Gaussian Variational Approximate Inference and Monte Carlo EM Algorithm for Generalized Linear Mixed Models

STAT 540 Project Presentation

Qianying Zhou

11/29/2018

College of Information Sciences and Technology

Generalized Linear Mixed Models (GLMM)

- A generalized linear mixed model is an extension of the generalized linear model in which the linear predictor contains random effects in addition to the usual fixed effects.
- GLMMs are widely applied to the analysis of grouped data, since the differences among groups (from different distributions) can be modelled as random effects.
- The general form of the model is:

$$y = X\beta + Z\mu + \epsilon,\ \mu \sim N(0, G)$$

- Fitting GLMMs via maximum likelihood involves integrating over the random effects. In general, those integrals cannot be expressed in analytical forms.

GLMM

- Consider the exponential family models of the form:

$$\mathbf{y}|\mathbf{u} \sim \exp\{\mathbf{y}^T(\mathbf{X}\boldsymbol{\beta}+\mathbf{Z}\mathbf{u}) - \mathbf{1}^T b(\mathbf{X}\boldsymbol{\beta}+\mathbf{Z}\mathbf{u}) + \mathbf{1}^T c(\mathbf{y})\}, \qquad \mathbf{u} \sim N(\mathbf{0}, \mathbf{G}),$$

- The parameters in the exponential family models are the fixed effects vector \beta and the random effects covariance matrix G. Their loglikelihood is:

$$\ell(\boldsymbol{\beta}, \boldsymbol{\Sigma}) = \sum_{i=1}^{m} \{\mathbf{y}_i^T \mathbf{X}_i \boldsymbol{\beta} + \mathbf{1}_i^T c(\mathbf{y}_i)\} - \frac{m}{2}\log|\boldsymbol{\Sigma}| - \frac{mK}{2}\log(2\pi)$$

$$+ \sum_{i=1}^{m} \log \int_{\mathbb{R}^K} \exp\left\{\mathbf{y}_i^T \mathbf{Z}_i \mathbf{u} - \mathbf{1}_i^T b(\mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\mathbf{u}) - \frac{1}{2}\mathbf{u}^T \boldsymbol{\Sigma}^{-1}\mathbf{u}\right\} d\mathbf{u}$$

- The K-dimensional integral in the loglikelihood cannot be solved analytically

Gaussian Variational Approximate (GVA)

- GVA introduces a pair of variational parameters μ_i, Λ_i. By Jensen's inequality and concavity of the logarithm function, we can get the lower bound:

$$
\begin{aligned}
\ell(\boldsymbol{\beta}, \boldsymbol{\Sigma}) &= \sum_{i=1}^{m} \{\mathbf{y}_i^T \mathbf{X}_i \boldsymbol{\beta} + \mathbf{1}_i^T c(\mathbf{y}_i)\} - \frac{m}{2} \log |\boldsymbol{\Sigma}| - \frac{mK}{2} \log(2\pi) \\
&\quad + \sum_{i=1}^{m} \log \int_{\mathbb{R}^K} \exp\left\{ \mathbf{y}_i^T \mathbf{Z}_i \mathbf{u} - \mathbf{1}_i^T b(\mathbf{X}_i \boldsymbol{\beta} + \mathbf{Z}_i \mathbf{u}) - \frac{1}{2} \mathbf{u}^T \boldsymbol{\Sigma}^{-1} \mathbf{u} \right\} \frac{\phi_{\boldsymbol{\Lambda}_i}(\mathbf{u} - \boldsymbol{\mu}_i)}{\phi_{\boldsymbol{\Lambda}_i}(\mathbf{u} - \boldsymbol{\mu}_i)} d\mathbf{u} \\
&\geq \sum_{i=1}^{m} \{\mathbf{y}_i^T \mathbf{X}_i \boldsymbol{\beta} + \mathbf{1}_i^T c(\mathbf{y}_i)\} - \frac{m}{2} \log |\boldsymbol{\Sigma}| - \frac{mK}{2} \log(2\pi) \\
&\quad + \sum_{i=1}^{m} E_{\mathbf{u} \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i)} \Big(\mathbf{y}_i^T \mathbf{Z}_i \mathbf{u} - \mathbf{1}_i^T b(\mathbf{X}_i \boldsymbol{\beta} + \mathbf{Z}_i \mathbf{u}) \\
&\quad - \frac{1}{2} \mathbf{u}^T \boldsymbol{\Sigma}^{-1} \mathbf{u} - \log(\phi_{\boldsymbol{\Lambda}_i}(\mathbf{u} - \boldsymbol{\mu}_i)) \Big) \\
&\equiv \underline{\ell}(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \boldsymbol{\mu}, \boldsymbol{\Lambda}),
\end{aligned}
$$

GVA

- The advantage of the lower-bound is that it no longer involves the integrals of size K. Hence, the computational speed is improved.
- We can use Newton-Raphson scheme to get the Gaussian variational approximate maximum likelihood estimators.

