Gradient-based Adaptive Markov Chain Monte Carlo

Michalis K. Titsias
DeepMind

Join work with Petros Dellaportas
UCL, AUEB, The Alan Turing Institute

Motivation

Markov chain Monte Carlo (MCMC) requires selecting the proposal distribution

Current adaptive MCMC methods, e.g. Haario et al. (2001) learn online by

- using the history of states $x_{t-1}, x_t, \ldots$
- ignoring important information, e.g. rejected states
- typically not using gradients of the target

Our goal: Learning by optimising a new max entropy regularised objective

- gradient-based stochastic optimisation
- learning from rejected states as well

Brief introduction to MCMC

Assumption: An intractable target distribution $\pi(x)$, $x \in \mathbb{R}^n$

Objective: Simulate samples from $\pi(x)$

Metropolis-Hastings (M-H): Form discrete time Markov chain $\{x_t\}_{t=0}^{\infty}$. At time t propose $y \sim q_\theta(y|x_t)$ and accept with probability

$$\alpha(x_t, y; \theta) = \min\left\{1, \frac{\pi(y)q_\theta(x_t|y)}{\pi(x_t)q_\theta(y|x_t)}\right\}$$

so that $x_{t+1} = y$, otherwise reject and $x_{t+1} = x_t$

Many schemes: Random walk Metropolis (RWM); Metropolis adjusted Langevin (MALA); Hamiltonian Monte Carlo (HMC); Gibbs sampling etc

Exploration vs exploitation

The sampling efficiency depends on the proposal $q_\theta(x|y)$ and the value θ

Adaptive MCMC $\Rightarrow$ online learning of θ as the algorithm runs

What could be a good proposal $q_\theta(x|y)$:

- Exploration: propose big jumps in the state space
- Exploitation: accept jumps with high probability

A simple way to balance between exploration and exploitation

- $\Rightarrow$ is the **speed measure**

Exploration vs exploitation and the speed measure

Speed function or measure (SM)

For RWM proposal $q_\sigma(y|x) = \mathcal{N}(y|x, \sigma^2 I)$, the SM is

$$s_\sigma(x) = \underbrace{\sigma^2}_{\textit{exploration}} \times \underbrace{\alpha(x;\sigma)}_{\textit{exploitation}} = \underbrace{\sigma^2}_{\textit{exploration}} \underbrace{\int \min\left\{1, \frac{\pi(y)q_\sigma(x_t|y)}{\pi(x_t)q_\sigma(y|x_t)}\right\} q_\sigma(y|x_t)dy}_{\textit{exploitation}}$$

where $\alpha(x;\sigma)$ is the average acceptance probability when starting at x

To balance between exploration/exploitation we could do:

- stochastic optimisation steps as the chain runs: x_t, x_{t+1}
- each time take a step towards maximising $s_\sigma(x_t)$ wrt σ

This connects to popular techniques for adaptive MCMC

Exploration vs exploitation and the speed measure

$$s_\sigma(x) = \underbrace{\sigma^2}_{\text{exploration}} \times \underbrace{\alpha(x;\sigma)}_{\text{exploitation}}$$

Say we further average under the stationary distribution $\pi(x)$

$$s_\sigma = \int \pi(x) s_\sigma(x) dx = \sigma^2 \times \int \pi(x)\alpha(x;\sigma) = \sigma^2 \times \underbrace{\alpha_\sigma}_{\text{overall accept. probab.}}$$

Series of theoretical work[1] produced analytical results (for certain targets) showing that when s_σ is maximised $\Rightarrow \alpha^*_\sigma = 0.234$

Similar results have been derived for other algorithms:

- Langevin/MALA $\Rightarrow \alpha^*_\sigma = 0.574$
- HMC $\Rightarrow \alpha^*_\sigma = 0.651$

This leads to popular heuristics $\Rightarrow$ tune step size σ^2 during burn-in to achieve a certain acceptance rate

[1] started with Roberts et al. Weak convergence and optimal scaling of random walk metropolis algorithms, 1997.

