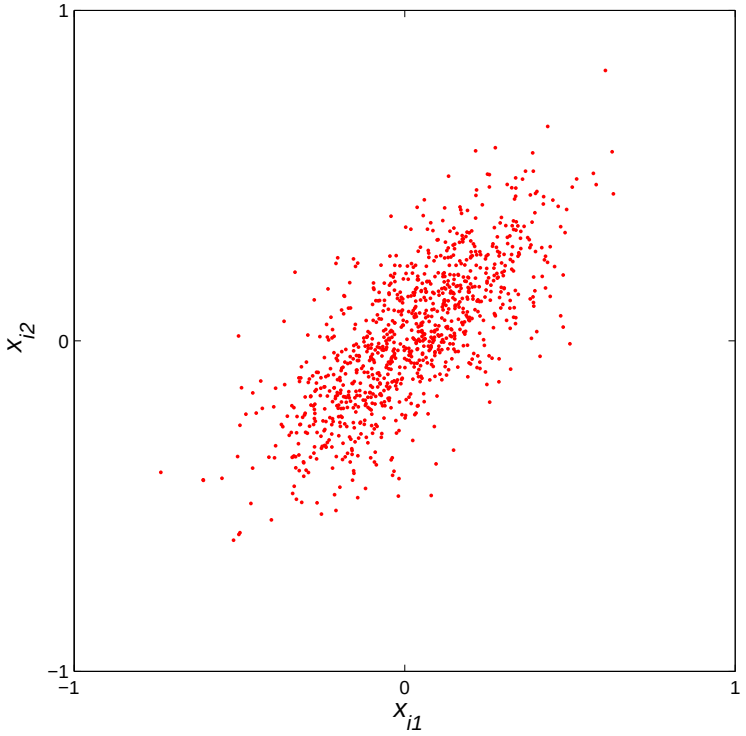
Probabilistic and Bayesian Machine Learning

Day 2: Simple Models

Yee Whye Teh
ywteh@gatsby.ucl.ac.uk

Gatsby Computational Neuroscience Unit
University College London

Multi-Variate Gaussian



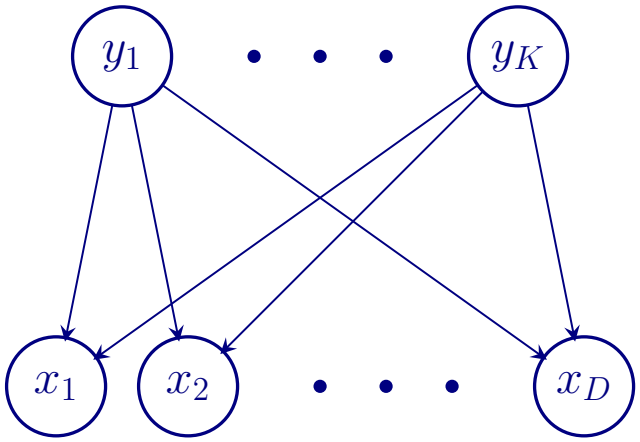
Assume:

- we have a data set $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
- each data point is a vector of D features: $\mathbf{x}_i = [x_{i1} \ldots x_{iD}]^\top$
- the data points are i.i.d. (independent and identically distributed).

One of the simplest probabilistic models is the multi-variate Gaussian (or normal) distribution. It models the **mean** of the data and the **correlations** between the D features in the data.

$$p(\mathbf{x}|\mu, \Sigma) = |2\pi\Sigma|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)\right\}$$

Factor Analysis, Probabilistic Principal Components Analysis



Observed vector data $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}; \mathbf{x}_i \in \mathbb{R}^D$

Assumed latent variables $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}; \mathbf{y}_i \in \mathbb{R}^K$

Linear generative model: $x_d = \sum_{k=1}^{K} \Lambda_{dk}\, y_k + \epsilon_d$

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

So, model for observations **x** is still Gaussian with:

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}; 0, I) \qquad p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}; \Lambda\mathbf{y}, \Psi)$$

