LINKÖPING UNIVERSITY
Department of Computer and Information Science
Division of Statistics and Machine Learning
Matias Quiroz and Mattias Villani

2016-10-07
Advanced Machine Learning
Master and PhD course

State Space Models - Computer Lab

Deadline: October 17 at midnight
Teacher: Matias Quiroz
Grades: Pass/Fail
Submission: Via LISAM

You should use R to solve the lab since the computer exam will be in R.
Output: an individual report and a group report to be presented on the seminar.
Attach your code in LISAM as separate files.

---

1. **Implementation of the linear Gaussian state space model**. Consider the linear Gaussian state space model

$$\begin{aligned} y_t &= Cx_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \Omega_\epsilon) \\ x_t &= Ax_{t-1} + Bu_t + v_t, \quad v_t \sim \mathcal{N}(0, \Omega_v) \ (v_t \perp\!\!\!\perp \epsilon_t), \quad x_0 \sim \mathcal{N}(\mu_0, \Sigma_0) \end{aligned}$$

where $y_t \in \mathbb{R}^{n_y \times 1}$ is the vector of observations at time t, $x_t \in \mathbb{R}^{n_x \times 1}$ is the unobserved state vector and $u_t \in \mathbb{R}^{n_u \times 1}$ is the control vector for x_t. The dimensions of the matrices of the system are $C \in \mathbb{R}^{n_y \times n_x}$, $\Omega_\epsilon \in \mathbb{R}^{n_y \times n_y}$, $A, \Omega_v, \mu_0, \Sigma_0 \in \mathbb{R}^{n_x \times n_x}$ and $B \in \mathbb{R}^{n_x \times n_u}$.

Your implementation will use simulated data to remind you that this is a good way to check that your code works. Moreover, real data can often give rise to numerical instabilities that have to be taken into account and this is outside the scope of this course. In this problem you will work with both a scalar measurement and state equation (i.e. $n_y = n_x = 1$), but you should write the code so that it can handle matrices. This is achieved by using the data type `matrix` in `R` when you define the variables (even if they are scalars!).

Check your results against those obtained using the `R` package `DLM`. This ensures that your code works properly, but more important it gives you a good practice for the computer exam.

   (a) **Simulate the model**. Use `set.seed(1234)` when you simulate data in this exercise. This is because (i) reproducibility facilitates the correction of the assignment and (ii) numerical instabilities do not appear (except the ones caused by bugs!) when implementing the filtering and smoothing recursions.

   Set $x_0 = 0$ and simulate $T = 100$ observations from the following model

$$\begin{aligned} y_t &= 0.93x_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \Omega_\epsilon = 0.5) \\ x_t &= x_{t-1} + v_t, \quad v_t \sim \mathcal{N}(0, \Omega_v = 0.1) \ (v_t \perp\!\!\!\perp \epsilon_t), \quad x_0 \sim \mathcal{N}(\mu_0 = 0, \Sigma_0 = 10). \end{aligned} \tag{1}$$

   Note that this model does not have any control variables, but you can still implement the code in the general matrix form and define B and u_t to be a matrix/vector of only zeros. With `set.seed(1234)` the simulated state and observations should look as in Figure 1. The data is also found in `simulated_data.Rda`.

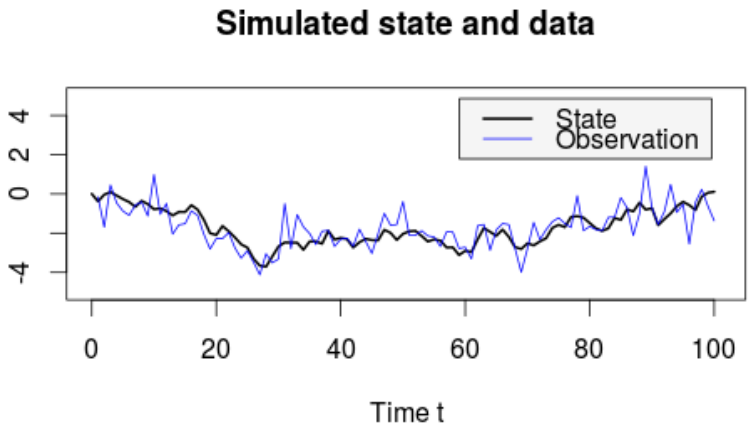

Figure 1: Simulated state and data with seed '1234'.

