Gradient-based Adaptive Markov Chain Monte Carlo

Summary of the method

Concepts of adaptive MCMC and the speed measure

The M-H algorithm defines a Markov chain that converges to $\pi(x)$

The sampling/statistical efficiency heavily depends on the proposal $q_\theta(x|y)$ and the parameters θ

Adaptive MCMC: online learning of θ as the algorithm runs

Intuition behind learning a good $q_\theta(x|y)$:

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

Concepts of adaptive MCMC and the speed measure

MCMC optimal adaptation/tuning[1] makes use of speed measure (SM)

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

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

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

$$\alpha(x;\sigma) = \int \alpha(x,y;\sigma) q_\sigma(y|x) dy$$

In optimal MCMC tuning, $s_\sigma(x)$ is averaged under the stationary distribution $\pi(x)$

$$s_\sigma = \int \pi(x) s_\sigma(x) dx$$

and then is maximised wrt σ

---

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

Concepts of adaptive MCMC and the speed measure

In optimal MCMC tuning, $s_\sigma(x)$ is averaged under the stationary distribution $\pi(x)$

$$s_\sigma = \int \pi(x) s_\sigma(x) dx$$

and then is maximised wrt σ

For certain targets and with increasing dimensions (Roberts et at, 1997) SM is maximised so that σ is set to a value that leads to acceptance probability 0.234

Similar results have been derived for other algorithms

- MALA: 0.54 is considered optimal
- HMC: 0.67 is considered optimal

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

Concepts of adaptive MCMC and the speed measure

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

While the current notion of speed measure is intuitive...

...it is only suitable for tuning proposals having a single step size

To learn arbitrary proposals $q_\theta(y|x)$, where θ is a vector of parameters, we need to generalise the speed measure

E.g. suppose $q_\Sigma(y|x) = \mathcal{N}(y|x, \Sigma)$. We need to generalise $s_\sigma(x)$ by replacing σ^2 with functional $\mathcal{F}(\Sigma)$ that depends on the full covariance

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

- it can lead to learning very poor proposals. E.g. 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 very low sampling efficiency or even non-ergodicity

The generalised speed measure

Intuition: A good functional $\mathcal{F}(\Sigma)$ must promote all components of x to jointly perform (relative to their scale) big jumps

This is better captured by the volume/determinant $|\Sigma|$ or more generally by the entropy

Generalised speed measure:

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

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

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

The hyperparameter β balances the relative strengths of the two terms

- As discussed next we can efficiently optimise β to achieve a desirable average acceptance rate

The generalised speed measure

Generalised speed measure:

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

Now that we have defined our objective we need to maximise it wrt θ

- This will be done in online fashion as the Markov chains runs
- At each step we collect data:
  $(\textit{current state}, \textit{proposed state}) = (x_t, y_t)$
- We will use (x_t, y_t) to make a stochastic gradient update

A crucial property of the method is that will learn even when y_t is rejected: in fact we will tend to learn more from rejections than acceptances!

Maximising the generalised speed measure using variational inference

Online learning of θ: Given the current x_t we wish to take a step towards maximising $s_\theta(x_t)$ or its logarithm,

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

The second term is just the (tractable) entropy of the proposal

For the first we work similarly to variational inference (apply Jensen's inequality)

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

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

Maximising the generalised speed measure using variational inference

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

To take a step towards maximising $\mathcal{F}_\theta$ we can apply stochastic gradient variational inference techniques

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

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

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

An unbiased estimate of $\nabla_\theta\mathcal{F}_\theta(x_t)$ can be obtained by

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

Maximising the generalised speed measure using variational inference

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

The first term is like differentiating through ReLu in neural networks:

- if $\log \frac{\pi(y_t)}{\pi(x_t)} + \log \frac{q_\theta(x_t|y_t)}{q_\theta(y_t|x_t)} \geq 0$ the gradient is 0 (this is the case when y_t is accepted with probability 1)
- Otherwise the gradient is

$$\nabla_\theta \log \pi(\mathcal{T}_\theta(x_t, \epsilon_t)) + \nabla_\theta \log \frac{q_\theta(x_t|\mathcal{T}_\theta(x_t, \epsilon_t))}{q_\theta(\mathcal{T}_\theta(x_t, \epsilon_t)|x_t)}$$

β trades off between large acceptance probability (exploitation) and large entropy (exploration):

