be performed on the kernel Gram matrix ⇒ Kernel PCA.

# Multidimensional Scaling

Suppose all we were given were distances or symmetric "dissimilarities" $\Delta_{ij}$.

$$\Delta = \begin{bmatrix} 0 & \Delta_{12} & \Delta_{13} & \Delta_{14} \\ \Delta_{12} & 0 & \Delta_{23} & \Delta_{24} \\ \Delta_{13} & \Delta_{23} & 0 & \Delta_{34} \\ \Delta_{14} & \Delta_{24} & \Delta_{34} & 0 \end{bmatrix}$$

**Goal**: Find vectors $\mathbf{y}_i$ such that $\|\mathbf{y}_i - \mathbf{y}_j\| \approx \Delta_{ij}$.

This is called **Multidimensional Scaling (MDS)**.

# Metric MDS

Assume the dissimilarities represent Euclidean distances between points in some high-D space.

$$\Delta_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| \text{ with } \sum_i \mathbf{x}_i = \mathbf{0}.$$

We have:

$$\begin{aligned}
\Delta_{ij}^2 &= \|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2 - 2\mathbf{x}_i \cdot \mathbf{x}_j \\
\sum_k \Delta_{ik}^2 &= m\|\mathbf{x}_i\|^2 + \sum_k \|\mathbf{x}_k\|^2 - \mathbf{0} \\
\sum_k \Delta_{kj}^2 &= \sum_k \|\mathbf{x}_k\|^2 + m\|\mathbf{x}_j\|^2 - \mathbf{0} \\
\sum_{kl} \Delta_{kl}^2 &= 2m \sum_k \|\mathbf{x}_k\|^2
\end{aligned}$$

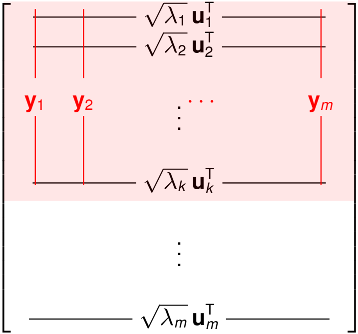
$$\Rightarrow G_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j \; = \frac{1}{2}\left(\frac{1}{m}\sum_k (\Delta_{ik}^2 + \Delta_{kj}^2) - \frac{1}{m^2}\sum_{kl}\Delta_{kl}^2 - \Delta_{ij}^2\right)$$

# Metric MDS and eigenvalues

We will actually minimize the error in the dot products:

$$\mathcal{E} = \sum_{ij} (G_{ij} - \mathbf{y}_i \cdot \mathbf{y}_j)^2$$

As in PCA, this is given by the top slice of the eigenvector matrix.



## Interpreting MDS

$$G = \frac{1}{2}\left(\frac{1}{m}(\Delta^2 1 + 1\Delta^2) - \Delta^2 - \frac{1}{m^2}1^T\Delta^2 1\right)$$

$$G = U\Lambda U^T; \qquad Y = [\Lambda^{1/2}U^T]_{1:k,1:m}$$

(1 is a matrix of ones.)

- **Eigenvectors.** Ordered, scaled and truncated to yield low-dimensional embedded points $\mathbf{y}_i$.
- **Eigenvalues.** Measure how much each dimension contributes to dot products.
- **Estimated dimensionality.** Number of significant (nonnegative – negative possible if $\Delta_{ij}$ are not metric) eigenvalues.

# MDS and PCA

**Dual matrices:**

| | | |
|---|---|---|
| $S = \frac{1}{m} X X^{\mathsf{T}}$ | scatter matrix | $(n \times n)$ |
| $G = X^{\mathsf{T}} X$ | Gram matrix | $(m \times m)$ |

- **Same eigenvalues** up to a constant factor.
- **Equivalent on metric data**, but MDS can run on non-metric dissimilarities.
- **Computational cost** is different.
  - PCA: $O((m + k)n^2)$
  - MDS: $O((n + k)m^2)$

# Non-metric MDS

MDS can be generalised to permit a monotonic mapping:

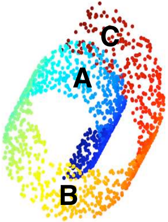
$$\Delta_{ij} \rightarrow g(\Delta_{ij}),$$

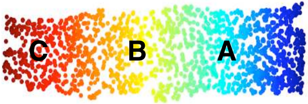
even if this violates metric rules (like the triangle inequality).

This can introduce a non-linear warping of the manifold.

