Probabilistic & Unsupervised Learning

Latent Variable Models for Time Series

Maneesh Sahani
maneesh@gatsby.ucl.ac.uk

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

Term 1, Autumn 2013

Modeling time series

Thus far, our data have been (marginally) iid. Now consider a sequence of observations:

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

that are *not* independent.

Modeling time series

Thus far, our data have been (marginally) iid. Now consider a sequence of observations:

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

that are *not* independent.

Examples:

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

Modeling time series

Thus far, our data have been (marginally) iid. Now consider a sequence of observations:

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

that are *not* independent.

Examples:

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

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

Modeling time series

Thus far, our data have been (marginally) iid. Now consider a sequence of observations:

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

that are *not* independent.

Examples:

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

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

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

Modeling time series

Thus far, our data have been (marginally) iid. Now consider a sequence of observations:

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

that are *not* independent.

Examples:

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

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

- 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)

Modeling time series

Thus far, our data have been (marginally) iid. Now consider a sequence of observations:

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

that are *not* independent.

Examples:

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

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

- 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 linking entire sequence

Markov models

In general:

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

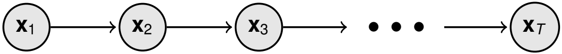
Markov models

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 ($\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}$).

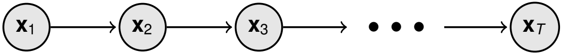
Markov models

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:

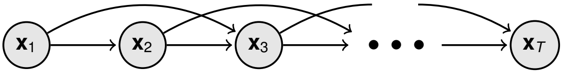
$$P(\mathbf{x}_1, \ldots, \mathbf{x}_t) = P(\mathbf{x}_1)P(\mathbf{x}_2|\mathbf{x}_1) \cdots P(\mathbf{x}_t|\mathbf{x}_{t-1})$$



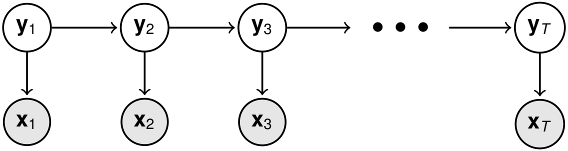
The term *Markov* refers to a conditional independence relationship. In this case, the Markov property is that, given the present observation ($\mathbf{x}_t$), the future ($\mathbf{x}_{t+1}, \ldots$) is independent of the past ($\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}$).

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



Temporal dependence captured by latents, with observations conditionally independent.

Speech recognition:

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

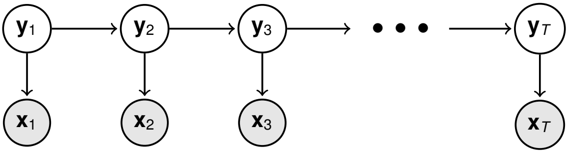
Vision:

- **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

Latent-Chain models



Joint probability factorizes:

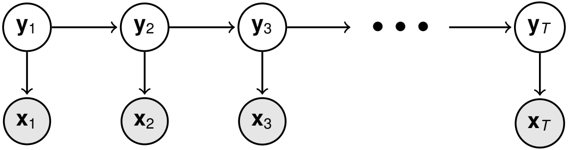
$$P(\mathbf{y}_{1:T}, \mathbf{x}_{1:T}) = P(\mathbf{y}_1)P(\mathbf{x}_1|\mathbf{y}_1)\prod_{t=2}^{T} P(\mathbf{y}_t|\mathbf{y}_{t-1})P(\mathbf{x}_t|\mathbf{y}_t)$$

where $\mathbf{y}_t$ and $\mathbf{x}_t$ are both real-valued vectors, and $\square_{1:T} \equiv \square_1, \ldots, \square_T$.

Two frequently-used tractable models:

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

Linear-Gaussian state-space models (SSMs)



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

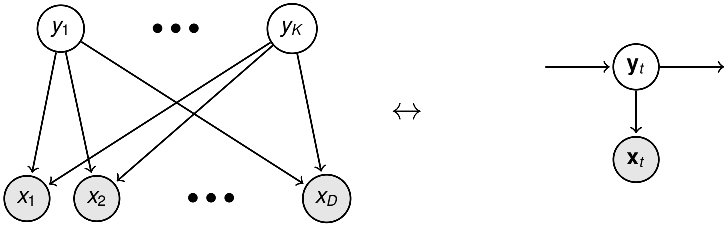
Output equation:	$\mathbf{x}_t = C\mathbf{y}_t + \mathbf{v}_t$
State dynamics equation:	$\mathbf{y}_t = A\mathbf{y}_{t-1} + \mathbf{w}_t$

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 $\mathbf{x}_{1:T}, \mathbf{y}_{1:T}$ is (one big) multivariate Gaussian.

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

From factor analysis to state space models



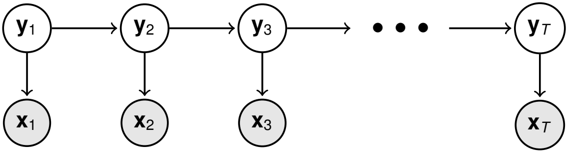
Factor analysis: $x_i = \sum_{j=1}^{K} \Lambda_{ij} \; y_j + \epsilon_i$

SSM output: $x_{t,i} = \sum_{j=1}^{K} C_{ij} \; y_{t,j} + v_i$.

Interpretation 1:

- Observations confined near low-dimensional subspace (as in FA/PCA).
- Successive observations are generated from correlated points in the latent space.
- However:
  - FA requires $K < D$ and Ψ diagonal; SSMs may have $K \geq D$ and arbitrary output noise. Why?
  - Thus ML estimates of subspace by FA and SSM may differ.

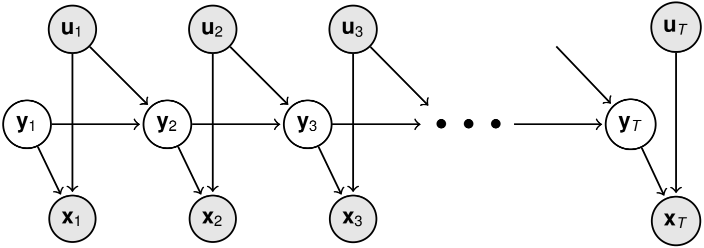
Linear dynamical systems



Interpretation 2:

- Markov chain with linear dynamics $\mathbf{y}_t = A\mathbf{y}_t \ldots$
- ...pertubed by Gaussian innovations noise – may describe stochasticity, unknown control, or model mismatch.
- Observations are a linear projection of the dynamical state, with additive iid Gaussian noise.
- Note:
  - Dynamical process ($\mathbf{y}_t$) may be higher dimensional than the observations ($\mathbf{x}_t$).
  - Observations do not form a Markov chain – longer-scale dependence reflects/reveals latent dynamics.

State Space Models with Control Inputs



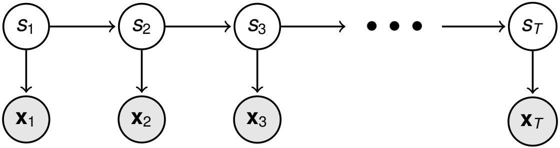
State space models can be used to model the input–output behaviour of controlled systems. The observed variables are divided into inputs ($\mathbf{u}_t$) and outputs ($\mathbf{x}_t$).

State dynamics equation: $\mathbf{y}_t = A\mathbf{y}_{t-1} + B\mathbf{u}_{t-1} + \mathbf{w}_t.$

Output equation: $\mathbf{x}_t = C\mathbf{y}_t + D\mathbf{u}_t + \mathbf{v}_t.$

Note that we can have many variants, e.g. $\mathbf{y}_t = A\mathbf{y}_{t-1} + B\mathbf{u}_t + \mathbf{w}_t$ or even $\mathbf{y}_t = A\mathbf{y}_{t-1} + B\mathbf{x}_{t-1} + \mathbf{w}_t$.

Hidden Markov models



Discrete hidden states $s_t \in \{1 \ldots, K\}$; outputs $\mathbf{x}_t$ can be discrete or continuous.
Joint probability factorizes:

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

Generative process:

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

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

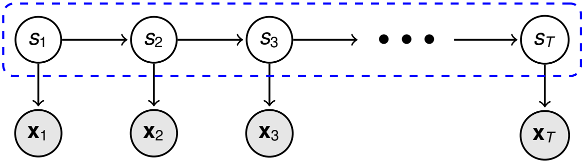
- A set of emission (output) distributions $A_j(\cdot)$ (one per state) converts state path to a sequence of observations $\mathbf{x}_t$.

$$A_j(\mathbf{x}) = P(\mathbf{x}_t = \mathbf{x}|s_t = j) \qquad \text{(for continuous } \mathbf{x}_t\text{)}$$
$$A_{jk} = P(\mathbf{x}_t = k|s_t = j) \qquad \text{(for discrete } \mathbf{x}_t\text{)}$$

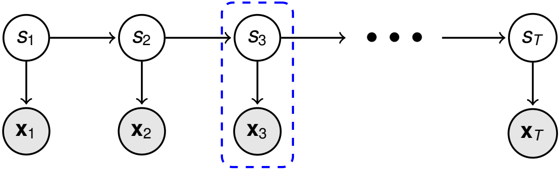
Hidden Markov models

Two interpretations:

- a Markov chain with stochastic measurements:



- or a mixture model with states coupled across time:

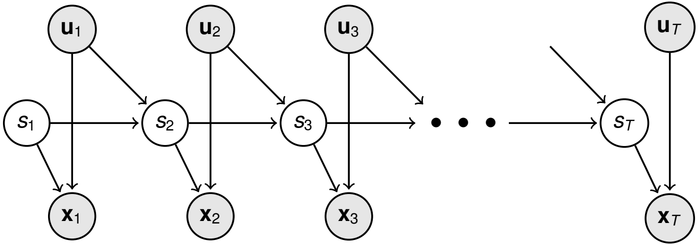


Even though hidden state sequence is first-order Markov, the output process may not be Markov of any order (for example: 1111121111311121111131 . . .).

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/automata.

Input-output hidden Markov models



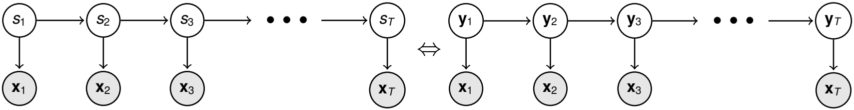
Hidden Markov models can also be used to model sequential input-output behaviour:

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

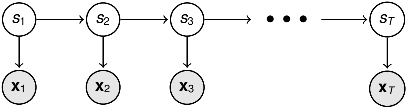
IOHMMs can capture arbitrarily complex input-output relationships, however the number of states required is often impractical.

HMMs and SSMs

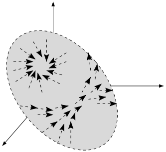
(Linear Gaussian) State space models are the continuous state analogue of hidden Markov models.



- A continuous vector state is a very powerful representation.
  For an HMM to communicate N bits of information about the past, it needs 2^N states! But a real-valued state vector can store an arbitrary number of bits in principle.

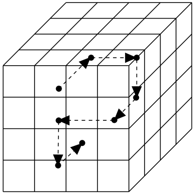


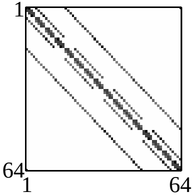
- Linear-Gaussian output/dynamics are very weak.
  The types of dynamics linear SSMs can capture is very limited. HMMs can in principle represent arbitrary stochastic dynamics and output mappings.



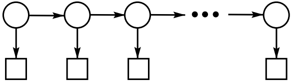
Many Extensions

- Constrained HMMs

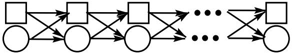




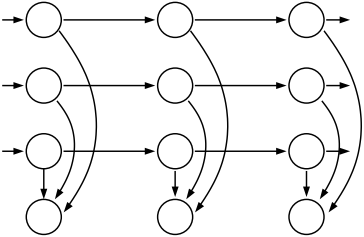
- Continuous state models with discrete outputs for time series and static data



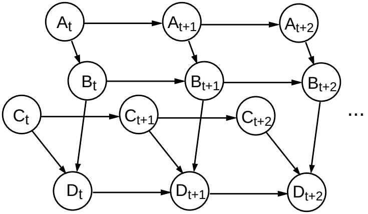
- Hierarchical models
- Hybrid systems ⇔ Mixed continuous & discrete states, switching state-space models



Richer state representations



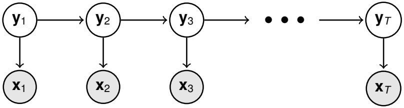
Factorial HMMs

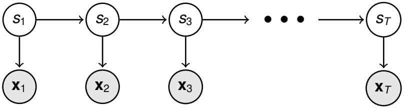


Dynamic Bayesian Networks

- These are hidden Markov models with many state variables (i.e. a distributed representation of the state).
- The state can capture many more bits of information about the sequence (linear in the number of state variables).

Chain models: ML Learning with EM



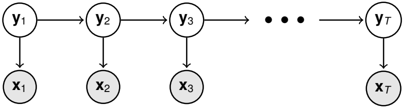


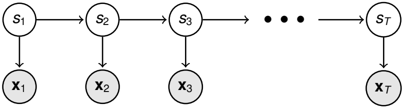
$$\mathbf{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_0, Q_0)$$
$$\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}(A\mathbf{y}_{t-1}, Q)$$
$$\mathbf{x}_t|\mathbf{y}_t \sim \mathcal{N}(C\mathbf{y}_t, R)$$

$$s_1 \sim \boldsymbol{\pi}$$
$$s_t|s_{t-1} \sim \Phi_{s_{t-1},\cdot}$$
$$\mathbf{x}_t|s_t \sim \mathsf{A}_{s_t}$$

The structure of learning and inference for both models is dictated by the factored structure.

Chain models: ML Learning with EM





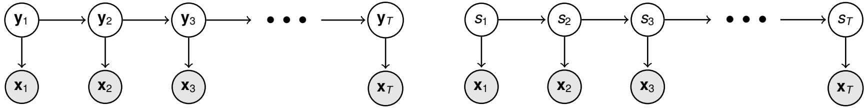
$$\mathbf{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_0, Q_0)$$
$$\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}(A\mathbf{y}_{t-1}, Q)$$
$$\mathbf{x}_t|\mathbf{y}_t \sim \mathcal{N}(C\mathbf{y}_t, R)$$

$$s_1 \sim \boldsymbol{\pi}$$
$$s_t|s_{t-1} \sim \Phi_{s_{t-1},\cdot}$$
$$\mathbf{x}_t|s_t \sim \mathrm{A}_{s_t}$$

The structure of learning and inference for both models is dictated by the factored structure.

$$P(\mathbf{x}_1,\ldots,\mathbf{x}_T,\mathbf{y}_1,\ldots,\mathbf{y}_T) = P(\mathbf{y}_1)\prod_{t=2}^{T} P(\mathbf{y}_t|\mathbf{y}_{t-1})\prod_{t=1}^{T} P(\mathbf{x}_t|\mathbf{y}_t)$$

Chain models: ML Learning with EM



$$\mathbf{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_0, Q_0)$$
$$\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}(A\mathbf{y}_{t-1}, Q)$$
$$\mathbf{x}_t|\mathbf{y}_t \sim \mathcal{N}(C\mathbf{y}_t, R)$$

$$s_1 \sim \boldsymbol{\pi}$$
$$s_t|s_{t-1} \sim \Phi_{s_{t-1},\cdot}$$
$$\mathbf{x}_t|s_t \sim \mathrm{A}_{s_t}$$

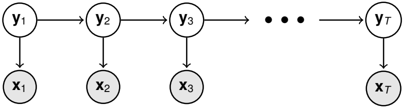
The structure of learning and inference for both models is dictated by the factored structure.

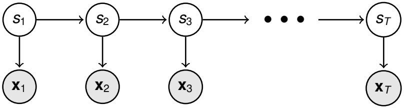
$$P(\mathbf{x}_1,\ldots,\mathbf{x}_T,\mathbf{y}_1,\ldots,\mathbf{y}_T) = P(\mathbf{y}_1)\prod_{t=2}^{T} P(\mathbf{y}_t|\mathbf{y}_{t-1})\prod_{t=1}^{T} P(\mathbf{x}_t|\mathbf{y}_t)$$

Learning (M-step):

$$\operatorname{argmax} \langle \log P(\mathbf{x}_1,\ldots,\mathbf{x}_T,\mathbf{y}_1,\ldots,\mathbf{y}_T)\rangle_{q(\mathbf{y}_1,\ldots,\mathbf{y}_T)} =$$
$$\operatorname{argmax}\left[\langle \log P(\mathbf{y}_1)\rangle_{q(\mathbf{y}_1)} + \sum_{t=2}^{T}\langle \log P(\mathbf{y}_t|\mathbf{y}_{t-1})\rangle_{q(\mathbf{y}_t,\mathbf{y}_{t-1})} + \sum_{t=1}^{T}\langle \log P(\mathbf{x}_t|\mathbf{y}_t)\rangle_{q(\mathbf{y}_t)}\right]$$

Chain models: ML Learning with EM





$$
\begin{aligned}
\mathbf{y}_1 &\sim \mathcal{N}(\boldsymbol{\mu}_0, Q_0) \\
\mathbf{y}_t | \mathbf{y}_{t-1} &\sim \mathcal{N}(A\mathbf{y}_{t-1}, Q) \\
\mathbf{x}_t | \mathbf{y}_t &\sim \mathcal{N}(C\mathbf{y}_t, R)
\end{aligned}
\qquad\qquad
\begin{aligned}
s_1 &\sim \boldsymbol{\pi} \\
s_t | s_{t-1} &\sim \Phi_{s_{t-1},\cdot} \\
\mathbf{x}_t | s_t &\sim \mathrm{A}_{s_t}
\end{aligned}
$$

The structure of learning and inference for both models is dictated by the factored structure.

$$
P(\mathbf{x}_1, \ldots, \mathbf{x}_T, \mathbf{y}_1, \ldots, \mathbf{y}_T) = P(\mathbf{y}_1) \prod_{t=2}^{T} P(\mathbf{y}_t | \mathbf{y}_{t-1}) \prod_{t=1}^{T} P(\mathbf{x}_t | \mathbf{y}_t)
$$

Learning (M-step):

$$
\operatorname{argmax} \left\langle \log P(\mathbf{x}_1, \ldots, \mathbf{x}_T, \mathbf{y}_1, \ldots, \mathbf{y}_T) \right\rangle_{q(\mathbf{y}_1, \ldots, \mathbf{y}_T)} =
$$

$$
\operatorname{argmax} \left[\left\langle \log P(\mathbf{y}_1) \right\rangle_{q(\mathbf{y}_1)} + \sum_{t=2}^{T} \left\langle \log P(\mathbf{y}_t | \mathbf{y}_{t-1}) \right\rangle_{q(\mathbf{y}_t, \mathbf{y}_{t-1})} + \sum_{t=1}^{T} \left\langle \log P(\mathbf{x}_t | \mathbf{y}_t) \right\rangle_{q(\mathbf{y}_t)} \right]
$$

So the expectations needed (in E-step) are derived from singleton and pairwise marginals.

Chain models: Inference

Three general inference problems:

Filtering: $P(\mathbf{y}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t)$

Smoothing: $P(\mathbf{y}_t|\mathbf{x}_1, \ldots, \mathbf{x}_T)$ (also $P(\mathbf{y}_t, \mathbf{y}_{t-1}|\mathbf{x}_1, \ldots, \mathbf{x}_T)$ for learning)

Prediction: $P(\mathbf{y}_t|\mathbf{x}_1, \ldots, \mathbf{x}_{t-\Delta t})$

Chain models: Inference

Three general inference problems:

Filtering:	$P(\mathbf{y}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_t)$	
Smoothing:	$P(\mathbf{y}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_T)$	(also $P(\mathbf{y}_t, \mathbf{y}_{t-1} \mid \mathbf{x}_1, \ldots, \mathbf{x}_T)$ for learning)
Prediction:	$P(\mathbf{y}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_{t-\Delta t})$	

Naively, these marginal posteriors seem to require very large integrals (or sums)

$$P(\mathbf{y}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_t) = \int \cdots \int d\mathbf{y}_1 \ldots d\mathbf{y}_{t-1} \; P(\mathbf{y}_1, \ldots, \mathbf{y}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_t)$$

Chain models: Inference

Three general inference problems:

Filtering: $P(\mathbf{y}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t)$

Smoothing: $P(\mathbf{y}_t|\mathbf{x}_1, \ldots, \mathbf{x}_T)$ (also $P(\mathbf{y}_t, \mathbf{y}_{t-1}|\mathbf{x}_1, \ldots, \mathbf{x}_T)$ for learning)

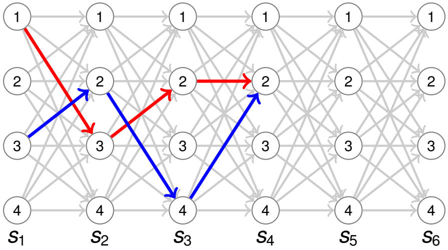
Prediction: $P(\mathbf{y}_t|\mathbf{x}_1, \ldots, \mathbf{x}_{t-\Delta t})$

Naively, these marginal posteriors seem to require very large integrals (or sums)

$$P(\mathbf{y}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t) = \int \cdots \int d\mathbf{y}_1 \ldots d\mathbf{y}_{t-1}\ P(\mathbf{y}_1, \ldots, \mathbf{y}_t|\mathbf{x}_1, \ldots, \mathbf{x}_t)$$

but again the factored structure of the distributions will help us. The algorithms rely on a form of temporal updating or message passing.

