State Space Model

Yan Shunxing and Liang Tiannuo

Department of Probability and Statistics
School of Mathematical Sciences
University of Science and Technology of China

yanshunxing@mail.ustc.edu.cn

December 17, 2020

Contents

State Space Model

A state space model or SSM is just like an HMM, but it is more general. The model can be written in the following generic form:

$$\mathbf{z}_t = g(\mathbf{u}_t, \mathbf{z}_{t-1}, \epsilon_t)$$

$$\mathbf{y}_t = h(\mathbf{z}_t, \mathbf{u}_t, \delta_t)$$

where

- $\mathbf{z}_t$ is the hidden state,
- $\mathbf{u}_t$ is an optional input or control signal,
- $\mathbf{y}_t$ is the observation,

State Space Model

and

- g is the transition model,
- h is the observation model,
- ϵ_t is the system noise at time t,
- δ_t is the observation noise at time t.

We assume that all parameters of the model, θ, are known; if not, they can be included into the hidden state.

State Space Model

Goals in using SSMs

- to recursively estimate $p(\mathbf{z}_t \mid \mathbf{y}_{1:t}, \mathbf{u}_{1:t}, \theta)$
- to convert our beliefs about the hidden state into predictions about future observables by computing the posterior predictive $p(\mathbf{y}_{t+1} \mid \mathbf{y}_{1:t})$.

Linear-Guassian SSM

An important special case of an SSM is where all factors are linear-Gaussian.
In other words, we assume
- The transition model is a linear function

$$\mathbf{z}_t = \mathbf{A}_t \mathbf{z}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \epsilon_t$$

- The observation model is a linear function

$$\mathbf{y}_t = \mathbf{C}_t \mathbf{z}_t + \mathbf{D}_t \mathbf{u}_t + \delta_t$$

Linear-Guassian SSM

- The system noise is Gaussian

$$\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$$

- The observation noise is Gaussian

$$\delta_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$$

This model is called a linear-Gaussian SSM (LG-SSM) or a linear dynamical system (LDS).

If the parameters $\theta_t = (\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t, \mathbf{D}_t, \mathbf{Q}_t, \mathbf{R}_t)$ are independent of time, the model is called stationary.

Application

The state-space model can be applied in subjects such as economics, statistics, computer science and electrical engineering,some of which we discuss in the sections below.

We mostly focus on LG-SSMs, for simplicity, although non-linear and/or non-Gaussian SSMs are even more widely used.

Outline

SSMs for object tracking

Consider a simple object moving in a 2D plane

- z_{1t} and z_{2t} are the horizontal and vertical locations of the object
- $\dot{z}_{1t}$ and $\dot{z}_{2t}$ are the corresponding velocity.
- Assume that the object is moving at constant velocity, but is perturbed by random Gaussian noise (e.g., due to the wind).
- We can observe the location of the object disturbed by Guassian noise

SSMs for object tracking

We can represent this as a state vector $\mathbf{z}_t \in \mathbb{R}^4$ as:

$$\mathbf{z}_t^T = \begin{pmatrix} z_{1t} & z_{2t} & \dot{z}_{1t} & \dot{z}_{2t} \end{pmatrix}$$

Thus system dynamics are follows:

$$\mathbf{z}_t = \mathbf{A}_t \mathbf{z}_{t-1} + \epsilon_t$$

$$\begin{pmatrix} z_{1t} \\ z_{2t} \\ \dot{z}_{1t} \\ \dot{z}_{2t} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} z_{1,t-1} \\ z_{2,t-1} \\ \dot{z}_{1,t-1} \\ \dot{z}_{2,t-1} \end{pmatrix} + \begin{pmatrix} \epsilon_{1t} \\ \epsilon_{2t} \\ \epsilon_{3t} \\ \epsilon_{4t} \end{pmatrix}$$

where $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ is the system noise, and Δ is the sampling period.

SSMs for object tracking

Let $\mathbf{y}_t \in \mathbb{R}^2$ represent our observation, which we assume is subject to Gaussian noise. Then

$$\mathbf{y}_t = \mathbf{C}_t \mathbf{z}_t + \delta_t$$

$$\begin{pmatrix} y_{1t} \\ y_{2t} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} z_{1t} \\ z_{2t} \\ \dot{z}_{1t} \\ \dot{z}_{2t} \end{pmatrix} + \begin{pmatrix} \delta_{1t} \\ \delta_{2t} \\ \delta_{3t} \\ \delta_{4t} \end{pmatrix}$$