- cannot be optimised by maximising $\mathcal{F}_\theta$
- needs to be updated to control the overall acceptance probability of the chain in order to achieve a certain desired value α_*

Gradient-based adaptive MCMC

Algorithm 1 Gradient-based Adaptive MCMC

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: $\beta \leftarrow \beta[1 + \rho_\beta(\alpha_t - \alpha_*)]$ # default value for $\rho_\beta = 0.02$.
end for

Gradient-based adaptive MCMC

Fit a full covariance Gaussian RWM proposal:

$$q_L(y|x) = \mathcal{N}(y|x, LL^\top), \quad L \text{ is Cholesky factor}$$

Reparametrisable: $y \equiv \mathcal{T}_L(x, \epsilon) = x + L\epsilon$, $\epsilon \sim \mathcal{N}(\epsilon|0, I)$

At t-th iteration when the state is x_t the lower bound becomes

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

By using the proposed $y_t = x_t + L\epsilon_t$ we can obtain an unbiased gradient estimate,

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

where $y_t = x_t + L\epsilon_t$ and operation $[A]_l$ zeros the upper triangular part

Gradient-based adaptive MCMC

Fit a full covariance MALA proposal:

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

This will require the Hessian of the target $\nabla_x \log \pi(x)$

But we can also do a very fast approximation without needing the Hessian

More details are in the paper

Experiments

We investigate two instances of gradient-based adaptive MCMC:

1. Gradient-based adaptive random walk (gadRWM)
2. Corresponding MALA (gadMALA). We consider two versions:
   - The exact algorithm that requires the evaluation of the Hessian (gadMALAe)
   - A fast approximate variant (gadMALAf)

Compare against:

1. RWM: $\mathcal{N}(y|x, \sigma^2 I)$
2. Traditional (non-gradient-based) adaptive MCMC (AM) that fits $\mathcal{N}(y|x, LL^\top)$
3. Standard MALA: $\mathcal{N}(y|x + (1/2)\sigma^2 \nabla \log \pi(x), \sigma^2 I)$
4. HMC with a fixed number of leap frog steps: either 5, or 10, or 20
5. No-U-turn sampler (NUTS): HMC that automatically determines the number of leap frog steps

Experiments

- A correlated 2-D Gaussian target with covariance matrix $\Sigma = [1\ 0.99; 0.99\ 1]$
- 51-D Gaussian target obtained by evaluating the squared exponential kernel plus small noise, i.e. $k(x_i, x_j) = \exp\{-\frac{1}{2}\frac{(x_i - x_j)^2}{0.16}\} + 0.01\delta_{i,j}$

In both experiments L was initialised to diagonal matrix with $0.1/\sqrt{n}$ in the diagonal

Experiments

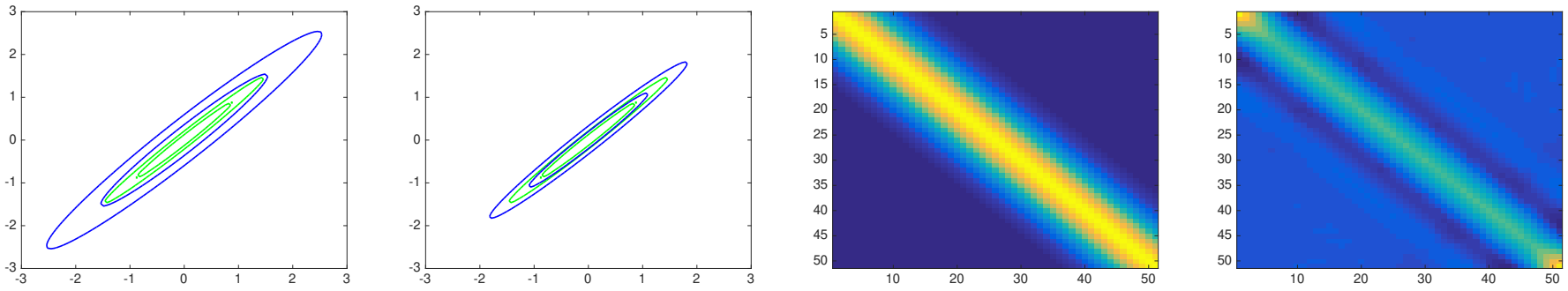

Figure: The green contours in the first two panels (from left to right) show the 2-D Gaussian target, while the blue contours show the learned covariance, $LL^\top$, after adapting for 2×10^4 iterations using gadRWM and targeting acceptance rates $\alpha_* = 0.25$ and $\alpha_* = 0.4$. For $\alpha_* = 0.25$ the adapted blue contours show that the proposal matches the shape of the target but it has higher entropy/variance and the hyperparameter β obtained the value 7.4. For $\alpha_* = 0.4$ the blue contours shrink a bit and β is reduced to 2.2 (since higher acceptance rate requires smaller entropy). The third panel shows the exact 51×51 covariance matrix and the last panel shows the adapted one, after running our most efficient gadMALAf scheme for 2×10^5 iterations.

