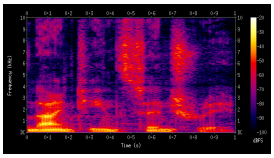
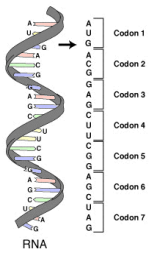
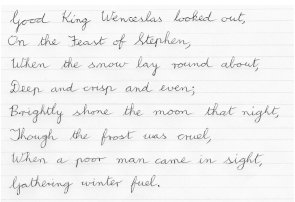
Sequence Modelling

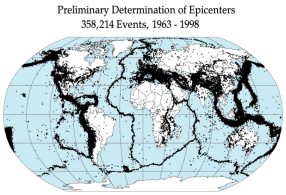
Rich Turner and José Miguel Hernández-Lobato

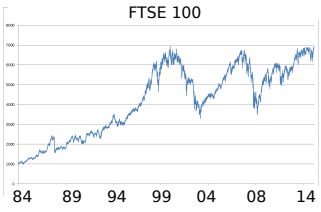
Sequence data

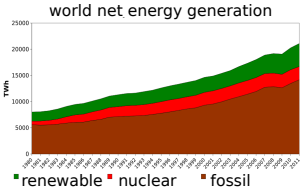












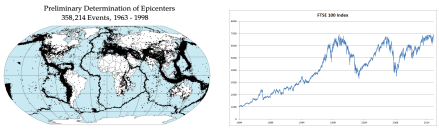
Some images taken from wikipedia

I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted. A. Turing

Goals of sequence modelling

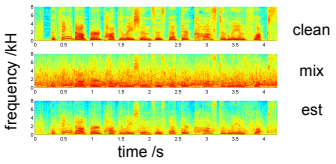
Predict future items in sequence

$$p(y_t|y_1,\ldots,y_{t-1})$$



Remove noise from a sequence

$$p(y'_1,\ldots,y'_t|y_1,\ldots,y_t)$$

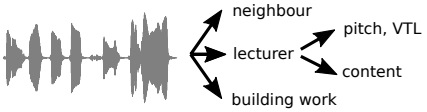


Predict one sequence from another

$$p(y'_1,\ldots,y'_t|y_1,\ldots,y_t)$$

Discover underlying latent variables

$$p(x_1,\ldots,x_t|y_1,\ldots,y_t)$$

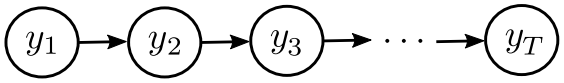


Markov models

First order Markov

parameters tied; ∞ number of variables; finite number of parameters

$$p(y_1,y_2,y_3,\ldots,y_T) = p(y_1)p(y_2|y_1)p(y_3|y_2)\ldots p(y_T|y_{T-1})$$



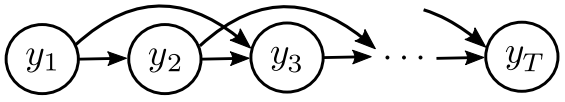
Markov model = conditional independence relationship + product rule

future; independent of past; given present

$$y_{t+1} \perp y_{1:t-1} | y_t \qquad p(y_{1:T}) = \prod_{t=1}^{T} p(y_t|y_{1:t-1})$$

Second order Markov

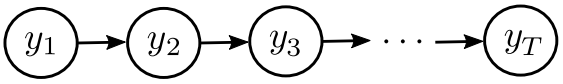
$$p(y_1,y_2,y_3,\ldots,y_T) = p(y_1)p(y_2|y_1)p(y_3|y_2,y_1)\ldots p(y_T|y_{T-1},y_{T-2})$$



Markov models for discrete data: n-gram models

First order Markov (bi-gram)

$$p(y_1, y_2, y_3, \ldots, y_T) = p(y_1)p(y_2|y_1)p(y_3|y_2)\ldots p(y_T|y_{T-1})$$