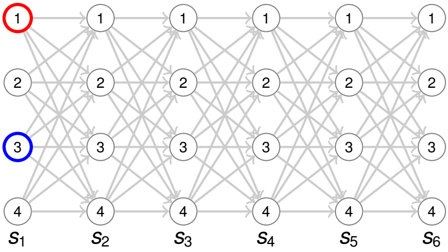
Crawling the HMM state-lattice



Consider an HMM, where we want to find $P(s_t{=}k|\mathbf{x}_1 \ldots \mathbf{x}_k) =$

$$\sum_{k_1,\ldots,k_{t-1}} P(s_1{=}k_1, \ldots, s_t{=}k|\mathbf{x}_1 \ldots \mathbf{x}_t) = \sum_{k_1,\ldots,k_{t-1}} \pi_{k_1} A_{k_1}(\mathbf{x}_1) \Phi_{k_1,k_2} A_{k_2}(\mathbf{x}_2) \ldots \Phi_{k_{t-1},k} A_k(\mathbf{x}_t)$$

Crawling the HMM state-lattice



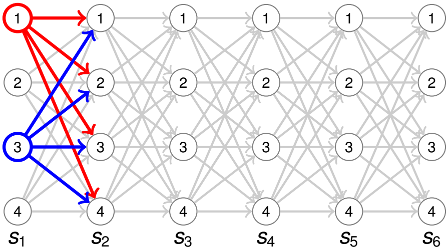
Consider an HMM, where we want to find $P(s_t{=}k|\mathbf{x}_1 \ldots \mathbf{x}_k) =$

$$\sum_{k_1,\ldots,k_{t-1}} P(s_1{=}k_1, \ldots, s_t{=}k|\mathbf{x}_1 \ldots \mathbf{x}_t) = \sum_{k_1,\ldots,k_{t-1}} \pi_{k_1} A_{k_1}(\mathbf{x}_1) \Phi_{k_1,k_2} A_{k_2}(\mathbf{x}_2) \ldots \Phi_{k_{t-1},k} A_k(\mathbf{x}_t)$$

Naïve algorithm:

- start a “bug” at each of the K states at $t = 1$ holding value 1

Crawling the HMM state-lattice

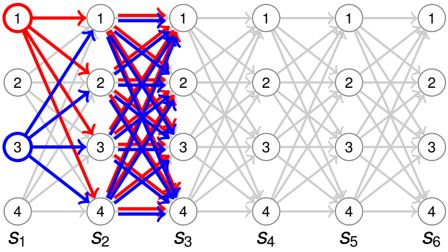


Consider an HMM, where we want to find $P(s_t{=}k|\mathbf{x}_1 \ldots \mathbf{x}_k) =$
$$\sum_{k_1,\ldots,k_{t-1}} P(s_1{=}k_1,\ldots,s_t{=}k|\mathbf{x}_1 \ldots \mathbf{x}_t) = \sum_{k_1,\ldots,k_{t-1}} \pi_{k_1} A_{k_1}(\mathbf{x}_1)\Phi_{k_1,k_2} A_{k_2}(\mathbf{x}_2)\ldots\Phi_{k_{t-1},k} A_k(\mathbf{x}_t)$$

Naïve algorithm:

- start a “bug” at each of the K states at $t = 1$ holding value 1
- move each bug forward in time: make copies of each bug to each subsequent state and multiply the value of each copy by transition prob. $\times$ output emission prob.

Crawling the HMM state-lattice



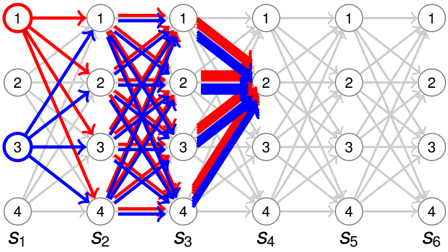
Consider an HMM, where we want to find $P(s_t{=}k|\mathbf{x}_1 \ldots \mathbf{x}_k) =$

$$\sum_{k_1,\ldots,k_{t-1}} P(s_1{=}k_1,\ldots,s_t{=}k|\mathbf{x}_1 \ldots \mathbf{x}_t) = \sum_{k_1,\ldots,k_{t-1}} \pi_{k_1} A_{k_1}(\mathbf{x}_1)\Phi_{k_1,k_2} A_{k_2}(\mathbf{x}_2)\ldots\Phi_{k_{t-1},k} A_k(\mathbf{x}_t)$$

Naïve algorithm:

- start a "bug" at each of the K states at $t = 1$ holding value 1
- move each bug forward in time: make copies of each bug to each subsequent state and multiply the value of each copy by transition prob. $\times$ output emission prob.
- repeat

Crawling the HMM state-lattice



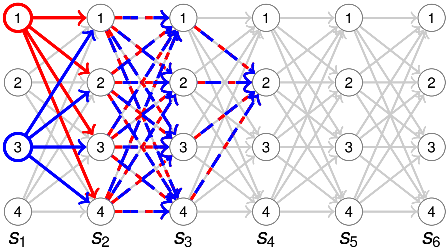
Consider an HMM, where we want to find $P(s_t{=}k|\mathbf{x}_1 \dots \mathbf{x}_k) =$

$$\sum_{k_1,\dots,k_{t-1}} P(s_1{=}k_1, \dots, s_t{=}k|\mathbf{x}_1 \dots \mathbf{x}_t) = \sum_{k_1,\dots,k_{t-1}} \pi_{k_1} A_{k_1}(\mathbf{x}_1)\Phi_{k_1,k_2} A_{k_2}(\mathbf{x}_2) \dots \Phi_{k_{t-1},k} A_k(\mathbf{x}_t)$$

Naïve algorithm:

- start a “bug” at each of the K states at $t = 1$ holding value 1
- move each bug forward in time: make copies of each bug to each subsequent state and multiply the value of each copy by transition prob. $\times$ output emission prob.
- repeat until all bugs have reached time t
- sum up values on all K^{t-1} bugs that reach state $s_t = k$ (one bug per state path)

Crawling the HMM state-lattice



Consider an HMM, where we want to find $P(s_t{=}k|\mathbf{x}_1 \dots \mathbf{x}_k) =$
$$\sum_{k_1,\dots,k_{t-1}} P(s_1{=}k_1,\dots,s_t{=}k|\mathbf{x}_1\dots\mathbf{x}_t) = \sum_{k_1,\dots,k_{t-1}} \pi_{k_1} A_{k_1}(\mathbf{x}_1)\Phi_{k_1,k_2}A_{k_2}(\mathbf{x}_2)\dots\Phi_{k_{t-1},k}A_k(\mathbf{x}_t)$$

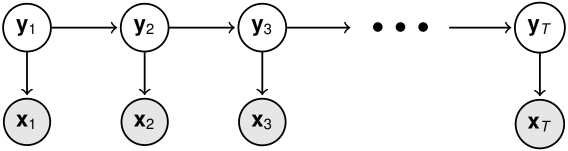
Naïve algorithm:

- start a "bug" at each of the K states at $t = 1$ holding value 1
- move each bug forward in time: make copies of each bug to each subsequent state and multiply the value of each copy by transition prob. $\times$ output emission prob.
- repeat until all bugs have reached time t
- sum up values on all K^{t-1} bugs that reach state $s_t = k$ (one bug per state path)

Clever recursion:

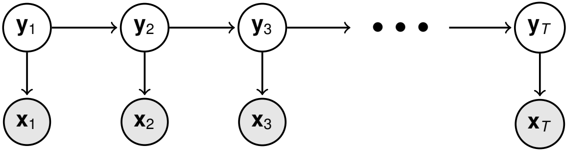
- at every step, replace bugs at each node with a single bug carrying sum of values

Probability updating: "Bayesian filtering"



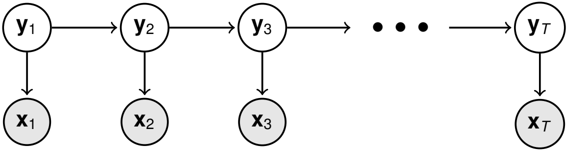
$$P(\mathbf{y}_t|\mathbf{x}_{1:t}) = \int P(\mathbf{y}_t, \mathbf{y}_{t-1}|\mathbf{x}_t,)\; d\mathbf{y}_{t-1}$$

Probability updating: "Bayesian filtering"



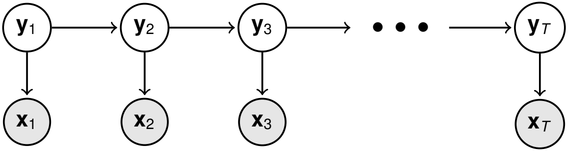
$$P(\mathbf{y}_t|\mathbf{x}_{1:t}) = \int P(\mathbf{y}_t, \mathbf{y}_{t-1}|\mathbf{x}_t, \mathbf{x}_{1:t-1})\, d\mathbf{y}_{t-1}$$

Probability updating: "Bayesian filtering"



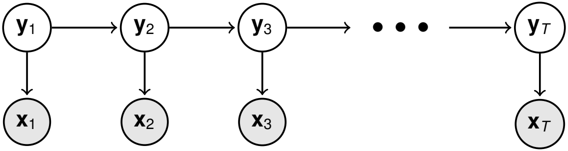
$$
\begin{aligned}
P(\mathbf{y}_t|\mathbf{x}_{1:t}) &= \int P(\mathbf{y}_t, \mathbf{y}_{t-1}|\mathbf{x}_t, \mathbf{x}_{1:t-1})\; d\mathbf{y}_{t-1} \\
&= \int \frac{P(\mathbf{x}_t, \mathbf{y}_t, \mathbf{y}_{t-1}|\mathbf{x}_{1:t-1})}{P(\mathbf{x}_t|\mathbf{x}_{1:t-1})}\; d\mathbf{y}_{t-1}
\end{aligned}
$$

Probability updating: "Bayesian filtering"



$$
\begin{aligned}
P(\mathbf{y}_t|\mathbf{x}_{1:t}) &= \int P(\mathbf{y}_t, \mathbf{y}_{t-1}|\mathbf{x}_t, \mathbf{x}_{1:t-1})\ d\mathbf{y}_{t-1} \\
&= \int \frac{P(\mathbf{x}_t, \mathbf{y}_t, \mathbf{y}_{t-1}|\mathbf{x}_{1:t-1})}{P(\mathbf{x}_t|\mathbf{x}_{1:t-1})}\ d\mathbf{y}_{t-1} \\
&\propto \int P(\mathbf{x}_t|\mathbf{y}_t, \mathbf{y}_{t-1}, \mathbf{x}_{1:t-1})P(\mathbf{y}_t|\mathbf{y}_{t-1}, \mathbf{x}_{1:t-1})P(\mathbf{y}_{t-1}|\mathbf{x}_{1:t-1})\ d\mathbf{y}_{t-1}
\end{aligned}
$$

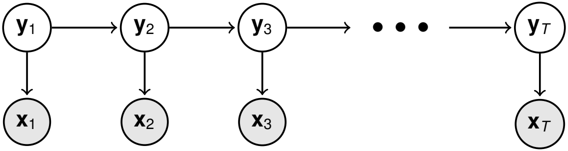
Probability updating: “Bayesian filtering”



$$
\begin{aligned}
{\color{blue}P(\mathbf{y}_t|\mathbf{x}_{1:t})} &= \int P(\mathbf{y}_t, \mathbf{y}_{t-1}|\mathbf{x}_t, \mathbf{x}_{1:t-1})\ d\mathbf{y}_{t-1} \\
&= \int \frac{P(\mathbf{x}_t, \mathbf{y}_t, \mathbf{y}_{t-1}|\mathbf{x}_{1:t-1})}{P(\mathbf{x}_t|\mathbf{x}_{1:t-1})}\ d\mathbf{y}_{t-1} \\
&\propto \int P(\mathbf{x}_t|\mathbf{y}_t, {\color{red}\mathbf{y}_{t-1}, \mathbf{x}_{1:t-1}})P(\mathbf{y}_t|\mathbf{y}_{t-1}, {\color{red}\mathbf{x}_{1:t-1}})P(\mathbf{y}_{t-1}|\mathbf{x}_{1:t-1})\ d\mathbf{y}_{t-1} \\
&= \int P(\mathbf{x}_t|\mathbf{y}_t)P(\mathbf{y}_t|\mathbf{y}_{t-1})P(\mathbf{y}_{t-1}|\mathbf{x}_{1:t-1})\ d\mathbf{y}_{t-1}
\end{aligned}
$$

Markov property

Probability updating: "Bayesian filtering"



$$
\begin{aligned}
\color{blue}{P(\mathbf{y}_t|\mathbf{x}_{1:t})} &= \int P(\mathbf{y}_t, \mathbf{y}_{t-1}|\mathbf{x}_t, \mathbf{x}_{1:t-1})\; d\mathbf{y}_{t-1} \\
&= \int \frac{P(\mathbf{x}_t, \mathbf{y}_t, \mathbf{y}_{t-1}|\mathbf{x}_{1:t-1})}{P(\mathbf{x}_t|\mathbf{x}_{1:t-1})}\; d\mathbf{y}_{t-1} \\
&\propto \int P(\mathbf{x}_t|\mathbf{y}_t, \color{red}{\mathbf{y}_{t-1}, \mathbf{x}_{1:t-1}})P(\mathbf{y}_t|\mathbf{y}_{t-1}, \color{red}{\mathbf{x}_{1:t-1}})P(\mathbf{y}_{t-1}|\mathbf{x}_{1:t-1})\; d\mathbf{y}_{t-1} \\
&= \int P(\mathbf{x}_t|\mathbf{y}_t)P(\mathbf{y}_t|\mathbf{y}_{t-1})\color{blue}{P(\mathbf{y}_{t-1}|\mathbf{x}_{1:t-1})}\; d\mathbf{y}_{t-1}
\end{aligned}
$$

Markov property

This is a **forward recursion** based on Bayes rule.

The HMM: Forward pass

The **forward recursion** for the HMM is a form of **dynamic programming**.

The HMM: Forward pass

The **forward recursion** for the HMM is a form of **dynamic programming**. Define:

$$\alpha_t(i) = P(\mathbf{x}_1, \ldots, \mathbf{x}_t, s_t = i|\theta)$$

The HMM: Forward pass

The **forward recursion** for the HMM is a form of **dynamic programming**. Define:

$$\alpha_t(i) = P(\mathbf{x}_1, \ldots, \mathbf{x}_t, s_t = i|\theta)$$

Then much like the Bayesian filtering updates, we have:

$$\alpha_1(i) = \pi_i A_i(\mathbf{x}_1) \qquad \alpha_{t+1}(i) = \left(\sum_{j=1}^{K} \alpha_t(j) \Phi_{ji} \right) A_i(\mathbf{x}_{t+1})$$

The HMM: Forward pass

The **forward recursion** for the HMM is a form of **dynamic programming**. Define:

$$\alpha_t(i) = P(\mathbf{x}_1, \ldots, \mathbf{x}_t, s_t = i | \theta)$$

Then much like the Bayesian filtering updates, we have:

$$\alpha_1(i) = \pi_i A_i(\mathbf{x}_1) \qquad \alpha_{t+1}(i) = \left(\sum_{j=1}^{K} \alpha_t(j) \Phi_{ji} \right) A_i(\mathbf{x}_{t+1})$$

We've defined $\alpha_t(i)$ to be a joint rather than a posterior.

The HMM: Forward pass

The **forward recursion** for the HMM is a form of **dynamic programming**. Define:

$$\alpha_t(i) = P(\mathbf{x}_1, \ldots, \mathbf{x}_t, s_t = i|\theta)$$

Then much like the Bayesian filtering updates, we have:

$$\alpha_1(i) = \pi_i A_i(\mathbf{x}_1) \qquad \alpha_{t+1}(i) = \left(\sum_{j=1}^{K} \alpha_t(j) \Phi_{ji} \right) A_i(\mathbf{x}_{t+1})$$

We've defined $\alpha_t(i)$ to be a joint rather than a posterior. It's easy to obtain the posterior by normalisation:

$$P(s_t = i|\mathbf{x}_1, \ldots, \mathbf{x}_t, \theta) = \frac{\alpha_t(i)}{\sum_k \alpha_t(k)}$$

The HMM: Forward pass

The **forward recursion** for the HMM is a form of **dynamic programming**. Define:

$$\alpha_t(i) = P(\mathbf{x}_1, \ldots, \mathbf{x}_t, s_t = i|\theta)$$

Then much like the Bayesian filtering updates, we have:

$$\alpha_1(i) = \pi_i A_i(\mathbf{x}_1) \qquad \alpha_{t+1}(i) = \left(\sum_{j=1}^{K} \alpha_t(j)\Phi_{ji}\right) A_i(\mathbf{x}_{t+1})$$

We've defined $\alpha_t(i)$ to be a joint rather than a posterior. It's easy to obtain the posterior by normalisation:

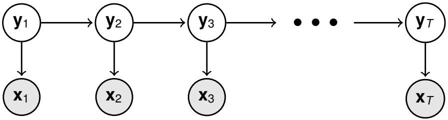
$$P(s_t = i|\mathbf{x}_1, \ldots, \mathbf{x}_t, \theta) = \frac{\alpha_t(i)}{\sum_k \alpha_t(k)}$$

This form enables us to compute the likelihood for $\theta = \{\mathrm{A}, \Phi, \boldsymbol{\pi}\}$ efficiently in $\mathcal{O}(TK^2)$ time:

$$P(\mathbf{x}_1 \ldots \mathbf{x}_T|\theta) = \sum_{s_1,\ldots,s_T} P(\mathbf{x}_1, \ldots, \mathbf{x}_T, s_1, \ldots, s_T, \theta) = \sum_{k=1}^{K} \alpha_T(k)$$

avoiding the exponential number of paths in the naïve sum (number of paths = K^T).

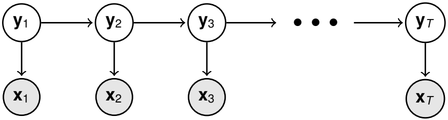
The LGSSM: Kalman Filtering



$$
\begin{aligned}
\mathbf{y}_1 &\sim \mathcal{N}(\boldsymbol{\mu}_0, Q_0) \\
\mathbf{y}_t|\mathbf{y}_{t-1} &\sim \mathcal{N}(A\mathbf{y}_{t-1}, Q) \\
\mathbf{x}_t|\mathbf{y}_t &\sim \mathcal{N}(C\mathbf{y}_t, R)
\end{aligned}
$$

For the SSM, the sums become integrals.

