Optimal scaling and dynamic-based methods

Lecturer: Riccardo Corradin

Optimal scaling

-

Random walk Metropolis-Hastrings

- Let us consider a **random walk Metropolis-Hastings** (RWMH) algorithm and let $\boldsymbol{\theta} = \boldsymbol{\theta}^{(r)}$ be the current status of the chain.

- It is called random walk when the proposal distribution is **symmetric** and **centered at the current state**, and going on with the sampling we "walk randomly" over the support of interest.

- For example, we can consider a Gaussian distribution centered on $\boldsymbol{\theta}^{(r)}$,

$$\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(r)} \sim N_p(\boldsymbol{\theta}^{(r)}, S), \qquad \text{implying that} \qquad q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}) = q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*).$$

- In this specific case, the **acceptance probability** simplifies and we get

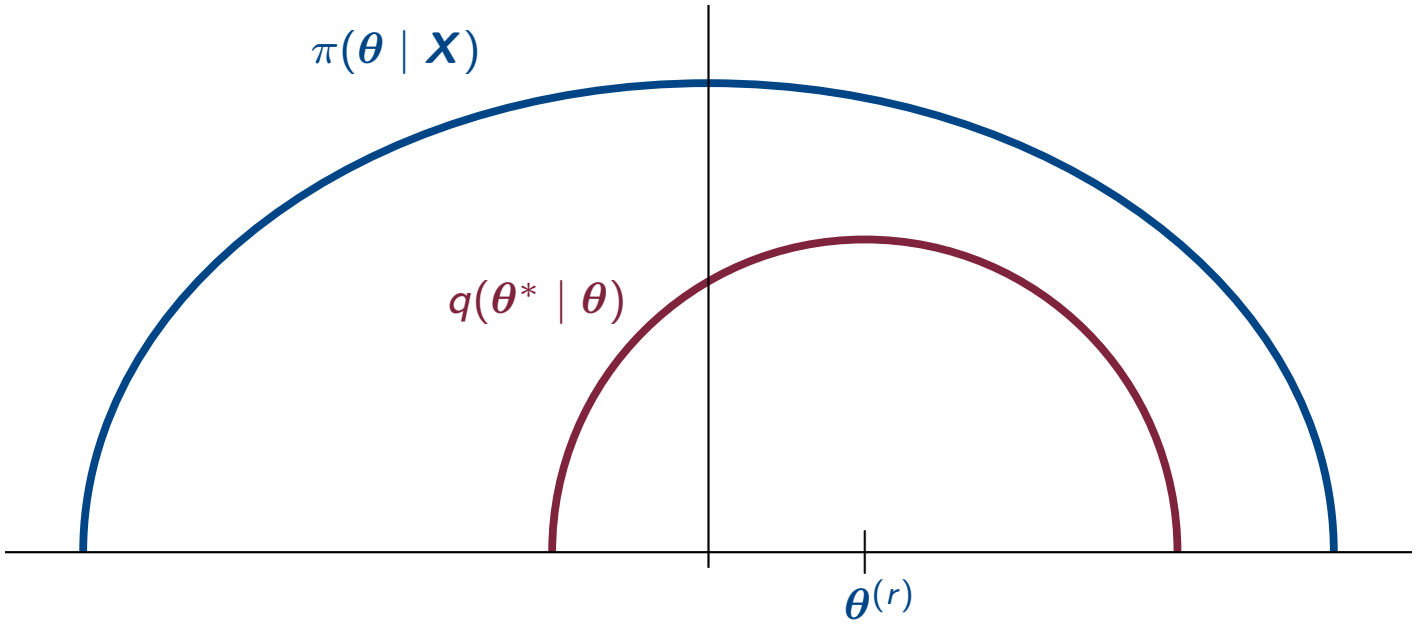
$$\alpha = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^* \mid \boldsymbol{X}) q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta} \mid \boldsymbol{X}) q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})}\right\} = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^* \mid \boldsymbol{X})}{\pi(\boldsymbol{\theta} \mid \boldsymbol{X})}\right\}$$

- The proposal distribution is a sensible choice especially whenever the support of $\boldsymbol{\theta}$ is unbounded.

Metropolis-Hastings algorithm

- **At each step**
  1. Generate y from $q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$
  2. If $U \leq \alpha$, with $U \sim \mathit{Unif}(0, 1)$, set $\boldsymbol{\theta}^{(r+1)} = \boldsymbol{\theta}^*$, otherwise $\boldsymbol{\theta}^{(r+1)} = \boldsymbol{\theta}^{(r)}$.