# But

**Rank ordering of Euclidean distances is NOT preserved in "manifold learning".**



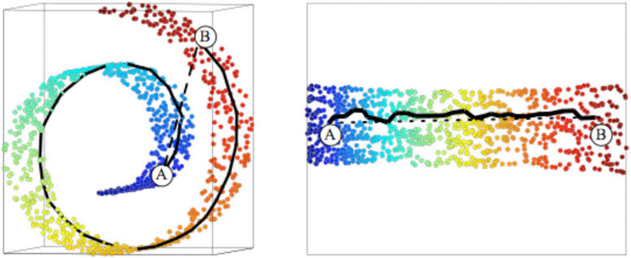




$d(A,C) < d(A,B)$

$d(A,C) > d(A,B)$

# Isomap

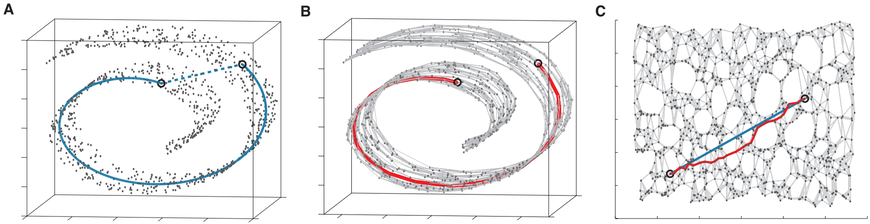
**Idea:** try to trace distance along the manifold. Use geodesic instead of (transformed) Euclidean distances in MDS.




- preserves local structure
- estimates "global" structure
- preserves information (MDS)

# Stages of Isomap

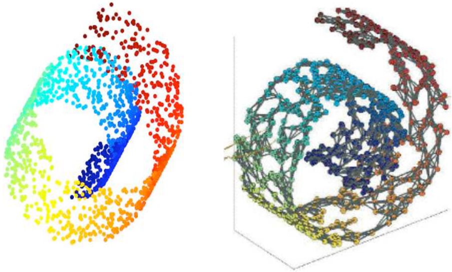
1. Identify neighbourhoods around each point (local points, assumed to be local on the manifold). Euclidean distances are preserved within a neighbourhood.
2. For points outside the neighbourhood, estimate distances by hopping between points within neighbourhoods.
3. Embed using MDS.



## Step 1: Adjacency graph

First we construct a graph linking each point to its neighbours.

- vertices represent input points
- undirected edges connect neighbours (weight = Euclidean distance)



Forms a discretised approximation to the submanifold, assuming:

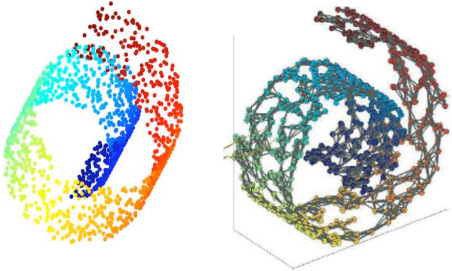
- Graph is singly-connected.
- Graph neighborhoods reflect manifold neighborhoods. No “short cuts”.

Defining the neighbourhood is critical: $k$-nearest neighbours, inputs within a ball of radius $r$, prior knowledge.

## Step 2: Geodesics

Estimate distances by shortest path in graph.

$$\Delta_{ij} = \min_{\text{path}(\mathbf{x}_i,\mathbf{x}_j)} \left\{ \sum_{e_i \in \text{path}(\mathbf{x}_i,\mathbf{x}_j)} \delta_i \right\}$$

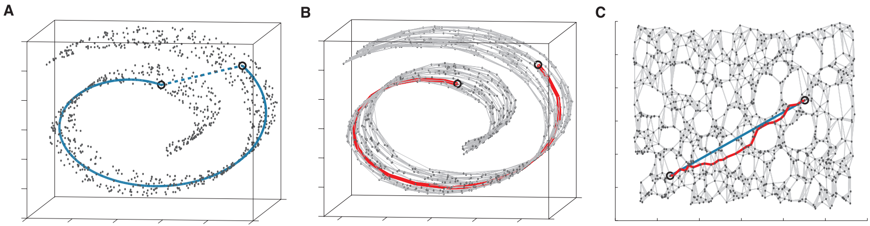


- Standard graph problem. Solved by Dijkstra's algorithm (and others).
- Better estimates for denser sampling.
- Short cuts very dangerous ("average" path distance?) .