$y_t \in \{1, ..., K\}$ discrete states

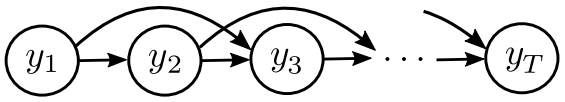
$p(y_1 = k) = \pi_k^0$ initial state probabilities

$p(y_t = k|y_{t-1} = l) = T_{k,l}$ transition probabilities (stochastic matrix)

$\sum_{k=1}^{K} T_{k,l} = 1$

Second order Markov (tri-gram)

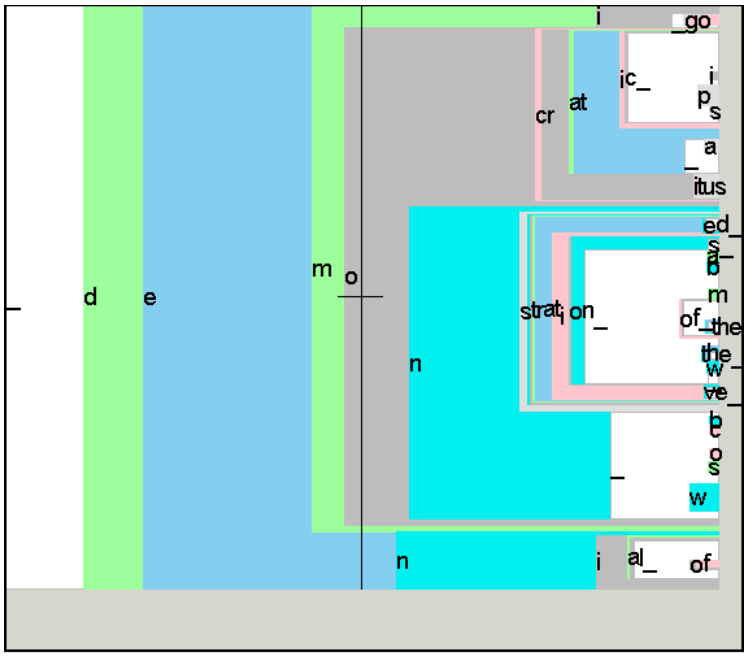
$$p(y_1, y_2, y_3, \ldots, y_T) = p(y_1)p(y_2|y_1)p(y_3|y_2, y_1)\ldots p(y_T|y_{T-1}, y_{T-2})$$



$p(y_t = k|y_{t-1} = l, y_{t-2} = m) = T_{k,l,m}$

n-grams require large multidimensional arrays

Example application of n-grams: text modelling for dasher



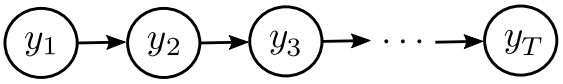
this_is_a_demo

http://www.inference.phy.cam.ac.uk/dasher/ https://www.youtube.com/watch?v=nr3s4613DX8

Markov models for continuous data: Auto-Regressive (AR) Gaussian models

First order Markov (AR(1))

$$p(y_1, y_2, y_3, \ldots, y_T) = p(y_1)p(y_2|y_1)p(y_3|y_2)\ldots p(y_T|y_{T-1})$$



$y_t \in \mathbb{R}^D$ continuous vector states

$p(y_1) = \mathcal{G}(y_1; \mu_0, \Sigma_0)$ initial state density

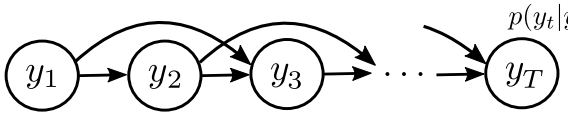
$p(y_t|y_{t-1}) = \mathcal{G}(y_t; \Lambda y_{t-1}, \Sigma)$ transition density

$$\mathcal{G}(y; \mu, \Sigma) = \frac{1}{(2\pi)^{D/2}\det(\Sigma)^{1/2}} \exp\left\{-\frac{1}{2}(y-\mu)^\top \Sigma^{-1}(y-\mu)\right\}$$