Monte Carlo EM algorithm

- Consider the random effects **u** to be the missing data. The complete data here is **W=(Y, u)**
- The monte carlo EM algorithm is as follows:
    1. Choosing starting values $\boldsymbol{\beta}^{(0)}$ and $\boldsymbol{\sigma}^{(0)}$, set **n=0**
    2. Generate *m* values, $\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)}, \boldsymbol{u}^{(3)}, \ldots, \boldsymbol{u}^{(m)}$ from the conditional distribution of **u|Y** using a Metropolis algorithm (use **f(y|u)** as the proposal distribution) and using the current parameter values
    3. Choose:
        (1) $\boldsymbol{\beta}^{(n+1)}$ to maximize a Monte Carlo estimate of **E[ln(f(y|u))]**
        (2) $\boldsymbol{\sigma}^{(n+1)}$ to maximize **E[ln(f(u|$\boldsymbol{\sigma}$))]**

Simulation Study

- Dataset: *Epilepsy* dataset
- Description: 59 epilepsy patients; each of the patients was assigned to a control group (placebo) or a treatment group. The experiment recorded the number of seizures experienced by each patient over 4 two-week periods.
- Structure: 59 x 4 observations on the following 7 variables: count(y), log(base/4), trt, trt*log(base/4), log(age), subject (u), v4
- Each patient can be seen as a group, and we use subject, which is the id of the patient as the random effect
- Consider the Poisson random intercept model:

$$y_{ij}|u_i \sim Poisson(\exp(\beta^T x_{ij} + u_i))$$

Results

- Use adaptive Gauss-Hermite quadrature (AGHQ) here as "gold standard" when the true values of the parameters are not known.

	β_0	β_{base}	β_{trt}	$\beta_{base*trt}$	β_{age}	β_{v4}	σ_0	Time (seconds)
AGHQ	-1.325	0.883	-0.933	0.481	-0.160	0.339	0.251	0.901
GVA (6 ITER)	-1.325	0.883	-0.933	0.481	-0.160	0.339	0.251	0.066
Monte Carlo EM	-1.325	0.883	-0.933	0.481	-0.160	0.339	0.251	108

Modified EM algorithms for High-Dimensional Clustering

Beomseok Seo

Penn State University

Nov. 29. 2018

Motivation

- For **unsupervised learning** techniques, model-based approach is one of the most popular methods.
- Model-based approach exploits latent variable Gaussian mixture model (**GMM**) and estimates the model through expectation and maximization (**EM**) algorithm.

$$\text{E-step}: \quad Q(\theta|\theta^{(t)}) = \hat{E}_{Z|X,\theta^{(t)}} l_c(\theta; X, Z)$$
$$= \sum_i^n \sum_k^K \underbrace{\hat{E}_{\theta^{(t)}}[z_{ki}|x_i]}_{\tau_{ki}^{(t)}} \{\log \pi_k + \log f_k(x_i; \theta_k)\}$$
$$\text{M-step}: \quad \theta^{(t+1)} = \arg\max_\theta Q(\theta|\theta^{(t)})$$

- However, when data is **high dimensional**, EM algorithm is confronted with identifiability, stability and computational efficiency problems.

Motivation

- Literature approached this issue by imposing sparsity through modification of either E-step or M-step in EM algorithm.

  ⇒ **Modifying E-step**

  $$Q(\theta|\theta^{(t)}) = \hat{E}_{Z|X,\theta^{(t)}}\{l_c(\theta; X, Z) + p(\theta)\}$$

  ⇒ **Modifying M-step**

  $$\theta^{(t+0.5)} = \arg\max_{\theta} Q(\theta|\theta^{(t)})$$
  $$\theta^{(t+1)} = s(\theta^{(t+0.5)})$$

- Different modifications have different performances in stability and computational efficiency.

