Joint Sampling of States and Parameters in State Space Models

William McCausland and Samuel Gingras

2023-08-30

Models under consideration

- State equation: x_t is univariate stationary Markov with

$$(x_{t+1} - \mu) = \phi(x_t - \mu) + \epsilon_t, \quad \epsilon_t \sim N(0, \omega^{-1}),$$

  where $\theta = (\log \omega, \tanh^{-1}(\phi), \mu)$ is the vector of parameters.

- Model for observed $y \equiv (y_1, \ldots, y_n)$ given $x \equiv (x_1, \ldots, x_n)$:

$$p(y|x, \theta, \theta_y) = \prod_{t=1}^{n} p(y_t|y_{1:t-1}, x_t, \theta_y),$$

  where θ_y is a vector of parameters for $y|x$.

- Observation model is flexible: y_t can be
  - univariate, multivariate, or of time-varying dimension,
  - discrete, continuous, or mixed.

- This model for (x, y) can be embedded in a larger model.

Comments about posterior simulation

Typically,

- the posterior autocorrelation of x_t is strong,
- the posterior dependence between x and θ is strong,
- the posterior dependence between x and θ_y is not as strong.

For high numerical and computational efficiency, want to

- draw (θ, x) as a block,
- do computations that are $O(n)$ in C, leaving rest to Matlab.

When (x, y) is in a larger model, $\theta, x|y, \theta_y, \ldots$ is a moving target.

HESSIAN method, McCausland (2012)

- Highly Efficient Simulation Smoothing, In A Nutshell.
- An approximation $q(x|\theta, y)$ of $p(x|\theta, y)$ based on Hessian of log smoothing density $\log p(x|\theta, y)$.
- Not model specific.
- Need code to evaluate five derivatives with respect to x_t of

$$\psi(x_t; \theta, y_{1:t}) \equiv \log p(y_t|x_t, \theta).$$

- HESSIAN method steps:
  1. Find mode x° of target distribution $x|\theta, y$.
  2. Forward pass to compute approximations of
     a. mode of $x_t|x_{t+1}, y, \theta$ as a polynomial in x_{t+1},
     b. five derivatives of $\log p(x_t|x_{t+1}, y, \theta)$ wrt x_t at $(x_t^\circ, x_{t+1}^\circ)$.
  3. Backward pass to
     a. draw non-Gaussian proposal $x_t^*|x_{t+1}^*, \theta, y$,
     b. evaluate proposal density $q(x_t^*|x_{t+1}^*, \theta, y)$.

Examples where derivatives are easy

Code for derivatives of $\psi(x_t) \equiv \log p(y_t|x_t, \theta)$, Gaussian SV model:

```
void derivative(double y_t, double x_t, double *psi_t)
{
  psi_t[4] = psi_t[2] = -0.5 * (y_t * y_t) * exp(-x_t);
  psi_t[1] = -psi_t[2] - 0.5;
  psi_t[5] = psi_t[3] = -psi_t[2];
}
```

Also quite straightforward:

1. Student's t stochastic volatility model
2. Poisson and gamma-Poisson state space model
3. Exponential, gamma, generalized gamma, Weibull state space models

Using Faà di Bruno I

- Important point: need to evaluate derivatives, not (necessarily) perform symbolic differentiation.
- Faà di Bruno's formula gives all derivatives of $(f \circ g)(x)$, for univariate f and g.
- For example, fourth derivative of $(f \circ g)(x)$:

$$\begin{aligned}(f \circ g)''''(x) = {} & f''''(g(x))g'(x)^4 + 6f'''(g(x))g''(x)g'(x)^2 \\ & + 3f''(g(x))g''(x)^2 + 4f''(g(x))g'''(x)g'(x) \\ & + f'(g(x))g''''(x).\end{aligned}$$

