Scalable Monte Carlo

Babak Shahbaba

Department of Statistics, UCI

May 4, 2017

Introduction

Bayesian inference

- In Bayesian statistics we make inference based on posterior probability distribution $P(\theta|y)$:

$$
\begin{aligned}
P(\theta|y) &= \frac{P(y|\theta)P(\theta)}{P(y)} \\
&\propto P(y|\theta)P(\theta)
\end{aligned}
$$

- For example, we can predict future observations, $\tilde{y}$, given the observed data y:

$$
P(\tilde{y}|y) = \int_\theta P(\tilde{y}|\theta)P(\theta|y)d\theta
$$

- **Main challenge:** inference almost always involves intractable integrals

Bayesian inference

- In Bayesian statistics we make inference based on posterior probability distribution $P(\theta|y)$:

$$
\begin{aligned}
P(\theta|y) &= \frac{P(y|\theta)P(\theta)}{P(y)} \\
&\propto P(y|\theta)P(\theta)
\end{aligned}
$$

- For example, we can predict future observations, $\tilde{y}$, given the observed data y:

$$
P(\tilde{y}|y) = \int_\theta P(\tilde{y}|\theta)P(\theta|y)d\theta
$$

- **Main challenge:** inference almost always involves intractable integrals

Monte Carlo approximation

- In general, we can approximate

$$\mu = \int_{\mathcal{X}} h(x) f(x) dx$$

using iid samples $x^{(1)}, x^{(2)}, ..., x^{(m)}$ from the distribution with density $f(x)$:

$$\hat{\mu} = \frac{1}{m}[h(x^{(1)}) + h(x^{(2)}) + ... + h(x^{(m)})]$$

- **Main challenge**: usually we cannot generate iid samples from the distribution

Monte Carlo approximation

- In general, we can approximate

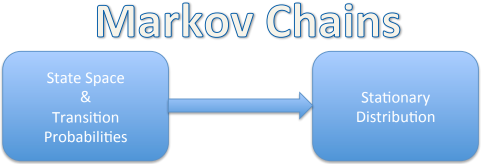
$$\mu = \int_{\mathcal{X}} h(x) f(x) dx$$

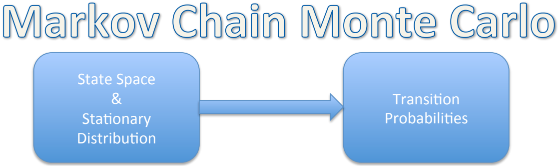
using iid samples $x^{(1)}, x^{(2)}, ..., x^{(m)}$ from the distribution with density $f(x)$:

$$\hat{\mu} = \frac{1}{m}[h(x^{(1)}) + h(x^{(2)}) + ... + h(x^{(m)})]$$

- **Main challenge**: usually we cannot generate iid samples from the distribution

- We use Markov chains to generate samples from the distribution

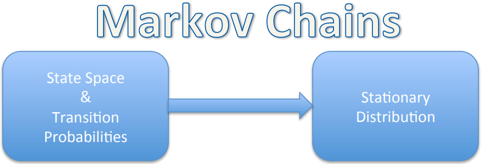


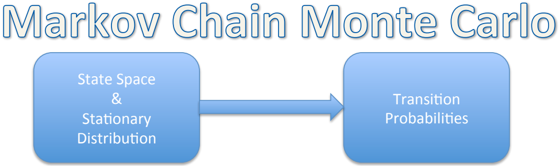


- **Main challenge**: finding a good transition probability

Markov chain Monte Carlo

- We use Markov chains to generate samples from the distribution





- **Main challenge**: finding a good transition probability

The Metropolis Algorithm

- Specify a symmetric transition probability $g(\theta, \theta^*)$ and repeat the following steps for many iterations:
  1. Given our current state θ, we propose a new state θ^* according to g.
  2. Calculated the acceptance probability,
  $$\begin{aligned} a(\theta, \theta^*) &= \min(1, \frac{f(\theta^*)}{f(\theta)}) \\ &= \min\{1, \exp(\log[f(\theta^*)] - \log[f(\theta)])\} \end{aligned}$$
  3. Accept the proposed state θ^* as the new state with probability $a(\theta, \theta^*)$ or remain at state θ.
- For asymmetrical proposal distribution, we use Metropolis-Hastings (MH),
$$a(\theta, \theta^*) = \min(1, \frac{f(\theta^*) g(\theta^*, \theta)}{f(\theta) g(\theta, \theta^*)})$$
- **Main challenge:** finding the right proposal-generating mechanism, $g(\theta, \theta^*)$

The Metropolis Algorithm

- Specify a symmetric transition probability $g(\theta, \theta^*)$ and repeat the following steps for many iterations:
  1. Given our current state θ, we propose a new state θ^* according to g.
  2. Calculated the acceptance probability,

$$
\begin{aligned}
a(\theta, \theta^*) &= \min(1, \frac{f(\theta^*)}{f(\theta)}) \\
&= \min\{1, \exp(\log[f(\theta^*)] - \log[f(\theta)])\}
\end{aligned}
$$