Second order Markov (AR(2))

$$p(y_1, y_2, y_3, \ldots, y_T) = p(y_1)p(y_2|y_1)p(y_3|y_2, y_1)\ldots p(y_T|y_{T-1}, y_{T-2})$$

$$p(y_t|y_{t-1}, y_{t-2}) = \mathcal{G}(y_t; \Lambda_1 y_{t-1} + \Lambda_2 y_{t-2}, \Sigma)$$

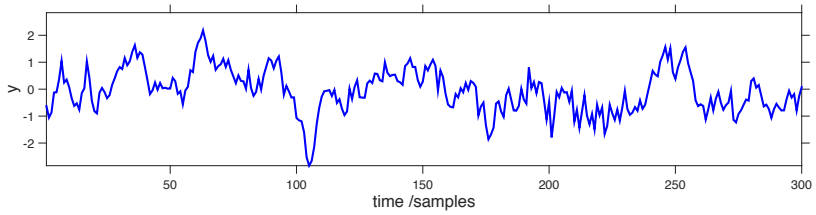


joint distribution over all variables is always multivariate Gaussian

Markov models for continuous data: Auto-Regressive (AR) Gaussian models

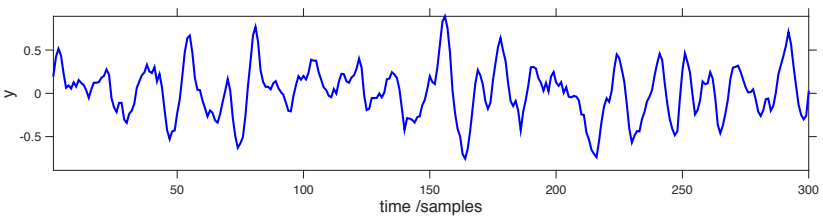
First order Markov (AR(1))

$$y_t \in \mathbb{R}^1 \quad p(y_t|y_{t-1}) = \mathcal{G}(y_t; \lambda y_{t-1}, \sigma^2) \quad \lambda = 0.9 \quad \sigma^2 = 0.01$$

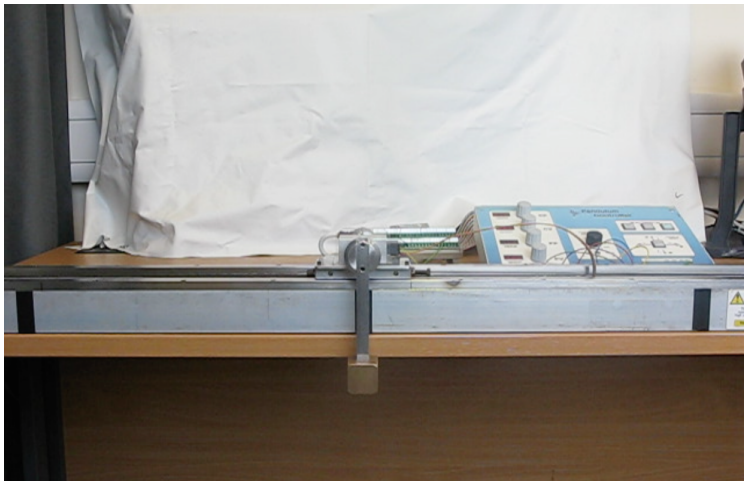


Second order Markov (AR(2))

$$y_t \in \mathbb{R}^1 \quad p(y_t|y_{t-1}, y_{t-2}) = \mathcal{G}(y_t; \lambda_1 y_{t-1} + \lambda_2 y_{t-2}, \sigma^2)$$
$$[\lambda_1, \lambda_2] = [1.57, -0.78] \quad \sigma^2 = 0.01$$