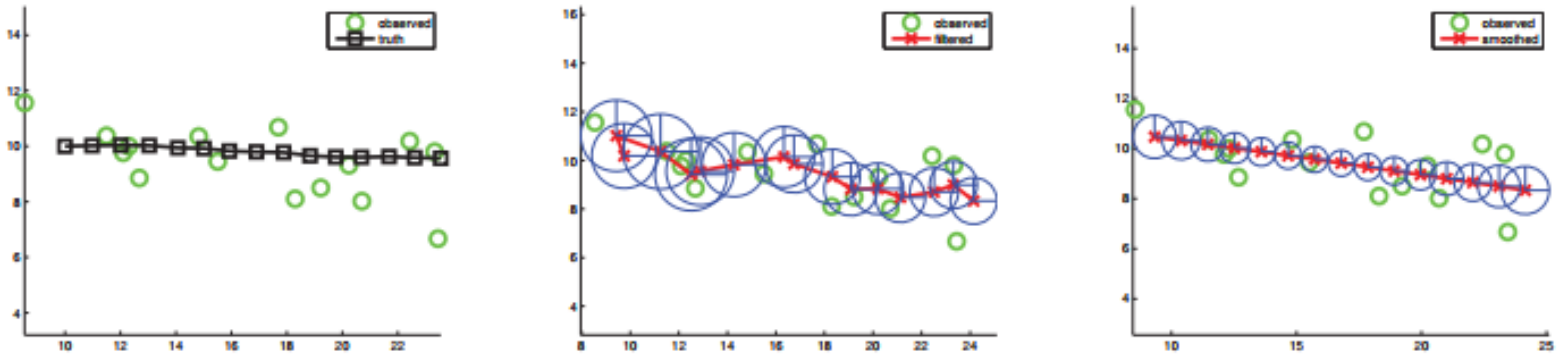
where $\delta_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ is the measurement noise.

SSMs for object tracking

Finally, we need to specify our initial (prior) beliefs about the state of the object $p(\mathbf{z}_1)$.

We will assume this is a Gaussian $p(\mathbf{z}_1) = \mathcal{N}\left(\mathbf{z}_1 \mid \boldsymbol{\mu}_{1|0}, \Sigma_{1|0}\right)$.

SSMs for object tracking



(a) (b) (c)

Figure 18.1 Illustration of Kalman filtering and smoothing. (a) Observations (green cirles) are generated by an object moving to the right (true location denoted by black squares). (b) Filtered estimated is shown by dotted red line. Red cross is the posterior mean, blue circles are 95% confidence ellipses derived from the posterior covariance. For clarity, we only plot the ellipses every other time step. (c) Same as (b), but using offline Kalman smoothing. Figure generated by `kalmanTrackingDemo`.

Outline

Online parameter learning via recursive least squares

The basic idea

- Define the prior to be $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\theta}_0, \boldsymbol{\Sigma}_0)$
- Let the hidden state be $\mathbf{z}_t = \boldsymbol{\theta}$
- Let the time-varying observation model be the current data vector

If we assume the regression parameters do not change, we can set $\mathbf{A}_t = \mathbf{I}$ and $\mathbf{Q}_t = 0$, so

$$p(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1}) = \mathcal{N}(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1}, 0) = \delta_{\boldsymbol{\theta}_{t-1}}(\boldsymbol{\theta}_t)$$

Online parameter learning via recursive least squares

Let $\mathbf{C}_t = \mathbf{x}_t^T$, and $\mathbf{R}_t = \sigma^2$, so the (non-stationary) observation model has the form

$$\mathcal{N}\left(\mathbf{y}_t \mid \mathbf{C}_t \mathbf{z}_t, \mathbf{R}_t\right) = \mathcal{N}\left(\mathbf{y}_t \mid \mathbf{x}_t^T \boldsymbol{\theta}_t, \sigma^2\right)$$

Applying the Kalman filter to this model provides a way to update our posterior beliefs about the parameters as the data streams in. This is known as the recursive least squares or RLS algorithm.

Outline

SSM for time series forecasting

The idea in the state-space approach to time series

- Create a generative model of the data in terms of latent processes, which capture different aspects of the signal.
- Then integrate out the hidden variables to compute the posterior predictive of the visibles.

We focus on the case of scalar (one dimensional) time series, for simplicity.

Time series model

In this part, we only consider the following models of time series

- Local level model
- Local linear model
- Seasonality
- ARMA models

Local level model

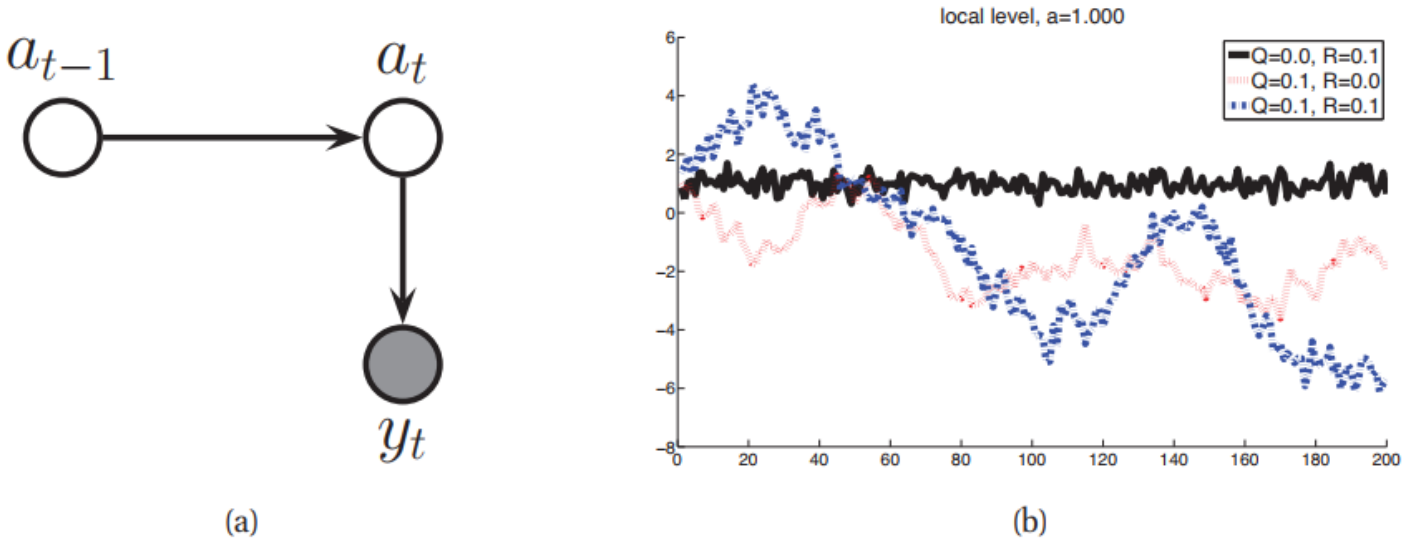
The simplest latent process is known as the local level model, which has the form