Experiments

Radford Neal's Gaussian target

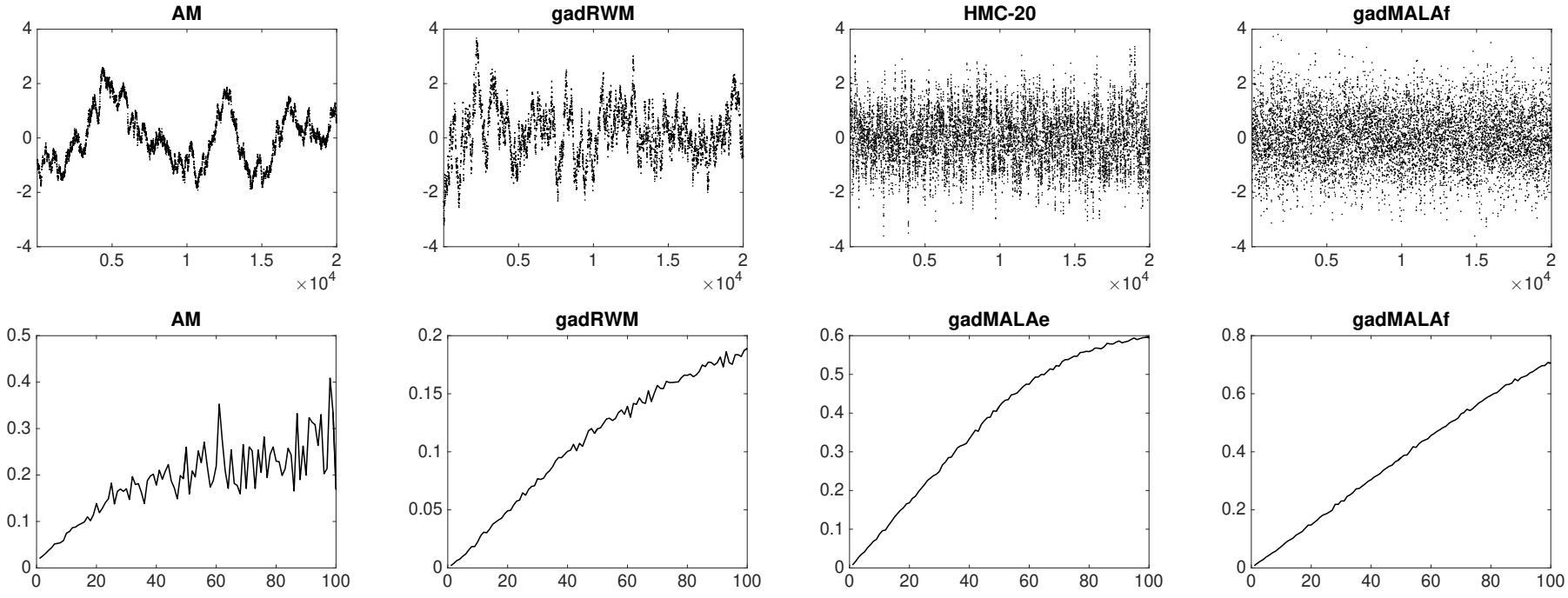

Figure: Panels in the first row show trace plots, obtained by different schemes, across the last 2×10^4 sampling iterations for the most difficult to sample x_{100} dimension. The panels in the second row show the estimated values of the diagonal of L obtained by different adaptive schemes. **The real Gaussian target has diagonal covariance matrix $\Sigma = \mathbf{diag}(s_1^2, \ldots, s_{100}^2)$ where s_i are uniform in the range $[0.01, 1]$.**

Experiments

Table 1: Comparison in Neal's Gaussian example (dimensionality was $n = 100$; see panel above) and Caravan binary classification dataset where the latter consists of 5822 data points (dimensionality was $n = 87$; see panel below). All numbers are averages across ten repeats where also one-standard deviation is given for the Min ESS/s score. From the three HMC schemes we report only the best one in each case.

