Review of Fast Kalman Filtering and implementation using R

Reetam Majumder

Department of Mathematics and Statistics, UMBC

STAT 617 Final Presentation
May 22, 2019

Outline

Original paper

- This is a partial review of 'Fast Kalman filtering and forward-backward smoothing via a low-rank perturbative approach' (Pnevmatikakis *et al.*)
- From the abstract:
  In this paper we note that if a relatively small number of observations are available per time step, the Kalman equations may be approximated in terms of a low-rank perturbation of the prior state covariance matrix in the absence of any observations. In many cases this approximation may be computed and updated very efficiently using fast methods from numerical linear algebra.
- We will look at how this works, and run some R simulations to see whether we can see any performance gain

The state space model

- Let x_t denote our d-dimensional state variable, and the b-dimensional observation y_t at time t
- We assume x_t and y_t satisfy the following linear, Gaussian dynamics and observation equations

$$x_{t+1} = Ax_t + u_t + \epsilon_t, \epsilon_t \sim N_d(0, V) \tag{1}$$

$$y_t = B_t x_t + \eta_t, \eta_t \sim N_b(\mu_t^\eta, W_t) \tag{2}$$

- Initial condition $x_0 \sim N_d(\mu_0, V_0)$
- Without loss of generality, μ_0, u_t and μ_t^η are 0 for all t

The Kalman Filter

- The Kalman Filter recursion computes the forward mean $\mu_t = \mathbb{E}(x_t | Y_{1:t})$ ($\mathbf{x}_t^t$ in our class notation) and $C_t = Cov(x_t | Y_{1:t})$ ($\mathbf{P}_t^t$ in our class notation)
- The Kalman recursions can be written as

$$C_t = (P_t^{-1} + B_t^T W_t^{-1} B_t)^{-1} \tag{3}$$

$$\mu_t = A\mu_{t-1} + P_t B_t^T (W_t + B_t P_t B_t^T)^{-1}(y_t - B_t A \mu_{t-1}) \tag{4}$$

$$with\ P_t = Cov(x_t | Y_{1:t-1}) = AC_{t-1}A^T + V \tag{5}$$

- Computing the inverses in the recursion for C_t requires $O(d^3)$ time in general, or $O(d^2)$ if the observation matrix is of low rank, i.e. $\rho(B_t) \ll d$
- In either case, $O(d^2)$ space is required to store C_t

Other assumptions and simplifications

- A key quantity is the prior covariance $C_{0,t}$, i.e. the covariance of x_t in the absence of any observations. Setting $B_t = B = 0$ in the Kalman filter recursion equations, we get

$$C_{0,t} = AC_{0,t-1}A^T + V \tag{6}$$

- If we assume that the state matrix A has a spectral norm less than 1 (stability), $C_{0,t}$ converges to the equilibrium prior covariance $C_0 = lim_{t \to \infty} C_{0,t}$
- This would then give us $AC_0A^T + V = C_0$
- If further we assume that A is normal $(AA^T = A^TA)$ and commutes with V, C_0 can be explicitly computed as

$$C_0 = V(I - AA^T)^{-1} \tag{7}$$

Fast Kalman Filtering

- The main idea is that when $\rho(B_t) \ll d$, C_t should be close to $C_{0,t}$. We approximate C_t as

$$C_t \approx \tilde{C}_t = C_{0,t} - L_t \Sigma_t L_t^T \tag{8}$$

where $L_t \Sigma_t L_t^T$ is a low-rank matrix that can be update directly (shown in the algorithm)

Fast Kalman Filtering

Algorithm 1 Fast Kalman filtering algorithm

$L_1 = C_{0,1}B_1^T, \quad \Sigma_1 = (W_1 + B_1 C_{0,1} B_1^T)^{-1}$ (cost $O(b_1^3 + b_1 K(d))$)

$\tilde{C}_1 = C_{0,1} - L_1 \Sigma_1 L_1^T$

$\tilde{\mu}_1 = L_1 \Sigma_1^{-1} y_1$

for $t = 2$ to T **do**

$C_{0,t} = A C_{0,t-1} A^T + V$

$\Phi_t = C_{0,t}^{-1} A L_{t-1}, \quad \Delta_t = (\Sigma_{t-1}^{-1} - L_{t-1}^T A^T C_{0,t}^{-1} A L_{t-1})^{-1}$ (cost $O(k_{t-1}^3 + k_{t-1} K(d))$)