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

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

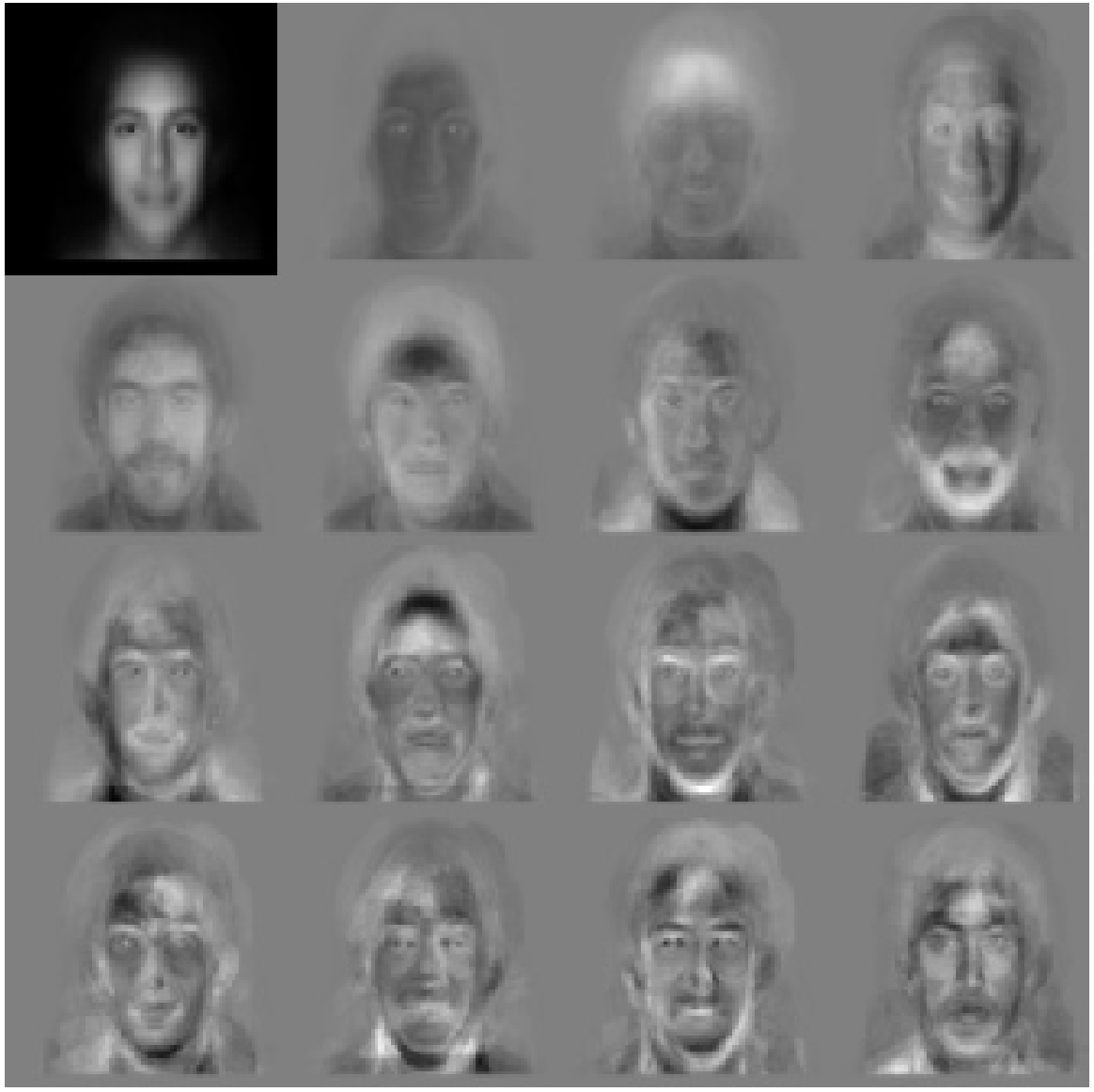
Probabilistic PCA: equivariance noise $\Psi_{dd} = \psi$.

Factor analysis: diagonal Ψ.

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

Psychometrics: factors of personality and intelligence.

Example principal components: Eigenfaces



from www-white.media.mit.edu/vismod/demos/facerec/basic.html

Matrix Factorization

Probabilistic PCA can be cast as a matrix factorization model.

Let $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ be a matrix with $\mathbf{x}_i$ in column i.
Similarly $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_N]$ be matrix of latent variables.

Generative model is:

$$X = \Lambda Y + \epsilon$$

With ϵ being matrix of iid Gaussian noise.

The model being symmetric in Λ and Y there is no reason to interpret them as different. Bayesian approach places prior over Λ and treat Λ and Y both as latent variables to be inferred from X.

Collaborative filtering: X is matrix of ratings, with X_{di} being rating of user i for item d, then $\Lambda_{d:}$ and $Y_{:i}$ are K dimensional latent properties of items and users to be inferred, with

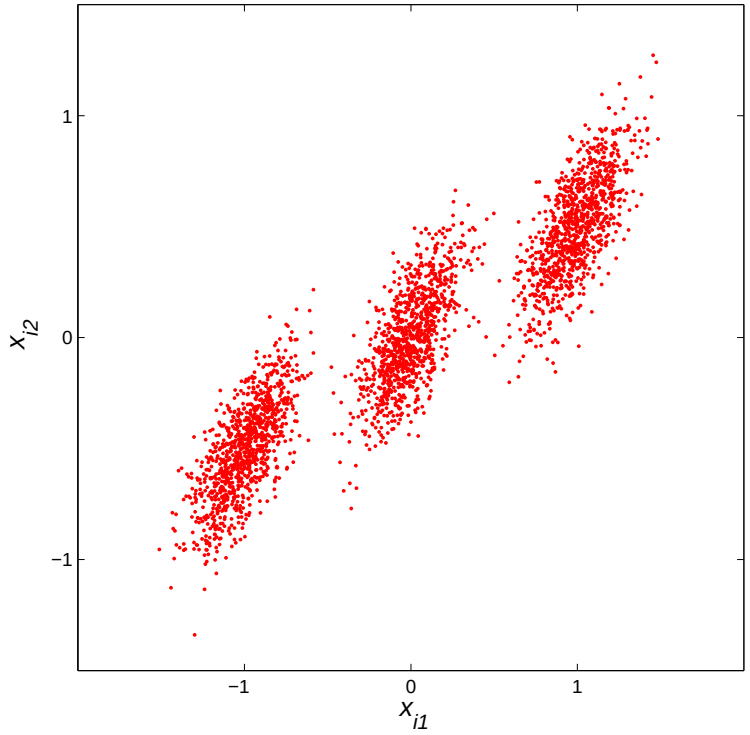
$$X_{di} = \sum_{k=1}^{K} \Lambda_{d:} Y_{:i} + \epsilon_{di}$$

Singular value decomposition (SVD) produces ML solution if X fully observed.