Learning more complex proposals

$$s_\sigma(x) = \underbrace{\sigma^2}_{\textit{exploration}} \times \underbrace{\alpha(x;\sigma)}_{\textit{exploitation}}$$

Speed measure is intuitive...

...but only suitable for tuning proposals having a single parameter σ^2

For arbitrary $q_\theta(y|x)$ where θ is a vector we need generalisations

E.g. let $q_\Sigma(y|x) = \mathcal{N}(y|x, \Sigma)$. We need to generalise $s_\sigma(x)$ by replacing σ^2 with some functional $\mathcal{F}(\Sigma)$

A bad choice: Average squared distance $\mathbb{E}[||y - x||^2] = \text{tr}(\Sigma) = \sum_i \sigma_i^2$

- *It can learn very poor proposals. Since the trace is the sum of variances it can obtain high values even when some of the components of x have very low variance, e.g. for some x_i it holds $\sigma_i^2 \approx 0$, which can result in low sampling efficiency or even non-ergodicity*

Learning more complex proposals

Intuition: Good $\mathcal{F}(\Sigma)$ must promote all components of x to jointly move

Better captured by volume/determinant $|\Sigma|$ or more generally by entropy

Generalisation of speed measure:

$$s_\theta(x) = \underbrace{\exp\{\beta \mathcal{H}_{q_\theta(y|x)}\}}_{\mathcal{F}(\theta)=\textit{exploration}} \times \underbrace{\int \alpha(x, y; \theta) q_\theta(y|x) dy}_{\textit{exploitation}}$$

where $\mathcal{H}_{q_\theta(y|x)}$ is the entropy $\mathcal{H}_{q_\theta(y|x)} = -\int q_\theta(y|x) \log q_\theta(y|x) dy$

For full Gaussian $q_\Sigma(y|x) = \mathcal{N}(y|x, \Sigma)$:

$$\mathcal{F}(\Sigma) = \exp\{\beta \mathcal{H}_{q(y|x)}\} = \text{const} \times |\Sigma|^{\frac{\beta}{2}}$$

Hyperparameter β balances the relative strengths of exploration/exploitation

Learning more complex proposals

Generalisation of speed measure:

$$s_\theta(x) = \underbrace{\exp\{\beta \mathcal{H}_{q_\theta(y|x)}\}}_{\textit{exploration}} \times \underbrace{\int \alpha(x, y; \theta) q_\theta(y|x) dy}_{\textit{exploitation}}$$

We need to maximise the objective wrt θ (and also tune β)

- Online as the Markov chain runs
- At each step we collect data:
  $(\textit{current state}, \textit{proposed state}, \textit{accept/reject decision}) = (x_t, y_t, \alpha_t)$
- We will use (x_t, y_t) to make a stochastic gradient update over θ
- Binary decision α_t will be used to update β

Online learning of θ

Apply variational inference + stochastic optimisation

Given current x_t we wish to take a step towards maximising $s_\theta(x_t)$ or its logarithm

$$\log s_\theta(x_t) = \log \int \alpha(x, y; \theta) q_\theta(y|x_t) dy + \beta \mathcal{H}_{q_\theta(y|x_t)}$$

2nd term is the (tractable) entropy of the proposal

We lower bound the 1st term as in variational inference

$\log s_\theta(x_t) \geq \mathcal{F}_\theta(x_t)$:

$$\begin{aligned}
\mathcal{F}_\theta(x_t) &= \int q_\theta(y|x_t) \log \min \left\{ 1, \frac{\pi(y) q_\theta(x_t|y)}{\pi(x_t) q_\theta(y|x_t)} \right\} dy + \beta \mathcal{H}_{q_\theta(y|x_t)} \\
&= \int q_\theta(y|x_t) \min \left\{ 0, \log \frac{\pi(y)}{\pi(x_t)} + \log \frac{q_\theta(x_t|y)}{q_\theta(y|x_t)} \right\} dy + \beta \mathcal{H}_{q_\theta(y|x_t)}
\end{aligned}$$