$$y_t = a_t + \epsilon_t^y, \quad \epsilon_t^y \sim \mathcal{N}(0, R)$$

$$a_t = a_{t-1} + \epsilon_t^a, \quad \epsilon_t^a \sim \mathcal{N}(0, Q)$$

where the hidden state is just $\mathbf{z}_t = a_t$. This model asserts that the observed data $y_t \in \mathbb{R}$ is equal to some unknown level term $a_t \in \mathbb{R}$, plus observation noise with variance R. In addition, the level a_t evolves over time subject to system noise with variance Q.

Local level model



(a) (b)

Figure 18.5 (a) Local level model. (b) Sample output, for $a_0 = 10$. Black solid line: $Q = 0$, $R = 1$ (deterministic system, noisy observations). Red dotted line: $Q = 0.1$, $R = 0$ (noisy system, deterministic observation). Blue dot-dash line: $Q = 0.1$, $R = 1$ (noisy system and observations). Figure generated by `ssmTimeSeriesSimple`.

Local linear trend

Many time series exhibit linear trends upwards or downwards, at least locally. We can model this by letting the level a_t change by an amount b_t at each step as follows:

$$y_t = a_t + \epsilon_t^y, \quad \epsilon_t^y \sim \mathcal{N}(0, R)$$

$$a_t = a_{t-1} + b_{t-1} + \epsilon_t^a, \quad \epsilon_t^a \sim \mathcal{N}(0, Q_a)$$

$$b_t = b_{t-1} + \epsilon_t^b, \quad \epsilon_t^b \sim \mathcal{N}(0, Q_b)$$

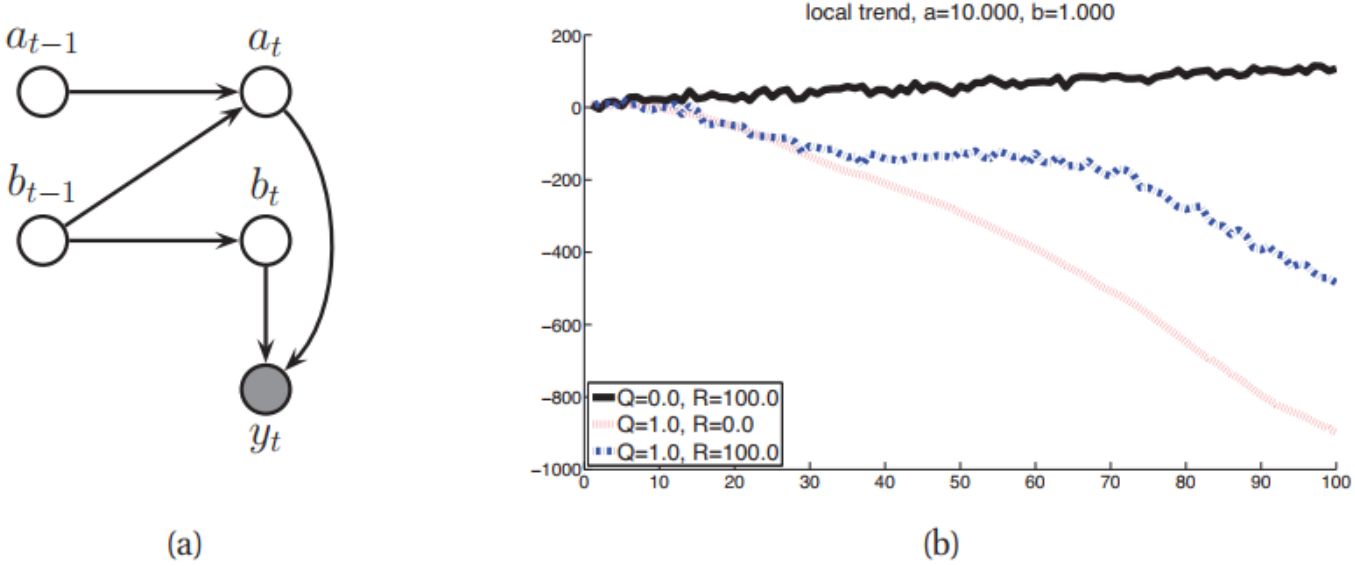
Local linear trend

We can write this in standard form by defining $\mathbf{z}_t = (a_t, b_t)$ and

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \mathbf{C} = \begin{pmatrix} 1 & 0 \end{pmatrix}, \mathbf{Q} = \begin{pmatrix} Q_a & 0 \\ 0 & Q_b \end{pmatrix}$$