Limitations of Gaussian, FA and PPCA models

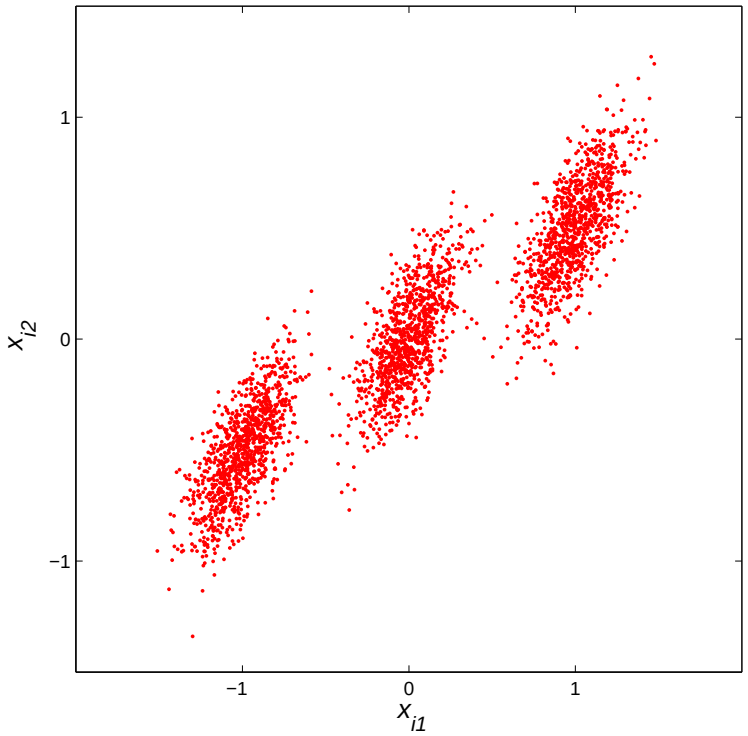
- Gaussian, FA and PCA models are easy to understand and use in practice.
- They are a convenient way to find interesting directions in very high dimensional data sets, eg as preprocessing
- Their problem is that they make very strong assumptions about the distribution of the data, only the mean and variance of the data are taken into account.

The class of densities which can be modelled is too restrictive.



By using mixtures of simple distributions, such as Gaussians, we can expand the class of densities greatly.

Mixture Distributions



A mixture distribution has a single discrete latent variable:

$$s_i \overset{\text{iid}}{\sim} Discrete(\boldsymbol{\pi})$$
$$\mathbf{x}_i \mid s_i = m \sim \mathcal{P}(\theta_m)$$

Mixtures arise naturally when observations from different sources have been collated.
They can also be used to approximate arbitrary distributions.

The Mixture Likelihood

The mixture model is

$$s_i \overset{\text{iid}}{\sim} Discrete(\boldsymbol{\pi})$$
$$\mathbf{x}_i \mid s_i = m \sim \mathcal{P}(\theta_m)$$

Under the discrete distribution

$$p(s_i = m) = \pi_m; \qquad \pi_m \geq 0, \sum_{m=1}^{k} \pi_m = 1$$

Thus, the probability (density) at a single data point $\mathbf{x}_i$ is

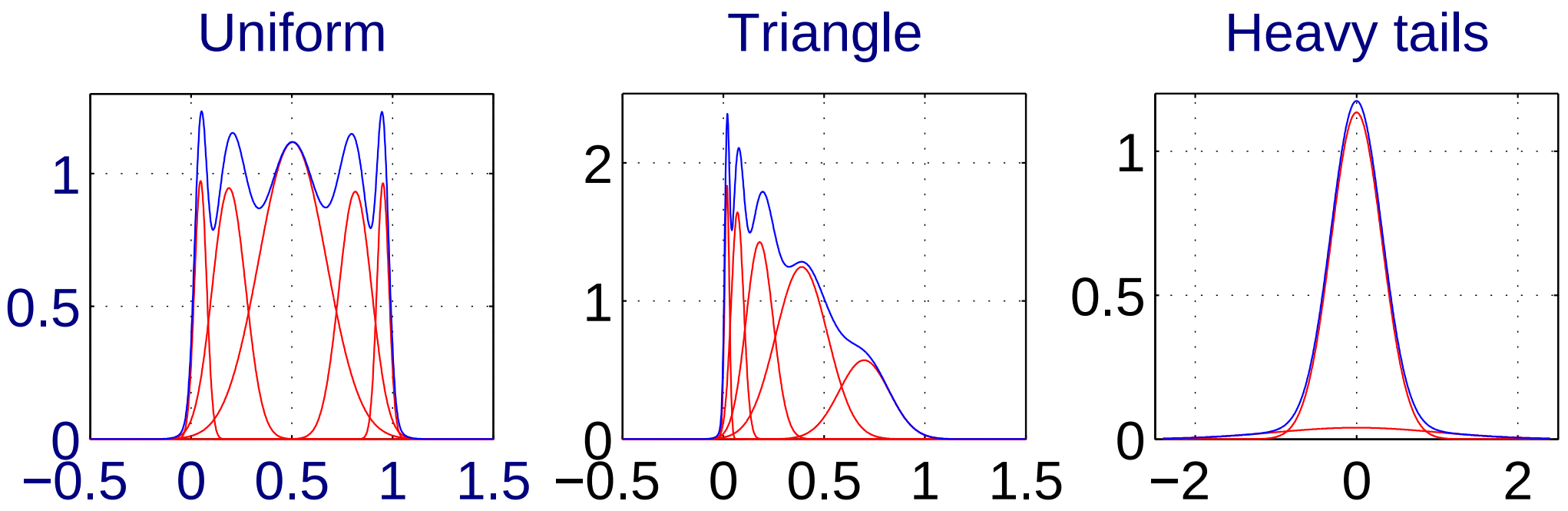
$$\begin{aligned} p(\mathbf{x}_i) &= \sum_{m=1}^{k} p(s_i = m) p(\mathbf{x}_i \mid s_i = m) \\ &= \sum_{m=1}^{k} \pi_m \mathcal{P}(\mathbf{x}_i; \theta_m) \end{aligned}$$

The mixture distribution (density) is a convex combination (or weighted average) of the component distributions (densities).