  3. Accept the proposed state θ^* as the new state with probability $a(\theta, \theta^*)$ or remain at state θ.

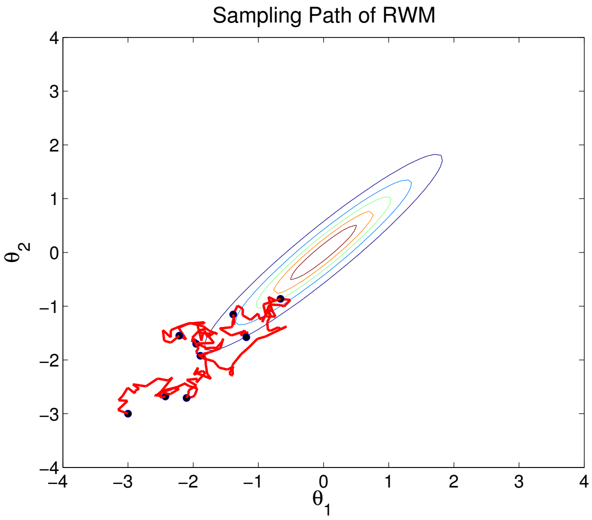
- For asymmetrical proposal distribution, we use Metropolis-Hastings (MH),

$$
a(\theta, \theta^*) = \min(1, \frac{f(\theta^*)g(\theta^*, \theta)}{f(\theta)g(\theta, \theta^*)})
$$

- **Main challenge:** finding the right proposal-generating mechanism, $g(\theta, \theta^*)$

Random walk Metropolis

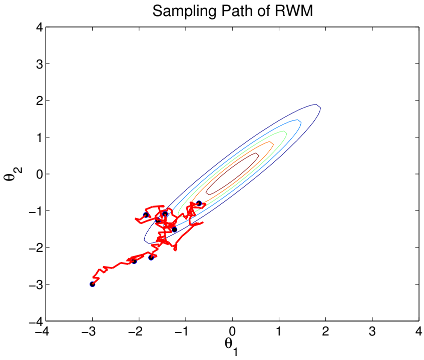
- A simple proposal generating mechanism is random walk: $\theta^* \sim N(\theta, \epsilon^2 I)$

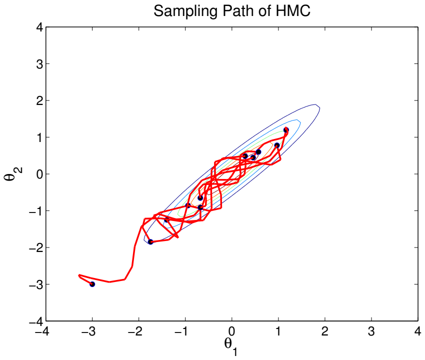


Hamiltonian Monte Carlo (HMC)

Hamiltonian Monte Carlo

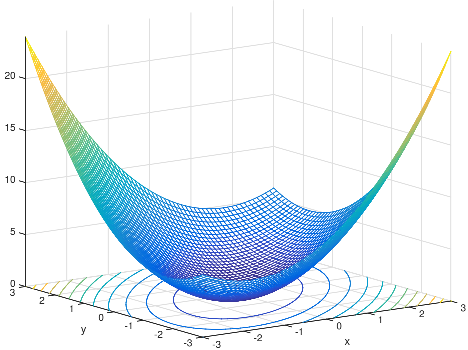
- HMC proposes states that are distant from the current state, but nevertheless have a high probability of acceptance.





Hamiltonian Monte Carlo

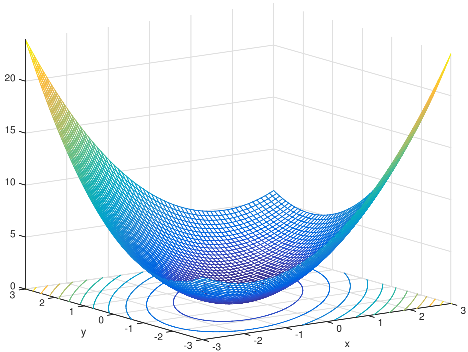
- The sampler is viewed as a dynamic system moving on a surface defined by the *energy* function U: negative log density of the target distribution



- Distant proposals are found by numerically simulating Hamiltonian dynamics for some specified amount of fictitious time

Hamiltonian Monte Carlo

- The sampler is viewed as a dynamic system moving on a surface defined by the *energy* function U: negative log density of the target distribution



- Distant proposals are found by numerically simulating Hamiltonian dynamics for some specified amount of fictitious time