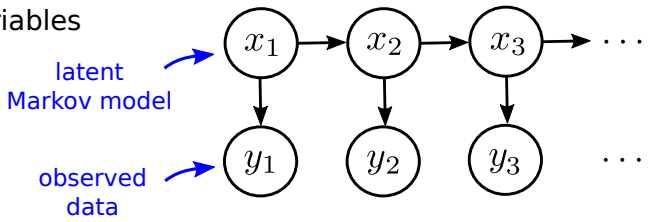


Example application of Markov Models: pendulum swing up control problem



Hidden Markov models

Real data depend on latent variables



ASR

x phonemes/words

y waveform/feature

Computer Vision

x objects, pose, lighting

y image pixel intensities

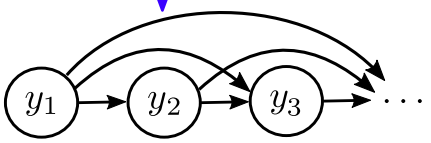
Natural Language Processing

x topics

y words

$$p(y_{1:T}, x_{1:T}) = \prod_{t=1}^{T} p(x_t|x_{t-1})p(y_t|x_t)$$

marginalise latents



$$p(y_{1:T})$$

fully connected

Two prevelant Examples:

Hidden Markov Models (discrete x)
Linear Gaussian State Space Models (Gaussian x and y)

Hidden Markov models: discrete hidden state

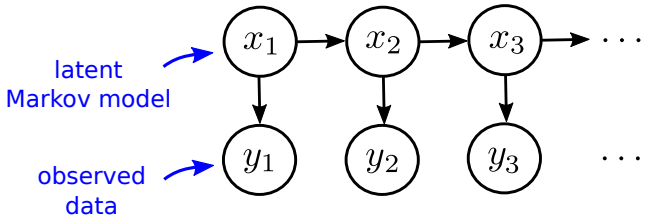
Discrete Hidden State

$x_t \in \{1, ..., K\}$

$p(x_t = k|x_{t-1} = l) = T_{k,l}$

E.g. in examples below $K = 2$

$T = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$

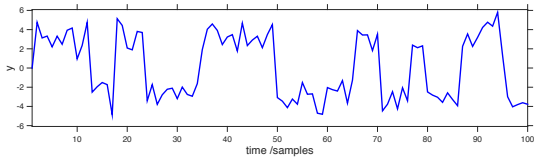


$p(y_{1:T}, x_{1:T}) = \prod_{t=1}^{T} p(x_t|x_{t-1})p(y_t|x_t)$

Continuous Observed State

$p(y_t|x_t = k) = \mathcal{G}(y_t; \mu_k, \Sigma_k)$

$\mu_1 = 3 \quad \mu_2 = -3 \quad \sigma_1^2 = \sigma_2^2 = 1$



Discrete Observed State

$p(y_t = l|x_t = k) = S_{l,k}$

$S = \begin{bmatrix} 0.5 & 0 \\ 0.5 & 0 \\ 0 & 1 \end{bmatrix}$

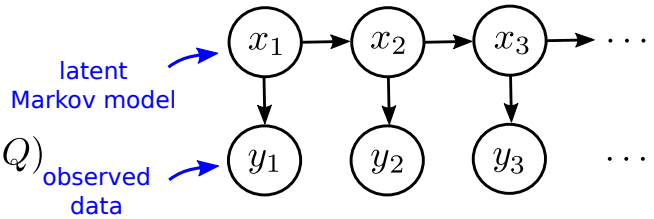
ABBBBAAABAAACCCCCBBBBBBCCCCCCCCCCC
AAABBBBAABAAABBCCCCCCCCCCCCCCCCCBBA
AACCCCCCBABCCCCCCCAABBAABABCCCCC

Hidden Markov models: continuous hidden state (LGSSMs)