Comparison between different modifications

- Where the modification is applied is not critical.
- Different modifications result in different update rule.

E.g. [Modification of M-step]: truncation step (Wang et al. 2014)
The update rule changes as follows.

$$\tau_{ki}^{(t)} = \frac{\pi_k^{(t)} f_k(x_i;\theta_k^{(t)})}{\sum_k^K \pi_k^{(t)} f_k(x_i;\theta_k^{(t)})}, \quad \pi_k^{(t+1)} = \frac{1}{n}\sum_i^n \tau_{ki}^{(t)} \quad \mu_k^{(t+0.5)} = \frac{\sum_i^n \tau_{ki}^{(t)} x_i}{\sum_i^n \tau_{ki}^{(t)}},$$

And additional step impose sparcity.

$$\begin{aligned} \hat{\mathcal{S}}^{(t+0.5)} &= \text{set of index } j\text{'s of the top } s \text{ largest } |\mu_{kj}^{(t+0.5)}| \\ \hat{\mu}_k^{(t+1)} &= \begin{cases} \mu_k^{(t+0.5)} & j \in \hat{\mathcal{S}}^{(t+0.5)} \\ 0 & j \notin \hat{\mathcal{S}}^{(t+0.5)} \end{cases} \end{aligned}$$

Comparison between different modifications

E.g. [Modification of E-step]: L_1 penalty (Pan and Shen 2007)

$$Q_p(\theta;\theta^{(t)}) = \sum_i^n \sum_k^K \tau_{ki}^{(t)} \{\log \pi_k + \log f_k(x_i;\theta_k)\} - \lambda \sum_k^K \sum_j^p |\mu_{kj}|$$

The update rule is same as previous example except that the additional step changes as follows.

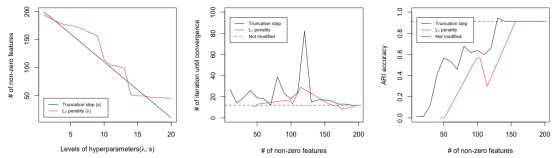
$$\hat{\mu}_k^{(t+1)} = sign(\mu_k^{(t+0.5)}) \left(|\mu_k^{(t+0.5)}| - \frac{\lambda}{\sum_i \tau_{ki}^{(t+1)}} V 1_p \right)_+ ,$$

For truncation step

$$\hat{\mu}_k^{(t+1)} = sign(\mu_k^{(t+0.5)}) \left(|\mu_k^{(t+0.5)}| - \mu_{k(n-s)}^{(t+0.5)} \right)_+ .$$

Simulation study

- $N = 100$, $p = 200$, $K = 3$(# of clusters), K-means initialization.

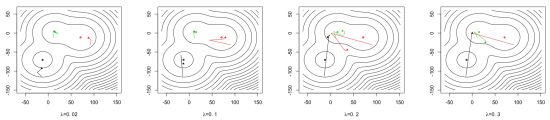


- Number of iteration till convergence is not proportional to dimension size.
- L1 penalized EM converges more quickly than EM with truncation step.
- Accuracy could not be improved by dim reduction.
- Any type of EM did not work well on random initialization.

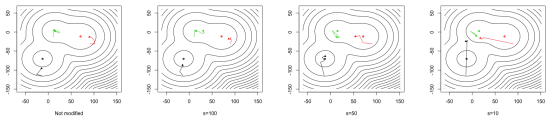
Simulation study

- Geometric interpretation on 2D principal components space.

L_1 penalty with different λ

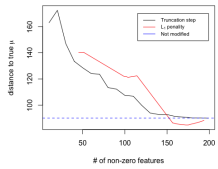


Truncation step with different thresholding s



Simulation study

- Geometric interpretation.



- Bias of estimated μ decreases at some level of hyperparameter and then increases as more dimensions are reduced.
- $\Rightarrow$ Bias corrected penalty such as SCAD, MCP may work better.

APPROXIMATE BAYESIAN COMPUTATION FOR ARCHIMEDEAN COPULA MODELS

Roopali Singh

Department of Statistics
Pennsylvania State University

APPROXIMATE BAYESIAN COMPUTING (ABC)

$$\pi(\theta|y) \propto p(y|\theta)\pi(\theta)$$

Problem: How to perform Bayesian inference when the likelihood function $p(y|\theta)$ is computationally intractable?