Posterior sampling

- For Bayesian inference, posterior distribution is the target distribution

Potential energy

$$U(\theta) = -\sum_{i=1}^{N} \log P(y_i|\theta) - \log P(\theta)$$

- We augment the parameter space with fictitious momentum variables

Kinetic energy

$$K(p) = \frac{1}{2} p^\top M^{-1} p$$

- Define the Hamiltonian function $H(\theta, p) = U(\theta) + K(p)$
- The joint density of (θ, p) is

$$P(\theta, p) \propto \exp\{-H(\theta, p)\} = \exp\{-U(\theta)\} \cdot \exp\{-K(p)\}$$

- The marginal distribution of θ is the posterior distribution

Hamiltonian Dynamics

- We can generate a proposal by starting from the current state at time 0 and moving to the state at time t:

$$(\theta, p) = (\theta^{(0)}, p^{(0)}) \xrightarrow{\text{HD}} (\theta^{(t)}, p^{(t)}) = (\theta^*, p^*)$$

- *Hamilton's equations* determine how θ and p change over [fictitious] time

Hamilton's equations

$$\begin{aligned} \frac{d\theta_j}{dt} &= \frac{\partial H}{\partial p_i} = [M^{-1}p]_i \\ \frac{dp_j}{dt} &= -\frac{\partial H}{\partial \theta_j} = -\frac{\partial U}{\partial \theta_j} \end{aligned}$$

- Important properties:
  - **Reversibility**: the target distribution remain invariant
  - **Volume preservation**: the Jacobin determinant is 1
  - **Conservation of Hamiltonian**: the acceptance rate is one; θ^* is the next sample if HD is analytically solvable

Numerical Integration

- Numerical integration is employed when analytic solution is not available

Leapfrog

$$
\begin{aligned}
p_j(t+\epsilon/2) &= p_j(t) - (\epsilon/2)\frac{\partial U}{\partial \theta_j}(\theta(t)) \\
\theta_j(t+\epsilon) &= \theta_j(t) + \epsilon\frac{\partial K}{\partial p_j}(p(t+\epsilon/2)) \\
p_j(t+\epsilon) &= p_j(t+\epsilon/2) - (\epsilon/2)\frac{\partial U}{\partial \theta_j}(\theta(t+\epsilon))
\end{aligned}
$$

- Important properties:
  - **Stability**: numerically stable if ε is appropriately chosen
  - **Reversibility and Volume preservation**: still hold
  - **Conservation of Hamiltonian**: broken, but can be corrected by MH correction step with acceptance rate

$$\alpha = \min[1, \exp(-H(\theta^*, p^*) + H(\theta, p))]$$

Algorithm 1: HMC algorithm

Initialize $\theta^{(0)} = \text{current } \theta$

Sample new momentum $p^{(0)} \sim \mathcal{N}(0, M = I)$

Calculate current $H(\theta^{(0)}, p^{(0)}) = U(\theta^{(0)}) + \frac{1}{2}(p^{(0)})^\top p^{(0)}$

for $\ell = 1$ to L (leapfrog steps) **do**

$p^{(\ell+\frac{1}{2})} = p^{(\ell)} - \epsilon/2 \nabla_\theta U(\theta^{(\ell)})$

$\theta^{(\ell+1)} = \theta^{(\ell)} + \epsilon p^{(\ell+\frac{1}{2})}$

$p^{(\ell+1)} = p^{(\ell+\frac{1}{2})} - \epsilon/2 \nabla_\theta U(\theta^{(\ell+1)})$

end for

Accept or reject according to the Metropolis acceptance probability

Example: logistic regression with $N(0, \sigma^2 I)$ prior

$$\nabla_{\beta_j} U(\beta) \quad = \quad -\sum_{i=1}^{N}[y_i - \frac{\exp(x_i\beta)}{1+\exp(x_i\beta)}]x_{ij} + \beta_j/\sigma^2$$

A special case: Langevin Monte Carlo

- A special case: $L = 1$ and $M = I$
- This is called Langevin Monte Carlo,

Langevin dynamics

$$\begin{aligned}
\theta^* &= \theta - \frac{\epsilon^2}{2}\nabla_\theta U(\theta) + \epsilon p \\
p^* &= p - \frac{\epsilon}{2}\nabla_\theta U(\theta) - \frac{\epsilon}{2}\nabla_\theta U(\theta^*)
\end{aligned}$$

- Alternatively, we could ignore the momentum variable p and use the following asymmetrical proposal with MH acceptance probability

$$\theta^* \sim N(\theta - \frac{\epsilon^2}{2}\nabla_\theta U(\theta), \epsilon^2 I)$$

- Dropping the accept/reject step leads to an approximate Langevin method (see Neal, 1993)