The LGSSM: Kalman Filtering

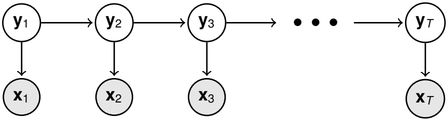


$$\mathbf{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_0, Q_0)$$
$$\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}(A\mathbf{y}_{t-1}, Q)$$
$$\mathbf{x}_t|\mathbf{y}_t \sim \mathcal{N}(C\mathbf{y}_t, R)$$

For the SSM, the sums become integrals. Let $\hat{\mathbf{y}}_1^0 = \boldsymbol{\mu}_0$ and $\hat{V}_1^0 = Q_0$; then (cf. FA)

$$P(\mathbf{y}_1|\mathbf{x}_1) = \mathcal{N}\big(\, \hat{\mathbf{y}}_1^0 + K_1(\mathbf{x}_1 - C\hat{\mathbf{y}}_i^0),\ \hat{V}_1^0 - K_1 C \hat{V}_1^0\big)$$

The LGSSM: Kalman Filtering

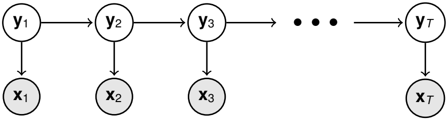


$$\mathbf{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_0, Q_0)$$
$$\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}(A\mathbf{y}_{t-1}, Q)$$
$$\mathbf{x}_t|\mathbf{y}_t \sim \mathcal{N}(C\mathbf{y}_t, R)$$

For the SSM, the sums become integrals. Let $\hat{\mathbf{y}}_1^0 = \boldsymbol{\mu}_0$ and $\hat{V}_1^0 = Q_0$; then (cf. FA)

$$P(\mathbf{y}_1|\mathbf{x}_1) = \mathcal{N}\left(\hat{\mathbf{y}}_1^0 + K_1(\mathbf{x}_1 - C\hat{\mathbf{y}}_i^0),\ \hat{V}_1^0 - K_1 C \hat{V}_1^0\right) \qquad K_1 = \hat{V}_1^0 C^\mathsf{T}(C\hat{V}_1^0 C^\mathsf{T} + R)^{-1}$$

The LGSSM: Kalman Filtering

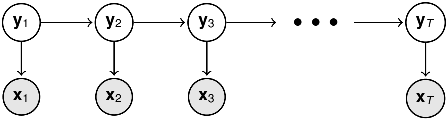


$$\mathbf{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_0, Q_0)$$
$$\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}(A\mathbf{y}_{t-1}, Q)$$
$$\mathbf{x}_t|\mathbf{y}_t \sim \mathcal{N}(C\mathbf{y}_t, R)$$

For the SSM, the sums become integrals. Let $\hat{\mathbf{y}}_1^0 = \boldsymbol{\mu}_0$ and $\hat{V}_1^0 = Q_0$; then (cf. FA)

$$P(\mathbf{y}_1|\mathbf{x}_1) = \mathcal{N}\big(\underbrace{\hat{\mathbf{y}}_1^0 + K_1(\mathbf{x}_1 - C\hat{\mathbf{y}}_i^0)}_{\hat{\mathbf{y}}_1^1}, \underbrace{\hat{V}_1^0 - K_1 C \hat{V}_1^0}_{\hat{V}_1^1}\big) \qquad K_1 = \hat{V}_1^0 C^{\mathsf{T}}(C\hat{V}_1^0 C^{\mathsf{T}} + R)^{-1}$$

The LGSSM: Kalman Filtering



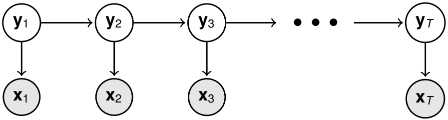
$$\mathbf{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_0, Q_0)$$
$$\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}(A\mathbf{y}_{t-1}, Q)$$
$$\mathbf{x}_t|\mathbf{y}_t \sim \mathcal{N}(C\mathbf{y}_t, R)$$

For the SSM, the sums become integrals. Let $\hat{\mathbf{y}}_1^0 = \boldsymbol{\mu}_0$ and $\hat{V}_1^0 = Q_0$; then (cf. FA)

$$P(\mathbf{y}_1|\mathbf{x}_1) = \mathcal{N}\big(\underbrace{\hat{\mathbf{y}}_1^0 + K_1(\mathbf{x}_1 - C\hat{\mathbf{y}}_i^0)}_{\hat{\mathbf{y}}_1^1}, \underbrace{\hat{V}_1^0 - K_1 C \hat{V}_1^0}_{\hat{V}_1^1}\big) \qquad K_1 = \hat{V}_1^0 C^{\mathsf{T}}(C\hat{V}_1^0 C^{\mathsf{T}} + R)^{-1}$$

In general, we define $\hat{\mathbf{y}}_t^T \equiv E[\mathbf{y}_t|\mathbf{x}_1, \ldots, \mathbf{x}_T]$ and $\hat{V}_t^T \equiv V[\mathbf{y}_t|\mathbf{x}_1, \ldots, \mathbf{x}_T]$.

The LGSSM: Kalman Filtering



$$\mathbf{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_0, Q_0)$$
$$\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}(A\mathbf{y}_{t-1}, Q)$$
$$\mathbf{x}_t|\mathbf{y}_t \sim \mathcal{N}(C\mathbf{y}_t, R)$$

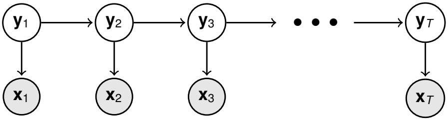
For the SSM, the sums become integrals. Let $\hat{\mathbf{y}}_1^0 = \boldsymbol{\mu}_0$ and $\hat{V}_1^0 = Q_0$; then (cf. FA)

$$P(\mathbf{y}_1|\mathbf{x}_1) = \mathcal{N}\Big(\underbrace{\hat{\mathbf{y}}_1^0 + K_1(\mathbf{x}_1 - C\hat{\mathbf{y}}_i^0)}_{\hat{\mathbf{y}}_1^1}, \underbrace{\hat{V}_1^0 - K_1 C\hat{V}_1^0}_{\hat{V}_1^1}\Big) \qquad K_1 = \hat{V}_1^0 C^{\mathsf{T}}(C\hat{V}_1^0 C^{\mathsf{T}} + R)^{-1}$$

In general, we define $\hat{\mathbf{y}}_t^T \equiv E[\mathbf{y}_t|\mathbf{x}_1, \ldots, \mathbf{x}_T]$ and $\hat{V}_t^T \equiv V[\mathbf{y}_t|\mathbf{x}_1, \ldots, \mathbf{x}_T]$. Then,

$$P(\mathbf{y}_t|\mathbf{x}_{1:t-1}) = \int d\mathbf{y}_{t-1} P(\mathbf{y}_t|\mathbf{y}_{t-1}) P(\mathbf{y}_{t-1}|\mathbf{x}_{1:t-1})$$

The LGSSM: Kalman Filtering



$$\mathbf{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_0, Q_0)$$
$$\mathbf{y}_t|\mathbf{y}_{t-1} \sim \mathcal{N}(A\mathbf{y}_{t-1}, Q)$$
$$\mathbf{x}_t|\mathbf{y}_t \sim \mathcal{N}(C\mathbf{y}_t, R)$$

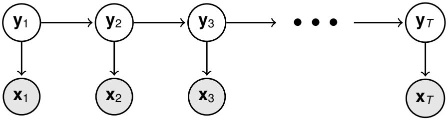
For the SSM, the sums become integrals. Let $\hat{\mathbf{y}}_1^0 = \boldsymbol{\mu}_0$ and $\hat{V}_1^0 = Q_0$; then (cf. FA)

$$P(\mathbf{y}_1|\mathbf{x}_1) = \mathcal{N}\big(\underbrace{\hat{\mathbf{y}}_1^0 + K_1(\mathbf{x}_1 - C\hat{\mathbf{y}}_i^0)}_{\hat{\mathbf{y}}_1^1}, \underbrace{\hat{V}_1^0 - K_1 C \hat{V}_1^0}_{\hat{V}_1^1}\big) \qquad K_1 = \hat{V}_1^0 C^\mathsf{T}(C\hat{V}_1^0 C^\mathsf{T} + R)^{-1}$$

In general, we define $\hat{\mathbf{y}}_t^T \equiv E[\mathbf{y}_t|\mathbf{x}_1, \dots, \mathbf{x}_T]$ and $\hat{V}_t^T \equiv V[\mathbf{y}_t|\mathbf{x}_1, \dots, \mathbf{x}_T]$. Then,

$$P(\mathbf{y}_t|\mathbf{x}_{1:t-1}) = \int d\mathbf{y}_{t-1} P(\mathbf{y}_t|\mathbf{y}_{t-1}) P(\mathbf{y}_{t-1}|\mathbf{x}_{1:t-1}) = \mathcal{N}(\underbrace{A\hat{\mathbf{y}}_{t-1}^{t-1}}_{\hat{\mathbf{y}}_t^{t-1}}, \underbrace{A\hat{V}_{t-1}^{t-1}A^\mathsf{T} + Q}_{V_t^{t-1}})$$

The LGSSM: Kalman Filtering



$$\mathbf{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_0, Q_0)$$
$$\mathbf{y}_t | \mathbf{y}_{t-1} \sim \mathcal{N}(A\mathbf{y}_{t-1}, Q)$$
$$\mathbf{x}_t | \mathbf{y}_t \sim \mathcal{N}(C\mathbf{y}_t, R)$$

For the SSM, the sums become integrals. Let $\hat{\mathbf{y}}_1^0 = \boldsymbol{\mu}_0$ and $\hat{V}_1^0 = Q_0$; then (cf. FA)

$$P(\mathbf{y}_1|\mathbf{x}_1) = \mathcal{N}\big(\underbrace{\hat{\mathbf{y}}_1^0 + K_1(\mathbf{x}_1 - C\hat{\mathbf{y}}_i^0)}_{\hat{\mathbf{y}}_1^1}, \underbrace{\hat{V}_1^0 - K_1 C \hat{V}_1^0}_{\hat{V}_1^1}\big) \qquad K_1 = \hat{V}_1^0 C^\mathsf{T}(C\hat{V}_1^0 C^\mathsf{T} + R)^{-1}$$

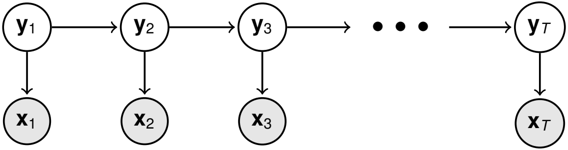
In general, we define $\hat{\mathbf{y}}_t^T \equiv E[\mathbf{y}_t|\mathbf{x}_1, \ldots, \mathbf{x}_T]$ and $\hat{V}_t^T \equiv V[\mathbf{y}_t|\mathbf{x}_1, \ldots, \mathbf{x}_T]$. Then,

$$P(\mathbf{y}_t|\mathbf{x}_{1:t-1}) = \int d\mathbf{y}_{t-1} P(\mathbf{y}_t|\mathbf{y}_{t-1}) P(\mathbf{y}_{t-1}|\mathbf{x}_{1:t-1}) = \mathcal{N}(\underbrace{A\hat{\mathbf{y}}_{t-1}^{t-1}}_{\hat{\mathbf{y}}_t^{t-1}}, \underbrace{A\hat{V}_{t-1}^{t-1}A^\mathsf{T} + Q}_{V_t^{t-1}})$$

$$P(\mathbf{y}_t|\mathbf{x}_{1:t}) = \mathcal{N}(\underbrace{\hat{\mathbf{y}}_t^{t-1} + K_t(\mathbf{x}_t - C\hat{\mathbf{y}}_t^{t-1})}_{\hat{\mathbf{y}}_t^t}, \underbrace{\hat{V}_t^{t-1} - K_t C \hat{V}_t^{t-1}}_{\hat{V}_t^t})$$

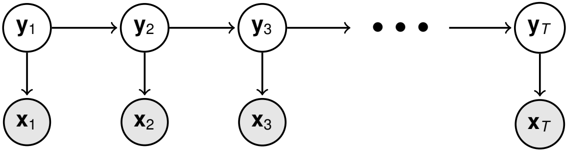
$$\underbrace{K_t = \hat{V}_t^{t-1} C^\mathsf{T}(C\hat{V}_t^{t-1} C^\mathsf{T} + R)^{-1}}_{\text{Kalman gain}}$$

The marginal posterior: “Bayesian smoothing”



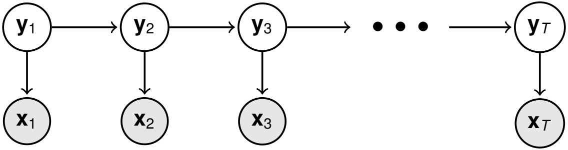
$P(\mathbf{y}_t|\mathbf{x}_{1:T})$

The marginal posterior: "Bayesian smoothing"



$$P(\mathbf{y}_t|\mathbf{x}_{1:T}) = \frac{P(\mathbf{y}_t, \mathbf{x}_{t+1:T}|\mathbf{x}_{1:t})}{P(\mathbf{x}_{t+1:T}|\mathbf{x}_{1:t})}$$

The marginal posterior: "Bayesian smoothing"



$$P(\mathbf{y}_t|\mathbf{x}_{1:T}) = \frac{P(\mathbf{y}_t, \mathbf{x}_{t+1:T}|\mathbf{x}_{1:t})}{P(\mathbf{x}_{t+1:T}|\mathbf{x}_{1:t})}$$

$$= \frac{P(\mathbf{x}_{t+1:T}|\mathbf{y}_t)P(\mathbf{y}_t|\mathbf{x}_{1:t})}{P(\mathbf{x}_{t+1:T}|\mathbf{x}_{1:t})}$$

The marginal combines a backward message with the forward message found by filtering.

The HMM: Forward–Backward Algorithm

State estimation: compute marginal posterior distribution over state at time t:

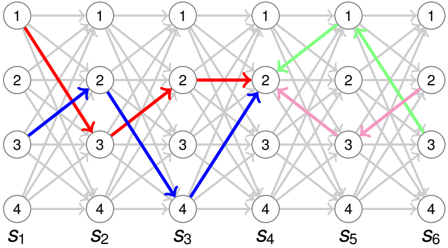
$$\gamma_t(i) \equiv P(s_t{=}i|\mathbf{x}_{1:T}) = \frac{P(s_t{=}i, \mathbf{x}_{1:t})P(\mathbf{x}_{t+1:T}|s_t{=}i)}{P(\mathbf{x}_{1:T})} = \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)}$$

where there is a simple backward recursion for

$$\beta_t(i) \equiv P(\mathbf{x}_{t+1:T}|s_t{=}i) = \sum_{j=1}^{K} P(s_{t+1}{=}j, \mathbf{x}_{t+1}, \mathbf{x}_{t+2:T}|s_t{=}i)$$

$$= \sum_{j=1}^{K} P(s_{t+1}{=}j|s_t{=}i)P(\mathbf{x}_{t+1}|s_{t+1}{=}j)P(\mathbf{x}_{t+2:T}|s_{t+1}{=}j) = \sum_{j=1}^{K} \Phi_{ij}A_j(\mathbf{x}_{t+1})\beta_{t+1}(j)$$

$\alpha_t(i)$ gives total *inflow* of probabilities to node (t, i); $\beta_t(i)$ gives total *outflow* of probabailties.



Bugs again: the bugs run forward from time 0 to t and backward from time T to t.

Viterbi decoding

- The numbers $\gamma_t(i)$ computed by forward-backward give the marginal posterior distribution over states at each time.

Viterbi decoding

- The numbers $\gamma_t(i)$ computed by forward-backward give the marginal posterior distribution over states at each time.
- By choosing the state i_t^* with the largest $\gamma_t(i)$ at each time, we can make a “best” state path. This is the path with the maximum expected number of correct states.

Viterbi decoding

- The numbers $\gamma_t(i)$ computed by forward-backward give the marginal posterior distribution over states at each time.
- By choosing the state i_t^* with the largest $\gamma_t(i)$ at each time, we can make a “best” state path. This is the path with the maximum expected number of correct states.
- But it is not the single path with the highest probability of generating the data. In fact it may be a path of probability zero!

Viterbi decoding

- The numbers $\gamma_t(i)$ computed by forward-backward give the marginal posterior distribution over states at each time.
- By choosing the state i_t^* with the largest $\gamma_t(i)$ at each time, we can make a "best" state path. This is the path with the maximum expected number of correct states.
- But it is not the single path with the highest probability of generating the data. In fact it may be a path of probability zero!
- To find the single best path, we use the *Viterbi decoding algorithm* which is just Bellman's dynamic programming algorithm applied to this problem. This is an inference algorithm which computes the most probable state sequences: $\underset{s_{1:T}}{\operatorname{argmax}} P(s_{1:T}|\mathbf{x}_{1:T}, \theta)$

Viterbi decoding

- The numbers $\gamma_t(i)$ computed by forward-backward give the marginal posterior distribution over states at each time.
- By choosing the state i_t^* with the largest $\gamma_t(i)$ at each time, we can make a "best" state path. This is the path with the maximum expected number of correct states.
- But it is not the single path with the highest probability of generating the data. In fact it may be a path of probability zero!
- To find the single best path, we use the *Viterbi decoding algorithm* which is just Bellman's dynamic programming algorithm applied to this problem. This is an inference algorithm which computes the most probable state sequences: $\underset{s_{1:T}}{\operatorname{argmax}} P(s_{1:T}|\mathbf{x}_{1:T}, \theta)$
- The recursions look the same as forward-backward, except with max instead of $\sum$.

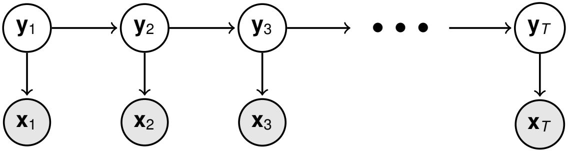
Viterbi decoding

- The numbers $\gamma_t(i)$ computed by forward-backward give the marginal posterior distribution over states at each time.
- By choosing the state i_t^* with the largest $\gamma_t(i)$ at each time, we can make a “best” state path. This is the path with the maximum expected number of correct states.
- But it is not the single path with the highest probability of generating the data. In fact it may be a path of probability zero!
- To find the single best path, we use the *Viterbi decoding algorithm* which is just Bellman’s dynamic programming algorithm applied to this problem. This is an inference algorithm which computes the most probable state sequences: $\underset{s_{1:T}}{\operatorname{argmax}} P(s_{1:T}|\mathbf{x}_{1:T}, \theta)$
- The recursions look the same as forward-backward, except with max instead of $\sum$.
- Bugs once more: same trick except at each step kill all bugs but the one with the highest value at the node.

Viterbi decoding

- The numbers $\gamma_t(i)$ computed by forward-backward give the marginal posterior distribution over states at each time.
- By choosing the state i_t^* with the largest $\gamma_t(i)$ at each time, we can make a "best" state path. This is the path with the maximum expected number of correct states.
- But it is not the single path with the highest probability of generating the data. In fact it may be a path of probability zero!
- To find the single best path, we use the *Viterbi decoding algorithm* which is just Bellman's dynamic programming algorithm applied to this problem. This is an inference algorithm which computes the most probable state sequences: $\underset{s_{1:T}}{\operatorname{argmax}} P(s_{1:T}|\mathbf{x}_{1:T}, \theta)$
- The recursions look the same as forward-backward, except with max instead of $\sum$.
- Bugs once more: same trick except at each step kill all bugs but the one with the highest value at the node.
- There is also a modified EM training based on the Viterbi decoder (assignment).

The LGSSM: Kalman smoothing



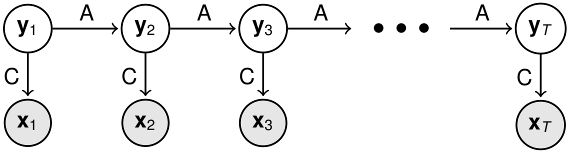
We use a slightly different decomposition:

$$
\begin{aligned}
P(\mathbf{y}_t|\mathbf{x}_{1:T}) &= \int P(\mathbf{y}_t, \mathbf{y}_{t+1}|\mathbf{x}_{1:T})\, d\mathbf{y}_{t+1} \\
&= \int P(\mathbf{y}_t|\mathbf{y}_{t+1}, \mathbf{x}_{1:T}) P(\mathbf{y}_{t+1}|\mathbf{x}_{1:T})\, d\mathbf{y}_{t+1} \\
&\underset{\text{Markov property}}{=} \int P(\mathbf{y}_t|\mathbf{y}_{t+1}, \mathbf{x}_{1:t}) P(\mathbf{y}_{t+1}|\mathbf{x}_{1:T})\, d\mathbf{y}_{t+1}
\end{aligned}
$$

This gives the additional **backward recursion**:

$$
\begin{aligned}
J_t &= \hat{V}_t^t A^\top (\hat{V}_{t+1}^t)^{-1} \\
\hat{\mathbf{y}}_t^T &= \hat{\mathbf{y}}_t^t + J_t(\hat{\mathbf{y}}_{t+1}^T - A\hat{\mathbf{y}}_t^t) \\
\hat{V}_t^T &= \hat{V}_t^t + J_t(\hat{V}_{t+1}^T - \hat{V}_{t+1}^t)J_t^\top
\end{aligned}
$$

ML Learning for SSMs using batch EM



Parameters: $\theta = \{\boldsymbol{\mu}_0, Q_0, A, Q, C, R\}$
Free energy:

$$\mathcal{F}(q, \theta) = \int d\mathbf{y}_{1:T}\ q(\mathbf{y}_{1:T})(\log P(x_{1:T}, \mathbf{y}_{1:T}|\theta) - \log q(\mathbf{y}_{1:T}))$$

E-step: Maximise $\mathcal{F}$ w.r.t. q with θ fixed: $q^*(\mathbf{y}) = p(\mathbf{y}|\mathbf{x}, \theta)$
This can be achieved with a two-state extension of the Kalman smoother.