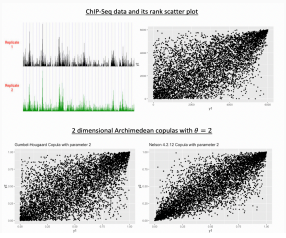
Solution: If we can easily simulate from the likelihood, ABC methods provide a possible way.

$$\pi_{ABC}(\theta|y) \propto \int \mathcal{I}(||y^* - y|| < \epsilon)\pi(\theta|y^*)dy^* \approx \pi(\theta|y)$$

MOTIVATION: ARCHIMEDEAN COPULA MODELS

J-dimensional copula: $C(u_1, \ldots, u_J) = P(U_1 \leq u_1, \ldots, U_J \leq u_J)$ where $U_1, \ldots, U_J$ are uniform random variables and $C : [0,1]^J \to [0,1]$.

J-dimensional Archimedean copula:
$\psi(C(u_1, \ldots, u_J); \theta) = \psi(u_1; \theta) + \ldots + \psi(u_J; \theta)$ where θ **is an association parameter** describing the strength of the dependence between U_j's, and ψ is a generator function specific to each Archimedean copula such that $\psi : [0,1] \to [0, \infty)$.



GUMBEL HOUGAARD COPULA

Gumbel Hougaard copula generator function:

$$\psi(u) = (-\log(u))^{\theta}\ ,\ \theta \in [1, \infty)$$

So its distribution function is,

$$C(u_1, ..., u_n) = \psi^{-1}(\psi(u_1)+...+\psi(u_n)) = \exp\{-((-\log(u_1))^{\theta}+...+(-\log(u_n))^{\theta})^{1/\theta}\}$$

Using ABC to estimate θ is possible because,

- likelihood is unavailable
- it easy to simulate from the model

ABC ALGORITHMS

Requirements:

- a proposal, q(.)
- the observed data, y
- a distance function, ||.||
- a tolerance level, ϵ

Rejection Sampling (Sisson et al. 2018)

1. Simulate $\theta^* \sim$ q(.)
2. Simulate y $\sim$ p(.|θ^*)
3. if $||y - y_{obs}|| < \epsilon$ then accept θ^* with probability $\frac{\pi(\theta^*)}{Kq(\theta^*)}$.
4. Repeat above steps N times.

MCMC (Marjoram et al. 2003)

1. Simulate $\theta^* \sim$ q(.)
2. Simulate y $\sim$ p(.|θ^*)
3. if $||y - y_{obs}|| < \epsilon$ then accept θ^* with probability $\min\{1, \frac{\pi(\theta^*)q(\theta|\theta^*)}{\pi(\theta)q(\theta^*|\theta)}\}$ else stay at θ.
4. Repeat above steps N times.

CHALLENGES OF ABC

- Should we use the entire dataset y or an appropriate summary statistic, s?
- Choice of distance function and summary statistic.
- Computationally expensive?
- Choice of tolerance level and other tuning parameters.

ABC MCMC: SIMULATION STUDY

- Prior on θ is **Gamma(shape=1, scale=2) truncated at 1**.
- Proposal is a random walk i.e. $N(\theta^{(i-1)}, \sigma^2)$ **truncated at 1**.
- Tolerance, ϵ and σ^2 are chosen such that the **acceptance rate was 20-35%**.
- **Absolute difference** is used as the distance function for two dimensions and for more than 2, **Frobenius norm** is used.
- For each simulation, y_{obs} consisted of 100 data points.
- Number of simulations = 100 and N=10000.

Dimensions:	2		3		5	
Summary statistic	Estimate	MSE	Estimate	MSE	Estimate	MSE
Spearman's rank ρ	3.06	1.32	2.94	1.12	2.19	0.07
Kendall's τ	2.60	0.47	2.55	0.39	2.16	0.05

Table: Results from ABC MCMC algorithm for Gumbel-Hougaard Copula with $\theta = 2$.

ABC MCMC WITH KERNEL FUNCTION

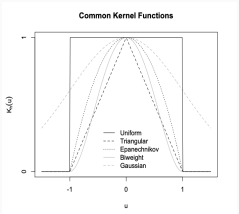
Kernel function: $K_\epsilon(u) = \frac{1}{\epsilon}K(\frac{u}{\epsilon})$ where $K(u) = \frac{1}{\sqrt{2\pi}}e^{-u^2/2}$

Previously: Accept θ^* with probability $\min\{1, \frac{\pi(\theta^*)q(\theta|\theta^*)}{\pi(\theta)q(\theta^*|\theta)}\}$.