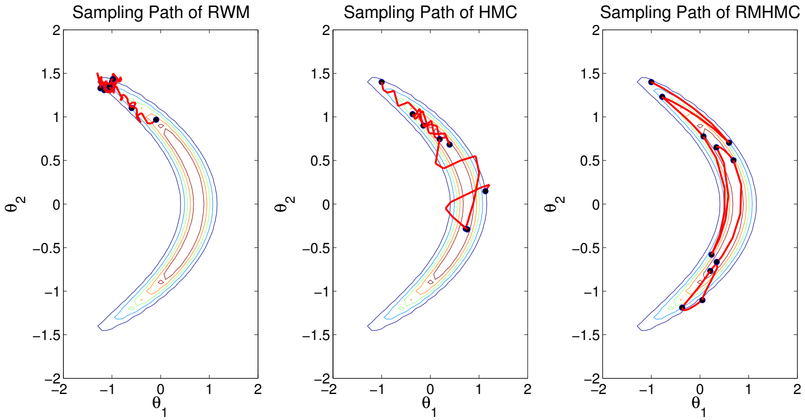
A more general case: Riemannian Manifold HMC

- Girolami and Calderhead (2011) have introduced a new method, called Riemannian Manifold HMC (RMHMC)
- They argue that it is more natural to put the Hamiltonian dynamic on Riemannian manifold of distributions rather than Euclidean space
- They follow Amari (2000) and use the Fisher information matrix, $G(\theta) = E\left[\nabla^2_\theta U(\theta)\right]$, as a metric on the manifold
- That is, they use position specific mass matrix, $M = G(\theta)$
- Example: logistic regression

$$G_{jk}(\beta) = \sum_{i=1}^{N} x_{ij} x_{ik} \frac{\exp(x_{i\beta})}{[1+\exp(x_i\beta)]^2}, \quad j \neq k$$

- We can explore the parameter space more efficiently by exploiting its geometric properties
- The resulting dynamics is non-separable so instead of the standard leapfrog method we need to use the *generalized* leapfrog method

HMC vs. RMHMC


Sampling Path of RWM
Sampling Path of HMC
Sampling Path of RMHMC
θ_2
θ_1
2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1
0
1
2

A main challenge: high computational cost

- For high-dimensional problems (big n and/or big d) and complex models, these methods tend to be computationally expensive

- We have proposed several variations of HMC:
  - Split HMC (S. et al., 2011)
  - Lagrangian Monte Carlo (Lan, et al., 2012)
  - Spherical HMC (Lan et al., 2013)
  - Wormhole HMC (Lan et al., 2013)
  - HMC with precomputing strategy (Zhang et al., 2015)
  - HMC with surrogate functions (Zhang et al., 2015)

Scalable HMC

Subsampling

- In recent years, computational methods based on mini-batches of data have been quite successful
  - The underlying assumption: there is redundancy in big data
  - The overall information can be retrieved from a small subset
  - We can approximate functions at low computational cost
- Welling and Teh (2011) used this approach (stochastic gradient) for Langevin dynamics using mini-batches of size n from N observations

$$\theta^* = \theta + \frac{\epsilon^2}{2}\left(\nabla_\theta P(\theta) + \frac{N}{n}\sum_{i=1}^{n}\nabla_\theta \log P(x_i|\theta)\right) + \epsilon p$$

- They also dropped the accept/reject step

Subsampling

- In recent years, computational methods based on mini-batches of data have been quite successful
  - The underlying assumption: there is redundancy in big data
  - The overall information can be retrieved from a small subset
  - We can approximate functions at low computational cost
- Welling and Teh (2011) used this approach (stochastic gradient) for Langevin dynamics using mini-batches of size n from N observations

$$\theta^* \quad = \quad \theta + \frac{\epsilon^2}{2}\Big(\nabla_\theta P(\theta) + \frac{N}{n}\sum_{i=1}^{n} \nabla_\theta \log P(x_i|\theta)\Big) + \epsilon p$$

- They also dropped the accept/reject step

Subsampling

- In recent years, computational methods based on mini-batches of data have been quite successful
  - The underlying assumption: there is redundancy in big data
  - The overall information can be retrieved from a small subset
  - We can approximate functions at low computational cost
- Welling and Teh (2011) used this approach (stochastic gradient) for Langevin dynamics using mini-batches of size n from N observations

$$\theta^* = \theta + \frac{\epsilon^2}{2}\left(\nabla_\theta P(\theta) + \frac{N}{n}\sum_{i=1}^{n} \nabla_\theta \log P(x_i|\theta)\right) + \epsilon p$$

- They also dropped the accept/reject step

Focusing on parameter space

- Finding optimum subsets by exploiting regularity in data space is difficult
- Using random subsets could lead to non-ignorable loss of information
- Therefore, we previously proposed to identify a subset of influential points to split the Hamiltonian function (Leimkuler and Reich, 2004) into two parts (S. et al., 2011)
- Recently, we have switched our focus from data space to parameter space
  - We exploit the smoothness and regularity of parameter space
  - We precompute functions (e.g., gradient) on a relatively small sample of parameters
  - MCMC algorithms use these precomputed values to approximate functions
  - We use the exact target distribution for the accept/reject step to ensure convergence to the right stationary distribution