M-step: Maximize $\mathcal{F}$ w.r.t. θ with q fixed.
This boils down to solving a few weighted least squares problems, since all the variables in:

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

form a multivariate Gaussian.

The M step for C

$$p(\mathbf{x}_t|\mathbf{y}_t) \propto \exp\left[-\frac{1}{2}(\mathbf{x}_t - C\mathbf{y}_t)^{\mathsf{T}} R^{-1} (\mathbf{x}_t - C\mathbf{y}_t)\right] \quad \Rightarrow$$

The M step for C

$$p(\mathbf{x}_t|\mathbf{y}_t) \propto \exp\left[-\frac{1}{2}(\mathbf{x}_t - C\mathbf{y}_t)^\mathsf{T} R^{-1}(\mathbf{x}_t - C\mathbf{y}_t)\right] \quad \Rightarrow$$

$$C_{\text{new}} = \underset{C}{\operatorname{argmax}} \left\langle \sum_t \ln p(\mathbf{x}_t|\mathbf{y}_t) \right\rangle_q$$

The M step for C

$p(\mathbf{x}_t|\mathbf{y}_t) \propto \exp\left[-\frac{1}{2}(\mathbf{x}_t - C\mathbf{y}_t)^\mathsf{T} R^{-1}(\mathbf{x}_t - C\mathbf{y}_t)\right] \quad \Rightarrow$

$$
\begin{aligned}
C_{\text{new}} &= \underset{C}{\operatorname{argmax}} \left\langle \sum_t \ln p(\mathbf{x}_t|\mathbf{y}_t) \right\rangle_q \\
&= \underset{C}{\operatorname{argmax}} \left\langle -\frac{1}{2} \sum_t (\mathbf{x}_t - C\mathbf{y}_t)^\mathsf{T} R^{-1} (\mathbf{x}_t - C\mathbf{y}_t) \right\rangle_q + \text{const}
\end{aligned}
$$

The M step for C

$$p(\mathbf{x}_t|\mathbf{y}_t) \propto \exp\left[-\tfrac{1}{2}(\mathbf{x}_t - C\mathbf{y}_t)^\top R^{-1}(\mathbf{x}_t - C\mathbf{y}_t)\right] \quad \Rightarrow$$

$$\begin{aligned}
C_{\text{new}} &= \underset{C}{\operatorname{argmax}} \left\langle \sum_t \ln p(\mathbf{x}_t|\mathbf{y}_t) \right\rangle_q \\
&= \underset{C}{\operatorname{argmax}} \left\langle -\frac{1}{2} \sum_t (\mathbf{x}_t - C\mathbf{y}_t)^\top R^{-1} (\mathbf{x}_t - C\mathbf{y}_t) \right\rangle_q + \text{const} \\
&= \underset{C}{\operatorname{argmax}} \left\{ -\frac{1}{2} \sum_t \mathbf{x}_t^\top R^{-1} \mathbf{x}_t - 2\mathbf{x}_t^\top R^{-1} C \langle \mathbf{y}_t \rangle + \langle \mathbf{y}_t^\top C^\top R^{-1} C \mathbf{y}_t \rangle \right\}
\end{aligned}$$

The M step for C

$$p(\mathbf{x}_t|\mathbf{y}_t) \propto \exp\left[-\tfrac{1}{2}(\mathbf{x}_t - C\mathbf{y}_t)^\mathsf{T} R^{-1}(\mathbf{x}_t - C\mathbf{y}_t)\right] \quad \Rightarrow$$

$$\begin{aligned}
C_{\text{new}} &= \underset{C}{\operatorname{argmax}} \left\langle \sum_t \ln p(\mathbf{x}_t|\mathbf{y}_t) \right\rangle_q \\
&= \underset{C}{\operatorname{argmax}} \left\langle -\frac{1}{2} \sum_t (\mathbf{x}_t - C\mathbf{y}_t)^\mathsf{T} R^{-1}(\mathbf{x}_t - C\mathbf{y}_t) \right\rangle_q + \text{const} \\
&= \underset{C}{\operatorname{argmax}} \left\{ -\frac{1}{2} \sum_t \mathbf{x}_t^\mathsf{T} R^{-1}\mathbf{x}_t - 2\mathbf{x}_t^\mathsf{T} R^{-1} C\langle \mathbf{y}_t\rangle + \langle \mathbf{y}_t^\mathsf{T} C^\mathsf{T} R^{-1} C\mathbf{y}_t\rangle \right\} \\
&= \underset{C}{\operatorname{argmax}} \left\{ \operatorname{Tr}\left[C \sum_t \langle \mathbf{y}_t\rangle \mathbf{x}_t^\mathsf{T} R^{-1} \right] - \frac{1}{2} \operatorname{Tr}\left[C^\mathsf{T} R^{-1} C \left\langle \sum_t \mathbf{y}_t \mathbf{y}_t^\mathsf{T} \right\rangle \right] \right\}
\end{aligned}$$

The M step for C

$p(\mathbf{x}_t|\mathbf{y}_t) \propto \exp\left[-\frac{1}{2}(\mathbf{x}_t - C\mathbf{y}_t)^\mathsf{T} R^{-1}(\mathbf{x}_t - C\mathbf{y}_t)\right] \quad \Rightarrow$

$$
\begin{aligned}
C_{\text{new}} &= \underset{C}{\operatorname{argmax}} \left\langle \sum_t \ln p(\mathbf{x}_t|\mathbf{y}_t) \right\rangle_q \\
&= \underset{C}{\operatorname{argmax}} \left\langle -\frac{1}{2} \sum_t (\mathbf{x}_t - C\mathbf{y}_t)^\mathsf{T} R^{-1} (\mathbf{x}_t - C\mathbf{y}_t) \right\rangle_q + \text{const} \\
&= \underset{C}{\operatorname{argmax}} \left\{ -\frac{1}{2} \sum_t \mathbf{x}_t^\mathsf{T} R^{-1} \mathbf{x}_t - 2\mathbf{x}_t^\mathsf{T} R^{-1} C \langle \mathbf{y}_t \rangle + \langle \mathbf{y}_t^\mathsf{T} C^\mathsf{T} R^{-1} C \mathbf{y}_t \rangle \right\} \\
&= \underset{C}{\operatorname{argmax}} \left\{ \operatorname{Tr}\left[C \sum_t \langle \mathbf{y}_t \rangle \mathbf{x}_t^\mathsf{T} R^{-1} \right] - \frac{1}{2} \operatorname{Tr}\left[C^\mathsf{T} R^{-1} C \left\langle \sum_t \mathbf{y}_t \mathbf{y}_t^\mathsf{T} \right\rangle \right] \right\}
\end{aligned}
$$

using $\frac{\partial \operatorname{Tr}[AB]}{\partial A} = B^\mathsf{T}$, we have $\dfrac{\partial\{\cdot\}}{\partial C} = R^{-1} \sum_t \mathbf{x}_t \langle \mathbf{y}_t \rangle^\mathsf{T} - R^{-1} C \left\langle \sum_t \mathbf{y}_t \mathbf{y}_t^\mathsf{T} \right\rangle$

The M step for C

$p(\mathbf{x}_t|\mathbf{y}_t) \propto \exp\left[-\frac{1}{2}(\mathbf{x}_t - C\mathbf{y}_t)^\top R^{-1}(\mathbf{x}_t - C\mathbf{y}_t)\right] \quad \Rightarrow$

$$\begin{aligned}
C_{\text{new}} &= \operatorname*{argmax}_C \left\langle \sum_t \ln p(\mathbf{x}_t|\mathbf{y}_t) \right\rangle_q \\
&= \operatorname*{argmax}_C \left\langle -\frac{1}{2} \sum_t (\mathbf{x}_t - C\mathbf{y}_t)^\top R^{-1} (\mathbf{x}_t - C\mathbf{y}_t) \right\rangle_q + \text{const} \\
&= \operatorname*{argmax}_C \left\{ -\frac{1}{2} \sum_t \mathbf{x}_t^\top R^{-1} \mathbf{x}_t - 2\mathbf{x}_t^\top R^{-1} C \langle \mathbf{y}_t \rangle + \langle \mathbf{y}_t^\top C^\top R^{-1} C \mathbf{y}_t \rangle \right\} \\
&= \operatorname*{argmax}_C \left\{ \text{Tr}\left[C \sum_t \langle \mathbf{y}_t \rangle \mathbf{x}_t^\top R^{-1} \right] - \frac{1}{2} \text{Tr}\left[C^\top R^{-1} C \left\langle \sum_t \mathbf{y}_t \mathbf{y}_t^\top \right\rangle \right] \right\}
\end{aligned}$$

using $\frac{\partial \text{Tr}[AB]}{\partial A} = B^\top$, we have $\dfrac{\partial\{\cdot\}}{\partial C} = R^{-1} \sum_t \mathbf{x}_t \langle \mathbf{y}_t \rangle^\top - R^{-1} C \left\langle \sum_t \mathbf{y}_t \mathbf{y}_t^\top \right\rangle$

$$\Rightarrow C_{\text{new}} = \left(\sum_t \mathbf{x}_t \langle \mathbf{y}_t \rangle^\top \right) \left(\sum_t \left\langle \mathbf{y}_t \mathbf{y}_t^\top \right\rangle \right)^{-1}$$

Notice that this is exactly the *same equation* as in factor analysis and linear regression!

The M step for A

$$p(\mathbf{y}_{t+1}|\mathbf{y}_t) \propto \exp\left\{-\frac{1}{2}(\mathbf{y}_{t+1} - A\mathbf{y}_t)^\mathsf{T} Q^{-1} (\mathbf{y}_{t+1} - A\mathbf{y}_t)\right\} \quad \Rightarrow$$

The M step for A

$$p(\mathbf{y}_{t+1}|\mathbf{y}_t) \propto \exp\left\{-\frac{1}{2}(\mathbf{y}_{t+1} - A\mathbf{y}_t)^{\mathsf{T}} Q^{-1} (\mathbf{y}_{t+1} - A\mathbf{y}_t)\right\} \quad \Rightarrow$$

$$A_{\text{new}} = \underset{A}{\operatorname{argmax}} \left\langle \sum_t \ln p(\mathbf{y}_{t+1}|\mathbf{y}_t) \right\rangle_q$$

The M step for A

$$p(\mathbf{y}_{t+1}|\mathbf{y}_t) \propto \exp\left\{-\tfrac{1}{2}(\mathbf{y}_{t+1} - A\mathbf{y}_t)^{\mathsf{T}} Q^{-1} (\mathbf{y}_{t+1} - A\mathbf{y}_t)\right\} \quad \Rightarrow$$

$$\begin{aligned} A_{\text{new}} &= \underset{A}{\operatorname{argmax}} \left\langle \sum_t \ln p(\mathbf{y}_{t+1}|\mathbf{y}_t) \right\rangle_q \\ &= \underset{A}{\operatorname{argmax}} \left\langle -\frac{1}{2} \sum_t (\mathbf{y}_{t+1} - A\mathbf{y}_t)^{\mathsf{T}} Q^{-1} (\mathbf{y}_{t+1} - A\mathbf{y}_t) \right\rangle_q + \text{const} \end{aligned}$$

The M step for A

$$p(\mathbf{y}_{t+1}|\mathbf{y}_t) \propto \exp\left\{-\tfrac{1}{2}(\mathbf{y}_{t+1} - A\mathbf{y}_t)^\mathsf{T} Q^{-1}(\mathbf{y}_{t+1} - A\mathbf{y}_t)\right\} \quad \Rightarrow$$

$$\begin{aligned}
A_{\text{new}} &= \underset{A}{\operatorname{argmax}} \left\langle \sum_t \ln p(\mathbf{y}_{t+1}|\mathbf{y}_t) \right\rangle_q \\
&= \underset{A}{\operatorname{argmax}} \left\langle -\frac{1}{2} \sum_t (\mathbf{y}_{t+1} - A\mathbf{y}_t)^\mathsf{T} Q^{-1}(\mathbf{y}_{t+1} - A\mathbf{y}_t) \right\rangle_q + \text{const} \\
&= \underset{A}{\operatorname{argmax}} \left\{ -\frac{1}{2} \sum_t \mathbf{y}_{t+1}^\mathsf{T} Q^{-1}\mathbf{y}_{t+1} - 2\left\langle \mathbf{y}_{t+1}^\mathsf{T} Q^{-1} A\mathbf{y}_t \right\rangle + \left\langle \mathbf{y}_t^\mathsf{T} A^\mathsf{T} Q^{-1} A\mathbf{y}_t \right\rangle \right\}
\end{aligned}$$

The M step for A

$$p(\mathbf{y}_{t+1}|\mathbf{y}_t) \propto \exp\left\{-\tfrac{1}{2}(\mathbf{y}_{t+1} - A\mathbf{y}_t)^\mathsf{T} Q^{-1}(\mathbf{y}_{t+1} - A\mathbf{y}_t)\right\} \quad \Rightarrow$$

$$\begin{aligned}
A_{\text{new}} &= \underset{A}{\operatorname{argmax}} \left\langle \sum_t \ln p(\mathbf{y}_{t+1}|\mathbf{y}_t) \right\rangle_q \\
&= \underset{A}{\operatorname{argmax}} \left\langle -\frac{1}{2}\sum_t (\mathbf{y}_{t+1} - A\mathbf{y}_t)^\mathsf{T} Q^{-1}(\mathbf{y}_{t+1} - A\mathbf{y}_t) \right\rangle_q + \text{const} \\
&= \underset{A}{\operatorname{argmax}} \left\{ -\frac{1}{2}\sum_t \mathbf{y}_{t+1}^\mathsf{T} Q^{-1}\mathbf{y}_{t+1} - 2\left\langle \mathbf{y}_{t+1}^\mathsf{T} Q^{-1} A\mathbf{y}_t \right\rangle + \left\langle \mathbf{y}_t^\mathsf{T} A^\mathsf{T} Q^{-1} A\mathbf{y}_t \right\rangle \right\} \\
&= \underset{A}{\operatorname{argmax}} \left\{ \operatorname{Tr}\left[A \sum_t \left\langle \mathbf{y}_t \mathbf{y}_{t+1}^\mathsf{T} \right\rangle Q^{-1} \right] - \frac{1}{2}\operatorname{Tr}\left[A^\mathsf{T} Q^{-1} A \sum_t \left\langle \mathbf{y}_t \mathbf{y}_t^\mathsf{T} \right\rangle \right] \right\}
\end{aligned}$$

The M step for A

$$p(\mathbf{y}_{t+1}|\mathbf{y}_t) \propto \exp\left\{-\tfrac{1}{2}(\mathbf{y}_{t+1} - A\mathbf{y}_t)^\top Q^{-1}(\mathbf{y}_{t+1} - A\mathbf{y}_t)\right\} \quad \Rightarrow$$

$$\begin{aligned}
A_{\text{new}} &= \operatorname*{argmax}_A \left\langle \sum_t \ln p(\mathbf{y}_{t+1}|\mathbf{y}_t) \right\rangle_q \\
&= \operatorname*{argmax}_A \left\langle -\frac{1}{2} \sum_t (\mathbf{y}_{t+1} - A\mathbf{y}_t)^\top Q^{-1}(\mathbf{y}_{t+1} - A\mathbf{y}_t) \right\rangle_q + \text{const} \\
&= \operatorname*{argmax}_A \left\{ -\frac{1}{2} \sum_t \mathbf{y}_{t+1}^\top Q^{-1}\mathbf{y}_{t+1} - 2\left\langle \mathbf{y}_{t+1}^\top Q^{-1} A\mathbf{y}_t \right\rangle + \left\langle \mathbf{y}_t^\top A^\top Q^{-1} A\mathbf{y}_t \right\rangle \right\} \\
&= \operatorname*{argmax}_A \left\{ \operatorname{Tr}\left[A \sum_t \left\langle \mathbf{y}_t \mathbf{y}_{t+1}^\top \right\rangle Q^{-1} \right] - \frac{1}{2} \operatorname{Tr}\left[A^\top Q^{-1} A \sum_t \left\langle \mathbf{y}_t \mathbf{y}_t^\top \right\rangle \right] \right\}
\end{aligned}$$

using $\frac{\partial \operatorname{Tr}[AB]}{\partial A} = B^\top$, we have $\dfrac{\partial\{\cdot\}}{\partial A} = Q^{-1} \sum_t \left\langle \mathbf{y}_{t+1}\mathbf{y}_t^\top \right\rangle - Q^{-1} A \sum_t \left\langle \mathbf{y}_t \mathbf{y}_t^\top \right\rangle$

The M step for A

$$p(\mathbf{y}_{t+1}|\mathbf{y}_t) \propto \exp\left\{-\tfrac{1}{2}(\mathbf{y}_{t+1} - A\mathbf{y}_t)^{\mathsf{T}} Q^{-1} (\mathbf{y}_{t+1} - A\mathbf{y}_t)\right\} \quad \Rightarrow$$

$$\begin{aligned}
A_{\text{new}} &= \underset{A}{\operatorname{argmax}} \left\langle \sum_t \ln p(\mathbf{y}_{t+1}|\mathbf{y}_t) \right\rangle_q \\
&= \underset{A}{\operatorname{argmax}} \left\langle -\frac{1}{2} \sum_t (\mathbf{y}_{t+1} - A\mathbf{y}_t)^{\mathsf{T}} Q^{-1} (\mathbf{y}_{t+1} - A\mathbf{y}_t) \right\rangle_q + \text{const} \\
&= \underset{A}{\operatorname{argmax}} \left\{ -\frac{1}{2} \sum_t \mathbf{y}_{t+1}^{\mathsf{T}} Q^{-1} \mathbf{y}_{t+1} - 2\left\langle \mathbf{y}_{t+1}^{\mathsf{T}} Q^{-1} A\mathbf{y}_t \right\rangle + \left\langle \mathbf{y}_t^{\mathsf{T}} A^{\mathsf{T}} Q^{-1} A\mathbf{y}_t \right\rangle \right\} \\
&= \underset{A}{\operatorname{argmax}} \left\{ \operatorname{Tr}\left[A \sum_t \left\langle \mathbf{y}_t \mathbf{y}_{t+1}^{\mathsf{T}} \right\rangle Q^{-1} \right] - \frac{1}{2} \operatorname{Tr}\left[A^{\mathsf{T}} Q^{-1} A \sum_t \left\langle \mathbf{y}_t \mathbf{y}_t^{\mathsf{T}} \right\rangle \right] \right\}
\end{aligned}$$

using $\frac{\partial \operatorname{Tr}[AB]}{\partial A} = B^{\mathsf{T}}$, we have $\displaystyle \frac{\partial\{\cdot\}}{\partial A} = Q^{-1} \sum_t \left\langle \mathbf{y}_{t+1} \mathbf{y}_t^{\mathsf{T}} \right\rangle - Q^{-1} A \sum_t \left\langle \mathbf{y}_t \mathbf{y}_t^{\mathsf{T}} \right\rangle$

$$\Rightarrow A_{\text{new}} = \left(\sum_t \left\langle \mathbf{y}_{t+1} \mathbf{y}_t^{\mathsf{T}} \right\rangle \right) \left(\sum_t \left\langle \mathbf{y}_t \mathbf{y}_t^{\mathsf{T}} \right\rangle \right)^{-1}$$

This is still analagous to factor analysis and linear regression, with expected correlations.