When $Q_b = 0$, we have $b_t = b_0$, which is some constant defining the slope of the line. If in addition we have $Q_a = 0$ then we have $a_t = a_{t-1} + b_0$. Unrolling this, we have $a_t = a_0 + b_0 t$, and hence $\mathbb{E}[y_t \mid \mathbf{y}_{1:t-1}] = a_0 + tb_0$. This is thus a generalization of the classic constant linear trend model.

Local linear model



(a) (b)

Figure 18.6 (a) Local Trend. (b) Sample output, for $a_0 = 10$, $b_0 = 1$. Color code as in Figure 18.5. Figure generated by `ssmTimeSeriesSimple`.

Seasonality

Many time series fluctuate periodically. This can be modeled by adding a latent process consisting of a series offset terms, c_t, which sum to zero (on average) over a complete cycle of S steps:

$$c_t = -\sum_{s=1}^{S-1} c_{t-s} + \epsilon_t^c, \epsilon_t^c \sim \mathcal{N}(0, Q_c)$$

and

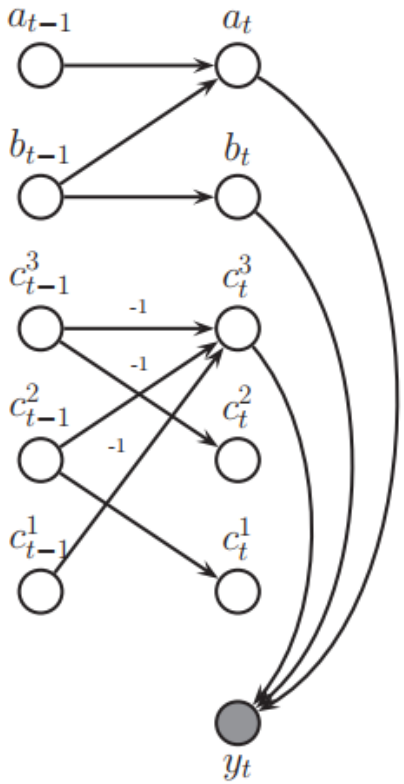
$$y_t = a_t + c_t + \epsilon_t^y, \quad \epsilon_t^y \sim \mathcal{N}(0, R)$$

where

$$a_t = a_{t-1} + b_{t-1} + \epsilon_t^a, \quad \epsilon_t^a \sim \mathcal{N}(0, Q_a)$$

$$b_t = b_{t-1} + \epsilon_t^b, \quad \epsilon_t^b \sim \mathcal{N}(0, Q_b)$$

Seasonality



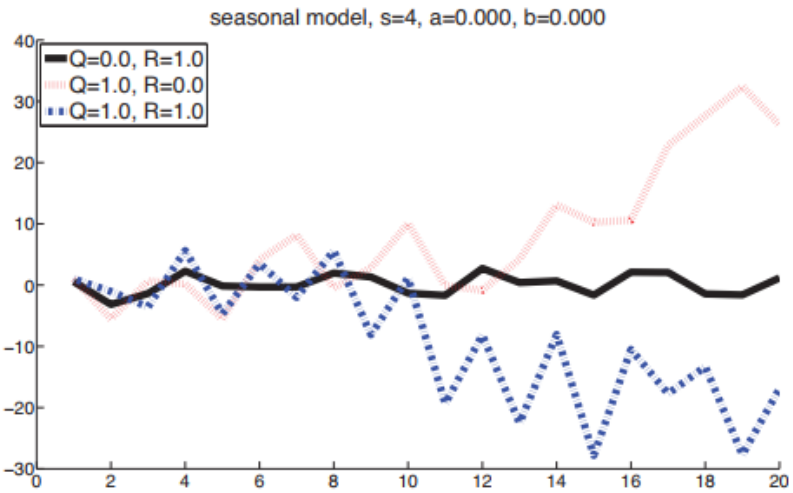


Figure 18.7 (a) Seasonal model. (b) Sample output, for $a_0 = b_0 = 0$, $\mathbf{c}_0 = (1, 1, 1)$, with a period of 4. Color code as in Figure 18.5. Figure generated by `ssmTimeSeriesSimple`.

ARMA models

The classical approach to time-series forecasting is based on ARMA models.
ARMA stands for auto-regressive moving-average, and has the form

$$x_t = \sum_{i=1}^{p} \alpha_i x_{t-i} + \sum_{j=1}^{q} \beta_j w_{t-j} + v_t$$

where $v_t, w_t \sim \mathcal{N}(0, 1)$ are independent Gaussian noise terms.