Online learning of θ

Apply stochastic optimisation as MCMC runs

Say $q_\theta(y|x_t)$ is reparametrisable: $y = \mathcal{T}_\theta(x_t, \epsilon)$, $\epsilon \sim p(\epsilon)$. The reparametrised bound is

$$\mathcal{F}_\theta(x_t) = \int p(\epsilon) \min \left\{ 0, \log \frac{\pi(\mathcal{T}_\theta(x_t, \epsilon))}{\pi(x_t)} + \log \frac{q_\theta(x_t|\mathcal{T}_\theta(x_t, \epsilon))}{q_\theta(\mathcal{T}_\theta(x_t, \epsilon)|x_t)} \right\} d\epsilon + \beta \mathcal{H}_{q_\theta(y|x_t)}.$$

MCMC at the t-th iteration proposes y_t: $\epsilon_t \sim p(\epsilon_t)$, $y_t = \mathcal{T}_\theta(x_t, \epsilon_t)$

Unbiased estimate of the gradient $\nabla_\theta \mathcal{F}_\theta(x_t)$:

$$\nabla_\theta \mathcal{F}_\theta(x_t, \epsilon_t) = \nabla_\theta \min \left\{ 0, \log \frac{\pi(\mathcal{T}_\theta(x_t, \epsilon_t))}{\pi(x_t)} + \log \frac{q_\theta(x_t|\mathcal{T}_\theta(x_t, \epsilon_t))}{q_\theta(\mathcal{T}_\theta(x_t, \epsilon_t)|x_t)} \right\} + \beta \nabla_\theta \mathcal{H}_{q_\theta(y|x_t)}$$

Online learning of θ

$$\nabla_\theta \mathcal{F}_\theta(x_t, \epsilon_t) = \nabla_\theta \min\left\{0, \log \frac{\pi(\mathcal{T}_\theta(x_t, \epsilon_t))}{\pi(x_t)} + \log \frac{q_\theta(x_t|\mathcal{T}_\theta(x_t, \epsilon_t))}{q_\theta(\mathcal{T}_\theta(x_t, \epsilon_t)|x_t)}\right\} + \beta \nabla_\theta \mathcal{H}_{q_\theta(y|x_t)}$$

It is like differentiating through ReLu in neural networks:

$$\nabla_\theta \mathcal{F}_\theta = \begin{cases} \nabla_\theta \log \pi(\mathcal{T}_\theta(x_t, \epsilon_t)) + \nabla_\theta \log \frac{q_\theta(x_t|\mathcal{T}_\theta(x_t,\epsilon_t))}{q_\theta(\mathcal{T}_\theta(x_t,\epsilon_t)|x_t)} + \beta \nabla_\theta \mathcal{H}_{q_\theta(y|x_t)}, & \text{if } \frac{\pi(y_t)}{\pi(x_t)} \frac{q_\theta(x_t|y_t)}{q_\theta(y_t|x_t)} < 1 \\ \beta \nabla_\theta \mathcal{H}_{q_\theta(y|x_t)}, & \text{otherwise} \end{cases}$$

Crucial property: We get stronger gradient when y_t is rejected (or there is a chance to be rejected). I.e. we tend to learn more from rejections than acceptances!

Online learning of hyperparameter β

$$\mathcal{F}_\theta(x_t) = \int q_\theta(y|x_t) \min\left\{0, \log\frac{\pi(y)}{\pi(x_t)} + \log\frac{q_\theta(x_t|y)}{q_\theta(y|x_t)}\right\} dy + \beta\mathcal{H}_{q_\theta(y|x_t)}$$

β trades off between exploitation and exploration

Cannot be optimised by maximising $\mathcal{F}_\theta$

Needs to be updated to control the average acceptance probability $\alpha_* = \mathbb{E}[\alpha_t]$:

- $\alpha_* = 0.25$ for RWM
- $\alpha_* = 0.6$ for Langevin/MALA

Gradient-based adaptive MCMC

Pseudo-code

Input: target $\pi(x)$; reparametrisable proposal $q_\theta(y|x)$ s.t. $y = \mathcal{T}_\theta(x, \epsilon)$, $\epsilon \sim p(\epsilon)$; initial x_0; desired average acceptance probability α_*.
Initialise θ, $\beta = 1$.
for $t = 1, 2, 3, \ldots,$ **do**
: Propose $\epsilon_t \sim p(\epsilon_t)$, $y_t = \mathcal{T}_\theta(x_t, \epsilon_t)$.
: Adapt θ: $\theta \leftarrow \theta + \rho_t \nabla_\theta \mathcal{F}_\theta(x_t, \epsilon_t)$.
: Accept or reject y_t using the standard M-H ratio to obtain x_{t+1}.
: Set $\alpha_t = 1$ if y_t was accepted and $\alpha_t = 0$ otherwise.
: Adapt hyperparameter β: $\beta \leftarrow \beta[1 + \rho_\beta(\alpha_t - \alpha_*)]$ # $\rho_\beta = 0.02$.
end for

We typically perform the two adapt steps during burn-in only. Then adaptation is switched off and we simply collect samples

Gradient-based adaptive MCMC: RWM

Fit a full covariance Gaussian RWM proposal:

$$q_L(y|x) = \mathcal{N}(y|x, LL^\top) \overset{\text{Reparametrisable}}{\longleftrightarrow} y = \mathcal{T}_L(x, \epsilon) = x + L\epsilon,\ \epsilon \sim \mathcal{N}(\epsilon|0, I)$$

When at state x_t the lower bound is

$$\mathcal{F}_L(x_t) = \int \mathcal{N}(\epsilon|0, I) \min\left\{0, \log \pi(x_t + L\epsilon) - \log \pi(x_t)\right\} d\epsilon + \beta \sum_{i=1}^{n} \log L_{ii}$$

By using the proposed $y_t = x_t + L\epsilon_t$ we can obtain the stochastic gradient

$$\nabla_L \mathcal{F}_L(x_t, \epsilon_t) = \begin{cases} \left[\nabla_{y_t} \log \pi(y_t) \times \epsilon_t^\top\right]_{lower} + \beta \text{diag}(\frac{1}{L_{11}}, \ldots, \frac{1}{L_{nn}}), & \text{if } \log \pi(y_t) < \log \pi(x_t) \\ \beta \text{diag}(\frac{1}{L_{11}}, \ldots, \frac{1}{L_{nn}}), & \text{otherwise} \end{cases}$$

where operation $[A]_{lower}$ zeros the upper triangular part of matrix A

Gradient-based adaptive MCMC: Langevin/MALA

Fit a full covariance MALA proposal:

$$q_L(y|x) = \mathcal{N}(y|x + (1/2)LL^\top \nabla_x \log \pi(x), LL^\top)$$

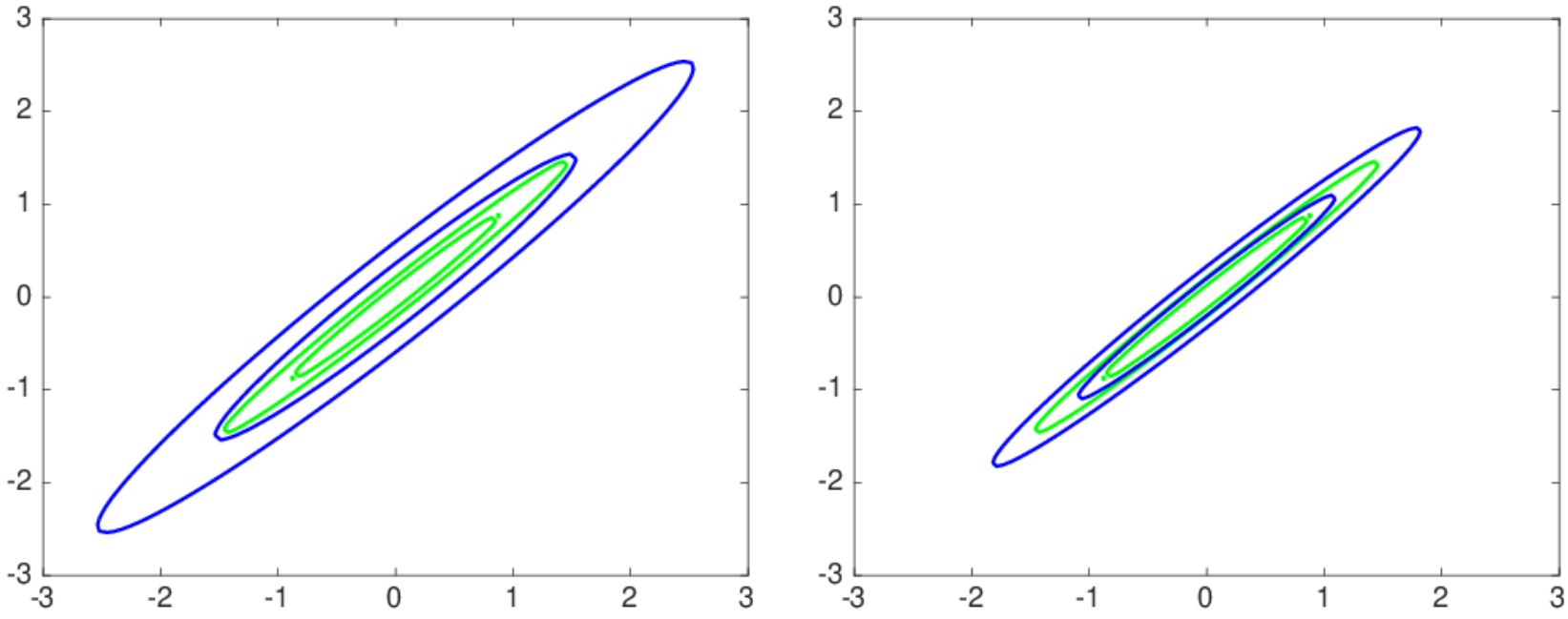
The corresponding stochastic gradient $\nabla_L \mathcal{F}_L(x_t, \epsilon_t)$ of the bound will require the Hessian $\nabla_x \nabla_x \log \pi(x)$

But we can also do a very fast approximation without needing the Hessian (by stopping gradient in the term requiring second derivatives)

Remarkably the fast approximation is not only faster, but tends to be significantly better in terms of sampling efficiency!

Experiments

A correlated 2-D Gaussian target with covariance matrix $\Sigma = [1\ 0.99; 0.99\ 1]$



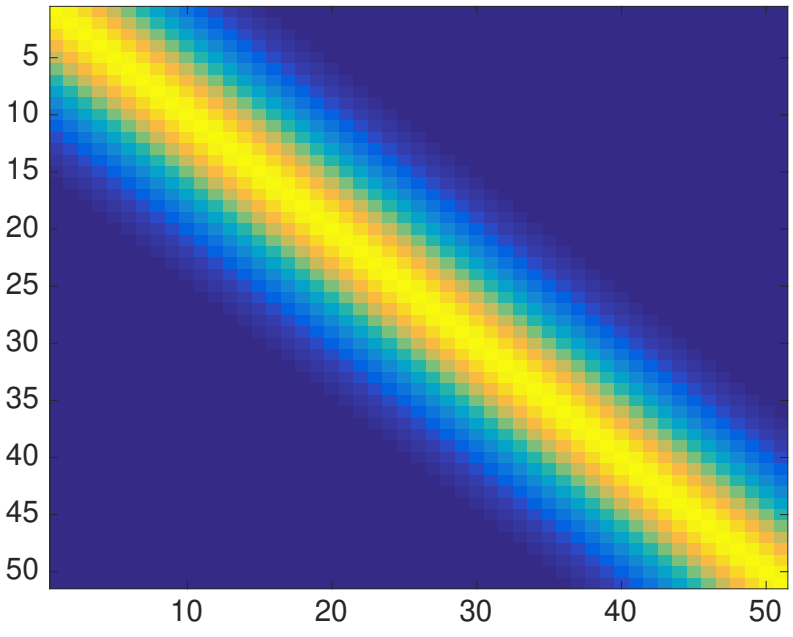
(a) (b)