Learning (online gradient)

Time series data must often be processed in real-time, and we may want to update parameters online as observations arrive. We can do so by updating a local version of the likelihood based on the Kalman filter estimates.
Consider the log likelihood contributed by each data point (ℓ_t):

$$\ell = \sum_{t=1}^{T} \ln p(\mathbf{x}_t | \mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) = \sum_{t=1}^{T} \ell_t$$

Then,

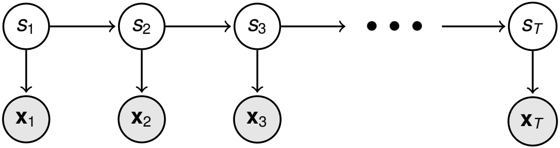
$$\ell_t = -\frac{D}{2} \ln 2\pi - \frac{1}{2} \ln |\Sigma| - \frac{1}{2} (\mathbf{x}_t - C\hat{\mathbf{y}}_t^{t-1})^{\mathsf{T}} \Sigma^{-1} (\mathbf{x}_t - C\hat{\mathbf{y}}_t^{t-1})$$

where D is dimension of $\mathbf{x}$, and:

$$\hat{\mathbf{y}}_t^{t-1} = A\hat{\mathbf{y}}_{t-1}^{t-1}$$
$$\Sigma = C\hat{V}_t^{t-1} C^{\mathsf{T}} + R$$
$$\hat{V}_t^{t-1} = A\hat{V}_{t-1}^{t-1} A^{\mathsf{T}} + Q$$

We differentiate ℓ_t to obtain gradient rules for A, C, Q, R. The size of the gradient step (learning rate) reflects our expectation about nonstationarity.

Learning HMMs using EM



Parameters: $\theta = \{\boldsymbol{\pi}, \Phi, \mathrm{A}\}$

Free energy:

$$\mathcal{F}(q, \theta) = \sum_{s_{1:T}} q(s_{1:T})(\log P(x_{1:T}, s_{1:T}|\theta) - \log q(s_{1:T}))$$

E-step: Maximise $\mathcal{F}$ w.r.t. q with θ fixed: $q^*(s_{1:T}) = P(s_{1:T}|\mathbf{x}_{1:T}, \theta)$

We will only need the marginal probabilities $q(s_t, s_{t+1})$, which can also be obtained from the forward–backward algorithm.

M-step: Maximize $\mathcal{F}$ w.r.t. θ with q fixed.

We can re-estimate the parameters by computing the expected number of times the HMM was in state i, emitted symbol k and transitioned to state j.

This is the **Baum-Welch algorithm** and it predates the (more general) EM algorithm.

M step: Parameter updates are given by ratios of expected counts

We can derive the following updates by taking derivatives of $\mathcal{F}$ w.r.t. θ.

M step: Parameter updates are given by ratios of expected counts

We can derive the following updates by taking derivatives of $\mathcal{F}$ w.r.t. θ.

- The initial state distribution is the expected number of times in state i at $t = 1$:

$$\hat{\pi}_i = \gamma_1(i)$$

M step: Parameter updates are given by ratios of expected counts

We can derive the following updates by taking derivatives of $\mathcal{F}$ w.r.t. θ.

- The initial state distribution is the expected number of times in state i at $t = 1$:

$$\hat{\pi}_i = \gamma_1(i)$$

- The expected number of transitions from state i to j which begin at time t is:

$$\xi_t(i \to j) \equiv P(s_t = i, s_{t+1} = j | \mathbf{x}_{1:T}) = \alpha_t(i) \Phi_{ij} A_j(\mathbf{x}_{t+1}) \beta_{t+1}(j) / P(x_{1:T})$$

  so the estimated transition probabilities are:

$$\widehat{\Phi}_{ij} = \sum_{t=1}^{T-1} \xi_t(i \to j) \Bigg/ \sum_{t=1}^{T-1} \gamma_t(i)$$

M step: Parameter updates are given by ratios of expected counts

We can derive the following updates by taking derivatives of $\mathcal{F}$ w.r.t. θ.

- The initial state distribution is the expected number of times in state i at $t = 1$:

$$\hat{\pi}_i = \gamma_1(i)$$

- The expected number of transitions from state i to j which begin at time t is:

$$\xi_t(i \to j) \equiv P(s_t = i, s_{t+1} = j | \mathbf{x}_{1:T}) = \alpha_t(i)\Phi_{ij}A_j(\mathbf{x}_{t+1})\beta_{t+1}(j)/P(x_{1:T})$$

  so the estimated transition probabilities are:

$$\widehat{\Phi}_{ij} = \sum_{t=1}^{T-1} \xi_t(i \to j) \Bigg/ \sum_{t=1}^{T-1} \gamma_t(i)$$

- The output distributions are the expected number of times we observe a particular symbol in a particular state:

$$\widehat{A}_{ik} = \sum_{t:\mathbf{x}_t=k} \gamma_t(i) \Bigg/ \sum_{t=1}^{T} \gamma_t(i)$$

  (or the state-probability-weighted mean and variance for a Gaussian output model).

HMM practicalities

- Numerical scaling: the conventional message definition is in terms of a large joint:

$$\alpha_t(i) = P(\mathbf{x}_{1:t}, s_t{=}i) \;\; \to 0 \text{ as } t \text{ grows, and so can easily underflow.}$$

Rescale:

$$\overline{\alpha}_t(i) = A_i(\mathbf{x}_t) \sum_j \tilde{\alpha}_{t-1}(j)\Phi_{ji} \qquad \rho_t = \sum_{i=1}^{K} \overline{\alpha}_t(i) \qquad \tilde{\alpha}_t(i) = \overline{\alpha}_t(i)/\rho_t$$

Exercise: show that:

$$\rho_t = P(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \theta) \qquad \prod_{t=1}^{T} \rho_t = P(\mathbf{x}_{1:T}|\theta)$$

What does this make $\tilde{\alpha}_t(i)$?

HMM practicalities

- Numerical scaling: the conventional message definition is in terms of a large joint:

  $\alpha_t(i) = P(\mathbf{x}_{1:t}, s_t{=}i) \;\; \to 0$ as t grows, and so can easily underflow.

  Rescale:

$$\overline{\alpha}_t(i) = A_i(\mathbf{x}_t) \sum_j \tilde{\alpha}_{t-1}(j)\Phi_{ji} \qquad \rho_t = \sum_{i=1}^{K} \overline{\alpha}_t(i) \qquad \tilde{\alpha}_t(i) = \overline{\alpha}_t(i)/\rho_t$$

  Exercise: show that:

$$\rho_t = P(\mathbf{x}_t | \mathbf{x}_{1:t-1}, \theta) \qquad \prod_{t=1}^{T} \rho_t = P(\mathbf{x}_{1:T} | \theta)$$

  What does this make $\tilde{\alpha}_t(i)$?

- Multiple observed sequences: average numerators and denominators in the ratios of updates.

HMM practicalities

- Numerical scaling: the conventional message definition is in terms of a large joint:

  $\alpha_t(i) = P(\mathbf{x}_{1:t}, s_t{=}i)$ $\rightarrow$ 0 as t grows, and so can easily underflow.

  Rescale:

$$\overline{\alpha}_t(i) = A_i(\mathbf{x}_t) \sum_j \tilde{\alpha}_{t-1}(j)\Phi_{ji} \qquad \rho_t = \sum_{i=1}^{K} \overline{\alpha}_t(i) \qquad \tilde{\alpha}_t(i) = \overline{\alpha}_t(i)/\rho_t$$

  Exercise: show that:

$$\rho_t = P(\mathbf{x}_t | \mathbf{x}_{1:t-1}, \theta) \qquad \prod_{t=1}^{T} \rho_t = P(\mathbf{x}_{1:T} | \theta)$$

  What does this make $\tilde{\alpha}_t(i)$?

- Multiple observed sequences: average numerators and denominators in the ratios of updates.
- Local optima (random restarts, annealing; see discussion later).

HMM pseudocode: inference (E step)

Forward-backward including scaling tricks.
[$\circ$ is the element-by-element (Hadamard/Schur) product: '$.*$' in matlab.]

$$\text{for } t = 1:T,\ i = 1:K \qquad p_t(i) = A_i(\mathbf{x}_t)$$

$$\alpha_1 = \boldsymbol{\pi} \circ p_1 \qquad \rho_1 = \textstyle\sum_{i=1}^K \alpha_1(i) \qquad \alpha_1 = \alpha_1/\rho_1$$

$$\text{for } t = 2:T \qquad \alpha_t = (\Phi^{\mathsf{T}} * \alpha_{t-1}) \circ p_t \qquad \rho_t = \textstyle\sum_{i=1}^K \alpha_t(i) \qquad \alpha_t = \alpha_t/\rho_t$$

$$\beta_T = 1$$

$$\text{for } t = T-1:1 \qquad \beta_t = \Phi * (\beta_{t+1} \circ p_{t+1})/\rho_{t+1}$$

$$\log P(\mathbf{x}_{1:T}) = \textstyle\sum_{t=1}^T \log(\rho_t)$$

$$\text{for } t = 1:T \qquad \gamma_t = \alpha_t \circ \beta_t$$

$$\text{for } t = 1:T-1 \qquad \xi_t = \Phi \circ (\alpha_t * (\beta_{t+1} \circ p_{t+1})^{\mathsf{T}})/\rho_{t+1}$$

HMM pseudocode: parameter re-estimation (M step)

Baum-Welch parameter updates:

For each sequence $l = 1 : L$, run forward–backward to get $\gamma^{(l)}$ and $\xi^{(l)}$, then

$$\pi_i = \frac{1}{L} \sum_{l=1}^{L} \gamma_1^{(l)}(i)$$

$$\Phi_{ij} = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}-1} \xi_t^{(l)}(ij)}{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}-1} \gamma_t^{(l)}(i)}$$

$$A_{ik} = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}} \delta(\mathbf{x}_t = k) \gamma_t^{(l)}(i)}{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}} \gamma_t^{(l)}(i)}$$

Degeneracies

Recall that the FA likelihood is conserved with respect to orthogonal transformations of $\mathbf{y}$:

$$
\begin{aligned}
P(\mathbf{y}) &= \mathcal{N}(\mathbf{0}, I) \\
P(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\Lambda\mathbf{y}, \Psi)
\end{aligned}
$$

Degeneracies

Recall that the FA likelihood is conserved with respect to orthogonal transformations of $\mathbf{y}$:

$$\begin{aligned} P(\mathbf{y}) &= \mathcal{N}(\mathbf{0}, I) \\ P(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\Lambda\mathbf{y}, \Psi) \end{aligned} \quad \& \quad \begin{aligned} \tilde{\mathbf{y}} &= U\mathbf{y} \\ \tilde{\Lambda} &= \Lambda U^{\mathsf{T}} \end{aligned}$$

Degeneracies

Recall that the FA likelihood is conserved with respect to orthogonal transformations of **y**:

$$\begin{aligned} P(\mathbf{y}) &= \mathcal{N}(\mathbf{0}, I) \\ P(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\Lambda\mathbf{y}, \Psi) \end{aligned} \quad \& \quad \begin{aligned} \tilde{\mathbf{y}} &= U\mathbf{y} \\ \tilde{\Lambda} &= \Lambda U^{\mathsf{T}} \end{aligned} \quad \Rightarrow \quad \begin{aligned} P(\tilde{\mathbf{y}}) &= \mathcal{N}\left(U\mathbf{0}, UIU^{\mathsf{T}}\right) = \mathcal{N}(\mathbf{0}, I) \\ P(\mathbf{x}|\tilde{\mathbf{y}}) &= \mathcal{N}\left(\Lambda U^{\mathsf{T}} U\mathbf{y}, \Psi\right) = \mathcal{N}\left(\tilde{\Lambda}\tilde{\mathbf{y}}, \Psi\right) \end{aligned}$$

Degeneracies

Recall that the FA likelihood is conserved with respect to orthogonal transformations of **y**:

$$\begin{array}{c} P(\mathbf{y}) = \mathcal{N}(\mathbf{0}, I) \\ P(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\Lambda\mathbf{y}, \Psi) \end{array} \quad \& \quad \begin{array}{c} \tilde{\mathbf{y}} = U\mathbf{y} \\ \tilde{\Lambda} = \Lambda U^{\mathsf{T}} \end{array} \quad \Rightarrow \quad \begin{array}{c} P(\tilde{\mathbf{y}}) = \mathcal{N}\left(U\mathbf{0}, UIU^{\mathsf{T}}\right) = \mathcal{N}(\mathbf{0}, I) \\ P(\mathbf{x}|\tilde{\mathbf{y}}) = \mathcal{N}\left(\Lambda U^{\mathsf{T}} U\mathbf{y}, \Psi\right) = \mathcal{N}\left(\tilde{\Lambda}\tilde{\mathbf{y}}, \Psi\right) \end{array}$$

Similarly, a mixture model is invariant to permutations of the latent.

Degeneracies

Recall that the FA likelihood is conserved with respect to orthogonal transformations of $\mathbf{y}$:

$$\begin{aligned} P(\mathbf{y}) &= \mathcal{N}(\mathbf{0}, I) \\ P(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\Lambda\mathbf{y}, \Psi) \end{aligned} \quad \& \quad \begin{aligned} \tilde{\mathbf{y}} &= U\mathbf{y} \\ \tilde{\Lambda} &= \Lambda U^{\mathsf{T}} \end{aligned} \quad \Rightarrow \quad \begin{aligned} P(\tilde{\mathbf{y}}) &= \mathcal{N}\left(U\mathbf{0}, UIU^{\mathsf{T}}\right) = \mathcal{N}(\mathbf{0}, I) \\ P(\mathbf{x}|\tilde{\mathbf{y}}) &= \mathcal{N}\left(\Lambda U^{\mathsf{T}} U\mathbf{y}, \Psi\right) = \mathcal{N}\left(\tilde{\Lambda}\tilde{\mathbf{y}}, \Psi\right) \end{aligned}$$

Similarly, a mixture model is invariant to permutations of the latent.

The LGSSM likelihood is conserved with respect to any invertible transform of the latent:

$$\begin{aligned} P(\mathbf{y}_{t+1}|\mathbf{y}_t) &= \mathcal{N}(A\mathbf{y}_t, Q) \\ P(\mathbf{x}_t|\mathbf{y}_t) &= \mathcal{N}(C\mathbf{y}, R) \end{aligned}$$

Degeneracies

Recall that the FA likelihood is conserved with respect to orthogonal transformations of $\mathbf{y}$:

$$\begin{array}{c} P(\mathbf{y}) = \mathcal{N}(\mathbf{0}, I) \\ P(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\Lambda\mathbf{y}, \Psi) \end{array} \quad \& \quad \begin{array}{c} \tilde{\mathbf{y}} = U\mathbf{y} \\ \tilde{\Lambda} = \Lambda U^{\mathsf{T}} \end{array} \quad \Rightarrow \quad \begin{array}{c} P(\tilde{\mathbf{y}}) = \mathcal{N}\left(U\mathbf{0}, UIU^{\mathsf{T}}\right) = \mathcal{N}(\mathbf{0}, I) \\ P(\mathbf{x}|\tilde{\mathbf{y}}) = \mathcal{N}\left(\Lambda U^{\mathsf{T}} U\mathbf{y}, \Psi\right) = \mathcal{N}\left(\tilde{\Lambda}\tilde{\mathbf{y}}, \Psi\right) \end{array}$$

Similarly, a mixture model is invariant to permutations of the latent.

The LGSSM likelihood is conserved with respect to any invertible transform of the latent:

$$\begin{array}{c} P(\mathbf{y}_{t+1}|\mathbf{y}_t) = \mathcal{N}(A\mathbf{y}_t, Q) \\ P(\mathbf{x}_t|\mathbf{y}_t) = \mathcal{N}(C\mathbf{y}, R) \end{array} \quad \& \quad \begin{array}{cc} \tilde{\mathbf{y}} = G\mathbf{y} & \tilde{A} = GAG^{-1} \\ \tilde{Q} = GAG^{\mathsf{T}} & \tilde{C} = CG^{-1} \end{array}$$

Degeneracies

Recall that the FA likelihood is conserved with respect to orthogonal transformations of $\mathbf{y}$:

$$\begin{aligned} P(\mathbf{y}) &= \mathcal{N}(\mathbf{0}, I) \\ P(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\Lambda\mathbf{y}, \Psi) \end{aligned} \quad \& \quad \begin{aligned} \tilde{\mathbf{y}} &= U\mathbf{y} \\ \tilde{\Lambda} &= \Lambda U^{\mathsf{T}} \end{aligned} \quad \Rightarrow \quad \begin{aligned} P(\tilde{\mathbf{y}}) &= \mathcal{N}\left(U\mathbf{0}, UIU^{\mathsf{T}}\right) = \mathcal{N}(\mathbf{0}, I) \\ P(\mathbf{x}|\tilde{\mathbf{y}}) &= \mathcal{N}\left(\Lambda U^{\mathsf{T}}U\mathbf{y}, \Psi\right) = \mathcal{N}\left(\tilde{\Lambda}\tilde{\mathbf{y}}, \Psi\right) \end{aligned}$$

Similarly, a mixture model is invariant to permutations of the latent.

The LGSSM likelihood is conserved with respect to any invertible transform of the latent:

$$\begin{aligned} P(\mathbf{y}_{t+1}|\mathbf{y}_t) &= \mathcal{N}(A\mathbf{y}_t, Q) \\ P(\mathbf{x}_t|\mathbf{y}_t) &= \mathcal{N}(C\mathbf{y}, R) \end{aligned} \quad \& \quad \begin{aligned} \tilde{\mathbf{y}} &= G\mathbf{y} & \tilde{A} &= GAG^{-1} \\ \tilde{Q} &= GAG^{\mathsf{T}} & \tilde{C} &= CG^{-1} \end{aligned}$$

$$\Rightarrow \quad \begin{aligned} P(\tilde{\mathbf{y}}_{t+1}|\tilde{\mathbf{y}}_t) &= \mathcal{N}\left(GAG^{-1}G\mathbf{y}_t, GQG^{\mathsf{T}}\right) = \mathcal{N}(\tilde{A}\tilde{\mathbf{y}}_t, \tilde{Q}) \\ P(\mathbf{x}_t|\tilde{\mathbf{y}}_t) &= \mathcal{N}\left(CG^{-1}G\mathbf{y}, R\right) = \mathcal{N}\left(\tilde{C}\tilde{\mathbf{y}}, R\right) \end{aligned}$$

Degeneracies

Recall that the FA likelihood is conserved with respect to orthogonal transformations of **y**:

$$\begin{array}{c} P(\mathbf{y}) = \mathcal{N}(\mathbf{0}, I) \\ P(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\Lambda\mathbf{y}, \Psi) \end{array} \quad \& \quad \begin{array}{c} \tilde{\mathbf{y}} = U\mathbf{y} \\ \tilde{\Lambda} = \Lambda U^{\mathsf{T}} \end{array} \quad \Rightarrow \quad \begin{array}{c} P(\tilde{\mathbf{y}}) = \mathcal{N}\left(U\mathbf{0}, UIU^{\mathsf{T}}\right) = \mathcal{N}(\mathbf{0}, I) \\ P(\mathbf{x}|\tilde{\mathbf{y}}) = \mathcal{N}\left(\Lambda U^{\mathsf{T}} U\mathbf{y}, \Psi\right) = \mathcal{N}\left(\tilde{\Lambda}\tilde{\mathbf{y}}, \Psi\right) \end{array}$$

Similarly, a mixture model is invariant to permutations of the latent.

The LGSSM likelihood is conserved with respect to any invertible transform of the latent:

$$\begin{array}{c} P(\mathbf{y}_{t+1}|\mathbf{y}_t) = \mathcal{N}(A\mathbf{y}_t, Q) \\ P(\mathbf{x}_t|\mathbf{y}_t) = \mathcal{N}(C\mathbf{y}, R) \end{array} \quad \& \quad \begin{array}{cc} \tilde{\mathbf{y}} = G\mathbf{y} & \tilde{A} = GAG^{-1} \\ \tilde{Q} = GAG^{\mathsf{T}} & \tilde{C} = CG^{-1} \end{array}$$

$$\Rightarrow \begin{array}{c} P(\tilde{\mathbf{y}}_{t+1}|\tilde{\mathbf{y}}_t) = \mathcal{N}\left(GAG^{-1}G\mathbf{y}_t, GQG^{\mathsf{T}}\right) = \mathcal{N}\left(\tilde{A}\tilde{\mathbf{y}}_t, \tilde{Q}\right) \\ P(\mathbf{x}_t|\tilde{\mathbf{y}}_t) = \mathcal{N}\left(CG^{-1}G\mathbf{y}, R\right) = \mathcal{N}\left(\tilde{C}\tilde{\mathbf{y}}, R\right) \end{array}$$

and the HMM is invariant to permutations (and to relaxations into something called an observable operator model).

The likelihood landscape

Besides the degenerate solutions, both LGSSMs and HMMs likelihood functions tend to have multiple local maxima. This complicates any ML-based approach to learning (including EM and gradient methods).

The likelihood landscape

Besides the degenerate solutions, both LGSSMs and HMMs likelihood functions tend to have multiple local maxima. This complicates any ML-based approach to learning (including EM and gradient methods).

Strategies:

- Restart EM/gradient ascent from different (often random) initial parameter values.