Focusing on parameter space

- Finding optimum subsets by exploiting regularity in data space is difficult
- Using random subsets could lead to non-ignorable loss of information
- Therefore, we previously proposed to identify a subset of influential points to split the Hamiltonian function (Leimkuler and Reich, 2004) into two parts (S. et al., 2011)
- Recently, we have switched our focus from data space to parameter space
  - We exploit the smoothness and regularity of parameter space
  - We precompute functions (e.g., gradient) on a relatively small sample of parameters
  - MCMC algorithms use these precomputed values to approximate functions
  - We use the exact target distribution for the accept/reject step to ensure convergence to the right stationary distribution

Focusing on parameter space

- Finding optimum subsets by exploiting regularity in data space is difficult
- Using random subsets could lead to non-ignorable loss of information
- Therefore, we previously proposed to identify a subset of influential points to split the Hamiltonian function (Leimkuler and Reich, 2004) into two parts (S. et al., 2011)
- Recently, we have switched our focus from data space to parameter space
  - We exploit the smoothness and regularity of parameter space
  - We precompute functions (e.g., gradient) on a relatively small sample of parameters
  - MCMC algorithms use these precomputed values to approximate functions
  - We use the exact target distribution for the accept/reject step to ensure convergence to the right stationary distribution

Focusing on parameter space

- Finding optimum subsets by exploiting regularity in data space is difficult
- Using random subsets could lead to non-ignorable loss of information
- Therefore, we previously proposed to identify a subset of influential points to split the Hamiltonian function (Leimkuler and Reich, 2004) into two parts (S. et al., 2011)
- Recently, we have switched our focus from data space to parameter space
  - We exploit the smoothness and regularity of parameter space
  - We precompute functions (e.g., gradient) on a relatively small sample of parameters
  - MCMC algorithms use these precomputed values to approximate functions
  - We use the exact target distribution for the accept/reject step to ensure convergence to the right stationary distribution

Naive Grid Approximation (GHMC)

$$\frac{dp_j}{dt} = -\frac{\partial U}{\partial \theta_j}$$
$$\frac{d\theta_j}{dt} = [M^{-1}p]_j$$

Denote

Force

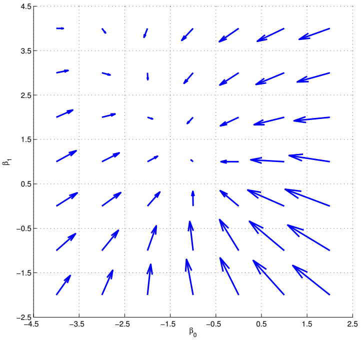
$$F = -\nabla U$$

- piecewise constant approximation

$$\tilde{F}(\theta) = F_{i,j} \triangleq F(c_{i,j}), \quad \text{if } \theta \in C_{i,j}$$

- piecewise linear approximation

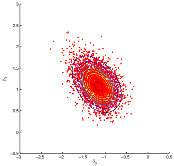
$$\tilde{F}(\theta) = F_{i,j} + \nabla F_{i,j} \cdot (q - c_{i,j}), \text{if } \theta \in C_{i,j}$$



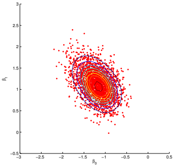
Force map of a logistic regression model

True $\beta = (-1, 1)$

GHMC–Examples

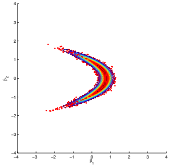


HMC

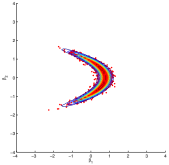


GHMC

Logistic regression



HMC



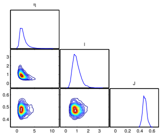
GHMC

Banana-shaped distribution

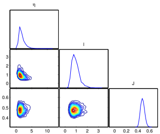
Experiment	Method	AR	s/Iteration	min(ESS)/s	Spped-up
LR	HMC	0.9225	7.0157E-4	1425.4	1
	GHMC	0.7981	3.318E-4	3013.9	**2.1**
BD	HMC	0.9353	3.8703E-4	962.1	1
	GHMC	0.6587	1.4498E-4	1651.6	**1.7**

Sparse Grid for Higher Dimensions (SGHMC)

- In higher dimensions, we use a sparse gird interpolation method based on *Smolyak's* formula (Barthelmann, 2000; Bungartz, 1998 & 2004)
- It employs $\mathcal{O}(N \cdot (\log(N))^{d-1})$ points only, with approximation accuracy preserved up to a logarithmic factor