(b) **Filtering: Sequential state inference.** We will learn x_t sequentially for the model in Equation (1), using only data up to t, i.e. $y_{1:t}$. In other words, we will compute the filtering distribution $p(x_t|y_{1:t})$ for $t = 1, \ldots, T$. Because of the linear relationship between y_t and x_t and the normality of ϵ_t and v_t, $p(x_t|y_{1:t})$ is Gaussian, where the mean and variance are obtained from the Kalman filter recursions.

Implement the Kalman filter. For each $t = 1, \ldots, T$, plot the point estimate $\mathrm{E}[x_t|y_{1:t}]$ and a 0.95 probability interval for $x_t|y_{1:t}$. In the same graph, plot the true state x_t (that you simulated in (a)) and y_t. Compare your filtering distribution to that obtained by the `DLM` package using `dlmFilter`. Notice that `dlmFilter` does not return $\Sigma_t = \mathrm{V}[x_t|y_{1:t}]$ but its Singular Value Decomposition (SVD) instead. You can obtain Σ_t using the function `dlmSvd2var`.

(c) **Prediction of state and data by simulation**. We first note that both the predictive distribution for the state and the data are analytically available for the linear Gaussian state space model. We will here approach it by simulation to get a good intuition about the uncertainties involved when simulating the data and the states.

Suppose we want to predict future values of the process y_{T+k} conditional on $y_{1:T}$. It is easy to show that (convince yourself, but no need to include in the report)

$$\begin{aligned} p(y_{T+k}|y_{1:T}) &= \int p(y_{T+k}, x_{T+k}|y_{1:T})dx_{T+k} \\ &= \int p(y_{T+k}|x_{T+k}, y_{1:T})p(x_{T+k}|y_{1:T})dx_{T+k} \\ &= \int p(y_{T+k}|x_{T+k})p(x_{T+k}|y_{1:T})dx_{T+k}, \end{aligned} \tag{2}$$

where the last equality follows from conditional independence in a general state space model (inspect the graphical model!). A fact that was often stressed during the course in Bayesian learning is that Bayes is all about averaging functions (here $p(y_{T+k}|x_{T+k})$) with respect to posterior uncertainty (here represented by $p(x_{T+k}|y_{1:T})$): notice that Equation (2) is nothing but

$$\mathrm{E}_{x_{T+k}|y_{1:T}}[p(y_{T+k}|x_{T+k})].$$

To simulate from $p(y_{T+k}|y_{1:T})$ we can sample

$$\begin{aligned} x_{T+k}^{(i)} &\sim p(x_{T+k}|y_{1:T}) \\ y_{T+k}^{(i)} &\sim p(y_{T+k}|x_{T+k}^{(i)}). \end{aligned}$$

We note that the first step requires to simulate the state at $t = T+k$ conditional on $y_{1:T}$. It is easy to show that

$$p(x_{T+k}|y_{1:T}) = \int p(x_{T+k}|x_{T+k-1})p(x_{T+k-1}|y_{1:T})dx_{T+k-1}.$$

Your task is to use simulation and compute 0.95 probability intervals for $x_{T+k}|y_{1:T}$ and $y_{T+k}|y_{1:T}$ for $k = 1, 2, 3, 4, 5$. **Hint:** Start with $k = 1$ and note that $p(x_{T+k-1}|y_{1:T}) = p(x_T|y_{1:T})$ which you can obtain from the Kalman filter. You can then easily simulate $x_{T+k}|y_{1:T}$ recursively, starting from $k = 1$ and, for each k, you can easily obtain a sample from $y_{T+k}|y_{1:T}$ as explained above. This gives you one realization of $y_{T+1}|y_{1:T}, \ldots, y_{T+5}|y_{1:T}$ (5 future observations) which you then repeat for a desired number of samples.