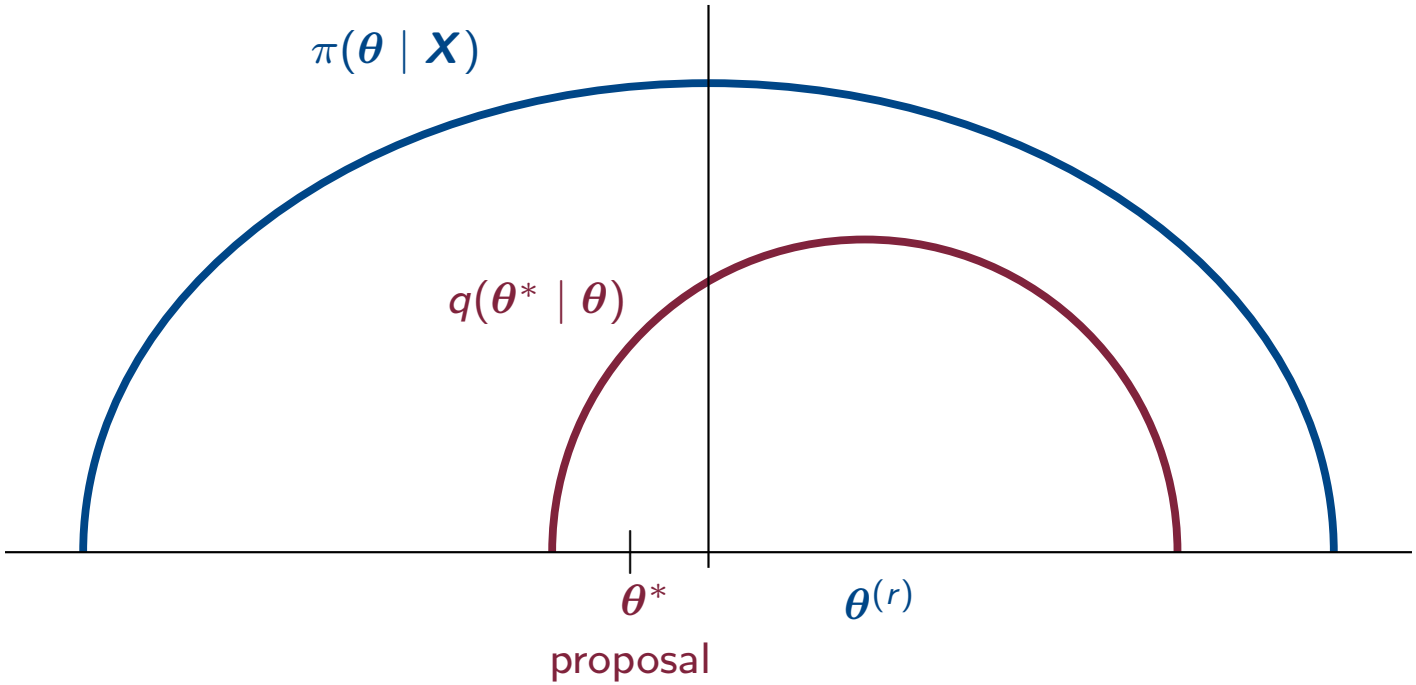
- **How it works**



Metropolis-Hastings algorithm

- **At each step**
  1. Generate y from $q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$
  2. If $U \leq \alpha$, with $U \sim \mathit{Unif}(0, 1)$, set $\boldsymbol{\theta}^{(r+1)} = \boldsymbol{\theta}^*$, otherwise $\boldsymbol{\theta}^{(r+1)} = \boldsymbol{\theta}^{(r)}$.