The likelihood landscape

Besides the degenerate solutions, both LGSSMs and HMMs likelihood functions tend to have multiple local maxima. This complicates any ML-based approach to learning (including EM and gradient methods).

Strategies:

- Restart EM/gradient ascent from different (often random) initial parameter values.
- Initialise output parameters with a stationary model:

  PCA $\rightarrow$ FA $\rightarrow$ LGSSM

  k-means $\rightarrow$ mixture $\rightarrow$ HMM

The likelihood landscape

Besides the degenerate solutions, both LGSSMs and HMMs likelihood functions tend to have multiple local maxima. This complicates any ML-based approach to learning (including EM and gradient methods).

Strategies:

- Restart EM/gradient ascent from different (often random) initial parameter values.
- Initialise output parameters with a stationary model:

  PCA $\rightarrow$ FA $\rightarrow$ LGSSM

  k-means $\rightarrow$ mixture $\rightarrow$ HMM

- Stochastic gradient methods and momentum. "Overstepping" EM.

The likelihood landscape

Besides the degenerate solutions, both LGSSMs and HMMs likelihood functions tend to have multiple local maxima. This complicates any ML-based approach to learning (including EM and gradient methods).

Strategies:

- Restart EM/gradient ascent from different (often random) initial parameter values.
- Initialise output parameters with a stationary model:

  PCA $\rightarrow$ FA $\rightarrow$ LGSSM

  k-means $\rightarrow$ mixture $\rightarrow$ HMM

- Stochastic gradient methods and momentum. "Overstepping" EM.
- Deterministic annealing.

The likelihood landscape

Besides the degenerate solutions, both LGSSMs and HMMs likelihood functions tend to have multiple local maxima. This complicates any ML-based approach to learning (including EM and gradient methods).

Strategies:

- Restart EM/gradient ascent from different (often random) initial parameter values.
- Initialise output parameters with a stationary model:

  PCA $\rightarrow$ FA $\rightarrow$ LGSSM

  k-means $\rightarrow$ mixture $\rightarrow$ HMM

- Stochastic gradient methods and momentum. "Overstepping" EM.
- Deterministic annealing.
- Non-ML learning (spectral methods).

Slow Feature Analysis (SFA)

Is there a zero-noise limit for an LGSSM analagous to PCA?

Slow Feature Analysis (SFA)

Is there a zero-noise limit for an LGSSM analagous to PCA?

We can take:

$$A = \mathrm{diag}\left[a_1 \dots a_K\right] \to I \quad (\text{with } a_1 \le a_2 \le \dots \le a_K \le 1)$$

$$Q = I - AA^{\mathsf{T}} \to 0 \quad (\text{setting the stationary latent covariance to } I)$$

$$R \to 0\,.$$

This limit yields a spectral algorithm called Slow Feature Analysis (or SFA).

Slow Feature Analysis (SFA)

Is there a zero-noise limit for an LGSSM analagous to PCA?

We can take:

$$A = \operatorname{diag}[a_1 \dots a_K] \to I \quad (\text{with } a_1 \leq a_2 \leq \cdots \leq a_K \leq 1)$$

$$Q = I - AA^{\mathsf{T}} \to 0 \quad (\text{setting the stationary latent covariance to } I)$$

$$R \to 0\,.$$

This limit yields a spectral algorithm called Slow Feature Analysis (or SFA).

SFA is conventionally defined as slowness pursuit (cf PCA and variance pursuit):
Given zero-mean series $\{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_T\}$ find a $K \times D$ matrix $W (= C^{\mathsf{T}})$ such that $\mathbf{y}_t = W\mathbf{x}_t$ changes as slowly as possible.

Slow Feature Analysis (SFA)

Is there a zero-noise limit for an LGSSM analagous to PCA?

We can take:

$$A = \mathrm{diag}\left[a_1 \ldots a_K\right] \to I \quad (\text{with } a_1 \leq a_2 \leq \cdots \leq a_K \leq 1)$$

$$Q = I - AA^{\mathsf{T}} \to 0 \quad (\text{setting the stationary latent covariance to } I)$$

$$R \to 0\,.$$

This limit yields a spectral algorithm called Slow Feature Analysis (or SFA).

SFA is conventionally defined as slowness pursuit (cf PCA and variance pursuit):
Given zero-mean series $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_T\}$ find a $K \times D$ matrix $W (= C^{\mathsf{T}})$ such that $\mathbf{y}_t = W\mathbf{x}_t$ changes as slowly as possible.

Specifically, find

$$W = \underset{W}{\mathrm{argmin}} \sum_t \|\mathbf{y}_t - \mathbf{y}_{t-1}\|^2 \quad \text{subject to} \quad \sum_t \mathbf{y}_t \mathbf{y}_t^{\mathsf{T}} = I\,.$$

The variance constraint prevents the trivial solutions $W = 0$ and $W = \mathbf{1}\mathbf{w}^{\mathsf{T}}$.

Slow Feature Analysis (SFA)

Is there a zero-noise limit for an LGSSM analagous to PCA?

We can take:

$$A = \operatorname{diag}[a_1 \dots a_K] \to I \quad (\text{with } a_1 \leq a_2 \leq \dots \leq a_K \leq 1)$$

$$Q = I - AA^{\mathsf{T}} \to 0 \quad (\text{setting the stationary latent covariance to } I)$$

$$R \to 0\,.$$

This limit yields a spectral algorithm called Slow Feature Analysis (or SFA).

SFA is conventionally defined as slowness pursuit (cf PCA and variance pursuit):
Given zero-mean series $\{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_T\}$ find a $K \times D$ matrix $W (= C^{\mathsf{T}})$ such that $\mathbf{y}_t = W\mathbf{x}_t$ changes as slowly as possible.

Specifically, find

$$W = \operatorname*{argmin}_W \sum_t \|\mathbf{y}_t - \mathbf{y}_{t-1}\|^2 \quad \text{subject to} \quad \sum_t \mathbf{y}_t \mathbf{y}_t^{\mathsf{T}} = I\,.$$

The variance constraint prevents the trivial solutions $W = 0$ and $W = \mathbf{1}\mathbf{w}^{\mathsf{T}}$.

W can be found by solving the generalised eigenvalue problem

$$WA = \Omega WB \quad \text{where } A = \textstyle\sum_t (\mathbf{x}_t - \mathbf{x}_{t-1})(\mathbf{x}_t - \mathbf{x}_{t-1})^{\mathsf{T}} \text{ and } B = \sum_t \mathbf{x}_t \mathbf{x}_t^{\mathsf{T}}\,.$$

See `http://www.gatsby.ucl.ac.uk/~maneesh/papers/turner-sahani-2007-sfa.pdf`.

(Generalised) Method of Moments estimators

What if we don't want the $A \to I$ limit?

(Generalised) Method of Moments estimators

What if we don't want the $A \to I$ limit?

The limit is necessary to make the spectral estimate and the (limiting) ML estimates coincide. However, if we give up on ML, then there are general spectral estimates available.

(Generalised) Method of Moments estimators

What if we don't want the $A \to I$ limit?

The limit is necessary to make the spectral estimate and the (limiting) ML estimates coincide. However, if we give up on ML, then there are general spectral estimates available.

The ML parameter estimates are defined:

$$\theta^{\text{ML}} = \operatorname*{argmax}_{\theta} \log P(\mathcal{X}|\theta)$$

and as you found, if $P \in$ ExpFam with sufficient statistic T then

$$\langle T(\mathbf{x}) \rangle_{\theta^{\text{ML}}} = \frac{1}{N} \sum_i T(\mathbf{x}_i) \qquad \text{(moment matching)}\,.$$

(Generalised) Method of Moments estimators

What if we don't want the $A \to I$ limit?

The limit is necessary to make the spectral estimate and the (limiting) ML estimates coincide. However, if we give up on ML, then there are general spectral estimates available.

The ML parameter estimates are defined:

$$\theta^{\mathrm{ML}} = \operatorname*{argmax}_{\theta} \log P(\mathcal{X}|\theta)$$

and as you found, if $P \in \text{ExpFam}$ with sufficient statistic T then

$$\langle T(\mathbf{x}) \rangle_{\theta^{\mathrm{ML}}} = \frac{1}{N} \sum_i T(\mathbf{x}_i) \qquad \text{(moment matching)}\,.$$

We can generalise this condition to arbitrary T even if $P \notin \text{ExpFam}$.

That is, we find estimate θ^* with

$$\langle T(\mathbf{x}) \rangle_{\theta^*} = \frac{1}{N} \sum_i T(\mathbf{x}_i) \quad \text{or} \quad \theta^* = \operatorname*{argmin}_{\theta} \left\| \langle T(\mathbf{x}) \rangle_{\theta} - \frac{1}{N} \sum_i T(\mathbf{x}_i) \right\|_{\mathcal{C}}$$

(Generalised) Method of Moments estimators

What if we don't want the $A \to I$ limit?

The limit is necessary to make the spectral estimate and the (limiting) ML estimates coincide. However, if we give up on ML, then there are general spectral estimates available.

The ML parameter estimates are defined:

$$\theta^{\text{ML}} = \underset{\theta}{\text{argmax}} \log P(\mathcal{X}|\theta)$$

and as you found, if $P \in \text{ExpFam}$ with sufficient statistic T then

$$\langle T(\mathbf{x}) \rangle_{\theta^{\text{ML}}} = \frac{1}{N} \sum_i T(\mathbf{x}_i) \qquad \text{(moment matching)}\,.$$

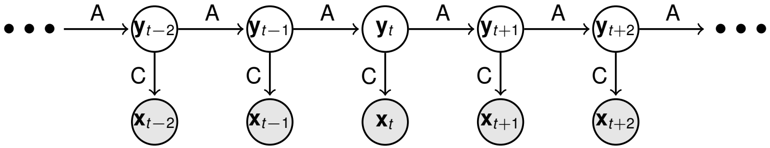
We can generalise this condition to arbitrary T even if $P \notin \text{ExpFam}$.

That is, we find estimate θ^* with

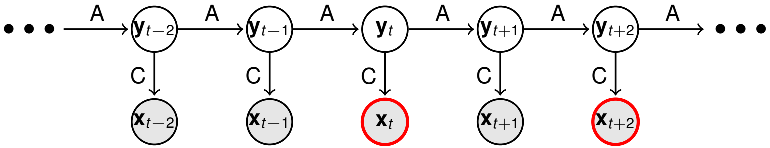
$$\langle T(\mathbf{x}) \rangle_{\theta^*} = \frac{1}{N} \sum_i T(\mathbf{x}_i) \quad \text{or} \quad \theta^* = \underset{\theta}{\text{argmin}} \|\langle T(\mathbf{x}) \rangle_\theta - \frac{1}{N} \sum_i T(\mathbf{x}_i)\|_{\mathcal{C}}$$

Judicious choice of T and metric $\mathcal{C}$ might make solution unique (no local optima) and consistent (correct given infinite within-model data).

Ho-Kalman SSID for LGSSMs

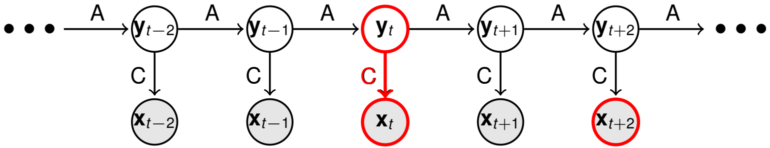


Ho-Kalman SSID for LGSSMs



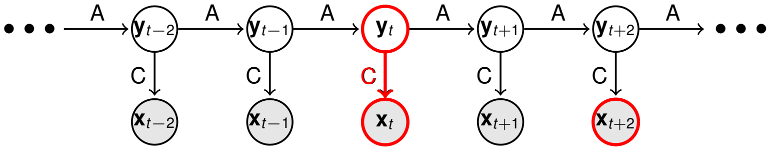
$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau} \mathbf{x}_t^\mathsf{T} \right\rangle$$

Ho-Kalman SSID for LGSSMs



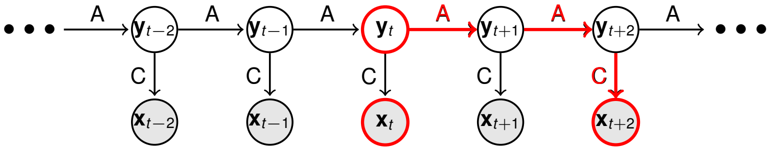
$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle = \left\langle \mathbf{x}_{t+\tau}(C\mathbf{y}_t + \boldsymbol{\eta})^\mathsf{T} \right\rangle$$

Ho-Kalman SSID for LGSSMs



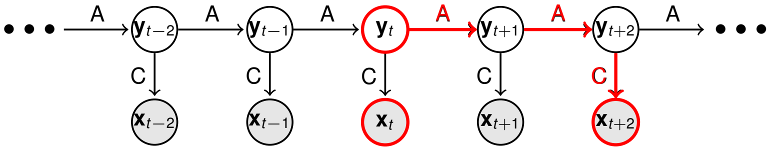
$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle = \left\langle \mathbf{x}_{t+\tau}\mathbf{y}_t^\mathsf{T} \right\rangle C^\mathsf{T}$$

Ho-Kalman SSID for LGSSMs



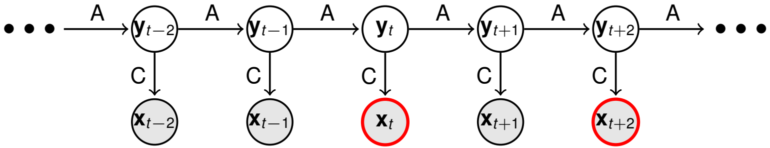
$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau} \mathbf{x}_t^\mathsf{T} \right\rangle = \left\langle \mathbf{x}_{t+\tau} \mathbf{y}_t^\mathsf{T} \right\rangle C^\mathsf{T} = \left\langle (CA^\tau \mathbf{y}_t + \boldsymbol{\eta}) \mathbf{y}_t^\mathsf{T} \right\rangle C^\mathsf{T}$$

Ho-Kalman SSID for LGSSMs



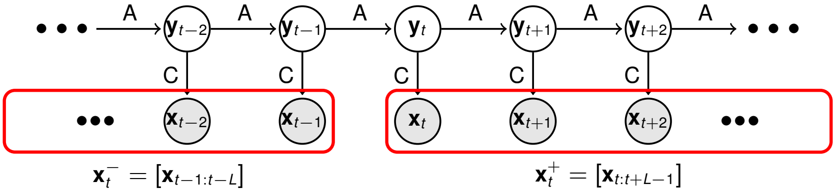
$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle = \left\langle \mathbf{x}_{t+\tau}\mathbf{y}_t^\mathsf{T} \right\rangle C^\mathsf{T} = CA^\tau \left\langle \mathbf{y}_t\mathbf{y}_t^\mathsf{T} \right\rangle C^\mathsf{T}$$

Ho-Kalman SSID for LGSSMs



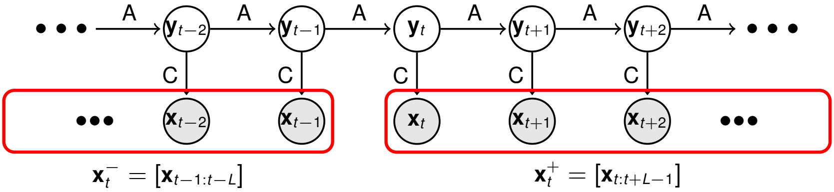
$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle = \left\langle \mathbf{x}_{t+\tau}\mathbf{y}_t^\mathsf{T} \right\rangle C^\mathsf{T} = CA^\tau \left\langle \mathbf{y}_t\mathbf{y}_t^\mathsf{T} \right\rangle C^\mathsf{T} = CA^\tau \Pi C^\mathsf{T}$$

Ho-Kalman SSID for LGSSMs



$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle = \left\langle \mathbf{x}_{t+\tau}\mathbf{y}_t^\mathsf{T} \right\rangle C^\mathsf{T} = CA^\tau \left\langle \mathbf{y}_t\mathbf{y}_t^\mathsf{T} \right\rangle C^\mathsf{T} = CA^\tau \Pi C^\mathsf{T}$$

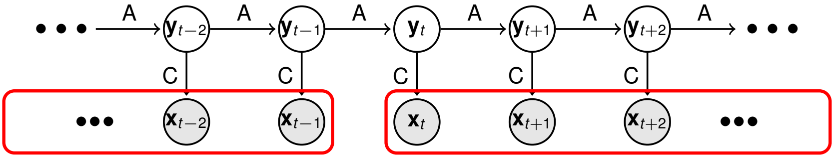
Ho-Kalman SSID for LGSSMs



$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle = \left\langle \mathbf{x}_{t+\tau}\mathbf{y}_t^\mathsf{T} \right\rangle C^\mathsf{T} = CA^\tau \left\langle \mathbf{y}_t\mathbf{y}_t^\mathsf{T} \right\rangle C^\mathsf{T} = CA^\tau \Pi C^\mathsf{T}$$

$$H \equiv \left\langle \mathbf{x}_t^+, \mathbf{x}_t^{-\,\mathsf{T}} \right\rangle = \begin{bmatrix} M_1 & M_2 & \cdots & M_L \\ M_2 & M_3 & & \\ \vdots & & & \vdots \\ M_L & & \cdots & M_{2L-1} \end{bmatrix} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{L-1} \end{bmatrix} \begin{bmatrix} A\Pi C^\mathsf{T} & \cdots & A^L \Pi C^\mathsf{T} \end{bmatrix}$$

Ho-Kalman SSID for LGSSMs

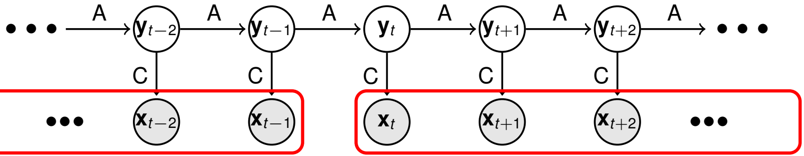


$$\mathbf{x}_t^- = [\mathbf{x}_{t-1:t-L}] \qquad \mathbf{x}_t^+ = [\mathbf{x}_{t:t+L-1}]$$

$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle = \left\langle \mathbf{x}_{t+\tau}\mathbf{y}_t^\mathsf{T} \right\rangle C^\mathsf{T} = CA^\tau \left\langle \mathbf{y}_t\mathbf{y}_t^\mathsf{T} \right\rangle C^\mathsf{T} = CA^\tau \Pi C^\mathsf{T}$$

$$H \equiv \left\langle \mathbf{x}_t^+, \mathbf{x}_t^{-\mathsf{T}} \right\rangle = \underbrace{\begin{bmatrix} M_1 & M_2 & \cdots & M_L \\ M_2 & M_3 & & \\ \vdots & & & \vdots \\ M_L & & \cdots & M_{2L-1} \end{bmatrix}}_{LD \times LD} = \underbrace{\begin{bmatrix} C \\ CA \\ \Xi \\ \vdots \\ CA^{L-1} \end{bmatrix}}_{LD \times K} \underbrace{\begin{bmatrix} A\Pi C^\mathsf{T} & \cdots \Upsilon & A^L \Pi C^\mathsf{T} \end{bmatrix}}_{K \times LD}$$

Ho-Kalman SSID for LGSSMs



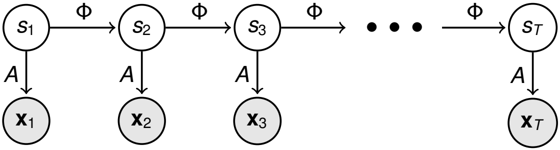
$\mathbf{x}_t^- = [\mathbf{x}_{t-1:t-L}]$ $\qquad$ $\mathbf{x}_t^+ = [\mathbf{x}_{t:t+L-1}]$

$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^{\mathsf{T}} \right\rangle = \left\langle \mathbf{x}_{t+\tau}\mathbf{y}_t^{\mathsf{T}} \right\rangle C^{\mathsf{T}} = CA^\tau \left\langle \mathbf{y}_t\mathbf{y}_t^{\mathsf{T}} \right\rangle C^{\mathsf{T}} = CA^\tau \Pi C^{\mathsf{T}}$$

$$H \equiv \left\langle \mathbf{x}_t^+, \mathbf{x}_t^{-\mathsf{T}} \right\rangle = \underbrace{\begin{bmatrix} M_1 & M_2 & \cdots & M_L \\ M_2 & M_3 & & \\ \vdots & & & \vdots \\ M_L & & \cdots & M_{2L-1} \end{bmatrix}}_{LD \times LD} = \underbrace{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{L-1} \end{bmatrix}}_{LD \times K \;(\Xi)} \underbrace{\begin{bmatrix} A\Pi C^{\mathsf{T}} & \cdots & A^L \Pi C^{\mathsf{T}} \end{bmatrix}}_{K \times LD \;(\Upsilon)}$$

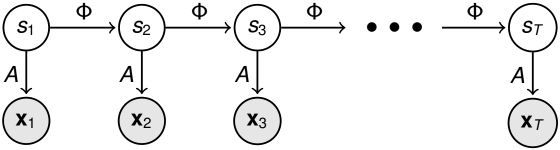
Off-diagonal correlation unaffected by noise. $SVD(\frac{1}{T}\sum \mathbf{x}_t^+ \mathbf{x}_t^{-\mathsf{T}})$ yields least-squares estimates of Ξ and Υ. Regression between blocks of Ξ yields $\widehat{A}$ and $\widehat{C}$.