Plot the estimated probability intervals in a time-extended plot that also includes the true state, the filtered state (with probability intervals) and the observations, see Figure 2 for an example. The function `dlmForecast` can be used to check your results.

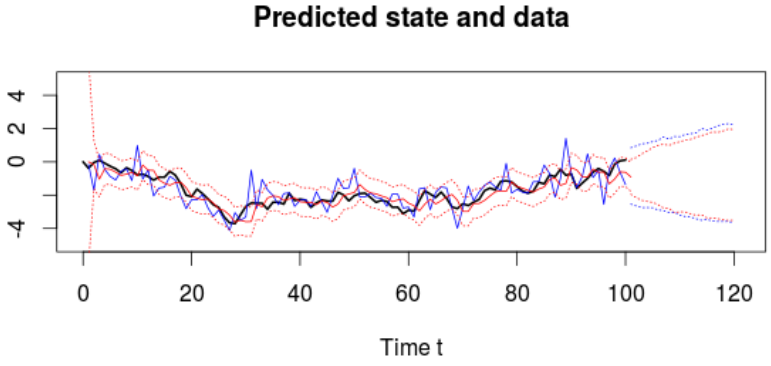

Figure 2: Example of the requested plot (with $k = 20$). True state (black), estimated state (red, filtered for $1 \leq t \leq 100$ and predicted for $t \geq 101$, dotted denotes probability intervals), observations (blue) and probability intervals for predicted distribution $p(y_{T+k}|y_{1:T})$ (dotted blue).

(d) **State and model parameter inference**. Up to now we have assumed that all model parameters are known and the task has been to estimate the state vector. We now also let the model parameters be unknown and estimate them together with the states from data. For simplicity we assume known variances in both the observation and state equation, but we remark that it is in principle straightforward to estimate them. Thus our model is

$$\begin{aligned} &y_t = \theta x_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \Omega_\epsilon = 0.5) \\ &x_t = x_{t-1} + v_t, \quad v_t \sim \mathcal{N}(0, \Omega_v = 0.1)\ (v_t \perp\!\!\!\perp \epsilon_t), \quad x_0 \sim \mathcal{N}(\mu_0 = 0, \Sigma_0 = 10), \end{aligned} \tag{3}$$

and since we are Bayesians we also need a prior for θ which we turn to in a short while. The posterior is now $p(\theta, x_{1:T}|y_{1:T})$ and we can use Gibbs sampling to explore it by simulation as follows. Initialize $\theta^{(0)}$ and, for $i = 1, \ldots, N$, repeat

(i) $x_{1:T}^{(i)} \sim p(x_{1:T}|\theta^{(i-1)}, y_{1:T})$

(ii) $\theta^{(i)} \sim p(\theta|x_{1:T}^{(i)}, y_{1:T})$.

Step (i) can be solved by Forward Filtering and Backward Sampling (FFBS), which in turn requires the Kalman Filter that you implemented earlier. The backward sampling use the mean and variance of $p(x_{t+1}|y_{1:t})$ ($\bar{\mu}_{t+1}$ and $\bar{\Sigma}_{t+1}$) that you already computed with the Kalman Filter and can thus be reused for computational efficiency. Note that you have to run the Kalman Filter (and the backward sampling) for every new θ! Step (ii) is easy to sample by noting that, once we condition on the simulated states from Step (i), the model for y_t in Equation (3) is an ordinary regression model with unknown slope but known variance. A normal prior for θ is conjugate for this model, so that the full conditional $p(\theta|x_{1:T}, y_{1_T})$ is also normal (with updated mean and variance).