Now: Accept θ^* with probability $\min\{1, \frac{K_\epsilon(||s^*-s_{obs}||)\pi(\theta^*)q(\theta|\theta^*)}{K_\epsilon(||s^{(i-1)}-s_{obs}||)\pi(\theta)q(\theta^*|\theta)}\}$.

Dimension	Estimate	MSE
2	2.24	0.12
3	2.14	0.09
5	2.18	0.10

Table: Results from ABC MCMC algorithm (using Kendall's τ) with kernel function for Gumbel Hougaard Copula with $\theta = 2$.



Note: Smoothing functions are discussed in Sisson et al. 2018 for ABC Rejection algorithms and something similar in Fernhead and Prangle 2011.

CONCLUSION

- For large data sets, using y_{obs} instead of summary statistic is not advisable due to computational costs.
- ABC MCMC requires a lot of tuning to run well. If possible, a different ABC algorithm can be preferred.
- Kendall's τ estimates better than Spearman's rank correlation.
- Estimation for higher dimensional copula works better because there is more information for a single parameter.
- ABC MCMC with kernel function seems like a better approach in this case, at least for lower dimensional copula.

Sequential Monte Carlo Estimation of High Dimensional Latent Variable Models

Application to Stochastic Volatility Models

Han Xiao

Nov. 2018

Pennsylvania State Unversity

Motivation

- **Estimate** high-dimensional latent variable models

  Stochastic volatility models: Widely used in practice and option pricing. Application Given asset returns, study the underlying volatility.

Motivation

- **Estimate** high-dimensional latent variable models

  Stochastic volatility models: Widely used in practice and option pricing. Application Given asset returns, study the underlying volatility.

$$y_t = \exp\left(\frac{h_t}{2}\right)\varepsilon_t \quad (1)$$

$$h_{t+1} = \mu + \phi(h_t - \mu) + \sigma\eta_t \quad (2)$$

  where ε and η are iid standard normal distribution.

  - y_t : asset return in reality $\rightarrow$ Observable
  - h_t: risk underlying the asset $\rightarrow$ Latent variable
  - Parameter: $\Theta = \{\mu, \phi, \sigma^2\}$

Motivation

- **Estimate** high-dimensional latent variable models

  Stochastic volatility models: Widely used in practice and option pricing. Application Given asset returns, study the underlying volatility.

$$y_t = \exp\left(\frac{h_t}{2}\right)\varepsilon_t \quad (1)$$

$$h_{t+1} = \mu + \phi(h_t - \mu) + \sigma\eta_t \quad (2)$$

  where ε and η are iid standard normal distribution.

  - y_t : asset return in reality $\rightarrow$ Observable
  - h_t: risk underlying the asset $\rightarrow$ Latent variable
  - Parameter: $\Theta = \{\mu, \phi, \sigma^2\}$

- **Goal**: learn parameters and latent variables recursively
  - use real time data and estimate efficiently

Computational challenge

The difficult part occurs in the latent variable (Equation (2)), i.e., likelihood function is not easily available:

$$f(h_t|y_t, \Theta) \propto \underbrace{f(y_t|h_t, \Theta)}_{\text{observed return}} \quad \underbrace{f(h_t|y_{t-1}, \Theta)}_{\text{latent variable: volatility}} \tag{3}$$

Computational challenge

The difficult part occurs in the latent variable (Equation (2)), i.e., likelihood function is not easily available:

$$f(h_t|y_t,\Theta) \propto \underbrace{f(y_t|h_t,\Theta)}_{\text{observed return}} \underbrace{f(h_t|y_{t-1},\Theta)}_{\text{latent variable: volatility}} \tag{3}$$

$$f(h_t|y_{t-1},\Theta) = \int \underbrace{f(h_t|h_{t-1},\Theta)}_{\text{proposed AR(1)}} \underbrace{f(h_{t-1}|y_{t-1},\Theta)}_{\text{hard closed form}} dh_{t-1} \tag{4}$$

$\Rightarrow$ *Difficult to evaluate the objective function*

Computational challenge

The difficult part occurs in the latent variable (Equation (2)), i.e., likelihood function is not easily available:

$$f(h_t|y_t,\Theta) \propto \underbrace{f(y_t|h_t,\Theta)}_{\text{observed return}} \underbrace{f(h_t|y_{t-1},\Theta)}_{\text{latent variable: volatility}} \tag{3}$$

$$f(h_t|y_{t-1},\Theta) = \int \underbrace{f(h_t|h_{t-1},\Theta)}_{\text{proposed AR(1)}} \underbrace{f(h_{t-1}|y_{t-1},\Theta)}_{\text{hard closed form}} dh_{t-1} \tag{4}$$