ARMA models

Some special cases

- $q = 0$, we have a pure AR model, where $x_t \perp x_i \mid x_{t-1;t-n}$, for $i < t - p$.
  - $p = 1$ and $q = 0$, we have the AR(l) model. It is just a first-order Markov chain.
- $p = 0$, we have a pure MA model, where $x_t \perp x_i$ for $i < t - q$.
  - $p = 0$ and $q = 1$, we have the MA(1) model.
- $p = q = 1$, we get the ARMA(1,1) model, which captures correlation at short and long time scales.

ARMA models

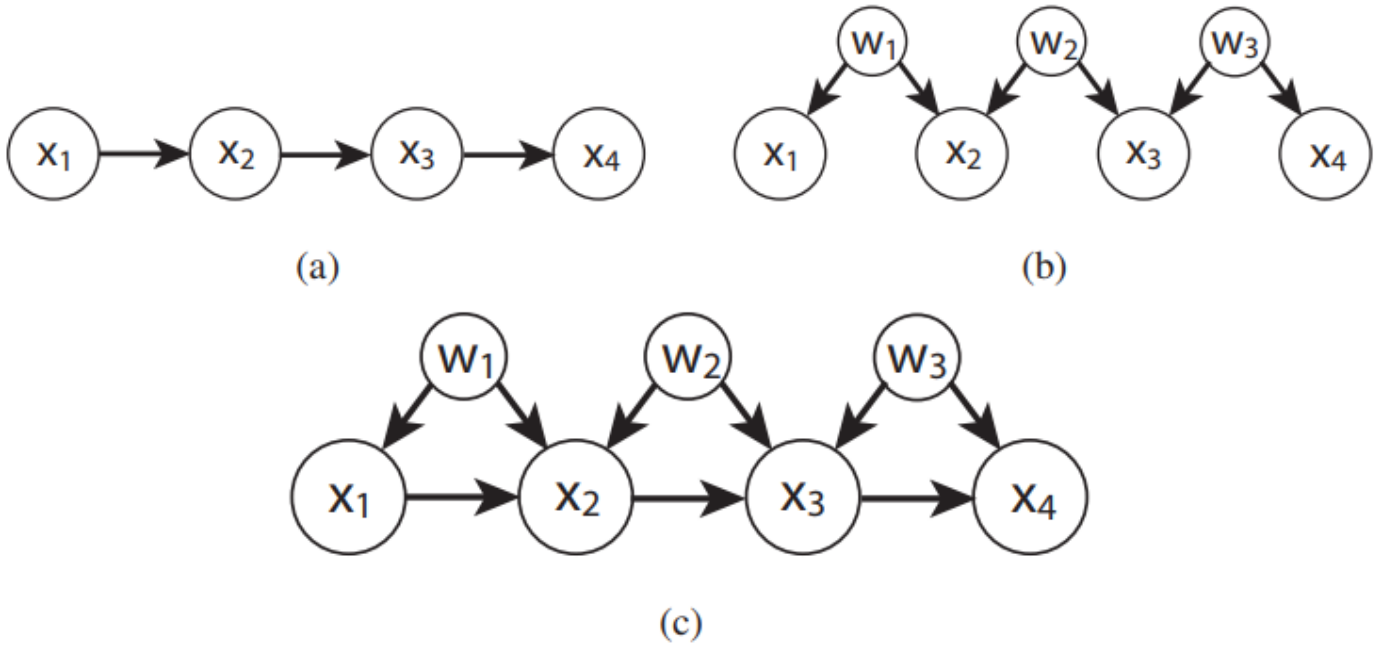


Figure 18.8 (a) An AR(1) model. (b) An MA(1) model represented as a bi-directed graph. (c) An ARMA(1,1) model. Source: Figure 5.14 of (Choi 2011). Used with kind permission of Myung Choi.

Inference

In this section, we discuss exact inference in LG-SSM models.

- We first consider the online case, which is analogous to the forwards algorithm for HMMs.
- Then consider the offline case, which is analogous to the forwards-backwards algorithm for HMMs.

Outline

The Kalman filtering algorithm

The Kalman filter is an algorithm for exact Bayesian filtering for linear-Gaussian state space models.
We will represent the marginal posterior at time t by

$$p(\mathbf{z}_t \mid \mathbf{y}_{1:t}, \mathbf{u}_{1:t}) = \mathcal{N}(\mathbf{z}_t \mid \boldsymbol{\mu}_t, \Sigma_t)$$

Since everything is Gaussian, we can perform the prediction and update steps in closed form.