Using Faà di Bruno II: computation

Code to compute up to five derivatives of $(f \circ g)(x)$:

```
void compute_Faa_di_Bruno(int n_derivs, double *f, double *g, double *fg)
{
    fg[0] = f[0];
    if( n_derivs >= 1) {
        fg[1] = f[1]*g[1];
        if( n_derivs >= 2 ) {
            double g1_2 = g[1]*g[1];
            fg[2] = f[1]*g[2] + f[2]*g1_2;
            if( n_derivs >= 3 ) {
                double g1_3 = g1_2*g[1];
                fg[3] = f[1]*g[3] + 3*f[2]*g[1]*g[2] + f[3]*g1_3;
                if( n_derivs >= 4 ) {
                    double g2_2 = g[2]*g[2];
                    double g1_4 = g1_3*g[1];
                    fg[4] = f[1]*g[4] + 4*f[2]*g[1]*g[3]
                        + 3*f[2]*g2_2 + 6*f[3]*g1_2*g[2] + f[4]*g1_4;
                    if( n_derivs >= 5 ) {
                        double g1_5 = g1_4*g[1];
                        fg[5] = f[1]*g[5] + 5*f[2]*g[1]*g[4] + 10*f[2]*g[2]*g[3]
                            + 15*f[3]*g2_2*g[1] + 10*f[3]*g[3]*g1_2 + 10*f[4]*g[2]*g1_3
                            + f[5]*g1_5;
                    }
                }
            }
        }
    }
}
```

Using Faà di Bruno III: Burr state space model

```
// Step 1: Direct computation h(x) = (lambda*y*exp(-x))^eta
h[0] = pow( lambda * y_t * exp(-x_t), eta );
h[1] = -h[0]*eta;  h[2] = -h[1]*eta;  h[3] = -h[2]*eta;
h[4] = -h[3]*eta;  h[5] = -h[4]*eta;

// Step 2a: Faa Di Bruno g(x) = q(h(x)), q(x) = log(1+x);
double z = 1+h[0];    double z_inv = 1/z;
q[0] = log(z);        q[1] = z_inv;
q[2] = q[1] * z_inv * (-1.0);
q[3] = q[2] * z_inv * (-2.0);
q[4] = q[3] * z_inv * (-3.0);
q[5] = q[4] * z_inv * (-4.0);
compute_Faa_di_Bruno(5, q, h, g);

// Step 2b: Direct computation of psi(x) = -eta*x - (kappa+
psi_t[1] = -(kappa+1) * g[1] - eta;
for (i=2; i<6; i++)
    psi_t[i] = -(kappa+1) * g[i];
```

Using Faà di Bruno IV: other examples

Models with two applications of Faà di Bruno:

1. Mixture of exponentials state space model
2. Mixture of gammas state space model
3. Mixture of Gaussians stochastic volatility model.

No need for data augmentation.

Joint draw (this paper)

- High level description of joint draw:
  1. Find shape of log likelihood $\log p(\theta|y)$ (x is integrated out).
  2. Draw $\theta^*|y$.
  3. Draw $x^*|\theta^*, y$ using HESSIAN method.
  4. Joint accept/reject of (θ^*, x^*).
- Steps 1 and 2 are the innovations of this paper.
- Will now outline how to approximate gradient and Hessian of $\log p(\theta|y)$, most of Step 1.

Outline 1

Analytic expression for gradient and Hessian of $\log f(x|\theta)$ with respect to $\theta \equiv (\log \omega, tanh^{-1}, \mu)$:

$$g_{x|\theta}(\theta) = \begin{bmatrix} \frac{n}{2} - \frac{\omega}{2} e^\top Q e \\ -\phi - \frac{\omega}{2} e^\top Q' e \\ \omega q e \end{bmatrix},$$

$$H_{x|\theta}(\theta) = \begin{bmatrix} -\frac{\omega}{2} e^\top Q e & -\frac{\omega}{2} e^\top Q' e & \omega q e \\ -\frac{\omega}{2} e^\top Q' e & -(1-\phi^2) - \frac{\omega}{2} e^\top Q'' e & \omega q' e \\ \omega q e & \omega q' e & -\omega q \iota \end{bmatrix},$$

where Q, Q' and Q'' are $n \times n$ tridiagonal, $e = x - \mu$.