Green contours show the Gaussian target; Blue contours show the adapted/learned covariance $LL^\top$

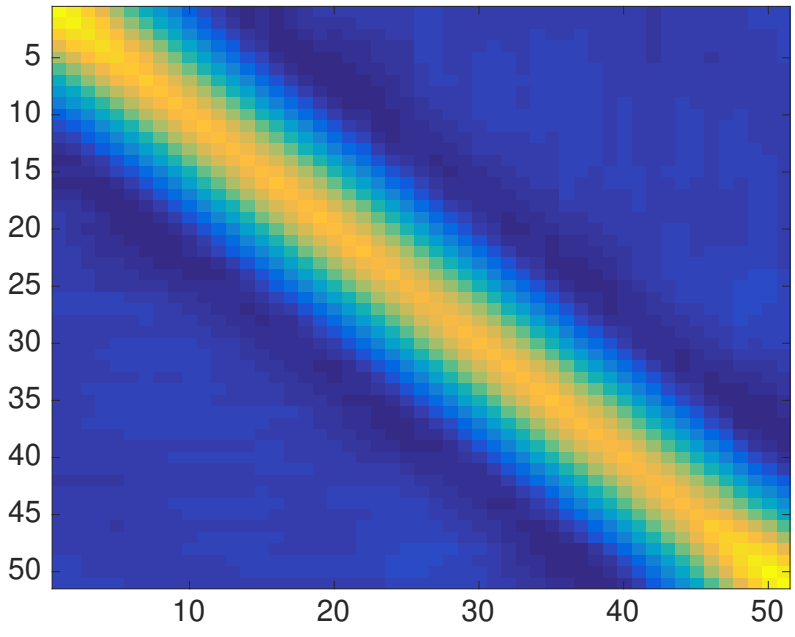
- (a) Targeting average acceptance rate $\alpha_* = 0.25 \Rightarrow$ optimised $\beta_* = 7.4$
- (b) Targeting average acceptance rate $\alpha_* = 0.4 \Rightarrow$ optimised $\beta_* = 2.2$

Experiments

51-D Gaussian target obtained by evaluating the squared exponential kernel plus small noise, i.e. $k(x_i, x_j) = \exp\{-\frac{1}{2}\frac{(x_i - x_j)^2}{0.16}\} + 0.01\delta_{i,j}$



(a)



(b)

- (a) The exact 51×51 covariance matrix
- (b) The adapted/learned covariance after running our most efficient MALA scheme

Experiments

Table: Bayesian binary classification in Caravan dataset consisting of 5822 data points (state/parameter dimensionality $n = 87$).

Method	Time(s)	Accept Rate	ESS (Min, Med, Max)	Min ESS/s (1 st.d.)
gadMALAf	23.1	0.621	(228.1, 750.3, 1114.7)	**9.94** (2.64)
gadMALAe	95.1	0.494	(66.6, 508.3, 1442.7)	0.70 (0.16)
gadRWM	22.6	0.234	(5.3, 34.3, 104.5)	0.23 (0.06)
AM	20.0	0.257	(3.2, 11.8, 62.5)	0.16 (0.01)
RWM	15.3	0.242	(3.0, 9.3, 52.5)	0.20 (0.03)
MALA	22.8	0.543	(4.4, 28.3, 326.0)	0.19 (0.05)
HMC-10	225.5	0.711	(248.3, 2415.7, 19778.7)	1.10 (0.12)
NUTS	1412.1	>0.7	(7469.5, 20000.0, 20000.0)	5.29 (0.38)

Min ESS/s score $= \frac{\textbf{Effective Sample Size}}{\textbf{Overall running time}}$

Experiments

Bayesian binary classification on MNIST digit "5" versus "6"