$\Rightarrow$ *Difficult to evaluate the objective function*

Challenge 1: Cannot directly draw **sample** from $f(h_t|y_{t-1},\Theta)$

Computational challenge

The difficult part occurs in the latent variable (Equation (2)), i.e., likelihood function is not easily available:

$$f(h_t|y_t,\Theta) \propto \underbrace{f(y_t|h_t,\Theta)}_{\text{observed return}} \underbrace{f(h_t|y_{t-1},\Theta)}_{\text{latent variable: volatility}} \tag{3}$$

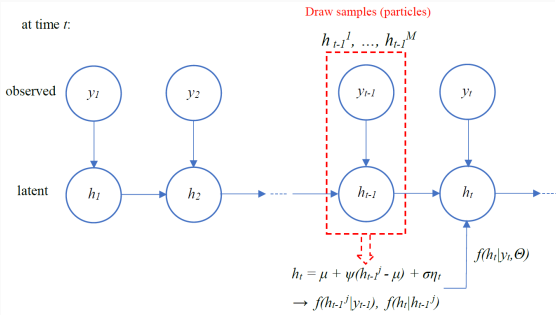
$$f(h_t|y_{t-1},\Theta) = \int \underbrace{f(h_t|h_{t-1},\Theta)}_{\text{proposed AR(1)}} \underbrace{f(h_{t-1}|y_{t-1},\Theta)}_{\text{hard closed form}} dh_{t-1} \tag{4}$$

$\Rightarrow$ *Difficult to evaluate the objective function*

Challenge 1: Cannot directly draw **sample** from $f(h_t|y_{t-1},\Theta)$

Challenge 2: Computational **complexity** of simulation increases with the number of time points in data

Intuition for sequential Monte Carlo Method



Sequential Monte Carlo: particle filter algorithm

- Main Idea: draw particles $\{h_{t-1}^{j}\}$ from filtered distribution $f(h_{t-1}|y_{t-1})$, and derive $f(h_t|y_t)$ using Equation (2). Then sequentially draw particles $\{h_t^{j}\}$

Sequential Monte Carlo: particle filter algorithm

- Main Idea: draw particles $\{h_{t-1}^j\}$ from filtered distribution $f(h_{t-1}|y_{t-1})$, and derive $f(h_t|y_t)$ using Equation (2). Then sequentially draw particles $\{h_t^j\}$
- Sequentially draw M particles at each time point t $\{h_{t-1}\} = \{h_{t-1}^1, \cdots, h_{t-1}^M\}$ from the filtered distributions: $f(h_{t-1}|y_{t-1})$, where M is the number of particles, then get

$$h_t = \mu + \phi(h_{t-1}^j - \mu)$$

Sequential Monte Carlo: particle filter algorithm

- Main Idea: draw particles $\{h_{t-1}^j\}$ from filtered distribution $f(h_{t-1}|y_{t-1})$, and derive $f(h_t|y_t)$ using Equation (2). Then sequentially draw particles $\{h_t^j\}$
- Sequentially draw M particles at each time point t $\{h_{t-1}\} = \{h_{t-1}^1, \cdots, h_{t-1}^M\}$ from the filtered distributions: $f(h_{t-1}|y_{t-1})$, where M is the number of particles, then get

$$h_t = \mu + \phi(h_{t-1}^j - \mu)$$

This gives Monte Carlo approximations

$$f(h_t|y_{t-1}, \Theta) \approx \frac{1}{M}\sum_{j=1}^{M} f(h_t|h_{t-1}^j, \Theta) \quad (5)$$

$$f(h_t|y_t, \Theta) \propto f(y_t|h_t, \Theta)\frac{1}{M}\sum_{j=1}^{M} f(h_t|h_{t-1}^j, \Theta) \quad (6)$$

