Advanced Machine Learning
State-space models
Lecture 2

Mattias Villani

Division of Statistics and Machine Learning
Department of Computer and Information Science
Linköping University

Lecture overview

- **Estimating model parameters**
- **Bayesian inference for the LGSS model**
- **Live demo of some R packages**

Estimating model parameters

- The **linear Gaussian state-space (LGSS) model**

$$\text{Measurement eq:} \quad \mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \varepsilon_t \qquad \varepsilon_t \overset{iid}{\sim} N(0, \Omega_\varepsilon)$$

$$\text{State eq:} \quad \mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t + \nu_t \qquad \nu_t \overset{iid}{\sim} N(0, \Omega_\nu)$$

- The elements in **A**, **B**, **C**, Ω_ε and Ω_ν may be unknown.
- Example: time-varying regression with p covariates $\mathbf{z}_t$ $(p \times 1)$

$$y_t = \mathbf{z}_t^T \beta_t + \varepsilon_t, \qquad \varepsilon_t \overset{iid}{\sim} N(0, \Omega_\varepsilon)$$

$$\beta_{1t} = a_1 \cdot \beta_{1,t-1} + \nu_t \qquad \nu_t \overset{iid}{\sim} N(0, \Omega_\nu)$$

$$\vdots$$

$$\beta_{pt} = a_p \cdot \beta_{p,t-1} + \nu_t \qquad \nu_t \overset{iid}{\sim} N(0, \Omega_\nu)$$

  - Here $C = \mathbf{z}_t^T$, $\mathbf{x}_t = \beta_t$ and $\mathbf{A} = \mathrm{Diag}(a_1, ..., a_p)$.
- The state space model's matrices (**A** etc) are parametrized by $\theta = (\theta_1, ..., \theta_s)$. To be explicit: $A(\theta)$, $B(\theta)$, ...,$\Omega_\nu(\theta)$.

Estimating model parameters

- Two options: Maximum likelihood estimate (MLE) or Bayesian.
- **Likelihood function**

$$p\left(\mathbf{y}_1, \ldots, \mathbf{y}_T | \theta\right) = \prod_{t=1}^{T} p\left(\mathbf{y}_t | \mathbf{y}_{1:t-1}, \theta\right)$$

- How compute $p\left(\mathbf{y}_t | \mathbf{y}_{1:t-1}, \theta\right)$? The trick: i) condition on $\mathbf{x}_t$, ii) exploit conditional independencies, iii) get rid of $\mathbf{x}_t$ by integrating it out:

$$\begin{aligned} p\left(\mathbf{y}_t | \mathbf{y}_{1:t-1}, \theta\right) &= \int p\left(\mathbf{y}_t | \mathbf{y}_{1:t-1}, \mathbf{x}_t, \theta\right) p\left(\mathbf{x}_t | \mathbf{y}_{1:t-1}, \theta\right) d\mathbf{x}_t \\ &= \int p\left(\mathbf{y}_t | \mathbf{x}_t, \theta\right) p\left(\mathbf{x}_t | \mathbf{y}_{1:t-1}, \theta\right) d\mathbf{x}_t \end{aligned}$$

- Note:
  - $p\left(\mathbf{x}_t | \mathbf{y}_{1:t-1}, \theta\right) = \overline{\mathrm{bel}}(\mathbf{x}_t)$ is Gaussian
  - $p\left(\mathbf{y}_t | \mathbf{x}_t, \theta\right)$ is Gaussian
  - $p\left(\mathbf{y}_t | \mathbf{y}_{1:t-1}, \theta\right)$ is then also Gaussian [not obvious, but expected].

Estimating model parameters

- Remember: we are looking for the Gaussian $p\left(\mathbf{y}_t | \mathbf{y}_{1:t-1}, \theta\right)$.