Outline 2

The gradient and Hessian of the log-likelihood $\log p(y|\theta)$ are

$$g_{y|\theta}(\theta) = E_{x|\theta,y}[g_{x|\theta}(\theta)],$$

$$H_{y|\theta}(\theta) = E_{x|\theta,y}[H_{x|\theta}(\theta)] + \mathrm{Var}_{x|\theta,y}[g_{x|\theta}(\theta)].$$

To compute approximations of these in $O(n)$ time we exploit:

1. Elements of $g_{x|\theta}(\theta)$, $H_{x|\theta}(\theta)$ are tridiagonal quadratic forms.
2. Apply laws of iterated expectations, total covariance to break down expectations and covariances into sequential moments.
3. $p(x|\theta, y)$ and its approximation $q(x|\theta, y)$ are Markov.
4. Use polynomial approximations of $\log q(x_t|x_{t+1}, \theta, y)$ that are by-products of the HESSIAN method state draw.

Application

Data, from ECB:

- 23 currencies against the Euro
- $n = 3139$ daily returns, per currency
- Used in Kastner and Frühwirth-Schnatter (2014) (ASIS paper).

Model:

- Basic Gaussian SV model, no leverage
- Same prior as KFS (2014)

Simulation:

- $M = 40000$ posterior draws, 5 burn-in draws.
- Acceptance probabilities from 0.51 to 0.80.
- Longest rejection runs 7 to 19.

Efficiency in KFS (2014) better than Gibbs with conditional draws

- $\theta|x, y$,
- $x|\theta, y$.

Efficiency I

Curr	$E[\sigma\|y]$	$\mathrm{sd}[\sigma\|y]$	rne	mult	$E[\phi\|y]$	$\mathrm{sd}[\phi\|y]$	rne	mult
AUD	0.166	0.023	0.50	48.9	0.976	0.007	0.44	29.8
CAD	0.085	0.018	0.32	38.9	0.987	0.006	0.25	22.0
CHF	0.209	0.020	0.51	37.6	0.985	0.004	0.31	10.1
CZK	0.277	0.034	0.42	40.5	0.953	0.011	0.44	31.4
DKK	0.429	0.042	0.23	16.8	0.902	0.018	0.27	15.5
GBP	0.101	0.013	0.44	38.6	0.991	0.003	0.22	8.7
HKD	0.066	0.011	0.36	27.2	0.993	0.003	0.16	5.9
IDR	0.230	0.035	0.35	49.1	0.966	0.010	0.34	38.3
JPY	0.119	0.016	0.49	44.2	0.989	0.004	0.28	13.3
KRW	0.140	0.017	0.54	42.5	0.987	0.004	0.30	12.2
MXN	0.163	0.021	0.57	48.9	0.977	0.007	0.43	25.9
MYR	0.079	0.014	0.30	27.2	0.990	0.004	0.20	12.0
NDK	0.175	0.022	0.55	41.9	0.970	0.008	0.44	23.1
NZD	0.174	0.031	0.26	35.4	0.963	0.012	0.28	31.4
PHP	0.148	0.023	0.44	70.3	0.975	0.008	0.42	51.4
PLN	0.190	0.021	0.55	38.2	0.975	0.006	0.42	18.3

Efficiency II

Curr	$E[\sigma\|y]$	$\mathrm{sd}[\sigma\|y]$	rne	mult	$E[\phi\|y]$	$\mathrm{sd}[\phi\|y]$	rne	mult
RON	0.309	0.026	0.51	30.8	0.970	0.006	0.43	13.8
RUB	0.149	0.017	0.54	44.4	0.988	0.004	0.27	10.1
SEK	0.111	0.012	0.46	27.6	0.991	0.003	0.21	4.9
SGD	0.068	0.011	0.28	28.3	0.995	0.003	0.12	5.5
THB	0.125	0.019	0.44	39.3	0.981	0.007	0.31	19.7
TRY	0.312	0.025	0.61	42.2	0.959	0.008	0.55	23.3
USD	0.066	0.010	0.36	26.4	0.993	0.003	0.18	6.6

Final notes

1. Have left out of this presentation how to use likelihood gradient and Hessian for drawing $\theta|y$.
2. Public Github site with Matlab code: samuelgingras/mhessian
    a. 12 basic state space models,
    b. getting it right code (Geweke, 2004) to test correctness of code,
    c. some posterior simulation examples.
3. We have other models on a private Github site to be made public as papers get published.