Sequential Monte Carlo: particle filter algorithm

- Main Idea: draw particles $\{h_{t-1}^j\}$ from filtered distribution $f(h_{t-1}|y_{t-1})$, and derive $f(h_t|y_t)$ using Equation (2). Then sequentially draw particles $\{h_t^j\}$
- Sequentially draw M particles at each time point t $\{h_{t-1}\} = \{h_{t-1}^1, \cdots, h_{t-1}^M\}$ from the filtered distributions: $f(h_{t-1}|y_{t-1})$, where M is the number of particles, then get

$$h_t = \mu + \phi(h_{t-1}^j - \mu)$$

This gives Monte Carlo approximations

$$f(h_t|y_{t-1}, \Theta) \approx \frac{1}{M}\sum_{j=1}^{M} f(h_t|h_{t-1}^j, \Theta) \quad (5)$$

$$f(h_t|y_t, \Theta) \propto f(y_t|h_t, \Theta)\frac{1}{M}\sum_{j=1}^{M} f(h_t|h_{t-1}^j, \Theta) \quad (6)$$

- I explore different sampling methods: importance-sampling, bootstrap, auxiliary variable Reference

Methodology: compare with MCMC

MCMC Algorithm

1. Initialize h_1 and $\Theta^{(1)}$
2. Sample h_t from $h_t^{(k+1)}|h_{t-1}^{(k)}, y, \Theta^{(k)}$ for $t = 1, \cdots, n$
3. Sample $\sigma^{2(k+1)}|y, h^{(k+1)}, \phi^{(k)}, \mu^{(k)}$
4. Sample $\phi^{(k+1)}|h^{(k+1)}, \mu^{(k)}, \sigma^{2(k+1)}$
5. Sample $\mu^{(k+1)}|h^{(k+1)}, \phi^{(k+1)}, \sigma^{2(k+1)}$
6. Goto 2

Methodology: compare with MCMC

MCMC Algorithm

1. Initialize h_1 and $\Theta^{(1)}$
2. Sample h_t from $h_t^{(k+1)}|h_{t-1}^{(k)}, y, \Theta^{(k)}$ for $t = 1, \cdots, n$
3. Sample $\sigma^{2(k+1)}|y, h^{(k+1)}, \phi^{(k)}, \mu^{(k)}$
4. Sample $\phi^{(k+1)}|h^{(k+1)}, \mu^{(k)}, \sigma^{2(k+1)}$
5. Sample $\mu^{(k+1)}|h^{(k+1)}, \phi^{(k+1)}, \sigma^{2(k+1)}$
6. Goto 2

Computing is more expensive

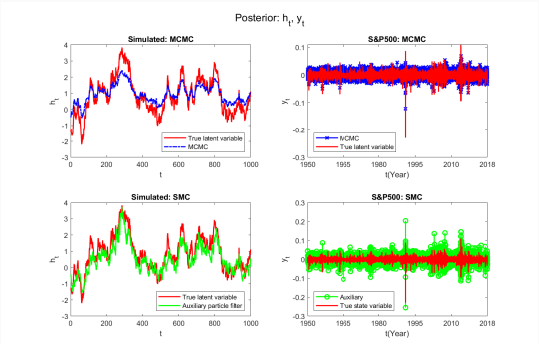
- Need to sample from $h_t^{(k+1)}|h_{t-1}^{(k)}, y, \Theta^{(k)}$ at each time point
- $\{h_t\}$,Θ are highly correlated, resulting in slow convergence ACF

Comparison: Efficiency and Accuracy

- Simulated data: $\mu_0 = 0.5$, $\phi_0 = 0.985$, $\sigma_0^2 = 0.04$
- Real data: Daily S&P 500 returns, 1950 – 2018

	Simulated Data				Real Data			
	MCMC	SISR	Boot	APF	MCMC	SISR	Boot	APF
Computational Efficiency								
ESS	5302	**724**	902	964	1107	**576**	756	826
time	113.106	**0.117**	0.228	0.311	193.733	**2.048**	3.766	4.274
MSE for last latent variable h_N								
mean	**0.013**	0.098	0.100	0.097	1.174	0.134	0.134	**0.132**
std	0.315	**0.041**	0.045	0.043	**0.047**	2.096	2.082	2.003
MSE for parameters								
μ	0.200	0.010	**0.000**	0.090	–	–	–	–
ϕ	0.006	**0.000**	0.000	0.000	–	–	–	–
σ^2	0.004	**0.001**	0.005	0.003	–	–	–	–

Comparison: Latent Variable Estimates



Conclusion

- SMC Strength
  - Efficient: large sample and iteration, more parameters
  - The algorithm can be easily parallelized
  - The computational complexity does not increase with increase in time
  - As the number of particles M increases, the accuracy increases.