HMMs $\rightarrow$ OOMs



Now consider an HMM with discrete output symbols.

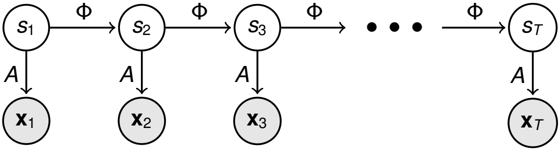
HMMs $\rightarrow$ OOMs



Now consider an HMM with discrete output symbols. The likelihood can be written:

$$P(x_{1:T}|\pi, \Phi, A) = \sum_i \pi_i A_i(x_1) \sum_j \Phi_{ji} A_j(x_2) \sum_k \Phi_{kj} A_K(x_3) \ldots$$

HMMs → OOMs

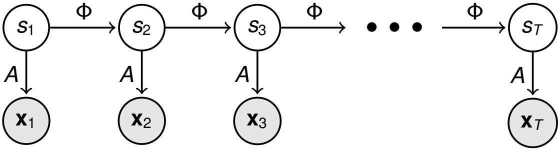


Now consider an HMM with discrete output symbols. The likelihood can be written:

$$P(x_{1:T}|\pi, \Phi, A) = \sum_i \pi_i A_i(x_1) \sum_j \Phi_{ji} A_j(x_2) \sum_k \Phi_{kj} A_K(x_3) \ldots$$

$$= \boldsymbol{\pi}^\top A_{x_1} \Phi^\top A_{x_2} \Phi^\top A_{x_3} \ldots \mathbf{1} \qquad [A_{x_1} = \mathrm{diag}\,[A_1(x_1), A_2(x_2), \ldots]]$$

HMMs $\rightarrow$ OOMs

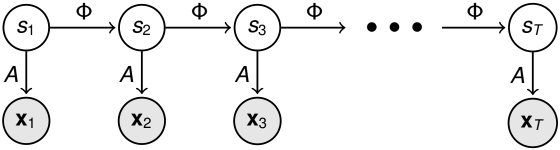


Now consider an HMM with discrete output symbols. The likelihood can be written:

$$
\begin{aligned}
P(x_{1:T}|\pi, \Phi, A) &= \sum_i \pi_i A_i(x_1) \sum_j \Phi_{ji} A_j(x_2) \sum_k \Phi_{kj} A_K(x_3) \dots \\
&= \boldsymbol{\pi}^\mathsf{T} A_{x_1} \Phi^\mathsf{T} A_{x_2} \Phi^\mathsf{T} A_{x_3} \dots \mathbf{1} \qquad [A_{x_1} = \mathrm{diag}\,[A_1(x_1), A_2(x_2), \dots]] \\
&= \mathbf{1}^\mathsf{T} O_{x_T} O_{x_{T-1}} \dots O_{x_1} \boldsymbol{\pi}
\end{aligned}
$$

where $O_a = \Phi A_a$ is a “propagation operator” on the latent belief that depends on observation.

HMMs → OOMs



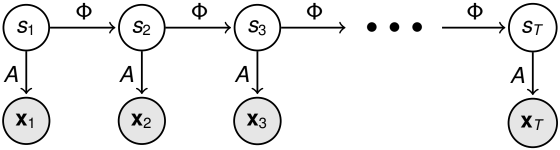
Now consider an HMM with discrete output symbols. The likelihood can be written:

$$
\begin{aligned}
P(x_{1:T}|\pi, \Phi, A) &= \sum_i \pi_i A_i(x_1) \sum_j \Phi_{ji} A_j(x_2) \sum_k \Phi_{kj} A_K(x_3) \ldots \\
&= \boldsymbol{\pi}^{\mathsf{T}} A_{x_1} \Phi^{\mathsf{T}} A_{x_2} \Phi^{\mathsf{T}} A_{x_3} \ldots \mathbf{1} \qquad [A_{x_1} = \mathrm{diag}\,[A_1(x_1), A_2(x_2), \ldots]] \\
&= \mathbf{1}^{\mathsf{T}} O_{x_T} O_{x_{T-1}} \ldots O_{x_1} \boldsymbol{\pi}
\end{aligned}
$$

where $O_a = \Phi A_a$ is a “propagation operator” on the latent belief that depends on observation.

Observable operator model (OOM) representation.

HMMs $\rightarrow$ OOMs



Now consider an HMM with discrete output symbols. The likelihood can be written:

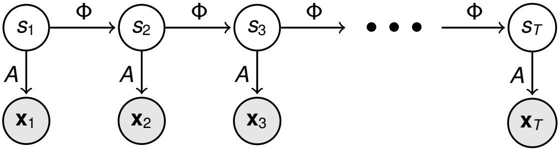
$$
\begin{aligned}
P(x_{1:T}|\pi, \Phi, A) &= \sum_i \pi_i A_i(x_1) \sum_j \Phi_{ji} A_j(x_2) \sum_k \Phi_{kj} A_K(x_3) \ldots \\
&= \boldsymbol{\pi}^{\mathsf{T}} A_{x_1} \Phi^{\mathsf{T}} A_{x_2} \Phi^{\mathsf{T}} A_{x_3} \ldots \mathbf{1} \qquad [A_{x_1} = \mathrm{diag}\,[A_1(x_1), A_2(x_2), \ldots]] \\
&= \mathbf{1}^{\mathsf{T}} O_{x_T} O_{x_{T-1}} \ldots O_{x_1} \boldsymbol{\pi}
\end{aligned}
$$

where $O_a = \Phi A_a$ is a “propagation operator” on the latent belief that depends on observation.

Observable operator model (OOM) representation.

- OOMs with arbitrary O matrices describe a larger class of distributions that HMMs.

HMMs → OOMs



Now consider an HMM with discrete output symbols. The likelihood can be written:

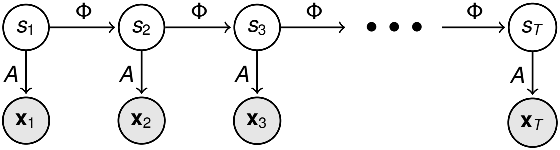
$$
\begin{aligned}
P(x_{1:T}|\pi, \Phi, A) &= \sum_i \pi_i A_i(x_1) \sum_j \Phi_{ji} A_j(x_2) \sum_k \Phi_{kj} A_K(x_3) \ldots \\
&= \boldsymbol{\pi}^{\mathsf{T}} A_{x_1} \Phi^{\mathsf{T}} A_{x_2} \Phi^{\mathsf{T}} A_{x_3} \ldots \mathbf{1} \qquad [A_{x_1} = \mathrm{diag}\,[A_1(x_1), A_2(x_2), \ldots]] \\
&= \mathbf{1}^{\mathsf{T}} O_{x_T} O_{x_{T-1}} \ldots O_{x_1} \boldsymbol{\pi}
\end{aligned}
$$

where $O_a = \Phi A_a$ is a “propagation operator” on the latent belief that depends on observation.

Observable operator model (OOM) representation.

- OOMs with arbitrary O matrices describe a larger class of distributions that HMMs.
- Not easy to normalise or even guarantee all assigned “probabilities” are positive.

HMMs $\rightarrow$ OOMs



Now consider an HMM with discrete output symbols. The likelihood can be written:

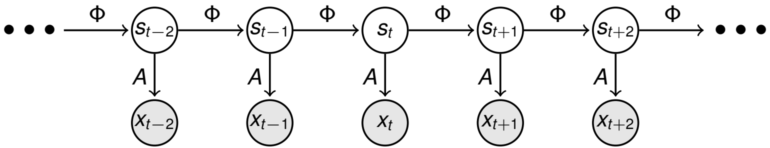
$$\begin{aligned} P(x_{1:T}|\pi, \Phi, A) &= \sum_i \pi_i A_i(x_1) \sum_j \Phi_{ji} A_j(x_2) \sum_k \Phi_{kj} A_K(x_3) \ldots \\ &= \boldsymbol{\pi}^{\mathsf{T}} A_{x_1} \Phi^{\mathsf{T}} A_{x_2} \Phi^{\mathsf{T}} A_{x_3} \ldots \mathbf{1} \qquad [A_{x_1} = \mathrm{diag}\,[A_1(x_1), A_2(x_2), \ldots]] \\ &= \mathbf{1}^{\mathsf{T}} O_{x_T} O_{x_{T-1}} \ldots O_{x_1} \boldsymbol{\pi} \end{aligned}$$

where $O_a = \Phi A_a$ is a "propagation operator" on the latent belief that depends on observation.

Observable operator model (OOM) representation.

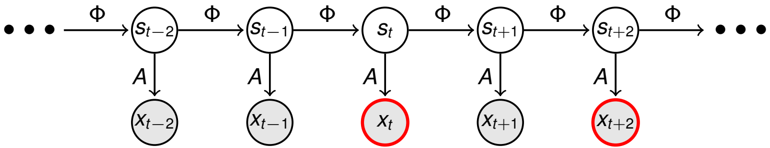
- OOMs with arbitrary O matrices describe a larger class of distributions that HMMs.
- Not easy to normalise or even guarantee all assigned "probabilities" are positive.
- Degenerate with respect to similarity transform $\widetilde{O} = GOG^{-1}$.

Spectral learning for HMMs



Write $\mathbf{x}_t = [\delta(x_t{=}i)]$; $\mathbf{s}_t = [\delta(s_t{=}i)]$; $A = [A_{ij}] = [P(x_t{=}i|s_t{=}j)]$

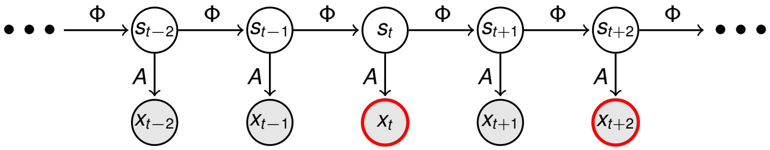
Spectral learning for HMMs



Write $\mathbf{x}_t = [\delta(x_t{=}i)]$; $\mathbf{s}_t = [\delta(s_t{=}i)]$; $A = [A_{ij}] = [P(x_t{=}i|s_t{=}j)]$

$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle$$

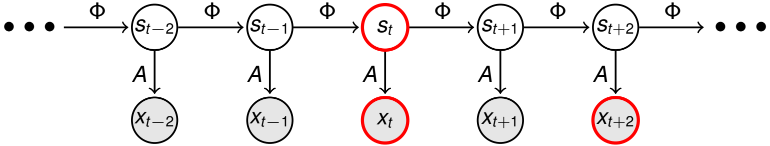
Spectral learning for HMMs



Write $\mathbf{x}_t = [\delta(x_t{=}i)]$; $\mathbf{s}_t = [\delta(s_t{=}i)]$; $A = [A_{ij}] = [P(x_t{=}i|s_t{=}j)]$

$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^{\mathsf{T}} \right\rangle = [P(x_{t+\tau}{=}i, x_t{=}j)]$$

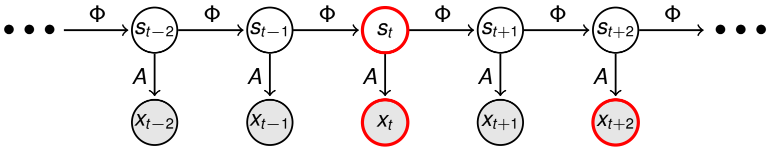
Spectral learning for HMMs



Write $\mathbf{x}_t = [\delta(x_t{=}i)]$; $\mathbf{s}_t = [\delta(s_t{=}i)]$; $A = [A_{ij}] = [P(x_t{=}i|s_t{=}j)]$

$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^{\mathsf{T}} \right\rangle = [\langle P(x_{t+\tau}{=}i, x_t{=}j|s_t)\rangle_{s_t}]$$

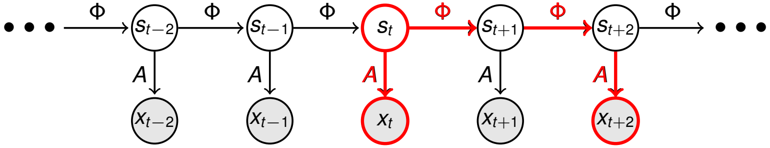
Spectral learning for HMMs



Write $\mathbf{x}_t = [\delta(x_t{=}i)]$; $\mathbf{s}_t = [\delta(s_t{=}i)]$; $A = [A_{ij}] = [P(x_t{=}i|s_t{=}j)]$

$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^{\mathsf{T}} \right\rangle = [\langle P(x_{t+\tau}{=}i|s_t)P(x_t{=}j|s_t)\rangle_{s_t}]$$

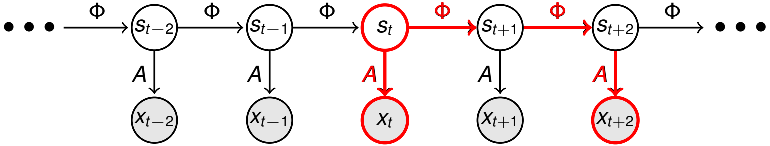
Spectral learning for HMMs



Write $\mathbf{x}_t = [\delta(x_t{=}i)]$; $\mathbf{s}_t = [\delta(s_t{=}i)]$; $A = [A_{ij}] = [P(x_t{=}i|s_t{=}j)]$

$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle = [\langle P(x_{t+\tau}{=}i|s_t)P(x_t{=}j|s_t)\rangle_{s_t}] = \left\langle A\Phi^\tau \mathbf{s}_t (A\mathbf{s}_t)^\mathsf{T} \right\rangle_{s_t}$$

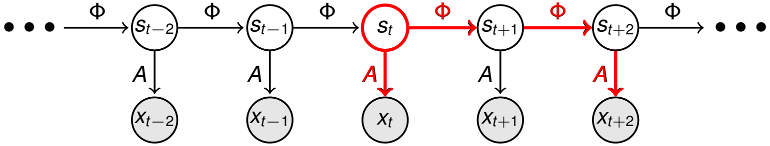
Spectral learning for HMMs



Write $\mathbf{x}_t = [\delta(x_t{=}i)]$; $\mathbf{s}_t = [\delta(s_t{=}i)]$; $A = [A_{ij}] = [P(x_t{=}i|s_t{=}j)]$

$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle = [\langle P(x_{t+\tau}{=}i|s_t)P(x_t{=}j|s_t)\rangle_{s_t}] = A\Phi^\tau \left\langle \mathbf{s}_t\mathbf{s}_t^\mathsf{T} \right\rangle A^\mathsf{T}$$

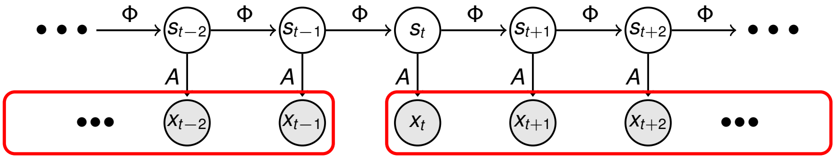
Spectral learning for HMMs



Write $\mathbf{x}_t = [\delta(x_t{=}i)]$; $\mathbf{s}_t = [\delta(s_t{=}i)]$; $A = [A_{ij}] = [P(x_t{=}i|s_t{=}j)]$

$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle = [\langle P(x_{t+\tau}{=}i|s_t)P(x_t{=}j|s_t)\rangle_{s_t}] = A\Phi^\tau \left\langle \mathbf{s}_t\mathbf{s}_t^\mathsf{T} \right\rangle A^\mathsf{T} = A\Phi^\tau \Pi A^\mathsf{T}$$

Spectral learning for HMMs



$\mathbf{x}_t^- = [\mathbf{x}_{t-1:t-L}]$ $\quad\quad$ $\mathbf{x}_t^+ = [\mathbf{x}_{t:t+L-1}]$

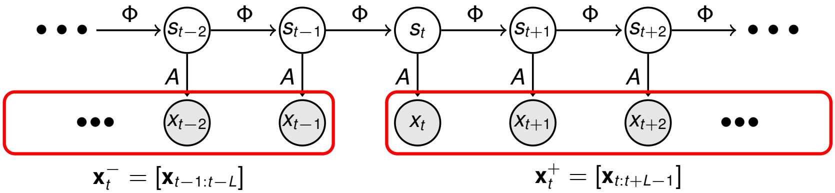
Write $\mathbf{x}_t = [\delta(x_t{=}i)]$; $\mathbf{s}_t = [\delta(s_t{=}i)]$; $A = [A_{ij}] = [P(x_t{=}i|s_t{=}j)]$

$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle = [\langle P(x_{t+\tau}{=}i|s_t)P(x_t{=}j|s_t)\rangle_{s_t}] = A\Phi^\tau \left\langle \mathbf{s}_t\mathbf{s}_t^\mathsf{T} \right\rangle A^\mathsf{T} = A\Phi^\tau \Pi A^\mathsf{T}$$

$$H \equiv \left\langle \mathbf{x}_t^+, {\mathbf{x}_t^-}^\mathsf{T} \right\rangle = \begin{bmatrix} M_1 & M_2 & \cdots & M_L \\ M_2 & M_3 & & \\ \vdots & & & \vdots \\ M_L & & \cdots & M_{2L-1} \end{bmatrix} = \begin{bmatrix} A \\ A\Phi \\ \vdots \\ A\Phi^{L-1} \end{bmatrix} \begin{bmatrix} \Phi\Pi A^\mathsf{T} & \cdots & \Phi^L \Pi A^\mathsf{T} \end{bmatrix}$$

(The conventional algorithm is written slightly differently, but follows similar logic).

Spectral learning for HMMs



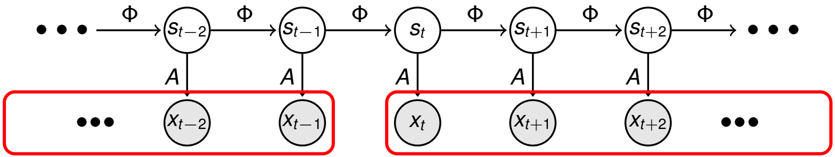
Write $\mathbf{x}_t = [\delta(x_t{=}i)]$; $\mathbf{s}_t = [\delta(s_t{=}i)]$; $A = [A_{ij}] = [P(x_t{=}i|s_t{=}j)]$

$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle = \left[\left\langle P(x_{t+\tau}{=}i|s_t)P(x_t{=}j|s_t)\right\rangle_{s_t}\right] = A\Phi^\tau \left\langle \mathbf{s}_t\mathbf{s}_t^\mathsf{T} \right\rangle A^\mathsf{T} = A\Phi^\tau \Pi A^\mathsf{T}$$

$$H \equiv \left\langle \mathbf{x}_t^+, \mathbf{x}_t^{-\mathsf{T}} \right\rangle = \underbrace{\begin{bmatrix} M_1 & M_2 & \cdots & M_L \\ M_2 & M_3 & & \\ \vdots & & & \vdots \\ M_L & & \cdots & M_{2L-1} \end{bmatrix}}_{LD \times LD} = \underbrace{\begin{bmatrix} A \\ A\Phi \\ \vdots \\ A\Phi^{L-1} \end{bmatrix}}_{LD \times K} \underbrace{\begin{bmatrix} \Phi\Pi A^\mathsf{T} & \cdots & \Phi^L \Pi A^\mathsf{T} \end{bmatrix}}_{K \times LD}$$

(The conventional algorithm is written slightly differently, but follows similar logic).
$\widehat{\Phi}$ and $\widehat{A}$ can be recovered as before . . .