Prediction step

The prediction step is straightforward to derive:

$$
\begin{aligned}
p\left(\mathbf{z}_t \mid \mathbf{y}_{1:t-1}, \mathbf{u}_{1:t}\right) &= \int \mathcal{N}\left(\mathbf{z}_t \mid \mathbf{A}_t \mathbf{z}_{t-1} + \mathbf{B}_t \mathbf{u}_t, \mathbf{Q}_t\right) \mathcal{N}\left(\mathbf{z}_{t-1} \mid \mu_{t-1}, \Sigma_{t-1}\right) d\mathbf{z} \\
&= \mathcal{N}\left(\mathbf{z}_t \mid \mu_{t|t-1}, \Sigma_{t|t-1}\right) \\
\mu_{t|t-1} &\triangleq \mathbf{A}_t \mu_{t-1} + \mathbf{B}_t \mathbf{u}_t \\
\Sigma_{t|t-1} &\triangleq \mathbf{A}_t \Sigma_{t-1} \mathbf{A}_t^T + \mathbf{Q}_t
\end{aligned}
$$

Measurement step

The measurement step can be computed using Bayes rule

$$p\left(\mathbf{z}_t \mid \mathbf{y}_t, \mathbf{y}_{1:t-1}, \mathbf{u}_{1:t}\right) \propto p\left(\mathbf{y}_t \mid \mathbf{z}_t, \mathbf{u}_t\right) p\left(\mathbf{z}_t \mid \mathbf{y}_{1:t-1}, \mathbf{u}_{1:t}\right)$$

After mathematical derivation of the formula

$$\begin{aligned} p\left(\mathbf{z}_t \mid \mathbf{y}_{1:t}, \mathbf{u}_t\right) &= \mathcal{N}\left(\mathbf{z}_t \mid \mu_t, \Sigma_t\right) \\ \mu_t &= \mu_{t|t-1} + \mathbf{K}_t \mathbf{r}_t \\ \Sigma_t &= \left(\mathbf{I} - \mathbf{K}_t \mathbf{C}_t\right) \Sigma_{t|t-1} \end{aligned}$$

where

$$\begin{aligned} \mathbf{r}_t &\triangleq \mathbf{y}_t - \hat{\mathbf{y}}_t \\ \hat{\mathbf{y}}_t &\triangleq \mathbb{E}\left[\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \mathbf{u}_{1:t}\right] = \mathbf{C}_t \mu_{t|t-1} + \mathbf{D}_t \mathbf{u}_t \end{aligned}$$

and $\mathbf{K}_t$ is the Kalman gain matrix, given by

$$\mathbf{K}_t \triangleq \Sigma_{t|t-1} \mathbf{C}_t^T \mathbf{S}_t^{-1}$$

Measurement step

where

$$\begin{aligned}\mathbf{S}_t &\triangleq \operatorname{cov}\left[\mathbf{r}_t \mid \mathbf{y}_{1:t-1}, \mathbf{u}_{1:t}\right] \\ &= \mathbb{E}\left[\left(\mathbf{C}_t\mathbf{z}_t + \delta_t - \hat{\mathbf{y}}_t\right)\left(\mathbf{C}_t\mathbf{z}_t + \delta_t - \hat{\mathbf{y}}_t\right)^T \mid \mathbf{y}_{1:t-1}, \mathbf{u}_{1:t}\right] \\ &= \mathbf{C}_t\Sigma_{t|t-1}\mathbf{C}_t^T + \mathbf{R}_t\end{aligned}$$

where $\delta_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$ is an observation noise term which is independent of all other noise sources. Note that by using the matrix inversion lemma, the Kalman gain matrix can also be written as

$$\mathbf{K}_t = \Sigma_{t|t-1}\mathbf{C}^T\left(\mathbf{C}\Sigma_{t|t-1}\mathbf{C}^T + \mathbf{R}\right)^{-1} = \left(\Sigma_{t|t-1}^{-1} + \mathbf{C}^T\mathbf{R}\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{R}^{-1}$$

Marginal likelihood

As a byproduct of the algorithm, we can also compute the log-likelihood of the sequence using

$$\log p\left(\mathbf{y}_{1:T} \mid \mathbf{u}_{1:T}\right) = \sum_t \log p\left(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \mathbf{u}_{1:t}\right)$$

where

$$p\left(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \mathbf{u}_{1:t}\right) = \mathcal{N}\left(\mathbf{y}_t \mid \mathbf{C}_t \boldsymbol{\mu}_{t|t-1}, \mathbf{S}_t\right)$$

Posterior predictive

The one-step-ahead posterior predictive density for the observations can be computed as follows

$$
\begin{aligned}
p\left(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \mathbf{u}_{1:t}\right) &= \int \mathcal{N}\left(\mathbf{y}_t \mid \mathbf{C}\mathbf{z}_t, \mathbf{R}\right) \mathcal{N}\left(\mathbf{z}_t \mid \boldsymbol{\mu}_{t|t-1}, \Sigma_{t|t-1}\right) d\mathbf{z}_t \\
&= \mathcal{N}\left(\mathbf{y}_t \mid \mathbf{C}\boldsymbol{\mu}_{t|t-1}, \mathbf{C}\Sigma_{t|t-1}\mathbf{C}^T + \mathbf{R}\right)
\end{aligned}
$$

This is useful for time series forecasting.

Outline

The Kalman smoothing algorithm

We have described the Kalman filter, which

- computes $p(z_t|y_{1:t})$ for each t.
- can be regarded as message passing on a graph, from left to right.
- is useful for online inference problems, such as tracking.

However, in an offline setting, we can

- wait until all the data has arrived, and then compute $p(z_t|y_{1:T})$.
- reduce uncertainty by conditioning on past and future data.
- work backwards, from right to left, sending information from the future back to the past, and them combining the two information sources.

