Estimation for diffusion processes

Yuta Koike

Mathematics and Informatics Center, The University of Tokyo
CREST JST

June 26, 2019

YSS2019

1. Setting
2. Parameter estimation
   - Quasi-log likelihood function
   - QMLE by `qmle()`
   - Adaptive Bayes Estimation by `adaBayes()`
   - Multi-dimensional process
3. Real data example
4. Effect of sampling schemes

Setting

- $B = (B_t)_{t \in [0,T]}$: Brownian motion
- We consider a diffusion process containing unknown parameters:

$$dX_t = a(X_t, \alpha)dt + b(X_t, \beta)dB_t, \quad X_0 = x_0, \qquad t \in [0, T]. \tag{1}$$

- α and β are unknown parameters (possibly multi-dimensional)
- The process $X = (X_t)_{t \in [0,T]}$ is observed at equi-spaced time points $t_i = i\Delta_n$ ($i = 0, 1, \dots, n$; $\Delta_n = T/n$)
- $\boxed{\text{Aim}}$ Estimate the parameters α and β from the observation data $X_{t_0}, X_{t_1}, \dots, X_{t_n}$
- YUIMA has two basic functions to accomplish this:
    - `qmle()`: Quasi-Maximum Likelihood Estimation (QMLE)
    - `adaBayes()`: Adaptive Bayes Estimation

Quasi-log likelihood function

- Euler-Maruyama type approximation:

$$\Delta X_i := X_{t_i} - X_{t_{i-1}} \approx a(X_{t_{i-1}}, \alpha)\Delta_n + b(X_{t_{i-1}}, \beta)(B_{t_i} - B_{t_{i-1}})$$

- Thus, the conditional pdf of ΔX_i given X_t $(0 \leq t \leq t_{i-1})$ is approximated by the normal density with mean $a(X_{t_{i-1}}, \alpha)\Delta_n$ and variance $b(X_{t_{i-1}}, \beta)^2\Delta_n$:

$$\frac{1}{\sqrt{2\pi b(X_{t_{i-1}}, \beta)^2\Delta_n}} \exp\left(-\frac{(\Delta X_i - \Delta_n a(X_{t_{i-1}}, \alpha))^2}{2b(X_{t_{i-1}}, \beta)^2\Delta_n}\right)$$

Quasi-log likelihood function

- The corresponding quasi-log likelihood function:

$$\mathbb{H}_n(\alpha,\beta) = -\frac{1}{2}\sum_{i=1}^{n}\left\{\log b(X_{t_{i-1}},\beta)^2 + \frac{(\Delta X_i - \Delta_n a(X_{t_{i-1}},\alpha))^2}{b(X_{t_{i-1}},\beta)^2\Delta_n}\right\}$$

- Using $\mathbb{H}_n(\alpha,\beta)$ as a standard log-likelihood function, we can implement ML type and Bayesian type estimation
- In the following, Θ_α and Θ_β denotes the parameter spaces for α and β, respectively

QMLE by `qmle()`

- The function `qmle()` computes the joint QMLE for α and β (when the option `joint = TRUE`):

$$(\hat{\alpha}_n, \hat{\beta}_n) = \arg \max_{(\alpha,\beta)\in\Theta_\alpha\times\Theta_\beta} \mathbb{H}_n(\alpha, \beta)$$

  - The optimization problem is solved by the function `optim()`; we need to set an initial value of the optimization to the option `start`
  - Currently, only hyperrectangles are supported for the parameter spaces Θ_α and Θ_β; they are set by the options `lower` and `upper`
  - The standard errors and asymptotic covariance matrix for the estimators are also available; `summary()` and `vcov()`

QMLE by `qmle()`

- When the option `joint = FALSE` (default), it computes the two stage QMLE:
  1. Given an initial value α^* for α, we estimate β by
  $$\check{\beta}_n = \arg\max_{\beta \in \Theta_\beta} \mathbb{H}_n(\alpha^*, \beta)$$
  2. We estimate α by
  $$\check{\alpha}_n = \arg\max_{\alpha \in \Theta_\alpha} \mathbb{H}_n(\alpha, \check{\beta}_n)$$
- The two-stage QMLE has the same asymptotic property as the standard QMLE and its computation is usually faster
- Of course, their finite sample performance could be different

QMLE by qmle()

- Summary: Basic formula for qmle():

```
qmle(yuima, start, method = "BFGS", lower, upper, joint
    = FALSE, rcpp = FALSE)
```

  - yuima: a yuima object
  - start: initial parameter values for optimization
  - method: optimization method used in optim()
  - lower: lower values for the parameter spaces
  - upper: upper values for the parameter spaces
  - joint: joint or two-stage?
  - rcpp: use C++ code or not?