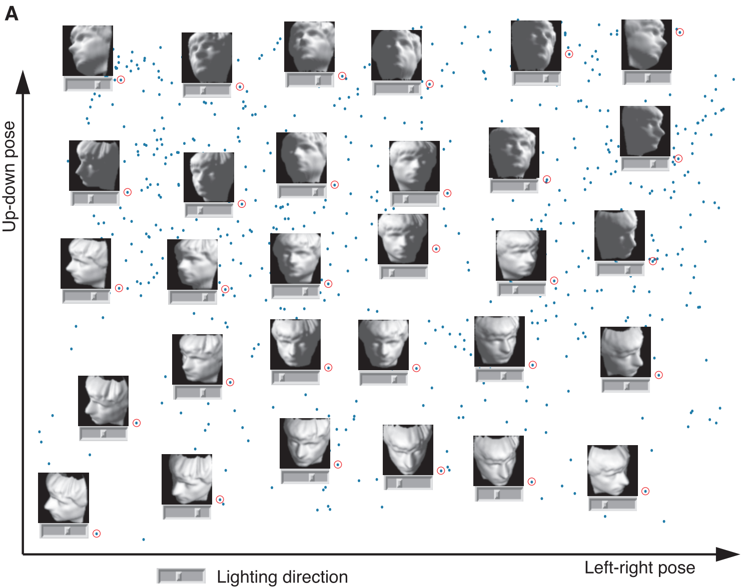
# Step 3: Embed

Embed using metric MDS (path distances obey the triangle inequality)

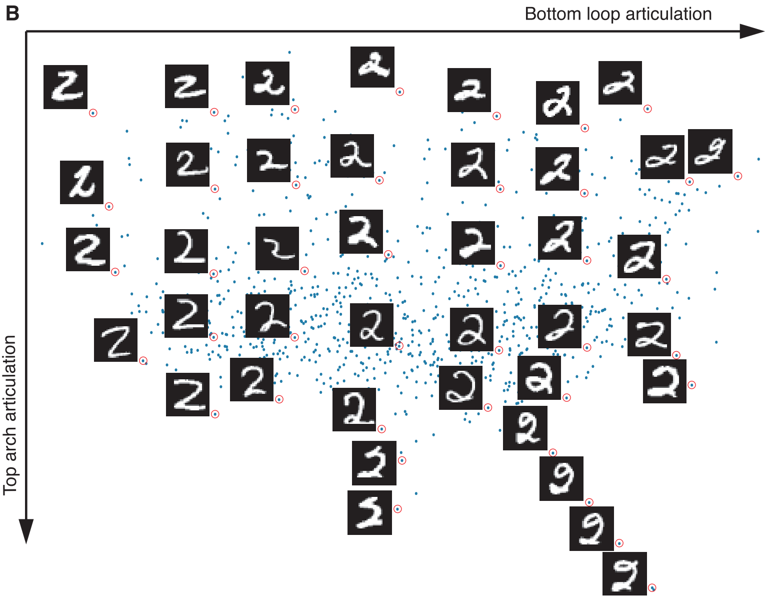
- Eigenvectors of Gram matrix yield low-dimensional embedding.
- Number of significant eigenvalues estimates dimensionality.



# Isomap example 1

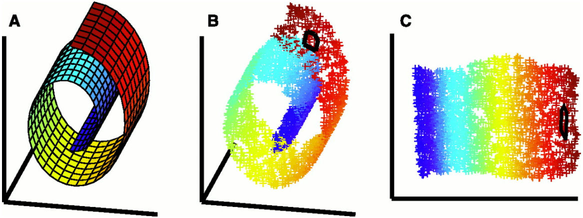


# Isomap example 2



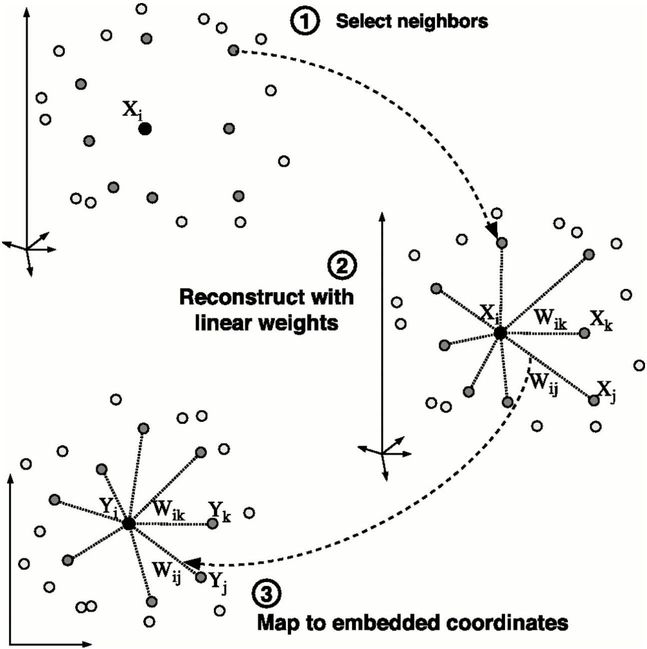
# Locally Linear Embedding (LLE)