$O_t = [\Phi_t \;\; B_t], \quad Q_t = \text{blkdiag}\{\Delta_t, W_t^{-1}\}$

$[\hat{L}_t, \hat{\Sigma}_t^{1/2}] = \text{svd}(C_{0,t} O_t (Q_t^{-1} + O_t^T C_{0,t} O_t)^{-1/2})$ (cost $O((b_t + k_{t-1})^2 d)$)

Truncate $\hat{L}_t$ and $\hat{\Sigma}_t$ to L_t and Σ_t. (effective rank $k_t \leq b_t + k_{t-1} \ll d$)

$\tilde{C}_t = C_{0,t} - L_t \Sigma_t L_t^T$

$\tilde{P}_t = C_{0,t} - A L_{t-1} \Sigma_{t-1} L_{t-1}^T A^T$ (cost $O(k_{t-1} K(d))$)

$\tilde{\mu}_t = A\tilde{\mu}_{t-1} + \tilde{P}_t B_t^T (W_t + B_t \tilde{P}_t B_t^T)^{-1} (y_t - B_t A \tilde{\mu}_{t-1})$ (cost $O(b_t^3 + b_t K(d))$)

Estimating the low rank perturbations from C_0

- L_t and Σ_t are obtained at each step by truncating a partial SVD

$$[\hat{L}_t, \hat{\Sigma}_t^{\frac{1}{2}}] = svd(C_{0,t}O_t(Q_t^{-1} + O_t^T C_{0,t} O^t)^{-\frac{1}{2}} \tag{9}$$

- L_t is chosen to be the first k_t columns of $\hat{L}_t$ and Σ_t as the first k_t diagonal elements of $\hat{\Sigma}_t$, where k_t is chosen to be large enough for accuracy and small enough for computational tractability.
- A reasonable choice is as the least solution of the inequality

$$\sum_{i \leq k_t} [\hat{\Sigma}_t]_{ii} \geq \theta \sum_i [\hat{\Sigma}_t]_{ii} \tag{10}$$

i.e. choose k_t to capture at least a large fraction θ of the term $\hat{L}_t \hat{\Sigma}_t^{\frac{1}{2}}$, the square root of the term perturbing $C_{0,t}$

When does this work well?

- If A or its inverse is banded/tree-banded, in the sense that $A_{ij} \neq 0$ only if i and j are neighbors on a tree, because then so is C_0^{-1}
- If A is defined in terms of a partial differential operator. A in these cases are typically sparse and has a specialized local structure
- If A has a Toeplitz (or block-Toeplitz) structure; e.g in whenever the state variable x_t has a spatial structure and the dynamics are spatially-invariant in some sense

This list is not exhaustive by any means

Simulation study

- I used a very simple case to test how this works
- While the parameters chosen are mentioned in the paper, they were originally used for a different purpose
    - T = 500 (total sample size)
    - d = 50, 100, 250, 1000 (dimension of the state matrix)
    - b = 1 (univariate y_t)
    - $A = 0.95 I_d$ (system dynamics matrix)
    - $V = 0.1 I_d$ (covariace matrix for the state)
    - $W = Var(y_t) = 0.5$
    - $B = (1, 1, ..., 1)_{b \times d}$ (observation gain matrix; observation is just sum of the states)
    - $V_0 = \sum_i^\infty A^i V {A^i}^T \approx 0.9256 I_d$

Simulation study

- Data was generated using these parameters, and A, B, V, W, V_0, y_t are fed back into the KF and FKF algorithms
- Since A is assumed to be stable, $C_{0,t} = C_0$. This simplified writing the code since we had one iterative variable less
- Further, b = 1 leads to another simplification, since this makes Σ_t potentially a scalar. This is most obviously seen in the first expression

$$\Sigma_1 = (W_1 + B_1 C_0 B_1^T)^{-1} \tag{11}$$

- While in the svd step Σ_t can be at most 2×2, I have kept Σ_t scalar

Run time for KF and FKF

Table: Time taken in seconds to run KF and FKF for different state dimensions

Dimension	KF	FKF
$d = 50$	0.12	2.31
$d = 100$	0.52	2.34
$d = 250$	6.41	3.00
$d = 1000$	542.44	14.59