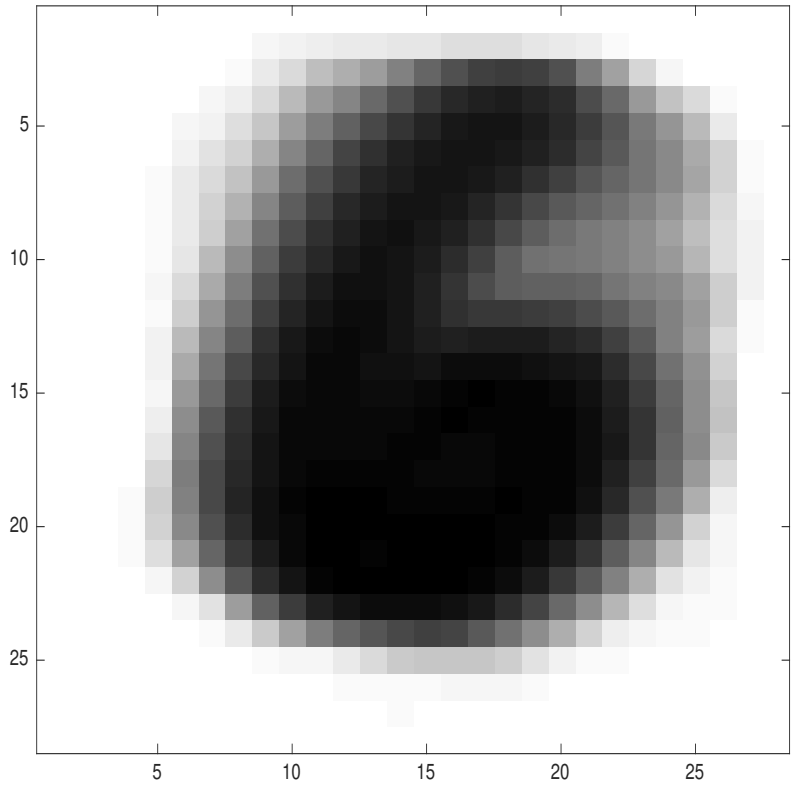


Figure: The 784 diagonal elements of L. Brighter/white colour means larger values. The most efficient gadMLAf was used

- For the border pixels the posterior of the corresponding parameters is close to the prior $\Rightarrow$ automatically learnt by the algorithm

Related work

Other gradient-based adaptive MCMC methods (using different objectives):

- Daniel Levy, Matt D. Hoffman, and Jascha Sohl-Dickstein. Generalizing hamiltonian monte carlo with neural networks. In International Conference on Learning Representations, 2018
- Kirill Neklyudov, Pavel Shvechikov, and Dmitry Vetrov. Metropolis-hastings view on variational inference and adversarial training. arXiv preprint arXiv:1810.07151, 2018

Also recent related methods trying to fit neural-based proposals:

- Jiaming Song, Shengjia Zhao, and Stefano Ermon. A-nice-mc: Adversarial training for mcmc. In Advances in Neural Information Processing Systems, pages 5140–5150, 2017
- Daniel Levy, Matt D. Hoffman, and Jascha Sohl-Dickstein. Generalizing hamiltonian monte carlo with neural networks. In International Conference on Learning Representations, 2018
- Raza Habib and David Barber. Auxiliary variational mcmc. International Conference on Learning Representations, 2019

Discussion

A new framework for gradient-based adaptive MCMC that makes use of an entropy-regularised objective

Future work:

- In high dimensions consider low rank covariance parametrisations
- Apply to HMC
- Learn proposals with state-dependent step-sizes or neural proposals
- Speed up learning using parallel computation
- Investigate theoretical properties/justification

Appendix

$$\int q_\theta(y|x_t) \log \min \left\{ 1, \frac{\pi(y) q_\theta(x_t|y)}{\pi(x_t) q_\theta(y|x_t)} \right\} dy \leq 0$$

For the independent Metropolis sampler the bound (without the entropic term) becomes tight when $q(y) = \pi(y)$.

Also the above is a lower bound of

$$\int q_\theta(y|x_t) \log \frac{\pi(y) q_\theta(x_t|y)}{\pi(x_t) q_\theta(y|x_t)} dy$$