Approximation with a Mixture of Gaussians (MoG)

The component densities may be viewed as elements of a basis which can be combined to approximate arbitrary distributions.

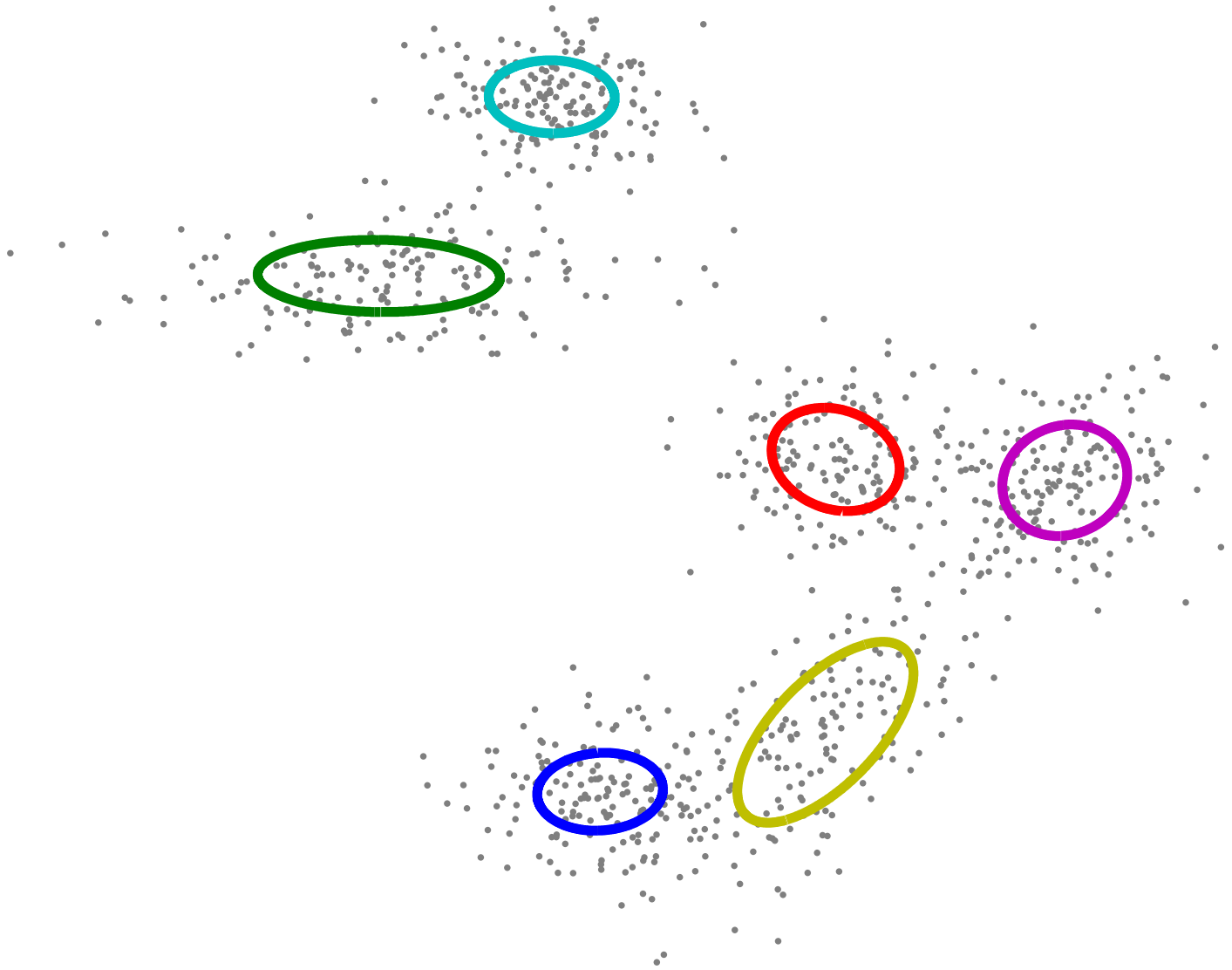
Here are examples where non-Gaussian densities are modelled (aproximated) as a mixture of Gaussians. The red curves show the (weighted) Gaussians, and the blue curve the resulting density.



Given enough mixture components we can model (almost) any density (as accurately as desired), but still only need to work with the well-known Gaussian form.

Clustering with a MoG

Clustering with a MoG



Clustering with a MoG

In clustering applications, the latent variable s_i represents the (unknown) identity of the cluster to which the ith observation belongs.

Thus, the latent distribution gives the prior probability of a data point coming from each cluster.

$$p(s_i = m \mid \pi) = \pi_m$$

Data from the mth cluster are distributed according to the mth component:

$$p(\mathbf{x}_i \mid s_i = m) = \mathcal{P}(\mathbf{x}_i; \theta_m)$$

Once we observe a data point, the posterior probability distribution for the cluster it belongs to is

$$p(s_i = m \mid \mathbf{x}_i) = \frac{\mathcal{P}(\mathbf{x}_i; \theta_m)\pi_m}{\sum_m \mathcal{P}(\mathbf{x}_i; \theta_m)\pi_m}$$

This is often called the responsibility of the mth cluster for the ith data point.