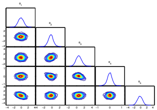


HMC

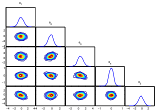


SGHMC

Gaussian Process model



HMC



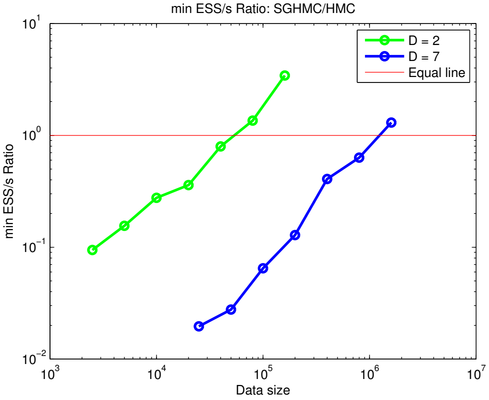
SGHMC

Elliptic PDE inverse problem

Experiment	Method	AR	s/Iteration	min(ESS)/s	Speed-up
GP	HMC	0.9472	2.3547E-1	1.3	1
	SGHMC	0.7066	2.9851E-2	**8.7**	**6.7**
ePDE	HMC	0.7719	2.02E-1	1.5	1
	SGHMC	0.6141	6.1952E-2	**4.3**	**2.9**

SGHMC–Examples

- However, this approach reaches its limit quite fast



HMC with Surrogate Functions

NNS-HMC

- In recent years, several methods have been proposed based on constructing *surrogate* Hamiltonians using Gaussian process models (Rasmussen, 2003; Meeds and Welling, 2015; Lan et. al., 2015)

- We have instead used a simple generalized additive model, which can be regarded as a shallow neural network,

$$\tilde{U}(\theta) = \sum_{i=1}^{s} v_i g(\boldsymbol{w_i} \cdot \theta + d_i) + d_0$$

with the softplus function: $g(z) = \log(1 + \exp(z))$

Extreme Learning Machine (ELM)

For training, we randomly assign input weights and biases, and then obtain the least-square estimates of the output weights v

ELM (Huang, 2006)

Given a training set $\mathcal{T} = \{(I_j, t_j) | I_j \in \mathbb{R}^n,\ t_j \in \mathbb{R}^m,\ j = 1, \ldots, N\}$, activation function $\sigma(x)$ and hidden node number s

1. Randomly assign input weight w_i and bias d_i, $i = 1, \ldots, s$
2. Calculate the hidden layer output matrix H

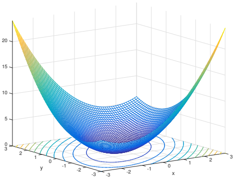
$$H_{ji} = \sigma(w_i I_j + d_i), \quad i = 1, \ldots, s,\ j = 1, \ldots, N$$

3. Calculate the output weight v

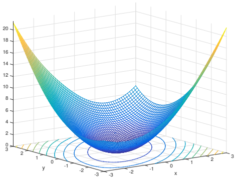
$$v = H^\dagger T, \quad T = [t_1, t_2, \ldots, t_N]^T$$

where $H^\dagger$ is the *Moore-Penrose generalized inverse* of matrix H

NNS-HMC



Target function



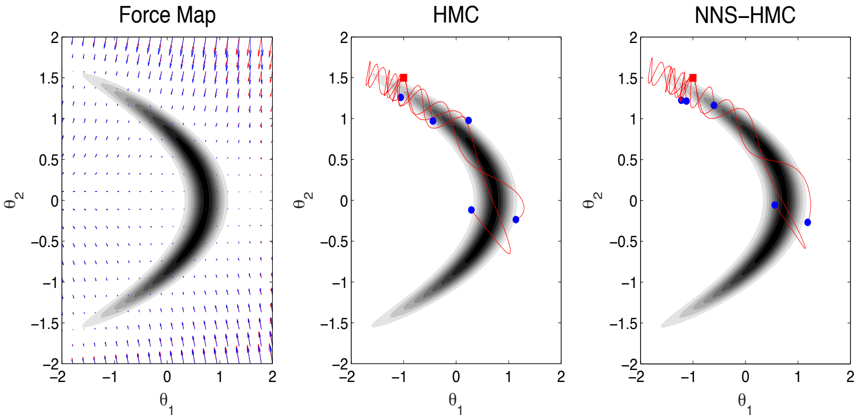
Neural network approximation

- The training process (using pre-convergence samples) and the approximation of functions in the sampling phase can be easily incorporated in HMC
- The approximate geometric information (e.g, gradient and Hessian) is obtained by differentiating the neural network directly,

$$\frac{\partial \tilde{U}}{\partial \theta} = \sum_{i=1}^{s} v_i g'(\boldsymbol{w}_i \cdot \boldsymbol{\theta} + d_i)\boldsymbol{w}_i$$

- Easy generalization to Riemannian Manifold HMC (NNS-RMHMC)