MDS and isomap preserve local and global (estimated, for isomap) distances. PCA preserves local and global structure.
**Idea:** estimate local (linear) structure of manifold. Preserve this as well as possible.



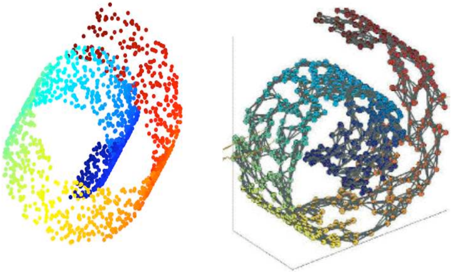

- preserves local structure (not just distance)
- not explicitly global
- preserves only local information

# Stages of LLE



## Step 1: Neighbourhoods

Just as in isomap, we first define neighbouring points for each input. Equivalent to the isomap graph, but we won't need the graph structure.



Forms a discretised approximation to the submanifold, assuming:

- Graph is singly-connected — although will "work" if not.
- Neighborhoods reflect manifold neighborhoods. No "short cuts".

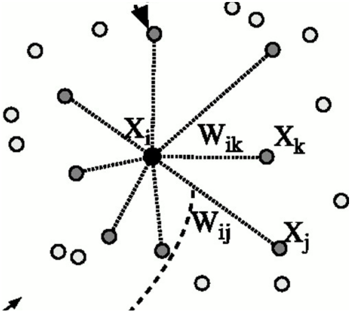
Defining the neighbourhood is critical: $k$-nearest neighbours, inputs within a ball of radius $r$, prior knowledge.

## Step 2: Local weights

Estimate local weights to minimize error

$$\Phi(W) = \sum_i \left\| \mathbf{x}_i - \sum_{j \in \mathrm{Ne}(i)} W_{ij} \mathbf{x}_j \right\|^2$$

$$\sum_{j \in \mathrm{Ne}(i)} W_{ij} = 1$$




- Linear regression – under- or over-constrained depending on $|\mathrm{Ne}(i)|$.
- Local structure – optimal weights are invariant to rotation, translation and scaling.
- Short cuts less dangerous (one in many).

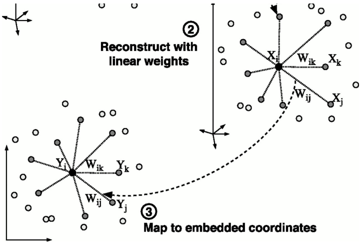
# Step 3: Embed

Minimise reconstruction errors in **y**-space under the same weights:

$$\psi(Y) = \sum_i \left\| \mathbf{y}_i - \sum_{j \in \mathrm{Ne}(i)} W_{ij} \mathbf{y}_j \right\|^2$$

subject to:

$$\sum_i \mathbf{y}_i = \mathbf{0}; \qquad \sum_i \mathbf{y}_i \mathbf{y}_i^{\mathsf{T}} = mI$$




We can re-write the cost function in quadratic form:

$$\psi(Y) = \sum_{ij} \Psi_{ij} [Y^{\mathsf{T}} Y]_{ij} \text{ with } \Psi = (I - W)^{\mathsf{T}} (I - W)$$

Minimise by setting $Y$ to equal the bottom $2 \ldots k+1$ eigenvectors of $\Psi$. (Bottom eigenvector always **1** – discard due to centering constraint)