- Mean by law of iterated expectations ($E = EE$)

$$\mathbb{E}\left(\mathbf{y}_t | \mathbf{y}_{1:t-1}, \theta\right) = \mathbf{C}\mathbb{E}\left(\mathbf{x}_t | \mathbf{y}_{1:t-1}, \theta\right) = \mathbf{C}\bar{\mu}_t$$

- Variance by conditional variance formula ($V = EV + VE$)

$$\begin{aligned}
\mathbb{V}\left(\mathbf{y}_t | \mathbf{y}_{1:t-1}, \theta\right) &= \mathbb{E}_{\mathbf{x}_t | \mathbf{y}_{1:t-1}, \theta}\left[\mathbb{V}\left(\mathbf{y}_t | \mathbf{x}_t, \mathbf{y}_{1:t-1}, \theta\right)\right] \\
&\quad + \mathbb{V}_{\mathbf{x}_t | \mathbf{y}_{1:t-1}, \theta}\left[\mathbb{E}\left(\mathbf{y}_t | \mathbf{x}_t, \mathbf{y}_{1:t-1}, \theta\right)\right] \\
&= \Omega_\varepsilon + \mathbb{V}_{\mathbf{x}_t | \mathbf{y}_{1:t-1}, \theta}\left(\mathbf{C}\mathbf{x}_t\right) = \Omega_\varepsilon + \mathbf{C}\bar{\Sigma}_t\mathbf{C}^T
\end{aligned}$$

Estimating model parameters

- In summary, the **likelihood function** is

$$p\left(\mathbf{y}_1, \ldots, \mathbf{y}_T | \theta\right) = \prod_{t=1}^{T} N\left(\mathbf{y}_t | \mathbf{C}\bar{\mu}_t, \mathbf{C}\bar{\Sigma}_t\mathbf{C}^T + \Omega_\varepsilon\right)$$

  where $\mathbf{C}$, Ω_ε, $\bar{\mu}_t$ and $\bar{\Sigma}_t$ all depend on θ generally.
- The Kalman filter gives us everything we need for $p\left(\mathbf{y}_1, \ldots, \mathbf{y}_T | \theta\right)$!
- **Numerical optimization** (e.g. `optim` in R) to find **MLE** $\hat{\theta}_{MLE}$.
- Approximate $\mathbb{V}\left(\hat{\theta}_{MLE}\right)$ from the numerical Hessian.
- Sampling from the **posterior distribution**

$$p\left(\theta | \mathbf{y}_1, \ldots, \mathbf{y}_T\right) \propto p\left(\mathbf{y}_1, \ldots, \mathbf{y}_T | \theta\right) p\left(\theta\right)$$

  by **Metropolis-Hastings**.

State smoothing

- **Filtering** (real time):

$$p(\mathbf{x}_t|\mathbf{y}_{1:t})$$

- **Smoothing** (retrospective):

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

- Start at the end $t = T$. We already have $p\left(\mathbf{x}_T|\mathbf{y}_{1:T}\right)$ from the last iteration of the Kalman filter. Work yourself backward in time to obtain $p(\mathbf{x}_{T-1}|\mathbf{y}_{1:T}), ..., p(\mathbf{x}_1|\mathbf{y}_{1:T})$.
- Note: the end result are the **marginal** densities at any t, $p(\mathbf{x}_t|\mathbf{y}_{1:T})$. More work to do if one also wants $p(\mathbf{x}_{t_1}, \mathbf{x}_{t_2}|\mathbf{y}_{1:T})$ for some times t_1 and t_2.

State smoothing

- **Algorithm** `Smoothing`($\mathbf{s}_{t+1}, \mathbf{S}_{t+1}, \mu_t, \Sigma_t, \bar{\mu}_{t+1}, \bar{\Sigma}_{t+1}$)
  - Mean update:
$$\mathbf{s}_t = \mu_t + \Sigma_t \mathbf{A}^T \bar{\Sigma}_{t+1}^{-1} \left(\mathbf{s}_{t+1} - \bar{\mu}_{t+1} \right)$$
  - Covariance update:
$$\mathbf{S}_t = \Sigma_t + \Sigma_t \mathbf{A}^T \bar{\Sigma}_{t+1}^{-1} \left(\mathbf{S}_{t+1} - \bar{\Sigma}_{t+1} \right) \bar{\Sigma}_{t+1}^{-1} \mathbf{A} \Sigma_t$$
  - Return $\mathbf{s}_t, \mathbf{S}_t$