Conclusion

- SMC Strength
  - Efficient: large sample and iteration, more parameters
  - The algorithm can be easily parallelized
  - The computational complexity does not increase with increase in time
  - As the number of particles M increases, the accuracy increases.
- SMC Weakness
  - Sensitive to outliers
  - Poor approximation at tails
  - Inference is dominated by a few particles with high weight: Potential solution is auxiliary particle filter

Conclusion

- SMC Strength
  - Efficient: large sample and iteration, more parameters
  - The algorithm can be easily parallelized
  - The computational complexity does not increase with increase in time
  - As the number of particles M increases, the accuracy increases.
- SMC Weakness
  - Sensitive to outliers
  - Poor approximation at tails
  - Inference is dominated by a few particles with high weight: Potential solution is auxiliary particle filter
- Auxiliary particle filter may be better at handling outliers and heavy tails
  - I found some tentative evidence from simulation Density

Thank you

Appendix

- Application
- Auxiliary particle filter algorithm
- MCMC simulation autocorrelation
- Sequence density

Application: Stochastic Volatility Models

1. Option price: Black-Scholes, Heston and Hull-White model
2. Long run risk model
3. Industry

Back

Methodology: Sequential Monte Carlo

SMC Algorithm: auxiliary particle filter Back

1. Given $\{h_{t-1}^1, \cdots, h_{t-1}^M\}$ from $f(h_{t-1}|y_{t-1}, \Theta)$ calculate

$$\begin{aligned} \widehat{h}_t^{*j} &= \mu + \phi(h_{t-1}^j - \mu) \\ w_j &= f_N(y_t|exp(\widehat{h}_t^{*j})), \quad j = 1, \cdots, M \end{aligned}$$

   and sample R times with probability $\{w_j\}$. Let the sampled index be $k_1, \cdots, k_R$, and associate these with $\widehat{h}_t^{*k_1}, \cdots, \widehat{h}_t^{*k_R}$

2. For each value of k_j from Step 1 simulate

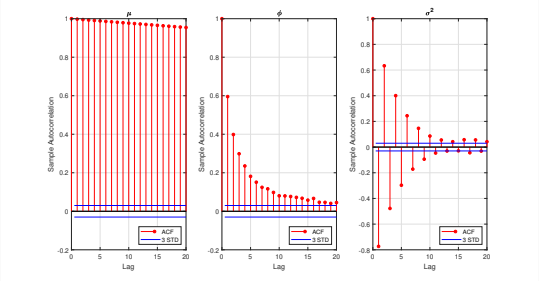
$$h_t^{*j} \sim \mathcal{N}(\mu + \phi(h_{t-1}^{*k_j} - \mu), \sigma^2), \quad j = 1, \cdots, R$$

3. Resample $\{h_t^{*1}, \cdots, h_t^{*R}\}$ with probability

$$\frac{\mathcal{N}(\mu + \phi(h_{t-1}^{*j} - \mu), \sigma^2)}{\mathcal{N}(\mu + \phi(h_{t-1}^{*k_j} - \mu), \sigma^2)}$$

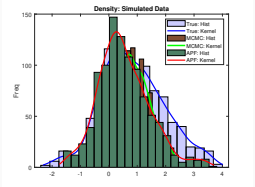
   to produce the filtered sample $\{h_t^1, \cdots, h_t^M\}$ from $f(h_t|y_t, \Theta)$

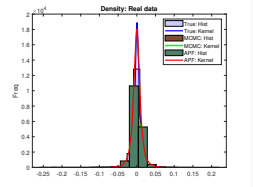
Slow convergence in MCMC: Autocorrelation



Back

Density analysis: simulated and real data





Back

Reference

- Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993), Novel approach to nonlinear/non-Gaussian Bayesian state estimation, IEEE Proceedings Part F: Communications, Radar and Signal Processing, 140, 107–113.
- Kim, S., Shephard, N., and Chib, S. (1998). Stochastic volatility: likelihood inference and comparison with ARCH models. The review of economic studies, 65(3), 361-393.
- Pitt, M. K. and Shephard, N. (1999), Filtering via simulation: auxiliary particle filters, Journal of the American Statistical Association, 94, 590–599.
- Creal, D. (2012). A survey of sequential Monte Carlo methods for economics and finance. Econometric reviews, 31(3), 245-296.

Back