Modeling time series

Consider a sequence of observations:

$$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_t$$

which are **not iid**.
For example:

- Sequence of images
- Speech signals, English sentences
- Stock prices
- Kinematic variables in a robot
- Sensor readings from an industrial process
- Amino acids, DNA, etc. . .

Goal: To build a probabilistic model of the data $p(\mathbf{x}_1, \ldots, \mathbf{x}_t)$. This can be used to:

- Predict $p(\mathbf{x}_t | \mathbf{x}_1, \ldots, \mathbf{x}_{t-1})$
- Detect abnormal/changed behaviour (if $p(\mathbf{x}_t, \mathbf{x}_{t+1}, \ldots | \mathbf{x}_1, \ldots, \mathbf{x}_{t-1})$ small)
- Recover underlying/latent/hidden causes

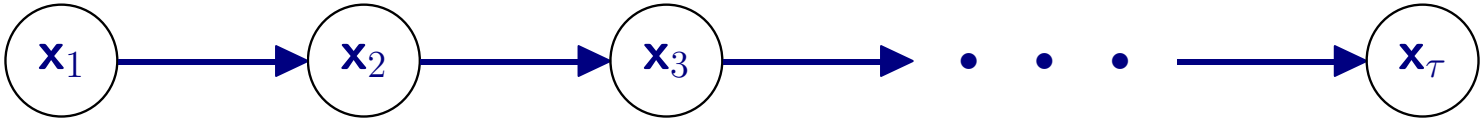
Markov models

In general:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_t) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_3|\mathbf{x}_1, \mathbf{x}_2) \cdots p(\mathbf{x}_t|\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_{t-1})$$

First-order Markov model:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_t) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1) \cdots p(\mathbf{x}_t|\mathbf{x}_{t-1})$$

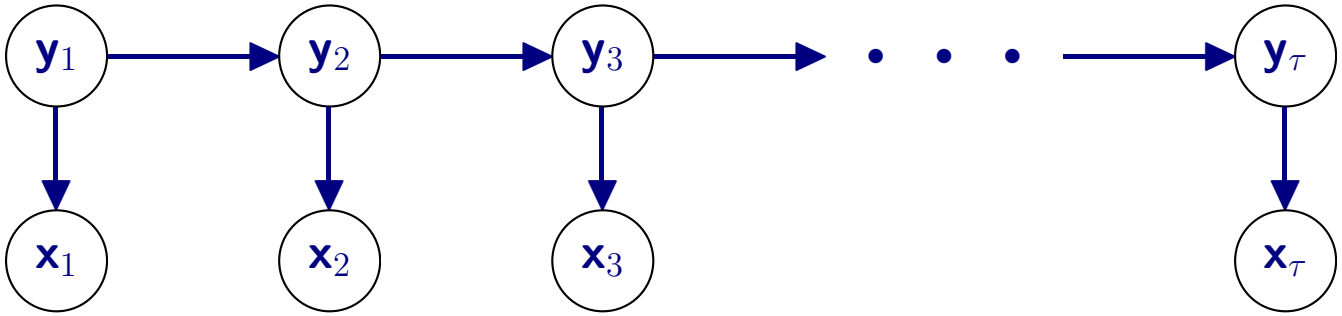


The term *Markov* refers to a conditional independence relationship. In this case, the Markov property is that, given the present observation ($\mathbf{x}_t$), the future ($\mathbf{x}_{t+1}, \ldots$) is independent of the past ($\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}$).

Second-order Markov model:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_t) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1) \cdots p(\mathbf{x}_{t-1}|\mathbf{x}_{t-3}, \mathbf{x}_{t-2})p(\mathbf{x}_t|\mathbf{x}_{t-2}, \mathbf{x}_{t-1})$$

Causal structure and latent variables



Speech recognition:

- **y** - underlying phonemes or words
- **x** - acoustic waveform

Vision:

- **y** - object identities, poses, illumination
- **x** - image pixel values

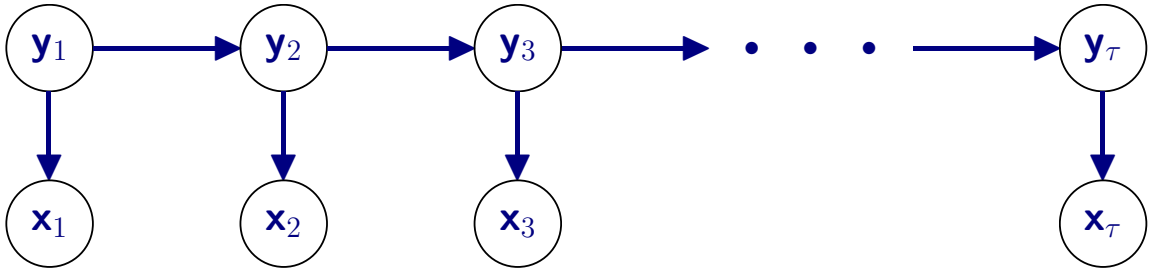
Industrial Monitoring:

- **y** - current state of molten steel in caster
- **x** - temperature and pressure sensor readings

Two frequently-used tractable models:

- Linear-Gaussian state-space models
- Hidden Markov models

Linear-Gaussian state-space models (SSMs)



Joint probability factorizes:

$$p(\mathbf{y}_{1:\tau}, \mathbf{x}_{1:\tau}) = p(\mathbf{y}_1)p(\mathbf{x}_1|\mathbf{y}_1) \prod_{t=2}^{\tau} p(\mathbf{y}_t|\mathbf{y}_{t-1})p(\mathbf{x}_t|\mathbf{y}_t)$$

where $\mathbf{y}_t$ and $\mathbf{x}_t$ are both real-valued vectors, and $\mathbf{z}_{1:\tau} \equiv \mathbf{z}_1, \ldots, \mathbf{z}_\tau$.