- Times were recorded in R using `proc.time()`

Estimation error for FKF

- We calculated the RMSE at each time point, and plotted the results

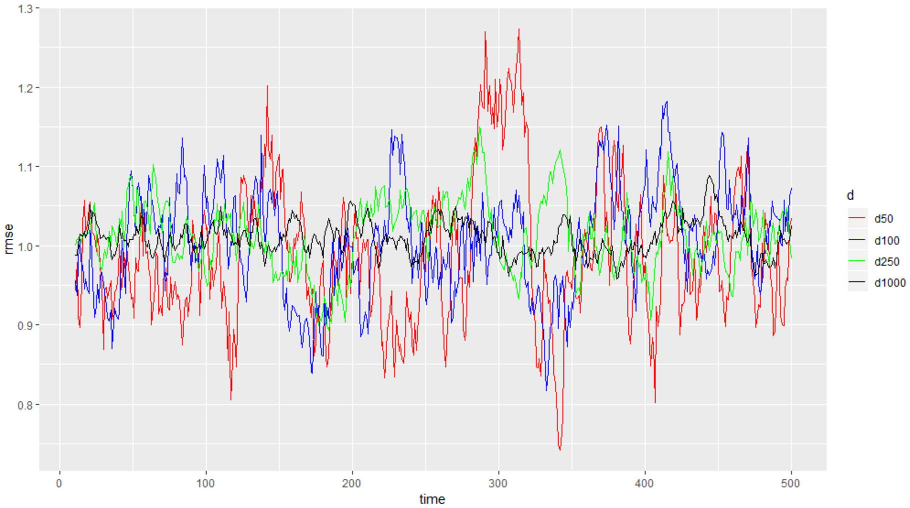
$$RMSE_t = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(X_{t,i} - \hat{X}_{t,i})^2} \tag{12}$$

- For small d, we get less bias, but also the most variability.
- Larger d has the smallest variability; I don't know if this is a feature of the choice of parameters, and/or the choice of k_t

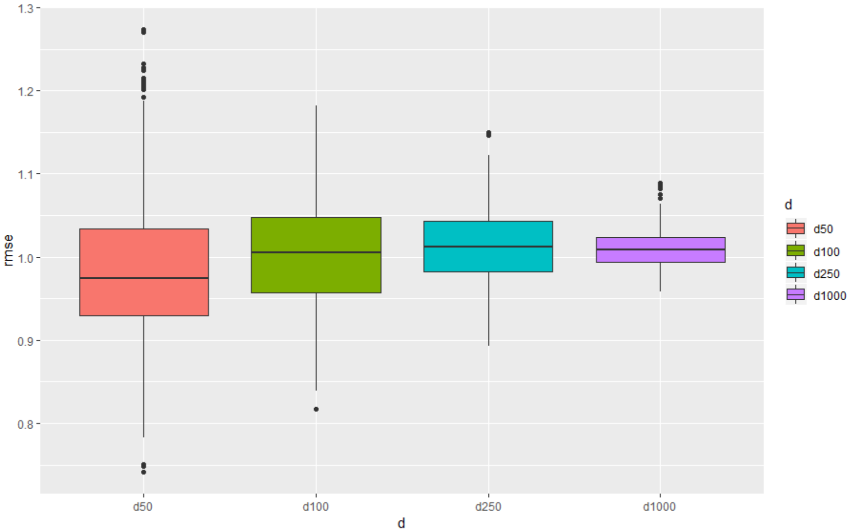
Table: RMSE for full data as well as truncated data (missing first 3 values)

Dimension	Full Data	Truncated data
d = 50	1355.641	490.6434
d = 100	2084.275	499.4935
d = 250	4287.014	503.0260
d = 1000	1877.39	501.5597