Surrogate Induced Hamiltonian Flow



Experiments

Experiment	Method	AP	s/Iter	$\min(\text{ESS})/s$	Spped-up
LR (Simulation)	HMC	0.6656	3.573E-01	1.45	1
	RMHMC	0.8032	3.794	0.06	0.04
	NNS-HMC	0.6726	1.364E-02	37.83	**26.09**
	NNS-RMHMC	0.8162	1.027E-01	2.17	1.50
LR (Bank Marketing)	HMC	0.8038	7.400E-02	6.51	1
	RMHMC	0.9210	6.305E-01	0.56	0.08
	NNS-HMC	0.7944	7.508E-03	58.22	**8.94**
	NNS-RMHMC	0.9064	2.741E-02	14.41	2.21
LR (Adult Data)	HMC	0.8300	7.898E-02	0.21	1
	RMHMC	0.8526	5.842E-01	1.06	4.81
	NNS-HMC	0.8096	9.914E-03	2.66	12.09
	NNS-RMHMC	0.8400	3.300E-02	18.68	**84.90**
Elliptic PDE	HMC	0.7077	1.568	0.061	1
	RMHMC	0.8014	4.388	0.228	3.74
	NNS-HMC	0.7138	7.419E-02	1.410	23.11
	NNS-RMHMC	0.6584	9.720E-02	4.375	**71.72**

Variational HMC

Free-form variational Bayes

- Alternatively, we can make inference based on an approximate distribution, similar to variational Bayes, but with a better and more flexible approximation (see for example, de Freitas et al., 2001; Salimans et al., 2015)
- For variational Bayes, we typically use a parametrized distribution $q_\eta(\theta)$ to approximate the target posterior $p(\theta|Y)$ by minimizing the KL divergence
- Here, we use the approximate distribution based our neural network model

$$Q_v(\theta) \propto \exp(-\tilde{U}(\theta)) = \exp[-\sum_{i=1}^{s} v_i g(\boldsymbol{w_i} \cdot \theta + d_i) + \phi(v)]$$

- This is simply a flexible exponential family model

Free-form variational Bayes

- To find Q_v, we follow Hyvarinen (2005) and minimize the score-matching distance

$$\tilde{D}_{SM}(P(\theta|Y)||Q_v(\theta)) = \frac{1}{2}\int Q_v(\theta)\|\nabla_\theta \tilde{U}(\theta) - \nabla_\theta U(\theta)\|^2 \, d\theta$$

- For this, we use HMC to generate samples from Q_v

$$\frac{d\theta}{dt} = \frac{\partial \tilde{H}}{\partial p} = M^{-1}p$$
$$\frac{dp}{dt} = -\frac{\partial \tilde{H}}{\partial \theta} = -\nabla_\theta \tilde{U}(\theta)$$

where the modified Hamiltonian is

$$\tilde{H}(\theta, p) = \tilde{U}(\theta) + K(p)$$

- Then minimize the regularized empirical distance

$$\hat{v} = \arg\min_v \frac{1}{2}\sum_{n=1}^{t} \|\nabla_\theta \tilde{U}(\theta_n) - \nabla_\theta U(\theta_n)\|^2 + \frac{\lambda}{2}\|v\|^2$$

Online updating of the weight vector

- Given the current weight vector $v^{(t)}$ and a new training data point $(\theta_{t+1}, \nabla_\theta U(\theta_{t+1}))$, the updating formula for the estimator is given by

$$v^{(t+1)} = v^{(t)} + W^{(t+1)}(\nabla_\theta U(\theta_{t+1}) - A_{t+1} v^{(t)})$$

where

$$W^{(t+1)} = C^{(t)} A'_{t+1} \left[I_d + A_{t+1} C^{(t)} A'_{t+1} \right]^{-1}$$
$$A_{t+1} = (A_1(\theta_{t+1}), \ldots, A_s(\theta_{t+1}))$$

with $A_i(\theta_{t+1}) := \sigma'(w_i \cdot \theta_{t+1} + d_i) w_i$, and $C^{(t)}$ can be updated by *Sherman-Morrison-Woodbury* formula:

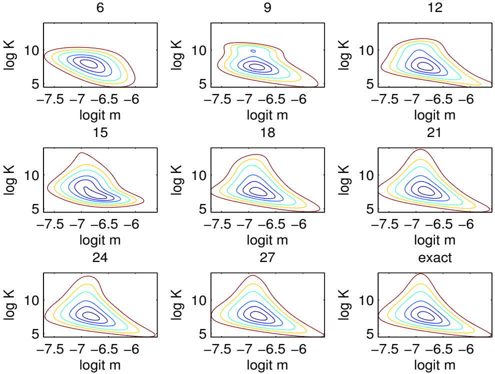
$$C^{(t+1)} = C^{(t)} - W^{(t+1)} A_{t+1} C^{(t)}$$