The Kalman smoothing algorithm

We have

$$
\begin{aligned}
p\left(\mathbf{z}_t \mid \mathbf{y}_{1:T}\right) &= \mathcal{N}\left(\boldsymbol{\mu}_{t|T}, \Sigma_{t|T}\right) \\
\boldsymbol{\mu}_{t|T} &= \boldsymbol{\mu}_{t|t} + \mathbf{J}_t\left(\boldsymbol{\mu}_{t+1|T} - \boldsymbol{\mu}_{t+1|t}\right) \\
\Sigma_{t|T} &= \boldsymbol{\Sigma}_{t|t} + \mathbf{J}_t\left(\Sigma_{t+1|T} - \Sigma_{t+1|t}\right)\mathbf{J}_t^T \\
\mathbf{J}_t &\triangleq \Sigma_{t|t}\mathbf{A}_{t+1}^T\Sigma_{t+1|t}^{-1}
\end{aligned}
$$

where $\mathbf{J}_t$ is the backwards Kalman gain matrix.

Derivation

The key idea

$$p(\mathbf{z}_t \mid \mathbf{y}_{1:T}) = \int p(\mathbf{z}_t \mid \mathbf{y}_{1:T}, \mathbf{z}_{t+1}) p(\mathbf{z}_{t+1} \mid \mathbf{y}_{1:T}) d\mathbf{z}_{t+1}$$
$$= \int p(\mathbf{z}_t \mid \mathbf{y}_{1:t}, \mathbf{y}_{t+1:T}, \mathbf{z}_{t+1}) p(\mathbf{z}_{t+1} \mid \mathbf{y}_{1:T}) d\mathbf{z}_{t+1}$$

By induction, assume we have already computed the smoothed distribution for $t+1$:

$$p(\mathbf{z}_{t+1} \mid \mathbf{y}_{1:T}) = \mathcal{N}\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|T}, \Sigma_{t+1|T}\right)$$

Derivation

The question is: how do we perform the integration?
First, we compute the filtered two-slice distribution $p(\mathbf{z}_t, \mathbf{z}_{t+1} \mid \mathbf{y}_{1:t})$ as follows:

$$p(\mathbf{z}_t, \mathbf{z}_{t+1} \mid \mathbf{y}_{1:t}) = \mathcal{N}\left(\begin{pmatrix} \mathbf{z}_t \\ \mathbf{z}_{t+1} \end{pmatrix} \mid \begin{pmatrix} \boldsymbol{\mu}_{t|t} \\ \boldsymbol{\mu}_{t+1|t} \end{pmatrix} \begin{pmatrix} \boldsymbol{\Sigma}_{t|t} & \boldsymbol{\Sigma}_{t|t}\mathbf{A}_{t+1}^T \\ \mathbf{A}_{t+1}\Sigma_{t|t} & \boldsymbol{\Sigma}_{t+1|t} \end{pmatrix}\right)$$

So

$$p(\mathbf{z}_t \mid \mathbf{z}_{t+1}, \mathbf{y}_{1:t}) = \mathcal{N}\left(\mathbf{z}_t \mid \boldsymbol{\mu}_{t|t} + \mathbf{J}_t\left(\mathbf{z}_{t+1} - \boldsymbol{\mu}_{t+1|t}\right), \Sigma_{t|t} - \mathbf{J}_t\Sigma_{t+1|t}\mathbf{J}_t^T\right)$$

Derivation

We can compute the smoothed distribution for t using the rules of iterated expectation and iterated covariance.
First, the mean:

$$\begin{aligned}
\boldsymbol{\mu}_{t|T} &= \mathbb{E}\left[\mathbb{E}\left[\mathbf{z}_t \mid \mathbf{z}_{t+1}, \mathbf{y}_{1:T}\right] \mid \mathbf{y}_{1:T}\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[\mathbf{z}_t \mid \mathbf{z}_{t+1}, \mathbf{y}_{1:t}\right] \mid \mathbf{y}_{1:T}\right] \\
&= \mathbb{E}\left[\boldsymbol{\mu}_{t|t} + \mathbf{J}_t\left(\mathbf{z}_{t+1} - \boldsymbol{\mu}_{t+1|t}\right) \mid \mathbf{y}_{1:T}\right] \\
&= \boldsymbol{\mu}_{t|t} + \mathbf{J}_t\left(\boldsymbol{\mu}_{t+1|T} - \boldsymbol{\mu}_{t+1|t}\right)
\end{aligned}$$