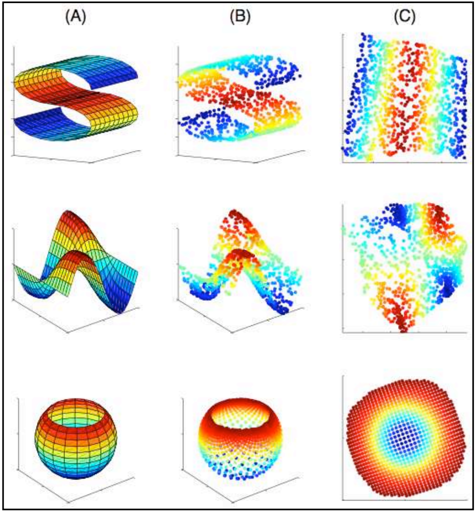
# LLE example 1

## Surfaces

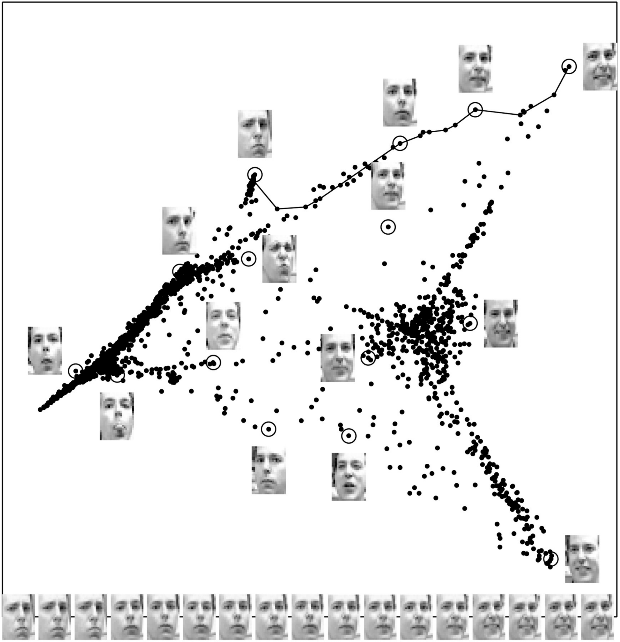
**N=1000**
**inputs**

**k=8**
**nearest neighbors**

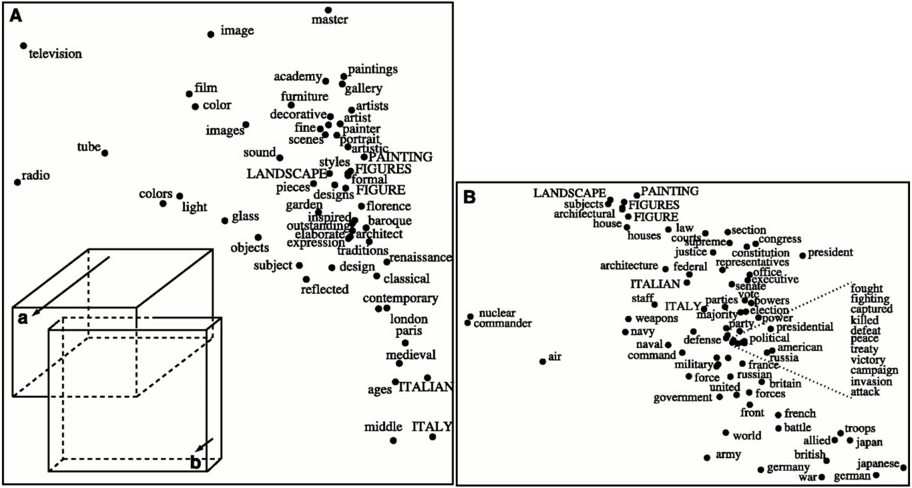
**D=3**
**d=2**
**dimensions**



# LLE example 2



# LLE example 3



# LLE and Isomap

**Many similarities**

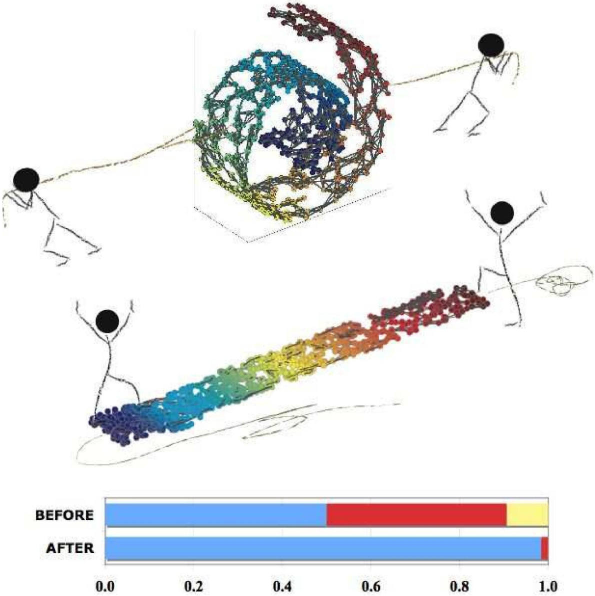
- Graph-based, spectral methods
- No local optima

**Essential differences**

- LLE does not estimate dimensionality
- Isomap can be shown to be consistent; no theoretical guarantees for LLE.
- LLE diagonalises a sparse matrix – more efficient than isomap.
- Local weights vs. local & global distances.