In a linear Gaussian SSM all conditional distributions are linear and Gaussian:

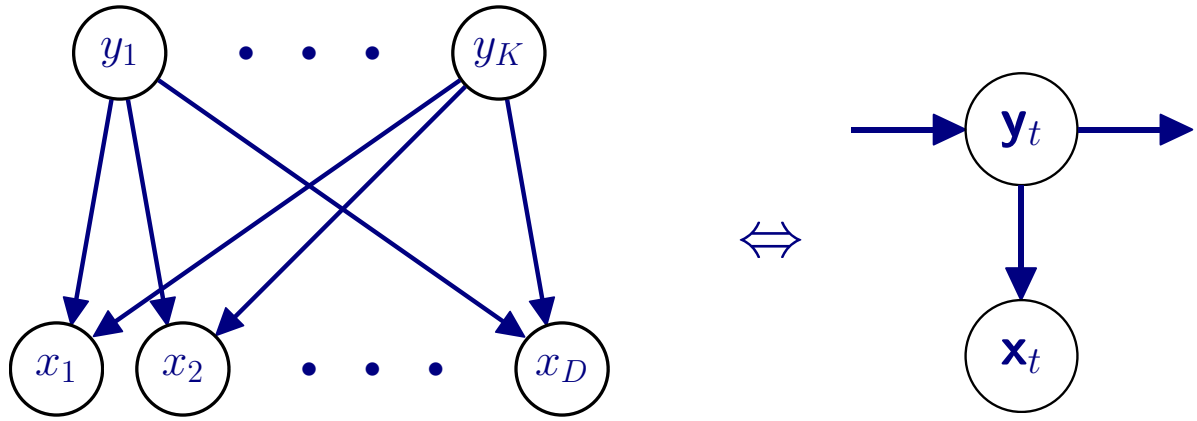
Output equation:	$\mathbf{x}_t = C\mathbf{y}_t + \mathbf{v}_t$
State dynamics equation:	$\mathbf{y}_t = A\mathbf{y}_{t-1} + \mathbf{w}_t$

where $\mathbf{v}_t$ and $\mathbf{w}_t$ are uncorrelated zero-mean multivariate Gaussian noise vectors.

Also assume $\mathbf{y}_1$ is multivariate Gaussian. The joint distribution over all variables $\mathbf{x}_{1:\tau}, \mathbf{y}_{1:\tau}$ is (one big) multivariate Gaussian. Why?

These models are also known as stochastic linear dynamical systems, Kalman filter models.

From factor analysis to state space models



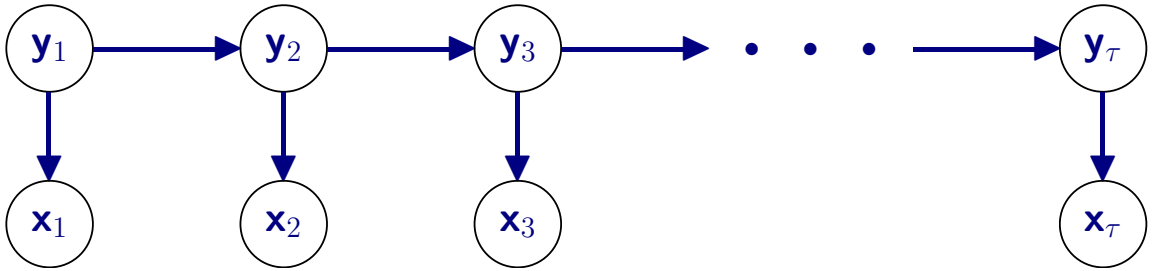
Factor analysis: $x_i = \sum_{j=1}^{K} \Lambda_{ij}\, y_j + \epsilon_i$ vs SSM output equation: $x_{t,i} = \sum_{j=1}^{K} C_{ij}\, y_{t,j} + v_i$.

Interpretation 1:

In both models the observations are linearly related to the hidden factors (state-variables) and all variables are Gaussian.

Linear Gaussian state-space models can therefore be seen as a dynamical generalization of factor analysis where $y_{t,j}$ can depend linearly on $y_{t-1,k}$.