Bayesian inference for the state

- How to **sample** from **posterior** of the **state** $p(\mathbf{x}_1, ..., \mathbf{x}_T | \mathbf{y}_{1:T}, \theta)$?
- Simulate the state trajectory **backward in time** starting with x_T:

$$p(\mathbf{x}_{1:T}|\mathbf{y}_{1:T},\theta) = p(\mathbf{x}_T|\mathbf{y}_{1:T},\theta)p(\mathbf{x}_{T-1}|\mathbf{x}_T,\mathbf{y}_{1:T},\theta)\cdots p(\mathbf{x}_1|\mathbf{x}_{T-1:2},\mathbf{y}_{1:T},\theta)$$

- **Forward Filtering Backward Sampling (FFBS)**:
  - Run the Kalman filter forward in time $t = 1, ..., T$.
  - Simulate x_T from $N(\mu_T, \Sigma_T)$.
  - Simulate states backward in time $t = T-1, T-2, ..., 1$:

$$\mathbf{x}_t|\mathbf{x}_{t+1:T}, y_{1:T}, \theta \sim N(\mathbf{h}_t, \mathbf{H}_t)$$

$$\mathbf{h}_t = \mu_t + \Sigma_t \mathbf{A}^T \bar{\Sigma}_{t+1}^{-1} (\mathbf{x}_{t+1} - \bar{\mu}_{t+1})$$

and

$$\mathbf{H}_t = \Sigma_t - \Sigma_t \mathbf{A}^T \bar{\Sigma}_{t+1}^{-1} \mathbf{A} \Sigma_t.$$

- Note: FFBS distributions conditions on $\mathbf{x}_{t+1:T}$.
- So the FFBS gives the *joint* (smoothing) posterior for $\mathbf{x}_{1:T}$, whereas the state smoothing gives the *marginal* posterior of $\mathbf{x}_t$ for all t.

DLM PACKAGE

- The **linear Gaussian state-space (LGSS) model**

$$\begin{aligned} \text{Measurement eq:} \quad \mathbf{y}_t &= \mathbf{C}\mathbf{x}_t + \varepsilon_t & \varepsilon_t &\overset{iid}{\sim} N(0, \Omega_\varepsilon) \\ \text{State eq:} \quad \mathbf{x}_t &= \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t + \nu_t & \nu_t &\overset{iid}{\sim} N(0, \Omega_\nu) \end{aligned}$$

- In the `dlm` package

$$\begin{aligned} \text{Measurement eq:} \quad Y_t &= F\theta_t + v_t & \varepsilon_t &\overset{iid}{\sim} N(0, V) \\ \text{State eq:} \quad \theta_t &= G\theta_{t-1} + w_t & \nu_t &\overset{iid}{\sim} N(0, W) \end{aligned}$$

- θ_t is the state vector in `dlm`. Y_t are the measurements (a vector).
- The `dlm` notation goes back to West and Harrison's yellow DLM bible.
- The state is an **unknown**, so it is a **greek letter**.
- Measurements is a **random variable** so it is a **capital letter**.
- `dlm` can also handle when F, G, V, W vary of over time.

DLM PACKAGE

- DLM

$$
\begin{aligned}
\text{Measurement eq:} \quad Y_t &= F\theta_t + v_t & \varepsilon_t &\overset{iid}{\sim} N(0, V) \\
\text{State eq:} \quad \theta_t &= G\theta_{t-1} + w_t & \nu_t &\overset{iid}{\sim} N(0, W)
\end{aligned}
$$

- Main functions:
  - `dlm` - creates the dlm model object
  - `dlmFilter` - Kalman filtering
  - `dlmSmooth` - State smoothing
  - `dlmLL` - computes the log-likelihood