Continuous Hidden State

$x_t \in \mathbb{R}^K$

$p(x_t|x_{t-1}) = \mathcal{G}(x_t; Ax_{t-1}, Q)$



Continuous Observed State

$y_t \in \mathbb{R}^D$

$p(y_t|x_t) = \mathcal{G}(y_t; Cx_t, R)$

$p(y_{1:T}, x_{1:T}) = \prod_{t=1}^{T} p(x_t|x_{t-1})p(y_t|x_t)$

E.g. simple example $K = 2 \quad D = 1$

$A = \lambda \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$

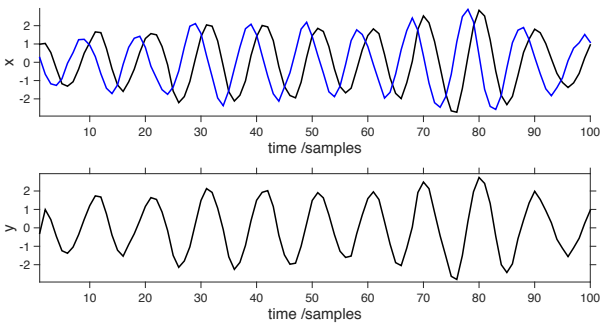
$\lambda = 0.99 \quad \theta = 2\pi/10$

$Q = (1 - \lambda^2) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

dynamics model

$C = [1, 0] \quad R = 0.01$

obs. model



Varieties of Inference

Distributional estimates

infer single state or sequence?

future data available?

	marginal	joint
filter	$p(x_t \mid y_{1:t})$	$p(x_{1:t} \mid y_{1:t})$
smoother	$p(x_t \mid y_{1:T})$	$p(x_{1:T} \mid y_{1:T})$

Point estimates

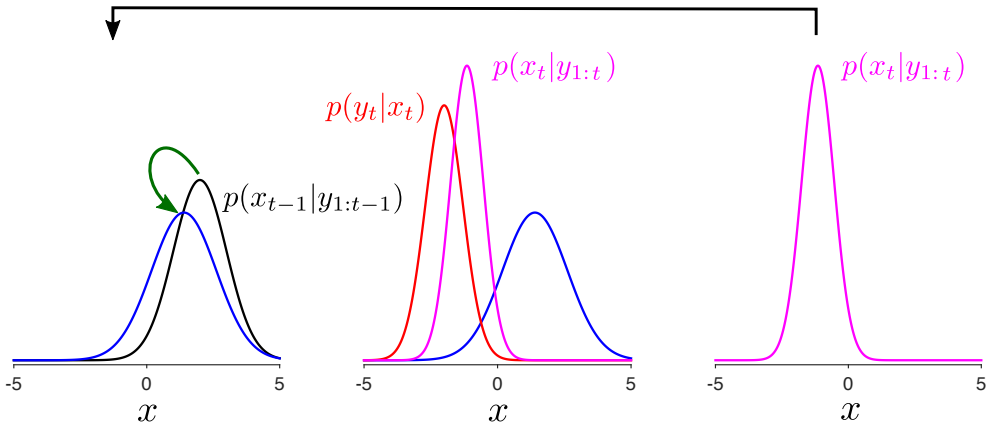
most probable state @ t

$$x_t^* = \arg\max_{x_t} p(x_t \mid y_{1:T})$$

most probable sequence

$$x'_{1:T} = \arg\max_{x_{1:T}} p(x_{1:T} \mid y_{1:T})$$

Inference: Kalman Filter



diffuse via dynamics

combine with likelihood

sum for discrete hidden state

$$p(x_{t-1} \mid y_{1:t-1})$$

$$p(x_t \mid y_{1:t-1}) = \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}) \mathrm{d}x_{t-1}$$

$$p(x_t \mid y_{1:t}) \propto p(x_t \mid y_{1:t-1}) p(y_t \mid x_t)$$

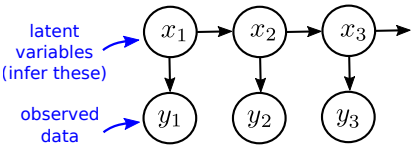
Bayes' Rule

prior

likelihood

Inference: Derivation of General Filtering Equations

Model



$$p(y_{1:T}, x_{1:T}) = \prod_{t=1}^{T} p(x_t|x_{t-1})p(y_t|x_t)$$

+

Rules of probability

product rule

$$p(A|B,C) = \frac{1}{p(B|C)}\, p(B|A,C)p(A|C)$$

sum rule

$$p(A|C) = \sum_B p(A,B|C)$$

Inference

= ?