Spectral learning for HMMs



$\mathbf{x}_t^- = [\mathbf{x}_{t-1:t-L}]$ $\qquad$ $\mathbf{x}_t^+ = [\mathbf{x}_{t:t+L-1}]$

Write $\mathbf{x}_t = [\delta(x_t{=}i)]$; $\mathbf{s}_t = [\delta(s_t{=}i)]$; $A = [A_{ij}] = [P(x_t{=}i|s_t{=}j)]$

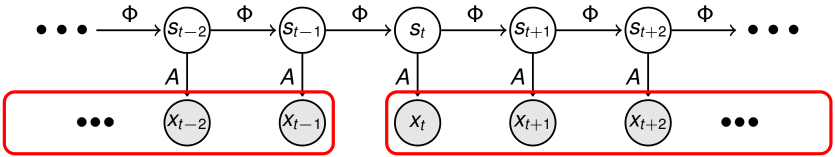
$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle = [\langle P(x_{t+\tau}{=}i|s_t)P(x_t{=}j|s_t)\rangle_{s_t}] = A\Phi^\tau \left\langle \mathbf{s}_t\mathbf{s}_t^\mathsf{T} \right\rangle A^\mathsf{T} = A\Phi^\tau \Pi A^\mathsf{T}$$

$$H \equiv \left\langle \mathbf{x}_t^+, \mathbf{x}_t^{-\mathsf{T}} \right\rangle = \underbrace{\begin{bmatrix} M_1 & M_2 & \cdots & M_L \\ M_2 & M_3 & & \\ \vdots & & & \vdots \\ M_L & & \cdots & M_{2L-1} \end{bmatrix}}_{LD \times LD} = \underbrace{\begin{bmatrix} A \\ A\Phi \\ \vdots \\ A\Phi^{L-1} \end{bmatrix}}_{LD \times K} \underbrace{\begin{bmatrix} \Phi\Pi A^\mathsf{T} & \cdots & \Phi^L \Pi A^\mathsf{T} \end{bmatrix}}_{K \times LD}$$

(The conventional algorithm is written slightly differently, but follows similar logic).

$\widehat{\Phi}$ and $\widehat{A}$ can be recovered as before upto arbitrary invertible transform of $\mathbf{s}$ $\rightarrow$ OOM.

Spectral learning for HMMs



$$\mathbf{x}_t^- = [\mathbf{x}_{t-1:t-L}] \qquad \mathbf{x}_t^+ = [\mathbf{x}_{t:t+L-1}]$$

Write $\mathbf{x}_t = [\delta(x_t{=}i)]$; $\mathbf{s}_t = [\delta(s_t{=}i)]$; $A = [A_{ij}] = [P(x_t{=}i|s_t{=}j)]$

$$M_\tau \equiv \left\langle \mathbf{x}_{t+\tau}\mathbf{x}_t^\mathsf{T} \right\rangle = [\langle P(x_{t+\tau}{=}i|s_t)P(x_t{=}j|s_t)\rangle_{s_t}] = A\Phi^\tau \left\langle \mathbf{s}_t\mathbf{s}_t^\mathsf{T} \right\rangle A^\mathsf{T} = A\Phi^\tau \Pi A^\mathsf{T}$$

$$H \equiv \left\langle \mathbf{x}_t^+, \mathbf{x}_t^{-\mathsf{T}} \right\rangle = \underbrace{\begin{bmatrix} M_1 & M_2 & \cdots & M_L \\ M_2 & M_3 & & \\ \vdots & & & \vdots \\ M_L & & \cdots & M_{2L-1} \end{bmatrix}}_{LD \times LD} = \underbrace{\begin{bmatrix} A \\ A\Phi \\ \vdots \\ A\Phi^{L-1} \end{bmatrix}}_{LD \times K} \underbrace{\begin{bmatrix} \Phi\Pi A^\mathsf{T} & \cdots & \Phi^L \Pi A^\mathsf{T} \end{bmatrix}}_{K \times LD}$$

(The conventional algorithm is written slightly differently, but follows similar logic).
$\widehat{\Phi}$ and $\widehat{A}$ can be recovered as before upto arbitrary invertible transform of $\mathbf{s}$ → OOM.
'Projection' to HHM space possible, but amplifies estimation errors.

ML and spectral learning

Spectral learning:

Maximum likelihood learning:

ML and spectral learning

Spectral learning:

- Efficient closed-form solution.

Maximum likelihood learning:

- Requires iterative maximisation.

ML and spectral learning

Spectral learning:

- Efficient closed-form solution finds global optimum.

Maximum likelihood learning:

- Requires iterative maximisation.
- Many local optima.

ML and spectral learning

Spectral learning:

- Efficient closed-form solution finds global optimum.
- Consistent (recovers true parameters upto degeneracies from infinite within-model data).

Maximum likelihood learning:

- Requires iterative maximisation.
- Many local optima.
- Consistent and asymptotically efficient (if the global maximum can be found).

ML and spectral learning

Spectral learning:

- Efficient closed-form solution finds global optimum.
- Consistent (recovers true parameters upto degeneracies from infinite within-model data).
- Eigen-/singular-value spectrum clue to latent dimensionality.

Maximum likelihood learning:

- Requires iterative maximisation.
- Many local optima.
- Consistent and asymptotically efficient (if the global maximum can be found).

ML and spectral learning

Spectral learning:

- Efficient closed-form solution finds global optimum.
- Consistent (recovers true parameters upto degeneracies from infinite within-model data).
- Eigen-/singular-value spectrum clue to latent dimensionality.
- Assumes stationarity. May be inappropriate for short sequences.

Maximum likelihood learning:

- Requires iterative maximisation.
- Many local optima.
- Consistent and asymptotically efficient (if the global maximum can be found).

ML and spectral learning

Spectral learning:

- Efficient closed-form solution finds global optimum.
- Consistent (recovers true parameters upto degeneracies from infinite within-model data).
- Eigen-/singular-value spectrum clue to latent dimensionality.
- Assumes stationarity. May be inappropriate for short sequences.
- HMM learning returns OOM parameters – may not correspond to any HMM or indeed to proper probabilistic model.

Maximum likelihood learning:

- Requires iterative maximisation.
- Many local optima.
- Consistent and asymptotically efficient (if the global maximum can be found).

ML and spectral learning

Spectral learning:

- Efficient closed-form solution finds global optimum.
- Consistent (recovers true parameters upto degeneracies from infinite within-model data).
- Eigen-/singular-value spectrum clue to latent dimensionality.
- Assumes stationarity. May be inappropriate for short sequences.
- HMM learning returns OOM parameters – may not correspond to any HMM or indeed to proper probabilistic model.
- Not easily generalised to nonlinear or more complex models (but see `http://www.gatsby.ucl.ac.uk/~maneesh/papers/buesing-etal-2012-nips.pdf`).

Maximum likelihood learning:

- Requires iterative maximisation.
- Many local optima.
- Consistent and asymptotically efficient (if the global maximum can be found).

ML and spectral learning

Spectral learning:

- Efficient closed-form solution finds global optimum.
- Consistent (recovers true parameters upto degeneracies from infinite within-model data).
- Eigen-/singular-value spectrum clue to latent dimensionality.
- Assumes stationarity. May be inappropriate for short sequences.
- HMM learning returns OOM parameters – may not correspond to any HMM or indeed to proper probabilistic model.
- Not easily generalised to nonlinear or more complex models (but see `http://www.gatsby.ucl.ac.uk/~maneesh/papers/buesing-etal-2012-nips.pdf`).
- In practice, error in recovered parameters is often large.

Maximum likelihood learning:

- Requires iterative maximisation.
- Many local optima.
- Consistent and asymptotically efficient (if the global maximum can be found).
- Generalises to “principled” approximate algorithms for nonlinear or complex models.

ML and spectral learning

Spectral learning:

- Efficient closed-form solution finds global optimum.
- Consistent (recovers true parameters upto degeneracies from infinite within-model data).
- Eigen-/singular-value spectrum clue to latent dimensionality.
- Assumes stationarity. May be inappropriate for short sequences.
- HMM learning returns OOM parameters – may not correspond to any HMM or indeed to proper probabilistic model.
- Not easily generalised to nonlinear or more complex models (but see `http://www.gatsby.ucl.ac.uk/~maneesh/papers/buesing-etal-2012-nips.pdf`).
- In practice, error in recovered parameters is often large.
- Often valuable as initialisation for ML methods.

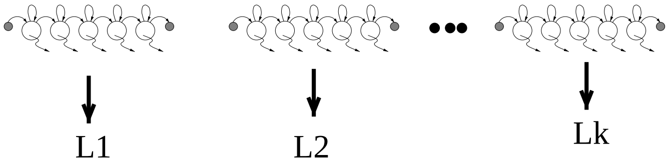
Maximum likelihood learning:

- Requires iterative maximisation.
- Many local optima.
- Consistent and asymptotically efficient (if the global maximum can be found).
- Generalises to "principled" approximate algorithms for nonlinear or complex models.

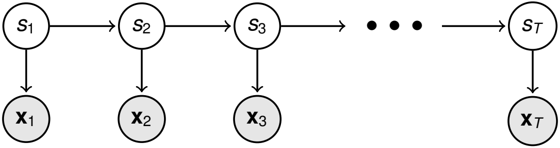
Recognition (classification) with HMMs

Multiple HMM models:

1. train one HMM for each class (requires each sequence to be labelled by the class)
2. evaluate the probability of an unknown sequence under each HMM
3. classify the unknown sequence by the HMM which gave it the highest likelihood



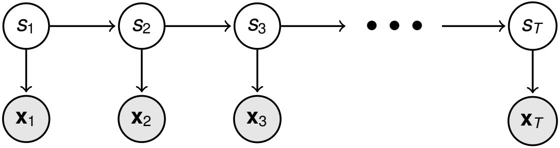
Recognition (labelling) with HMMs



Use a single HMM to label sequences:

1. train a single HMM on sequences of data $\mathbf{x}_1, \ldots, \mathbf{x}_T$ and corresponding labels $s_1, \ldots, s_T$.
2. On an unlabelled test sequence, compute the posterior distribution over label sequences $P(s_1, \ldots, s_T | \mathbf{x}_1, \ldots, \mathbf{x}_T)$.
3. Return the label sequence either with highest expected number of correct states, or highest probability under the posterior (Viterbi).

Recognition (labelling) with HMMs



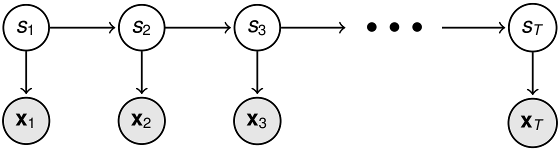
Use a single HMM to label sequences:

1. train a single HMM on sequences of data $\mathbf{x}_1, \ldots, \mathbf{x}_T$ and corresponding labels $s_1, \ldots, s_T$.
2. On an unlabelled test sequence, compute the posterior distribution over label sequences $P(s_1, \ldots, s_T | \mathbf{x}_1, \ldots, \mathbf{x}_T)$.
3. Return the label sequence either with highest expected number of correct states, or highest probability under the posterior (Viterbi).

Models the whole joint distribution $P(\mathbf{x}_{1:T}, s_{1:T})$, but only uses $P(s_{1:T}|\mathbf{x}_{1:T})$.

- May be more accurate and more efficient use of data to model $P(s_{1:T}|x_{1:T})$ directly.

Recognition (labelling) with HMMs



Use a single HMM to label sequences:

1. train a single HMM on sequences of data $\mathbf{x}_1, \ldots, \mathbf{x}_T$ and corresponding labels $s_1, \ldots, s_T$.
2. On an unlabelled test sequence, compute the posterior distribution over label sequences $P(s_1, \ldots, s_T | \mathbf{x}_1, \ldots, \mathbf{x}_T)$.
3. Return the label sequence either with highest expected number of correct states, or highest probability under the posterior (Viterbi).

Models the whole joint distribution $P(\mathbf{x}_{1:T}, s_{1:T})$, but only uses $P(s_{1:T}|\mathbf{x}_{1:T})$.

- May be more accurate and more efficient use of data to model $P(s_{1:T}|x_{1:T})$ directly.
- This leads to a model called a **Conditional Random Field**.

Conditional distribution in a HMM

Conditional distribution over label sequences of a HMM:

$$
\begin{aligned}
P(s_{1:T}|\mathbf{x}_{1:T},\theta) &= \frac{P(s_{1:T},\mathbf{x}_{1:T}|\theta)}{\sum_{s_{1:T}} P(s_{1:T},\mathbf{x}_{1:T}|\theta)} \\
&\propto P(s_1|\pi)\prod_{t=1}^{T-1} P(s_{t+1}|s_t,\mathrm{T})\prod_{t=1}^{T} P(\mathbf{x}_t|s_t,\mathrm{A}) \\
&= \exp\Bigg(\sum_i \delta(s_1{=}i)\log\pi_i + \sum_{t=1}^{T-1}\sum_{ij}\delta(s_t{=}i,s_{t+1}{=}j)\log\Phi_{ij} \\
&\qquad + \sum_{t=1}^{T}\sum_{ik}\delta(s_t{=}i,\mathbf{x}_t{=}k)\log A_{ik}\Bigg).
\end{aligned}
$$

This functional form gives a well-defined conditional distribution, even if we do not enforce the constraints

$$
\Phi_{ij} \geq 0 \qquad \sum_j \Phi_{ij} = 1
$$

or the similar ones for $\boldsymbol{\pi}$ and A (cf. OOMs). The forward-backward algorithm can still be applied to compute the conditional distribution.

This is an example of a conditional random field.

Conditional random fields

Define two sets of functions: single label and label-pair functions.
Single label functions:

$$f_i(s_t, \mathbf{x}_t) \qquad \text{for } i = 1, \ldots, I$$

Label-pair functions:

$$g_j(s_t, s_{t+1}, \mathbf{x}_t, \mathbf{x}_{t+1}) \qquad \text{for } j = 1, \ldots, J$$

Each function is associated with a real-valued parameter: λ_i, κ_j.
A conditional random field defines a conditional distribution over $s_{1:T}$ given $\mathbf{x}_{1:T}$ as follows:

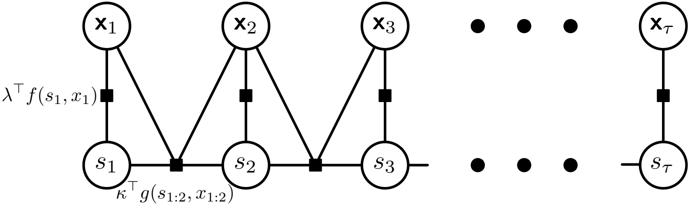
$$P(s_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\lambda}, \boldsymbol{\kappa}) \propto \exp\left(\sum_{t=1}^{T}\sum_{i} \lambda_i f_i(s_t, \mathbf{x}_t) + \sum_{t=1}^{T-1}\sum_{j} \kappa_j g_j(s_t, s_{t+1}, \mathbf{x}_t, \mathbf{x}_{t+1})\right)$$

The forward-backward algorithm can be used to compute:

$$P(s_t|\mathbf{x}_{1:T}, \boldsymbol{\lambda}, \boldsymbol{\kappa}) \qquad P(s_t, s_{t+1}|\mathbf{x}_{1:T}, \boldsymbol{\lambda}, \boldsymbol{\kappa}) \qquad \underset{s_{1:T}}{\operatorname{argmax}}\, P(s_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\lambda}, \boldsymbol{\kappa})$$

Factor graph notation for CRFs

$$P(s_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\lambda}, \boldsymbol{\kappa}) \propto \exp\left(\sum_{t=1}^{T}\sum_{i}\lambda_i f_i(s_t, \mathbf{x}_t) + \sum_{t=1}^{T-1}\sum_{j}\kappa_j g_j(s_t, s_{t+1}, \mathbf{x}_t, \mathbf{x}_{t+1})\right)$$



Discriminative vs generative modelling

Labelled training data comes from a true underlying distribution $\tilde{P}(s_{1:T}, \mathbf{x}_{1:T})$.

Generative modelling: train a HMM by maximizing likelihood:

$$\theta_{\text{Joint}} = \underset{\theta}{\text{argmax}}\, E_{\tilde{P}}[\log P(s_{1:T}, \mathbf{x}_{1:T} | \theta)]$$

(note do not need EM here, since no latent variables)

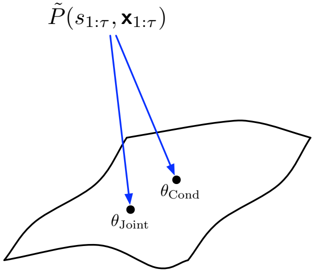
Discriminative modelling: train another HMM by maximizing conditional likelihood:

$$\theta_{\text{Cond}} = \underset{\theta}{\text{argmax}}\, E_{\tilde{P}}[\log P(s_{1:T} | \mathbf{x}_{1:T}, \theta)]$$

By construction:

$$E_{\tilde{P}}[\log P(s_{1:T} | \mathbf{x}_{1:T}, \theta_{\text{Cond}})] \geq E_{\tilde{P}}[\log P(s_{1:T} | \mathbf{x}_{1:T}, \theta_{\text{Joint}})]$$

If $\tilde{P}$ belongs to model class, $P(\cdot | \theta_{\text{Joint}}) = \tilde{P}$ and equality holds.



Caveats:

- Underlying distribution $\tilde{P}$ not usually in model class.
- training set differs from $\tilde{P}$.
- Overfitting easier in discriminative setting.
- Generative modelling often much simpler (fits each conditional probability separately, not iterative).

Major point of debate in machine learning.

Structured generalized linear models

$$P(s_{1:T}|\mathbf{x}_{1:T}, \boldsymbol{\lambda}, \boldsymbol{\kappa}) \propto \exp\left(\sum_{t=1}^{T}\sum_{i}\lambda_i f_i(s_t, \mathbf{x}_t) + \sum_{t=1}^{T-1}\sum_{j}\kappa_j g_j(s_t, s_{t+1}, \mathbf{x}_t, \mathbf{x}_{t+1})\right)$$

The conditional distribution over $s_{1:T}$ forms an exponential family parameterized by $\boldsymbol{\lambda}, \boldsymbol{\kappa}$ and dependent on $\mathbf{x}_{1:T}$.

CRFs are a multivariate generalization of generalized linear models (GLMs).

The labels s_t in a CRF are not independently predicted, but they have a Markov property: $s_{1:t-1}$ is independent of $s_{t+1:T}$ given s_t and $\mathbf{x}_{1:T}$.
This allows efficient inference using the forward-backward algorithm.

CRFs are models for structured prediction (another major machine learning frontier).

CRFs are very flexible.

CRFs have found wide spread applications across a number of fields: natural language processing (part-of-speech tagging, named-entity recognition, coreference resolution), information retrieval (information extraction), computer vision (image segmentation, object recognition, depth perception), bioinformatics (protein structure prediction, gene finding)...

Learning CRFs

$$P(s_{1:T}|\mathbf{x}_{1:T},\boldsymbol{\lambda},\boldsymbol{\kappa}) \propto \exp\left(\sum_{t=1}^{T}\sum_{i}\lambda_i f_i(s_t,\mathbf{x}_t) + \sum_{t=1}^{T-1}\sum_{j}\kappa_j g_j(s_t,s_{t+1},\mathbf{x}_t,\mathbf{x}_{t+1})\right)$$

Given labelled data $\{s_{1:T}^{(c)},\mathbf{x}_{1:T}^{(c)}\}_{c=1}^{N}$, we train CRFs by maximum likelihood:

$$\frac{\partial\sum_c \log P(s_{1:T}^{(c)}|\mathbf{x}_{1:T}^{(c)},\boldsymbol{\lambda},\boldsymbol{\kappa})}{\partial\lambda_i} = \sum_{c=1}^{N}\sum_{t=1}^{T} f_i(s_t^{(c)},\mathbf{x}_t^{(c)}) - E_{P(s_{1:T}|\mathbf{x}_{1:T}^{(c)})}[f_i(s_t^{(c)},\mathbf{x}_t^{(c)})]$$

$$\frac{\partial\sum_c \log P(s_{1:T}^{(c)}|\mathbf{x}_{1:T}^{(c)},\boldsymbol{\lambda},\boldsymbol{\kappa})}{\partial\kappa_j} = \sum_{c=1}^{N}\sum_{t=1}^{T-1} g_j(s_{t:t+1}^{(c)},\mathbf{x}_{t:t+1}^{(c)}) - E_{P(s_{1:T}|\mathbf{x}_{1:T}^{(c)})}[g_j(s_{t:t+1},\mathbf{x}_{t:t+1}^{(c)})]$$

There is no closed-form solution for the parameters, so we use gradient ascent instead.

Note: expectations are computed using the forward-backward algorithm.

The log likelihood is concave, so unlike EM we will get to global optimum (another major frontier in machine learning).