QMLE by `qmle()`: An example

- Let us estimate the following SDE model:

$$dX_t = (-\alpha_1 X_t + \alpha_2)dt + \beta dB_t, \quad x_0 = 0.3$$

  - Known as the **Ornstein-Uhlenbeck (OU) process**
  - The true parameter values: $\alpha_1 = 3$, $\alpha_2 = 1$, $\beta = 0.3$
  - The parameters for the sampling schemes: $n = 1000$ and $T = 3n^{1/3}$
- **R example**: `qmle-ex.r`

Adaptive Bayes Estimation by `adaBayes()`

- The function `adaBayes()` computes the adaptive Bayes-type estimator for α and β as follows:
  1. Given an initial value α^* for α, we estimate β by

$$\tilde{\beta}_n = \frac{\int_{\Theta_\beta} \beta \exp(\mathbb{H}_n(\alpha^*, \beta))\pi_1(\beta)d\beta}{\int_{\Theta_\beta} \exp(\mathbb{H}_n(\alpha^*, \beta))\pi_1(\beta)d\beta}$$

  2. We estimate α by

$$\tilde{\alpha}_n = \frac{\int_{\Theta_\alpha} \alpha \exp(\mathbb{H}_n(\alpha, \tilde{\beta}_n))\pi_2(\alpha)d\alpha}{\int_{\Theta_\alpha} \exp(\mathbb{H}_n(\alpha, \tilde{\beta}_n))\pi_2(\alpha)d\alpha}$$

  - π_1 and π_2 are prior densities for β and α, respectively (they can be improper)

- Basic formula for adaBayes():

```
adaBayes(yuima, start, prior, lower, upper, method = "
   mcmc", mcmc = 1000, rcpp = FALSE, algorithm = "
   randomwalk")
```

  - yuima: a yuima object
  - start: initial parameter values for optimization
  - prior: prior densities
  - lower: lower values for the parameter spaces
  - upper: upper values for the parameter spaces
  - method: How to compute the integrals? ("mcmc" for MCMC and "nomcmc" for numerical integration by the package cubature)
  - mcmc: number of MCMC iterations
  - rcpp: use C++ code or not?
  - algorithm: MCMC algorithm: "randomwalk" and "MpCN" are available

- **R example**: adaBayes-ex.r

Multi-dimensional process

- Multi-dimensional SDEs can be handled analogously
- As an illustration, we estimate the unknown parameters of the following two-dimensional SDE:

$$\begin{cases} dX_{1,t} &= -\alpha_1 X_{1,t}dt + \beta_1 dB_{1,t} + X_{2,t}dB_{3,t}, \\ dX_{2,t} &= -(\alpha_2 X_{1,t} + \alpha_3 X_{2,t})dt + X_{1,t}dB_{1,t} + \beta_2 dB_{2,t}, \end{cases} \tag{2}$$

  where $(B_{1,t})_{t\in[0,T]}$, $(B_{2,t})_{t\in[0,T]}$, $(B_{3,t})_{t\in[0,T]}$ are three independent Brownian motions
- **R example**: `est-multi.r`

Real data example

- The functions `qmle()` and `adaBayes()` can be used to fit a SDE model to real data
- As an illustration, we fit the OU model to the dataset `LogSPX` contained in the package `yuima`
- **R example**: `est-spx.r`

Effect of sampling schemes

- The convergence rates of QMLEs/adaptive Bayes estimators for α and β are given by $1/\sqrt{T}$ and $\sqrt{\Delta_n}$
- Thus, regarding the sampling scheme, the estimation accuracy of α is affected by T and the effect of Δ_n is not large, at least asymptotically
- The roles of T and Δ_n are reversed for the estimation accuracy of β
- We check this effect in the previous OU model example
- **R example**: `est-sampling.r`