$$\begin{aligned}
p(x_t|y_{1:t}) &= p(x_t|y_t, y_{1:t-1}) \\
&= \frac{1}{p(y_t|y_{1:t-1})}\, p(y_t|x_t, y_{1:t-1})p(x_t|y_{1:t-1}) && \text{product rule } A = x_t\ B = y_t\ C = y_{1:t-1} \\
&= \frac{1}{p(y_t|y_{1:t-1})}\, p(y_t|x_t)p(x_t|y_{1:t-1}) && \text{conditional independence from model } y_t \perp y_{1:t-1}|x_t \\
&\propto p(y_t|x_t)p(x_t|y_{1:t-1}) && \text{constant of proportionality } p(y_t|y_{1:t-1}) \text{ (see learning)}
\end{aligned}$$

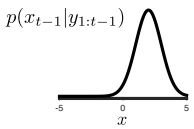
$$\begin{aligned}
p(x_t|y_{1:t-1}) &= \int p(x_t, x_{t-1}|y_{1:t-1})\mathrm{d}x_{t-1} && \text{sum rule } A = x_t\ B = x_{t-1}\ C = y_{1:t-1} \\
&= \int p(x_t|x_{t-1}, y_{1:t-1})p(x_{t-1}|y_{1:t-1})\mathrm{d}x_{t-1} && \text{product rule} \\
&= \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})\mathrm{d}x_{t-1} && \text{conditional independence from model}
\end{aligned}$$

Inference: Kalman Filter

$$p(x_{t-1}|y_{1:t-1}) = \mathcal{G}(x_{t-1}; \mu_{t-1}^{t-1}, V_{t-1}^{t-1})$$

most recent data used in prediction

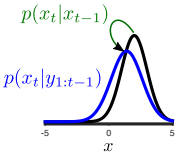
variable being predicted



diffuse via dynamics

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})\mathrm{d}x_{t-1}$$

diffuses toward 0

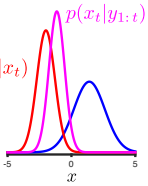


$$p(x_t|y_{1:t-1}) = \mathcal{G}(x_t; \mu_t^{t-1}, V_t^{t-1}) \qquad \mu_t^{t-1} = A\mu_{t-1}^{t-1}$$