Linear dynamical systems

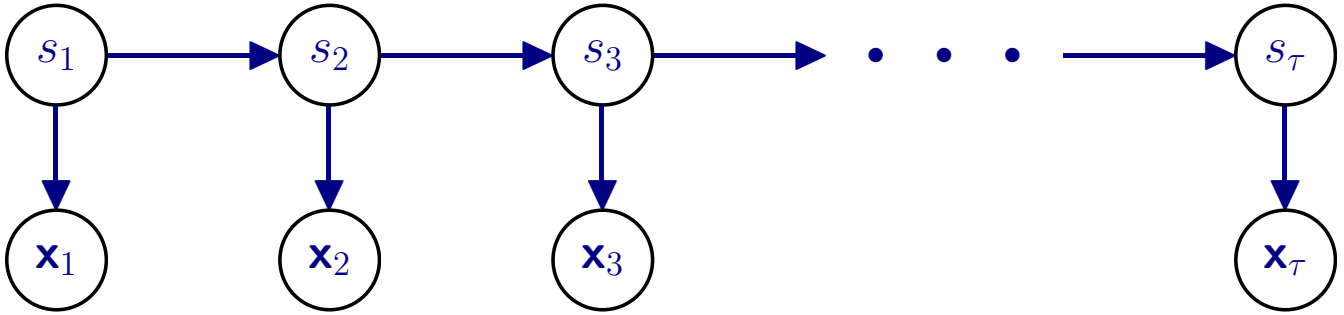


Interpretation 2:

Markov chain with linear Gaussian dynamics $\mathbf{y}_{t-1} \rightarrow \mathbf{y}_t$.

Observation variables $\mathbf{x}_t$ are a linear projection of latent variables $\mathbf{y}_t$, with Gaussian observation noise.

Hidden Markov models



Joint probability factorizes:

$$p(s_{1:\tau}, \mathbf{x}_{1:\tau}) = p(s_1)p(\mathbf{x}_1|s_1) \prod_{t=2}^{\tau} p(s_t|s_{t-1})p(\mathbf{x}_t|s_t)$$

Discrete hidden states $s_t \in \{1 \ldots, K\}$, while outputs $\mathbf{x}_t$ can be discrete or continuous.

Generative process:

1. A first-order Markov chain generates the hidden state sequence (path):

   initial state probs: $\pi_j = p(s_1 = j)$ transition matrix: $T_{ij} = p(s_{t+1} = j|s_t = i)$

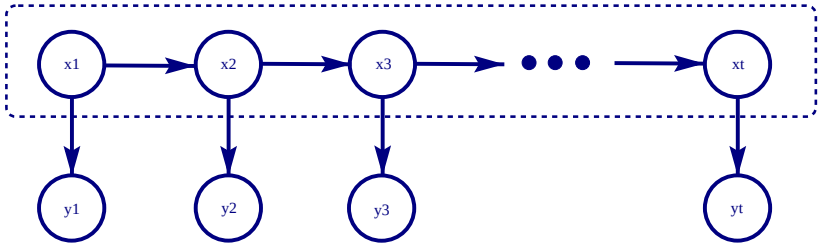
2. A set of emission (output) distributions $A_j(\cdot)$ (one per state) converts this state path into a sequence of observations $\mathbf{x}_t$.

$$\begin{aligned} A_j(\mathbf{x}) &= p(\mathbf{x}_t = \mathbf{x}|s_t = j) \quad \text{(for continuous } \mathbf{x}_t\text{)} \\ A_{jk} &= p(\mathbf{x}_t = k|s_t = j) \quad \text{(for discrete } \mathbf{x}_t\text{)} \end{aligned}$$

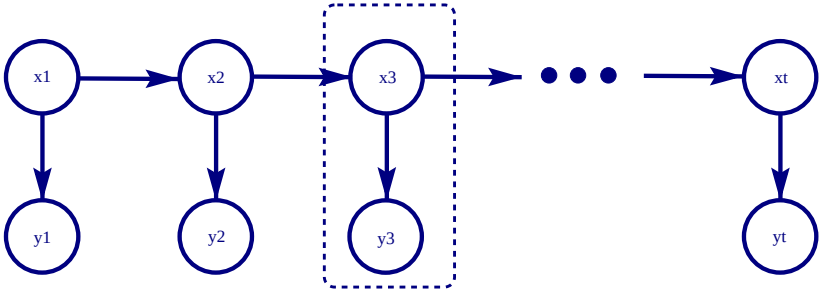
Hidden Markov models

Two interpretations:

- a Markov chain with stochastic measurements:



- or a mixture model with states coupled across time:



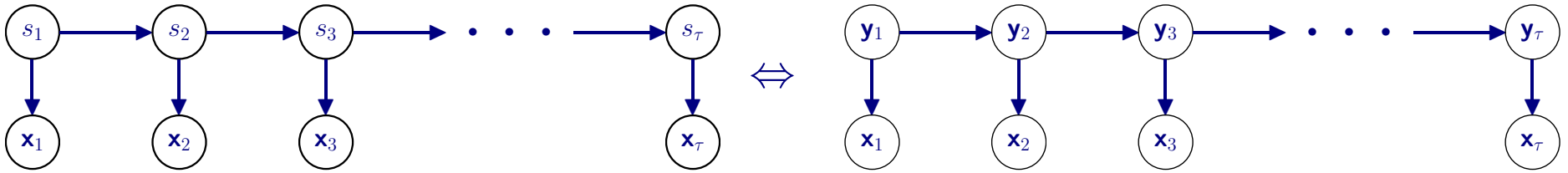
Even though hidden state sequence is first-order Markov, the output process may not be Markov of **any** order (for example: $1111121111311121111131\ldots$).

Discrete state, discrete output models can approximate any continuous dynamics and observation mapping even if nonlinear; however this is usually not practical.

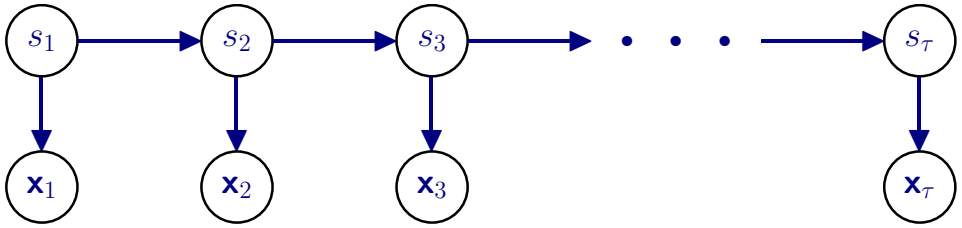
HMMs are related to stochastic finite state machines/automatas.