# Maximum Variance Unfolding

Unfold neighbourhood graph preserving local structure.



# Maximum Variance Unfolding

Unfold neighbourhood graph preserving local structure.

1. Build the neighbourhood graph.
2. Find $\{\mathbf{y}_i\} \subset \mathbb{R}^n$ (points in **high-D** space) with maximum variance, preserving local distances. Let $K_{ij} = \mathbf{y}_i^\top \mathbf{y}_j$. Then:

   Maximise $\mathrm{Tr}\left[K\right]$ subject to:

   $$\begin{aligned} \textstyle\sum_{ij} K_{ij} &= 0 && \text{(centered)} \\ K &\succeq 0 && \text{(positive definite)} \\ \underbrace{K_{ii} - 2K_{ij} + K_{jj}}_{\|\mathbf{y}_i - \mathbf{y}_j\|^2} &= \|\mathbf{x}_i - \mathbf{x}_j\|^2 \text{ for } j \in \mathrm{Ne}(i) && \text{(locally metric)} \end{aligned}$$

   This is a **semi-definite program**: convex optimisation with unique solution.
3. Embed $\mathbf{y}_i$ in $\mathbb{R}^k$ using linear methods (PCA/MDS).

# Stochastic Neighbour Embedding

Softer "probabilistic" notions of neighbourhood and consistency.

High-D "transition" probabilities:

$$p_{j|i} = \frac{e^{-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}}{\sum_{k \neq i} e^{-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_k\|^2/\sigma^2}} \quad \text{for } j \neq i, \qquad p_{i|i} = 0$$

Find $\{\mathbf{y}_i\} \subset \mathbb{R}^k$ to:

$$\text{minimise} \sum_{ij} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \qquad \text{with } q_{j|i} = \frac{e^{-\frac{1}{2}\|\mathbf{y}_i - \mathbf{y}_j\|^2}}{\sum_{k \neq i} e^{-\frac{1}{2}\|\mathbf{y}_i - \mathbf{y}_k\|^2}}.$$

Nonconvex optimisation is initialisation dependent.

Scale $\sigma$ plays a similar role to neighbourhood definition:

- Fixed $\sigma$: resembles a fixed-radius ball.
- Choose $\sigma_i$ to maintain consistent entropy in $p_{j|i}$ of $\log_2 k$: similar to $k$-nearest neighbours.

# SNE variants

- Symmetrise probabilities ($p_{ij} = p_{ji}$)

$$p_{ij} = \frac{e^{-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2}}{\sum_{k \neq l} e^{-\frac{1}{2}\|\mathbf{x}_l - \mathbf{x}_k\|^2/\sigma^2}} \quad \text{for } j \neq i$$

- Gaussian Process Latent Variable Models. Lawrence. Advances in Neural Information Processing Systems, 2004.
  Define $q_{ij}$ analagously, optimise joint KL.

- Heavy-tailed embedding distributions allow embedding to lower dimensions than true manifold:

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

  Student-t distribution defines "t-SNE".

Focus is on visualisation, rather than manifold discovery.

# Gaussian Process Latent Variable Models

Recap: probabilistic PCA

$$\mathbf{y}_i | \mathbf{x}_i, \Lambda \sim \mathcal{N}(\Lambda \mathbf{x}_i, \beta^{-1} I)$$
$$\mathbf{x}_i \sim \mathcal{N}(0, I)$$

## Gaussian Process Latent Variable Models

Recap: probabilistic PCA

$$\mathbf{y}_i | \mathbf{x}_i, \Lambda \sim \mathcal{N}(\Lambda \mathbf{x}_i, \beta^{-1} I)$$
$$\mathbf{x}_i \sim \mathcal{N}(0, I)$$

Usually: compute posterior over $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$, maximizing likelihood over $\Lambda$.

# Gaussian Process Latent Variable Models

Recap: probabilistic PCA

$$\mathbf{y}_i | \mathbf{x}_i, \Lambda \sim \mathcal{N}(\Lambda \mathbf{x}_i, \beta^{-1} I)$$
$$\mathbf{x}_i \sim \mathcal{N}(0, I)$$

Usually: compute posterior over $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$, maximizing likelihood over $\Lambda$.