$$V_t^{t-1} = AV_{t-1}^{t-1}A^\top + Q$$

variance inflates

combine with likelihood



$$p(x_t|y_{1:t}) \propto p(x_t|y_{1:t-1})p(y_t|x_t)$$

prior likelihood

$$p(x_t|y_{1:t}) = \mathcal{G}(x_t; \mu_t^t, V_t^t)$$

prediction correction

$$\mu_t^t = \mu_t^{t-1} + K_t(y_t - C\mu_t^{t-1})$$

$$V_t^t = V_t^{t-1} - K_t C V_t^{t-1}$$

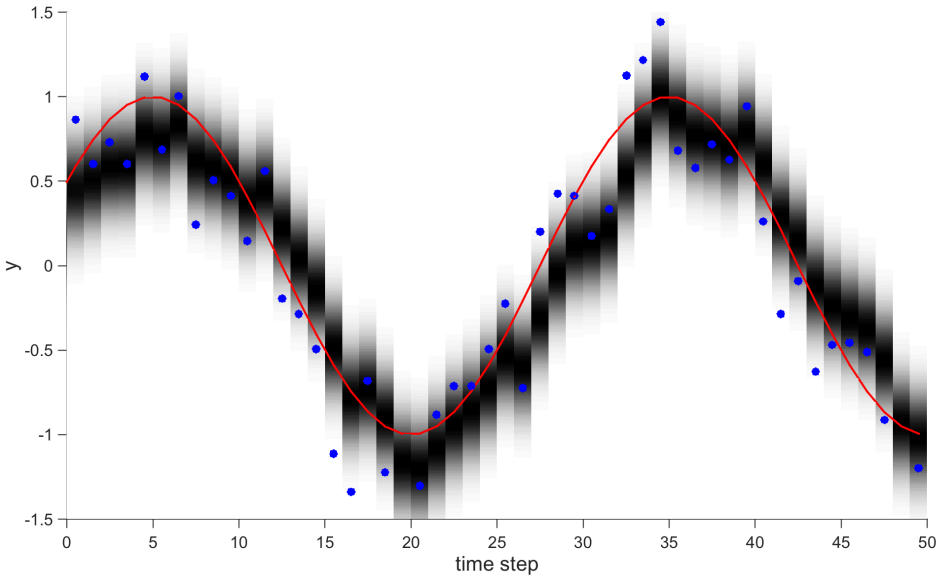
Kalman gain $K_t = V_t^{t-1}C^\top(CV_t^{t-1}C^\top + R)^{-1}$

Kalman Filter Demo

- data: $y_t = \sin(\omega t) + \sigma_y \epsilon_t$ where $\sigma_y^2 = 0.1$
- model: $x_t = \lambda x_{t-1} + \sigma\eta$ and $y_t = x_t + \sigma_y \eta'_t$
  where $\lambda = 0.99$ and $\sigma^2 = 1 - \lambda^2$
- demo shows how the Kalman filter processes the data to form estimates of the hidden state at each time point $p(x_t|y_{1:t})$

Kalman Filter Demo

observed noisy data y_t, ground truth sinusoid

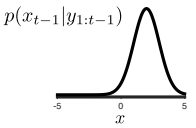


Inference: Forward Algorithm

$$p(x_{t-1} = k|y_{1:t-1}) = \rho_{t-1}^{t-1}(k)$$

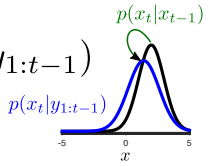
most recent data used in prediction

variable being predicted



diffuse via dynamics

$$p(x_t = k|y_{1:t-1}) = \sum_{l=1}^{K} p(x_t = k|x_{t-1} = l)p(x_{t-1} = l|y_{1:t-1})$$

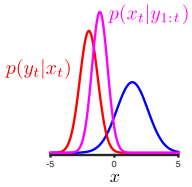


$$\rho_t^{t-1}(k) = \sum_{l=1}^{K} T(k,l)\rho_{t-1}^{t-1}(l)$$

combine with likelihood

$$p(x_t = k|y_{1:t}) \propto p(x_t = k|y_{1:t-1})p(y_t|x_t = k)$$

prior likelihood



$$\rho_t^t(k) \propto \rho_t^{t-1}(k)p(y_t|x_t = k)$$

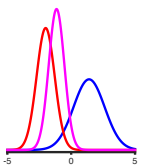
When implementing, take care with numerical underflow/overflow.

Computing the likelihood

How can we compute the likelihood efficiently?

$$p(y_{1:T}) = \prod_{t=1}^{T} p(y_t|y_{1:t-1})$$

already retuned by Kalman Filter/Forward algorithm



$$p(x_t|y_{1:t}) = \frac{1}{p(y_t|y_{1:t-1})} p(y_t|x_t)p(x_t|y_{1:t-1})$$

$$\propto p(y_t|x_t)p(x_t|y_{1:t-1})$$

$p(y_t|y_{1:t-1})$ is normaliser of filter/forward algorithm update

How can we compute the smoothing estimate?

$$p(x_t|y_{1:T})$$

LGSSM: Kalman Smoother
HMM: Forward-Backward= Algorithm

How can we compute the most probable sequence?