Derivation

Then the covariance:

$$
\begin{aligned}
\boldsymbol{\Sigma}_{t|T} &= \operatorname{cov}\left[\mathbb{E}\left[\mathbf{z}_t \mid \mathbf{z}_{t+1}, \mathbf{y}_{1:T}\right] \mid \mathbf{y}_{1:T}\right] + \mathbb{E}\left[\operatorname{cov}\left[\mathbf{z}_t \mid \mathbf{z}_{t+1}, \mathbf{y}_{1:T}\right] \mid \mathbf{y}_{1:T}\right] \\
&= \operatorname{cov}\left[\mathbb{E}\left[\mathbf{z}_t \mid \mathbf{z}_{t+1}, \mathbf{y}_{1:t}\right] \mid \mathbf{y}_{1:T}\right] + \mathbb{E}\left[\operatorname{cov}\left[\mathbf{z}_t \mid \mathbf{z}_{t+1}, \mathbf{y}_{1:t}\right] \mid \mathbf{y}_{1:T}\right] \\
&= \operatorname{cov}\left[\boldsymbol{\mu}_{t|t} + \mathbf{J}_t\left(\mathbf{z}_{t+1} - \boldsymbol{\mu}_{t+1|t}\right) \mid \mathbf{y}_{1:T}\right] + \mathbb{E}\left[\boldsymbol{\Sigma}_{t|t} - \mathbf{J}_t \boldsymbol{\Sigma}_{t+1|t} \mathbf{J}_t^T \mid \mathbf{y}_{1:T}\right] \\
&= \mathbf{J}_t \operatorname{cov}\left[\mathbf{z}_{t+1} - \boldsymbol{\mu}_{t+1|t} \mid \mathbf{y}_{1:T}\right] \mathbf{J}_t^T + \boldsymbol{\Sigma}_{t|t} - \mathbf{J}_t \boldsymbol{\Sigma}_{t+1|t} \mathbf{J}_t^T \\
&= \mathbf{J}_t \boldsymbol{\Sigma}_{t+1|T} \mathbf{J}_t^T + \boldsymbol{\Sigma}_{t|t} - \mathbf{J}_t \boldsymbol{\Sigma}_{t+1|t} \mathbf{J}_t^T \\
&= \boldsymbol{\Sigma}_{t|t} + \mathbf{J}_t\left(\boldsymbol{\Sigma}_{t+1|T} - \boldsymbol{\Sigma}_{t+1|t}\right) \mathbf{J}_t^T
\end{aligned}
$$

The algorithm can be initialized from $\mu_{T|T}$ and $\Sigma_{T|T}$ from the last step of the filtering algorithm.

The Kalman smoothing algorithm

The Kalman smoothing algorithm

- The algorithm can be initialized from $\mu_{T|T}$ and $\Sigma_{T|T}$ from the Kalman filter.
- The backwards pass does not need access to the data (not need $y_{1:T}$)
- It allows us to "throw away" potentially high dimensional observation vectors, and just keep the filtered belief states.

Comparison to the forwards-backwards algorithm for HMMs

In both the forwards and backwards passes for LDS, we always worked with normalized distributions.
Furthermore, the backwards pass depends on the results of the forwards pass. It is different from the usual presentation of forwards-backwards for HMMs, where the backwards pass can be computed independently of the forwards pass .

Comparison to the forwards-backwards algorithm for HMMs

It turns out that we can rewrite the Kalman smoother in a modified form which makes it more similar to forwards-backwards for HMMs. In particular, we have

$$\begin{aligned} p(\mathbf{z}_t \mid \mathbf{y}_{1:T}) &= \int p(\mathbf{z}_t \mid \mathbf{y}_{1:t}, \mathbf{z}_{t+1})\, p(\mathbf{z}_{t+1} \mid \mathbf{y}_{1:T})\, d\mathbf{z}_{t+1} \\ &= \int p(\mathbf{z}_t, \mathbf{z}_{t+1} \mid \mathbf{y}_{1:t}) \frac{p(\mathbf{z}_{t+1} \mid \mathbf{y}_{1:T})}{p(\mathbf{z}_{t+1} \mid \mathbf{y}_{1:t})} d\mathbf{z}_{t+1} \end{aligned}$$

Now

$$p(\mathbf{z}_{t+1} \mid \mathbf{y}_{1:T}) = \frac{p(\mathbf{y}_{t+1:T} \mid \mathbf{z}_{t+1}, \mathbf{y}_{1:t})\, p(\mathbf{z}_{t+1} \mid \mathbf{y}_{1:t})}{p(\mathbf{y}_{t+1:T} \mid \mathbf{y}_{1:t})}$$

Comparison to the forwards-backwards algorithm for HMMs

so

$$\frac{p\left(\mathbf{z}_{t+1} \mid \mathbf{y}_{1:T}\right)}{p\left(\mathbf{z}_{t+1} \mid \mathbf{y}_{1:t}\right)} = \frac{p\left(\mathbf{z}_{t+1} \mid \mathbf{y}_{1:t}\right) p\left(\mathbf{y}_{t+1:T} \mid \mathbf{z}_{t+1}\right)}{p\left(\mathbf{z}_{t+1} \mid \mathbf{y}_{1:t}\right) p\left(\mathbf{y}_{t+1:T} \mid \mathbf{y}_{1:t}\right)} \propto p\left(\mathbf{y}_{t+1:T} \mid \mathbf{z}_{t+1}\right)$$

which is the conditional likelihood of the future data. This backwards message can be computed independently of the forwards message.

Comparison to the forwards-backwards algorithm for HMMs

However, this approach has several disadvantages:

- it needs access to the original observation sequence;
- the backwards message is a likelihood, not a posterior, so it need not to integrate to 1 over z_t;
- when exact inference is not possible, it makes more sense to try to approximate the smoothed distribution rather than the backwards likelihood term.