Suppose we know the values of the latent $X$, then we can integrate out $\Lambda$ (c.f. linear regression), giving a conditional probability of $Y = [\mathbf{y}_1 \ldots \mathbf{y}_N]^\top$:

$$\Lambda \sim \mathcal{N}(0, \alpha^{-1} I)$$
$$p(Y|X) \sim |2\pi K|^{-\frac{D}{2}} \exp\left(-\frac{1}{2}\mathrm{Tr}[K^{-1} Y Y^\top]\right) \qquad K = \alpha X X^\top + \beta I$$

# Gaussian Process Latent Variable Models

Recap: probabilistic PCA

$$\mathbf{y}_i | \mathbf{x}_i, \Lambda \sim \mathcal{N}(\Lambda \mathbf{x}_i, \beta^{-1} I)$$
$$\mathbf{x}_i \sim \mathcal{N}(0, I)$$

Usually: compute posterior over $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$, maximizing likelihood over $\Lambda$.

Suppose we know the values of the latent $X$, then we can integrate out $\Lambda$ (c.f. linear regression), giving a conditional probability of $Y = [\mathbf{y}_1 \ldots \mathbf{y}_N]^\top$:

$$\Lambda \sim \mathcal{N}(0, \alpha^{-1} I)$$
$$p(Y|X) \sim |2\pi K|^{-\frac{D}{2}} \exp\left(-\frac{1}{2}\mathrm{Tr}[K^{-1} Y Y^\top]\right) \qquad K = \alpha X X^\top + \beta I$$

This is just $D$ independent Gaussian processes, one for each dimension of $Y$! Each Gaussian process describes a mapping from latent space $\mathbf{x}$ to one dimension of $\mathbf{y}$.

# Gaussian Process Latent Variable Models

Recap: probabilistic PCA

$$\mathbf{y}_i | \mathbf{x}_i, \Lambda \sim \mathcal{N}(\Lambda \mathbf{x}_i, \beta^{-1} I)$$
$$\mathbf{x}_i \sim \mathcal{N}(0, I)$$

Usually: compute posterior over $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$, maximizing likelihood over $\Lambda$.

Suppose we know the values of the latent $X$, then we can integrate out $\Lambda$ (c.f. linear regression), giving a conditional probability of $Y = [\mathbf{y}_1 \ldots \mathbf{y}_N]^\top$:

$$\Lambda \sim \mathcal{N}(0, \alpha^{-1} I)$$
$$p(Y|X) \sim |2\pi K|^{-\frac{D}{2}} \exp\left(-\frac{1}{2}\mathrm{Tr}[K^{-1} Y Y^\top]\right) \qquad K = \alpha X X^\top + \beta I$$

This is just $D$ independent Gaussian processes, one for each dimension of $Y$! Each Gaussian process describes a mapping from latent space $\mathbf{x}$ to one dimension of $\mathbf{y}$.

Replacing the linear kernel with nonlinear kernels gives nonlinear mappings—nonlinear dimensionality reduction.

# Gaussian Process Latent Variable Models

Recap: probabilistic PCA

$$\mathbf{y}_i | \mathbf{x}_i, \Lambda \sim \mathcal{N}(\Lambda \mathbf{x}_i, \beta^{-1} I)$$
$$\mathbf{x}_i \sim \mathcal{N}(0, I)$$

Usually: compute posterior over $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$, maximizing likelihood over $\Lambda$.

Suppose we know the values of the latent $X$, then we can integrate out $\Lambda$ (c.f. linear regression), giving a conditional probability of $Y = [\mathbf{y}_1 \ldots \mathbf{y}_N]^\top$:

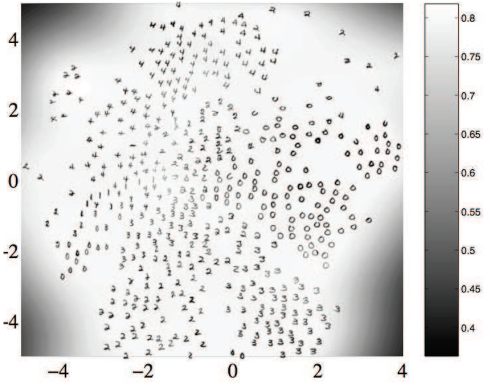
$$\Lambda \sim \mathcal{N}(0, \alpha^{-1} I)$$
$$p(Y|X) \sim |2\pi K|^{-\frac{D}{2}} \exp\left(-\frac{1}{2}\mathrm{Tr}[K^{-1} Y Y^\top]\right) \qquad K = \alpha X X^\top + \beta I$$

This is just $D$ independent Gaussian processes, one for each dimension of $Y$! Each Gaussian process describes a mapping from latent space $\mathbf{x}$ to one dimension of $\mathbf{y}$.