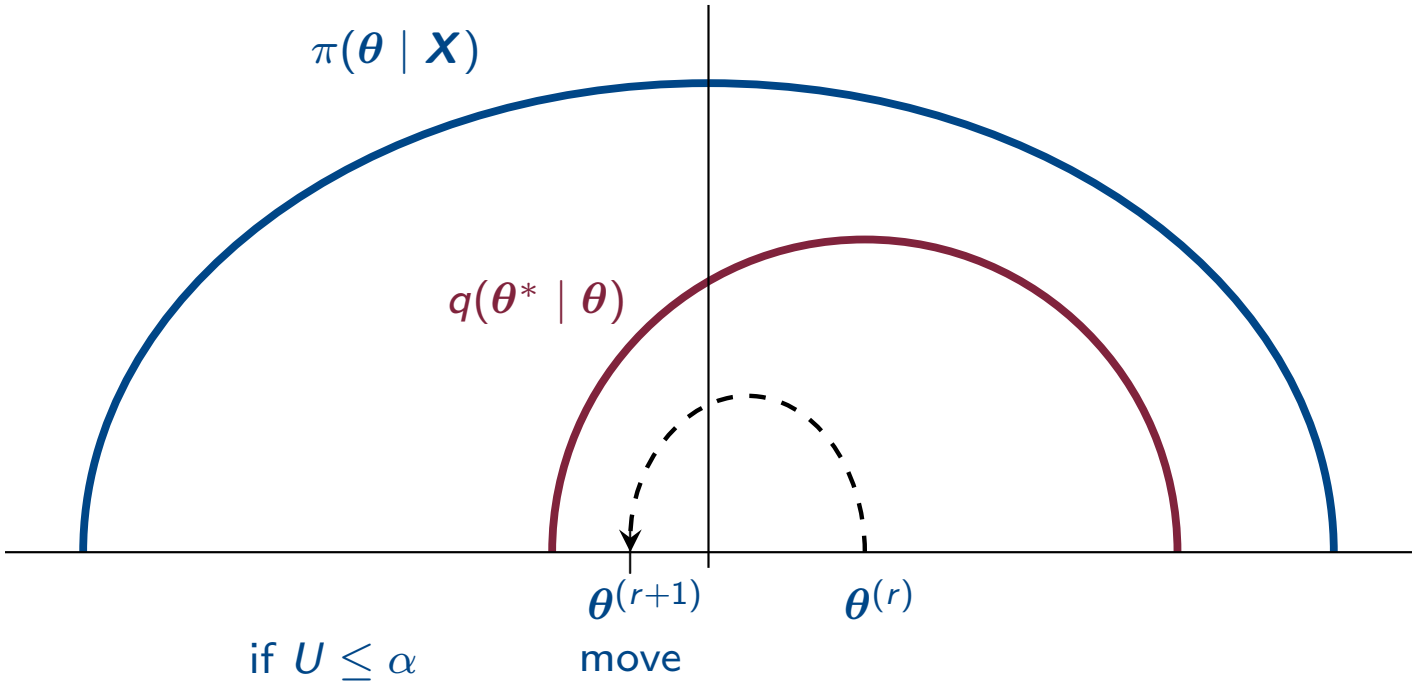
- **How it works**



Metropolis-Hastings algorithm

- **At each step**
  1. Generate y from $q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$
  2. If $U \leq \alpha$, with $U \sim \mathit{Unif}(0, 1)$, set $\boldsymbol{\theta}^{(r+1)} = \boldsymbol{\theta}^*$, otherwise $\boldsymbol{\theta}^{(r+1)} = \boldsymbol{\theta}^{(r)}$.