Implement a Gibbs sampler that samples from $p(\theta, x_{1:T}|y_{1:T})$ following the instructions above. Take the vague prior $p(\theta) = \mathcal{N}(0, 100)$. Implement your own FFBS and, to check that it is correct, you can compare to `dlmBSample`. **Warning**: For some starting values of θ you can run into numerical problems as the Kalman filter can become numerically which our implementation, as already noticed, does not take into account. Change starting values and try again. I found that this problem can arise with a negative starting value or a very small one. Plot the posterior $p(\theta|y_{1:T})$ using e.g. a histogram or a kernel density estimate (don't forget to discard burn-in). Can you estimate θ accurately (recall that the true value is 0.93)?

2. **Regression with time-varying parameters**. The Capital Asset Pricing Model (CAPM) is useful for determining the risk of an asset. Let $y_t = r_t - r_t^f$, where r_t is the return of the asset of interest at time t and r_t^f is the return of a risk-free asset (e.g. a bond). Likewise, $z_t = r_t^M - r_t^f$ is the excess return at time t but for the market portfolio with return r_t^M. The CAPM model assumes that

$$y_t = \alpha + \beta z_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2).$$

The parameter α captures the excess returns not explained by z_t. The main parameter β measures the sensitivity of the asset w.r.t changes in the market: $\beta > 1$ (< 1) amplifies (dampens) changes in markets return and is considered an aggressive (conservative) investment.

The dynamic CAPM models both the "base"-return and sensitivity as time-varying. We here consider the following dynamic model

$$\begin{aligned} y_t &= \alpha_t + z_t\beta_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2) \\ \alpha_t &= \alpha_{t-1} + v_t^{(1)}, \quad v_t^{(1)} \sim \mathcal{N}(0, \sigma_{v^{(1)}}^2) \quad p(\alpha_0) = \mathcal{N}(0, 100) \\ \beta_t &= \beta_{t-1} + v_t^{(2)}, \quad v_t^{(2)} \sim \mathcal{N}(0, \sigma_{v^{(2)}}^2) \quad p(\beta_0) = \mathcal{N}(1, 0.5) \text{ and } \epsilon_t \perp\!\!\!\perp v_t^{(1)} \perp\!\!\!\perp v_t^{(2)}. \end{aligned}$$

Note that we have a vector of states $x_t = (\alpha_t, \beta_t)$ and, in contrast to the previous problem, the system matrix C of the observation equation is time-dependent (here $C = z_t$).

You will use the data set `CAPM_data.Rda` to estimate the model. The variables of interest in the data set are r_t^f = RKFREE, r_t^M = MARKET and r_t = IBM. The frequency of the data is monthly, from January 1978 to December 1987, a total of $T = 120$ data points. Use the `DLM` package in R to solve the filtering, smoothing and backward sampling problems in this exercise (your code from above cannot have a time-varying system matrix). In the `DLM` package, a time-varying system matrix z_t can be incorporated using the `JFF` input to the `dlm` function, see the

help. As an alternative, the function `dlmModReg` allows you to specify a regression model with time-varying parameters directly. However, **do not** use `dlmModReg`. Learning how to use the `dlm` function with time-varying system matrices is a good exercise as it allows you to treat far more general models.

(a) **Estimating the variance components by MLE.** Find the Maximum Likelihood Estimate (MLE) of $\sigma^2_\epsilon, \sigma^2_{v^{(1)}}, \sigma^2_{v^{(2)}}$ (using the `DLM` package). For problems (b)-(c) you will use the MLE values obtain here.

(b) **Filtering and smoothing**. Compute the filtering and smoothing distribution, respectively. In both cases, plot the result for both α and β (on separate graphs), together with 0.95 probability intervals.

(c) **Average sensitivity pre and post 1982 - a retrospective analysis**. Do a retrospective analysis of the average sensitivity before and after year 1982 (year 1982 starts at $t = 49$). Did the average sensitivity increase after 1982? Motivate your answer. **Hints:** (i) recall that retrospective conditions on all available data and (ii) think like a Bayesian!

(d) **Bayesian inference for the variance components.** Describe in detail how you would do Bayesian inference on the variance components. No need to implement it, but clearly motivate your answer.

Good luck!