Replacing the linear kernel with nonlinear kernels gives nonlinear mappings—nonlinear dimensionality reduction.

But now dependence on $X$ is complicated—instead of computing a posterior over $X$ we must find point values that maximise the likelihood (jointly with the hyperparameters), or use a variational approximation (cf also the Locally-Linear Latent Variable Model) .

# Gaussian Process Latent Variable Models

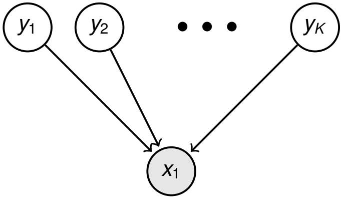


# Intractability

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

There are three (main) reasons:

- Distributions may have complicated forms (e.g. non-linearities in generative model).
- "Explaining away": observing the value of a child induces dependencies amongst its parents.




- Even with simple models, Bayesian computation of the full posterior over both latent variables and parameters is made complicated by the strong coupling between latent variables and parameters.

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

# Approximate Inference

- Linearisation: Approximate nonlinearities by Taylor series expansion about a point (e.g. the approximate mean or mode of the hidden variable distribution). Linear approximations are particularly useful since Gaussian distributions are closed under linear transformations (e.g., EKF). Also Laplace's approximation.
- Monte Carlo Sampling: Approximate posterior distribution over unobserved variables by a set of random samples. We often need Markov chain Monte carlo or sequential Monte Carlo methods to sample from difficult distributions.
- Variational Methods: Approximate the hidden variable posterior $p(H)$ with a tractable form $q(H)$, such that $\mathbf{KL}[q||p]$ is minimised. This gives a lower bound on the likelihood that can be maximised with respect to the parameters of $q(H)$.
- Local Message Passing Methods: Approximate the hidden variable posterior $p(H)$ with a tractable form $q(H)$ or with a set of locally consistent tractable forms by other means (loopy belief propagation, expectation propagation).
- Recognition Models and Autoencoders: Approximate the hidden variable posterior distribution using an explicit *bottom-up* recognition model/network.

# References


- Pattern Classification. Duda, Hart and Stork. Wiley, 2000.
- A Unifying Review of Linear Gaussian Models. Roweis and Ghaharamani. Neural Computation, 1999.
- Independent Component Analysis. Hyvarinen, Karhunan and Oja. John Wiley and Sons, 2001.
- Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images. Olshausen & Field Nature, 1996.
- A Learning Algorithm for Boltzmann Machines. Ackley, Hinton and Sejnowski. Cognitive Science, 1985.
- Connectionist Learning of Belief Networks. Neal. Artificial Intelligence, 1992.
- Latent Dirichlet Allocation. Blei, Ng and Jordan. Journal of Machine Learning Research, 2003.
- Factorial Hidden Markov Models. Ghahramani and Jordan. Machine Learning, 1997.
- Dynamic Bayesian Networks: Representation, Inference and Learning. Kevin Murphy. PhD Thesis, 2002.

## References


- Isomap. Tenenbaum, de Silva & Langford, Science, **290**(5500):2319–23 (2000).
- LLE. Roweis & Saul, Science, **290**(5500):2323–6 (2000).
- Laplacian Eigenmaps. Belkin & Niyogi, Neural Comput **23**(6):1373–96 (2003).
- Hessian LLE. Donoho & Grimes, PNAS **100**(10): 5591–6 (2003).
- Maximum variance unfolding. Weinberger & Saul, Int J Comput Vis **70**(1):77–90 (2006).
- Conformal eigenmaps. Sha & Saul ICML **22**:785–92 (2005).
- SNE Hinton & Roweis, NIPS, 2002; t-SNE van der Maaten & Hinton, JMLR, 9:2579–2605, 2008.
- Gaussian Process Latent Variable Models Lawrence. Advances in Neural Information Processing Systems, 2004.
- Locally-Linear Latent Variable Models Park et al. Advances in Neural Information Processing Systems, 2015.


More at: `http://www.gatsby.ucl.ac.uk/~maneesh/dimred/`