Method	Time(s)	Accept Rate	ESS (Min, Med, Max)	Min ESS/s (1 st.d.)
(Neal's Gaussian)				
gadMALAf	8.7	0.563	(1431.2, 2001.2, 2470.5)	**165.08** (18.80)
gadMALAe	10.1	0.535	(940.5, 2006.3, 2747.6)	93.67 (10.28)
gadRWM	6.3	0.252	(24.9, 65.4, 122.2)	3.93 (0.71)
AM	2.3	0.257	(8.7, 48.6, 829.1)	3.83 (0.89)
RWM	2.1	0.261	(2.9, 8.4, 2547.6)	1.38 (0.06)
MALA	3.0	0.530	(2.9, 10.0, 12489.2)	0.99 (0.03)
HMC-20	47.4	0.694	(306.1, 1537.8, 19732.4)	6.47 (3.52)
NUTS	360.5	>0.7	(18479.6, 20000.0, 20000.0)	51.28 (1.64)
(Caravan)				
gadMALAf	24.8	0.599	(204.8, 785.9, 1129.5)	**8.30** (2.34)
gadMALAe	91.5	0.492	(56.4, 496.4, 1420.2)	0.62 (0.16)
gadRWM	23.0	0.227	(5.6, 35.5, 98.8)	0.24 (0.04)
AM	18.2	0.257	(3.2, 11.8, 62.5)	0.18 (0.02)
RWM	17.4	0.242	(3.0, 9.3, 52.5)	0.17 (0.02)
MALA	22.5	0.543	(4.4, 28.3, 326.0)	0.20 (0.06)
HMC-10	225.8	0.711	(248.3, 2415.7, 19778.7)	1.10 (0.14)
NUTS	1412.1	>0.7	(7469.5, 20000.0, 20000.0)	5.29 (0.38)

Experiments

Bayesian logistic regression on MNIST: "5" versus "6", consisted of 11339 data points. The size of the state x was $n = 785$

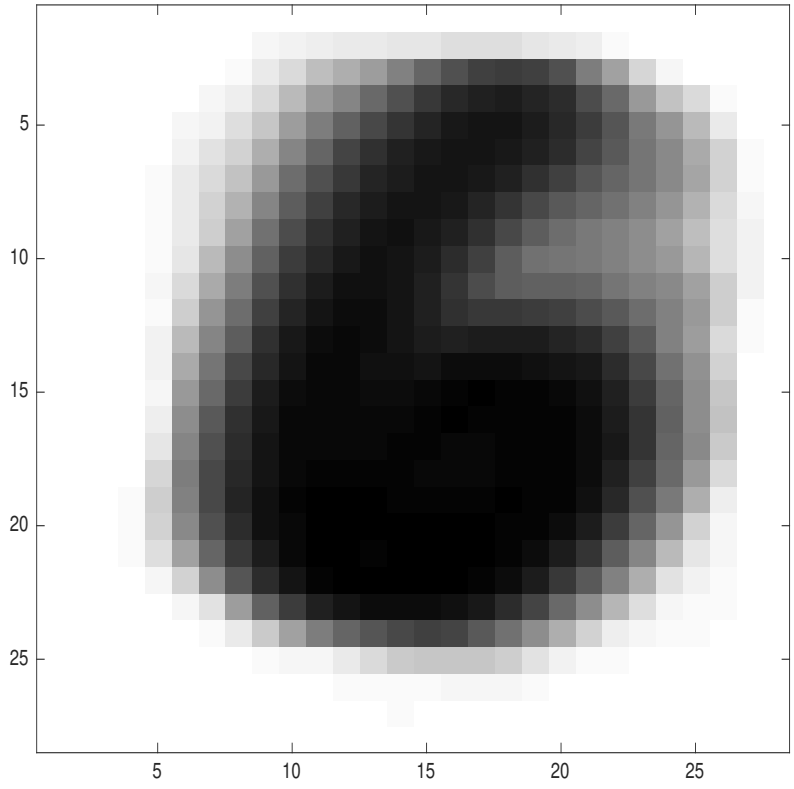

Figure: The first 784 diagonal elements (i.e. excluding the bias component of x) of the full 785×785 Cholesky factor L found after 5×10^4 adapting iterations by gadMALAf. Brighter/white colour means larger values.