Example: a beta-binomial model

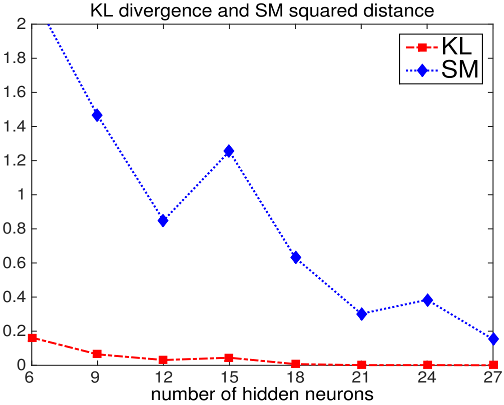
- For illustration, we consider the following beta-binomial model:

$$P(y_j|m, K) = \binom{n_j}{y_j} \frac{B(Km + y_j, K(1-m) + n_j - y_j)}{B(Km, K(1-m))}$$

- The following plot shows approximate posterior distributions for different numbers of hidden neurons (basis functions)

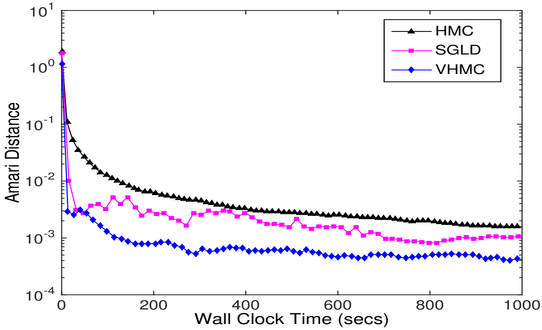


Example: beta-binomial model



Example: Independent Component Analysis

- In this example, we apply ICA to MEG data
- The following plot compares our method to HMC and SGLD (Welling and Teh, 2011) using the Amari distance (Amari et al., 1996), $d_A(\overline{W}, W_0)$, for the unmixing matrix W



Conclusion

- It is essential to develop scalable Bayesian inference methods as high-dimensional problems and complex models become commonplace in scientific studies
- Subsampling strategies have provided promising results
- However, we believe that focusing on parameter space, as opposed to data space, and exploiting its structure and regularity would lead to more reliable methods
- While methods based on surrogate functions could scale well, they might not be very effective for big data analysis
- Our variational HMC method provides a framework to bring together MCMC and Variational Bayes in order to construct robust and scalable Bayesian inference methods with both approximation accuracy and computational efficiency

Conclusion

- It is essential to develop scalable Bayesian inference methods as high-dimensional problems and complex models become commonplace in scientific studies
- Subsampling strategies have provided promising results
- However, we believe that focusing on parameter space, as opposed to data space, and exploiting its structure and regularity would lead to more reliable methods
- While methods based on surrogate functions could scale well, they might not be very effective for big data analysis
- Our variational HMC method provides a framework to bring together MCMC and Variational Bayes in order to construct robust and scalable Bayesian inference methods with both approximation accuracy and computational efficiency

Conclusion

- It is essential to develop scalable Bayesian inference methods as high-dimensional problems and complex models become commonplace in scientific studies
- Subsampling strategies have provided promising results
- However, we believe that focusing on parameter space, as opposed to data space, and exploiting its structure and regularity would lead to more reliable methods
- While methods based on surrogate functions could scale well, they might not be very effective for big data analysis
- Our variational HMC method provides a framework to bring together MCMC and Variational Bayes in order to construct robust and scalable Bayesian inference methods with both approximation accuracy and computational efficiency

Conclusion

- It is essential to develop scalable Bayesian inference methods as high-dimensional problems and complex models become commonplace in scientific studies
- Subsampling strategies have provided promising results
- However, we believe that focusing on parameter space, as opposed to data space, and exploiting its structure and regularity would lead to more reliable methods
- While methods based on surrogate functions could scale well, they might not be very effective for big data analysis
- Our variational HMC method provides a framework to bring together MCMC and Variational Bayes in order to construct robust and scalable Bayesian inference methods with both approximation accuracy and computational efficiency

Conclusion

- It is essential to develop scalable Bayesian inference methods as high-dimensional problems and complex models become commonplace in scientific studies

- Subsampling strategies have provided promising results

- However, we believe that focusing on parameter space, as opposed to data space, and exploiting its structure and regularity would lead to more reliable methods

- While methods based on surrogate functions could scale well, they might not be very effective for big data analysis

- Our variational HMC method provides a framework to bring together MCMC and Variational Bayes in order to construct robust and scalable Bayesian inference methods with both approximation accuracy and computational efficiency

Acknowledgements

- Collaborators:
  - Hongkai Zhao
  - Cheng Zhang
- Fundings:

DMS 1622490

R01 AI107034

Thank you!