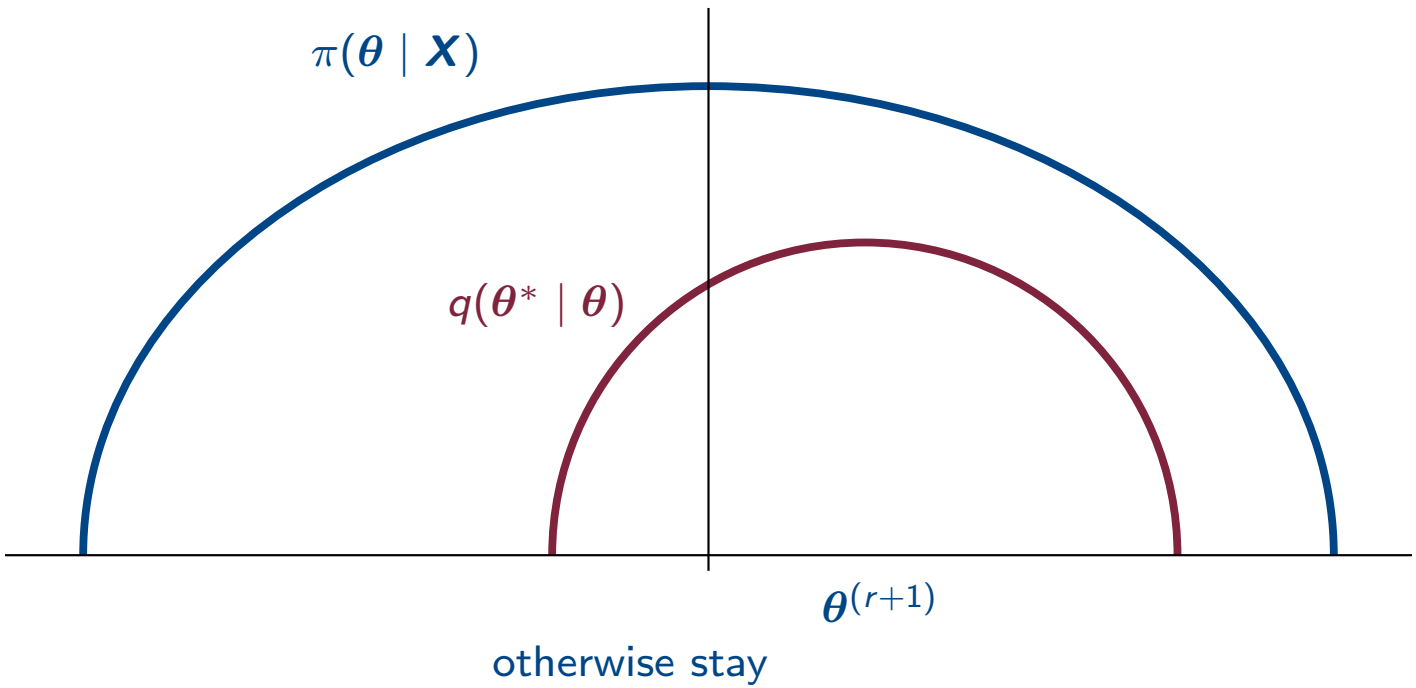
- **How it works**



Metropolis-Hastings algorithm

- **At each step**
  1. Generate y from $q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$
  2. If $U \leq \alpha$, with $U \sim Unif(0, 1)$, set $\boldsymbol{\theta}^{(r+1)} = \boldsymbol{\theta}^*$, otherwise $\boldsymbol{\theta}^{(r+1)} = \boldsymbol{\theta}^{(r)}$.

- **How it works**



Optimal choice of the proposal distribution

- Among all the possible proposal densities for $q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$, we restrict our focus on multivariate Gaussians centered on $\boldsymbol{\theta}$.
- Despite this important simplification, choosing the covariance matrix S remains a difficult task and crucially affects the performance.
- In the **univariate / bivariate** cases, one could tune the variance of the proposal distribution S by **trial and error** and with some patience.
- Unfortunately, whenever the parameter's dimension is large, the "manual" elicitation of the matrix S is almost impossible.
- Can we identify an **ideal covariance** matrix S that is optimal in some sense? Can we "estimate" it from the data?

Asymptotic variance

- For an arbitrary squared-integrable function $g(\cdot) : \mathbb{R}^p \to \mathbb{R}$, let us consider the following Monte Carlo estimator

$$\hat{\eta}_g = \frac{1}{R} \sum_{r=1}^{R} g(\boldsymbol{\theta}^{(r)}),$$

  for the posterior expectation $\eta_g = \mathrm{E}[g(\boldsymbol{\theta}) \mid \boldsymbol{X}]$.

- If a central limit theorem holds, we have the following limit in distribution

$$\sqrt{R} \frac{\hat{\eta}_g - \eta_g}{\sigma_g} \xrightarrow{d} Z, \qquad Z \sim N(0, 1),$$

  where σ_g^2 is the so-called **asymptotic variance** of the MCMC.

- Intuitively, we seek a covariance matrix S that minimizes the asymptotic variance σ_g^2.

- Other measures of "optimality" can be defined, but it can be shown they are all equivalent asymptotically (for large values of p).

Asymptotic variance

- Let $\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)}, \ldots$ be sample from a Markov chain sample, with $\boldsymbol{\theta}^{0)} \sim \pi(\boldsymbol{\theta} \mid \boldsymbol{X})$ being a sample from the stationary distribution.
- The asymptotic variance can be written as follows

$$\sigma_g^2 = \mathrm{Var}(g(\boldsymbol{\theta}^{(0)}) \mid \boldsymbol{X})\tau_g = \mathrm{Var}(g(\boldsymbol{\theta}^{(0)}) \mid \boldsymbol{X}) \left[1 + 2\sum_{j=1}^{\infty} \mathrm{Corr}(g(\boldsymbol{\theta}^{(0)}), g(\boldsymbol{\theta}^{(r)}))\right].$$

- The quantity τ_g is sometimes called **integrated autocorrelation** and measures the **loss of efficiency** with respect to independent (iid) sampling ($\tau_g = 1$).
- When $\tau_g = 1$, the MCMC algorithm is "optimal" and there is no autocorrelation.
- Rarely, one could obtain $\tau_g < 1$, which is indeed an improvement over iid sampling.
- The effectiveSize R command produces an estimate of $R\tau_g^{-1}$ from the empirical samples of the chain.

Optimal scaling

- The relationship between the matrix S and the asymptotic variance σ_g^2 **is unclear**. In addition, the variance σ_g^2 depends on the chosen function $g(\cdot)$.