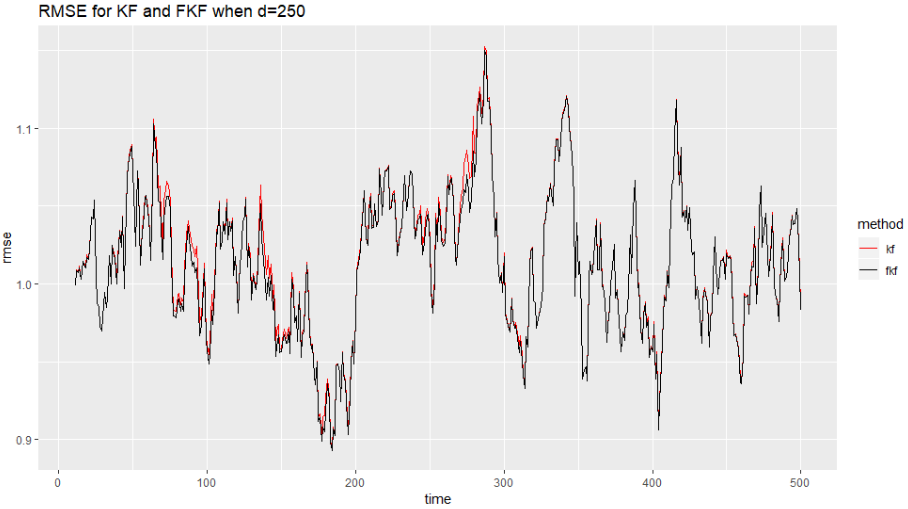
Estimation error for FKF



Estimation error for FKF



Comparison of RMSE for KF and FKF



Extension to fast Kalman smoothing

Algorithm 3 Low-Rank Block-Thomas Algorithm

$\tilde{D}_1 = D_1, L_1 = D_1^{-1}B_1^T$ (cost $O(b_1 d)$, $k_1 = b_1$)

$\Sigma_1 = (W_1 + B_1 D_1^{-1} B_1^T)^{-1}$ (cost $O(b_1^3)$)

$\tilde{\boldsymbol{q}}_1 = (-D_1^{-1} + L_1 \Sigma_1 L_1^T)\nabla_1 \quad (= -\tilde{M}_1^{-1}\nabla_1)$ (cost $O(b_1 K(d))$)

for $t = 2$ to T **do**

  $\tilde{D}_t = D_t - E_{t-1}\tilde{D}_{t-1}^{-1}E_{t-1}^T$

  $O_t = [B_t^T \quad E_{t-1}L_{t-1}], \quad Q_t = \text{blkdiag}\{W_t^{-1}, \Sigma_{t-1}\}$

  $[\hat{L}_t, \hat{\Sigma}_t^{1/2}] = \text{svd}(\tilde{D}_t^{-1}O_t(Q_t^{-1} + O_t^T\tilde{D}_t^{-1}O_t)^{-1/2})$ (cost $O((b_t + k_{t-1})^2 K(d))$)

  Truncate $\hat{L}_t$ and $\hat{\Sigma}_t$ to L_t and Σ_t. (effective rank $k_t \le b_t + k_{t-1} \ll d$)

  $\tilde{\boldsymbol{q}}_t = -(\tilde{D}_t^{-1} - L_t\Sigma_t L_t^T)(\nabla_t - E_{t-1}^T\tilde{\boldsymbol{q}}_{t-1}) \quad (= -\tilde{M}_t^{-1}(\nabla_t - E_{t-1}^T\tilde{\boldsymbol{q}}_{t-1}))$ (cost $O(k_t K(d))$)

$\tilde{\mathbf{s}}_T = \tilde{\boldsymbol{q}}_T$

for $i = T-1$ to 1 **do**

  $\tilde{\mathbf{s}}_t = \tilde{\boldsymbol{q}}_t + (\tilde{D}_t^{-1}E_t^T - L_t\Sigma_t L_t^T E_t^T)\tilde{\mathbf{s}}_{t+1} \quad (= \tilde{\boldsymbol{q}}_t + \tilde{\Gamma}_t\tilde{\mathbf{s}}_{t+1})$ (cost $O(k_t K(d))$)

Discussion

- Both these methods work well for a 'few-observations' setting
- There is also an asumption made of a low SNR ratio. In case of high SNR, $C_0 \approx 0$, i.e. perturbations happen around the 0 matrix
- The fast low-rank methods can also facilitate hyperparameter selection in the smoothing setting
- The assumption of normality of A might not hold; standard direct methods for dealing with this involve an orthogonalization step that takes $O(d^3)$ time in general
- Finally, if we have $C_{0,t}$ instead of C_0, the methods demonstrated can be applied when $C_{0,t}$ can be updated efficiently and used for fast matrix-vector operations