$$x'_{1:T} = \arg\max_{x_{1:T}} p(x_{1:T}|y_{1:T})$$

LGSSM: Kalman Smoother
HMM: Viterbi Decoding

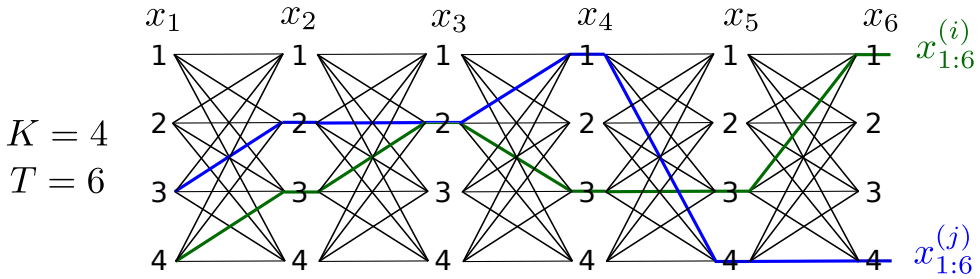
The magic of the Forward Algorithm: Dynamic Programming

What's going on here?

In discrete case, likelihood involves sum over all sequences: $x_{1:T}^{(k)}$

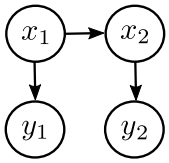
$$p(y_{1:T}) = \sum_{\text{all sequences } k} p(y_{1:T}, x_{1:T}^{(k)})$$

Trellis diagram represents possible sequences:



$K = 4$
$T = 6$

Exponential number of sequences: K^T



But Forward algorithm had linear complexity in time (loop over t)

Markov property means we can forget history of previous states:
just remember last one (dynamic programming/belief propagation)

Maximum Likelihood Learning of HMMs: simple once inference is solved

log-likelihood: $\log p(y_{1:T}|\theta) = \log \int p(y_{1:T}, x_{1:T}|\theta) dx_{1:T}$

gradient of log-likelihood: $\frac{\mathrm{d}}{\mathrm{d}\theta} \log p(y_{1:T}|\theta) = \frac{1}{p(y_{1:T}|\theta)} \int \frac{\mathrm{d}}{\mathrm{d}\theta} p(y_{1:T}, x_{1:T}|\theta) dx_{1:T}$

simple form: e.g. quadratic in x for LGSSMs

$E(\theta; x_{1:T}, y_{1:T}) = \sum_t [\log p(y_t|x_t, \theta) + \log p(x_t|x_{t-1}, \theta)]$

show gradient depends on simple moments of posterior:

$$\frac{\mathrm{d}}{\mathrm{d}\theta} \log p(y_{1:T}|\theta) = \frac{1}{p(y_{1:T}|\theta)} \int \frac{\mathrm{d}}{\mathrm{d}\theta} \exp(\overbrace{\log p(y_{1:T}, x_{1:T}|\theta)}^{E(\theta; x_{1:T}, y_{1:T})}) dx_{1:T}$$

$$\frac{\mathrm{d}}{\mathrm{d}\theta} \log p(y_{1:T}|\theta) = \frac{1}{p(y_{1:T}|\theta)} \int p(y_{1:T}, x_{1:T}|\theta) \frac{\mathrm{d}}{\mathrm{d}\theta} E(\theta; x_{1:T}, y_{1:T}) dx_{1:T}$$

$$\frac{\mathrm{d}}{\mathrm{d}\theta} \log p(y_{1:T}|\theta) = \int p(x_{1:T}|y_{1:T}, \theta) \frac{\mathrm{d}}{\mathrm{d}\theta} E(\theta; x_{1:T}, y_{1:T}) dx_{1:T}$$

$$\frac{\mathrm{d}}{\mathrm{d}\theta} \log p(y_{1:T}|\theta) = \left\langle \frac{\mathrm{d}}{\mathrm{d}\theta} E(\theta; x_{1:T}, y_{1:T}) \right\rangle_{p(x_{1:T}|y_{1:T}, \theta)}$$

requires posterior moments: marginals and pairwise marginals