HMMs and SSMs

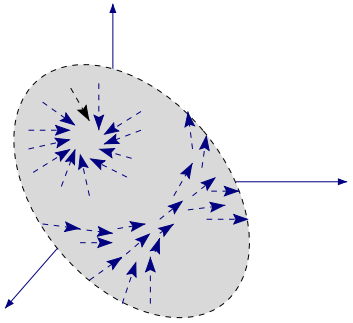
State space models (linear dynamical systems with Gaussian noise) are exactly the continuous state analogue of hidden Markov models.



- Continuous vector of states is a very powerful representation.
  For an HMM to communicate N bits of information about the past, it needs 2^N states.

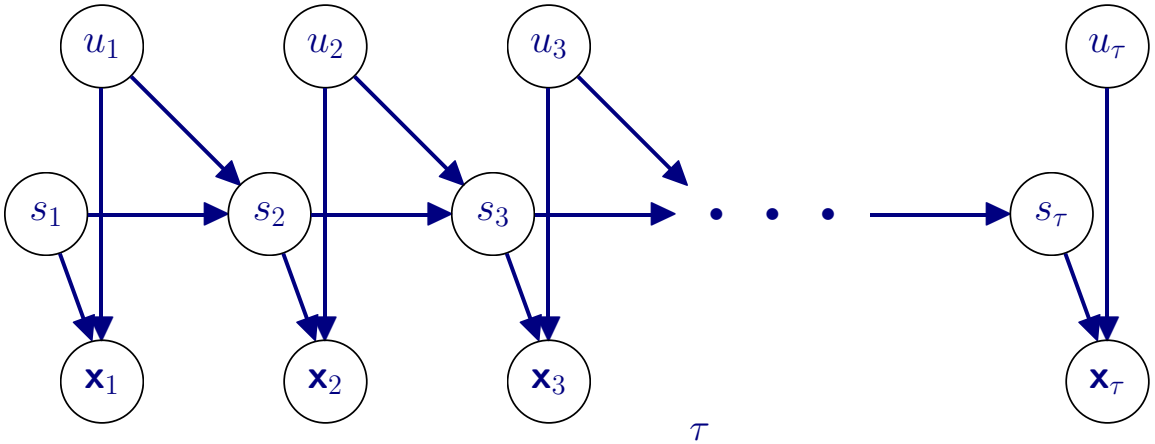


- Linear-Gaussian output/dynamics are very weak.
  The types of dynamics linear SSMs can capture is very limited. However, HMMs can in principle represent arbitrary stochastic dynamics and output mappings.



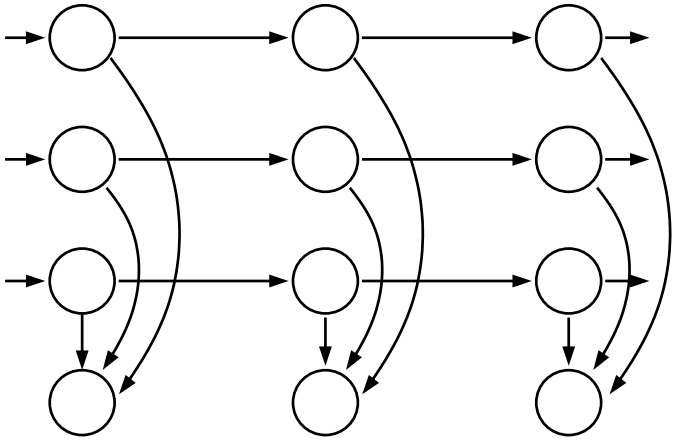
Some Extensions

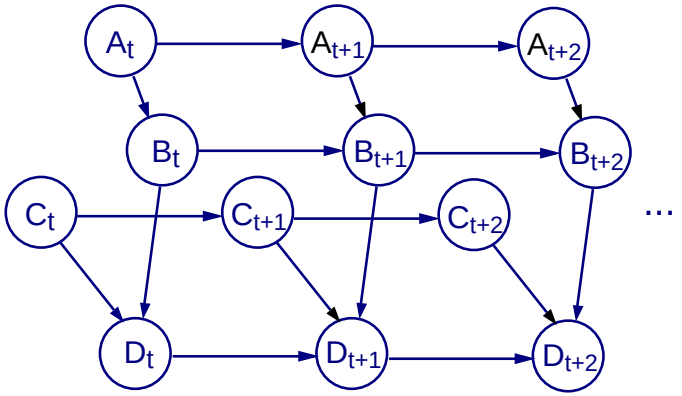
- Input-output state-space models



$$p(s_{1:\tau}, \mathbf{x}_{1:\tau}|u_{1:\tau}) = p(s_1|u_1)p(\mathbf{x}_1|s_1, u_1) \prod_{t=2}^{\tau} p(s_t|s_{t-1}, u_{t-1})p(\mathbf{x}_t|s_t, u_t)$$

- Factorial hidden Markov models and dynamic Bayesian networks





- Hierarchical HMMs

End Notes

Great review for these basic models are from a paper by Roweis & Ghahramani (1999): A Unifying Review of Linear Gaussian Models. Neural Computation 11(2):305–345.

End Notes