- Let us initially assume that the posterior distribution has the following form

$$\pi(\boldsymbol{\theta} \mid \boldsymbol{X}) = \prod_{j=1}^{p} f(\theta_j), \qquad \mathrm{Var}(\boldsymbol{\theta} \mid \boldsymbol{X}) = \sigma I_p,$$

  meaning that the components of $\boldsymbol{\theta}$ are independent and identically distributed from some density f.

- Moreover, we consider the following proposal distribution

$$\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(r)} \sim N_p(\boldsymbol{\theta}^{(r)}, s_p^2 I_p), \qquad s_p^2 = \ell^2/p^2.$$

- In this simplified setting, we seek an **optimal scaling** value for ℓ^2.

Diffusion processes

- This problem simplifies remarkably in the **asymptotic** regime, as p diverges.
- Let us define a continuous-time stochastic process from the first component θ_1 of the $\boldsymbol{\theta} = (\theta_1, ..., \theta_p)$, namely

$$Z^{(t)} = \theta_1^{([tp])},$$

  where $[\cdot]$ denotes the integer part function.
- That is, $Z^{(t)}$ is a **speeded-up** continuous-time version of the original Markov chain, parametrized to make jumps every p time units.
- We need some smoothness conditions on the density f (Roberts et al., 1997), and in particular we assume that

$$\mathcal{I} = \mathrm{E}\left[\left(\frac{f'(\theta_1)}{f(\theta_1)}\right)^2\right] < \infty.$$

- Note that, in the Gaussian case, $\mathcal{I} = \sigma^{-2}$ corresponds to the precision.

Diffusion processes

Theorem (Roberts and Rosenthal, 1997)

Let $B^{(t)}$ be the standard Brownian motion and let $\Phi(\cdot)$ be the cdf of a standard Gaussian. The continuous-time stochastic process Z weakly converges to

$$Z \xrightarrow{d} W, \qquad p \to \infty,$$

where W is a diffusion process satisfying the stochastic differential equation

$$\mathrm{d}W^{(t)} = h(\ell)^{1/2} \dot{B}^{(t)} + \frac{h(\ell) \nabla \log f(W^{(t)})}{2} \mathrm{d}t,$$

where the speed of the diffusion is

$$h(\ell) = \ell^2 2\Phi\left(-\frac{\mathcal{I}^{1/2}\ell}{2}\right),$$

for some constant $\mathcal{I}$ that depends solely on f.

- All the involved quantities have a clear interpretation in terms of the original RWMH algorithm.

Speed of the diffusion $h(\ell)$

- The speed of the diffusion $h(\ell)$is strictly related to the asymptotic variance of the MCMC algorithm.
- Recall that we aim at finding an **optimal value** for ℓ that minimizes the autocorrelation.
- In the first place, note that for small $\epsilon > 0$, it holds that

$$\mathrm{Corr}(g(W^{(0)}), g(W^{(\epsilon)})) \approx 1 - \epsilon B_g h(\ell),$$

  where B_g is a **constant term** not depending on ℓ.
- Let $\tau_g(\ell)$ be the integrated autocorrelation time of the RWMH. Then, for large p it holds that

$$\tau_g(\ell)^{-1} \approx h(\ell) e_g p^{-1},$$

  where $e_g > 0$ is some constant depending only on $g(\cdot)$.
- The maximization of the diffusion speed is equivalent to the minimization of the autocorrelation for any function $g(\cdot)$.

Speed of diffusion and acceptance rate

- Let us define the acceptance rate of the original p-dimensional RWMH as

$$A_p(\ell) = \lim_{R\to\infty} \frac{\text{"\# of accepted moves}}{R},$$

  namely the long-term proportion of accepted moves.

- Then, it can be shown that

$$\lim_{p\to\infty} A_p(\ell) = A_\ell = 2\Phi\left(-\frac{\mathcal{I}^{1/2}\ell}{2}\right).$$

- Moreover, recall the definition of the speed of diffusion

$$h(\ell) = \ell^2 2\Phi\left(-\frac{\mathcal{I}^{1/2}\ell}{2}\right) = \ell^2 A(\ell),$$

  implying that the **speed of the diffusion** is strictly related to the **acceptance rate**.

Important consequences

- Hence, we consider ℓ maximizing the speed of diffusion $h(\ell)$, obtaining the **optimal scaling**

$$\ell_{\text{opt}} \approx \frac{2.38}{\mathcal{I}^{1/2}}.$$

- The acceptance rate, evaluated at the optimal scaling, is

$$A(\ell_{\text{opt}}) \approx 0.234$$

  corresponding to the **optimal acceptance rate**.

- This suggests the following optimal proposal variance for $\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(r)} \sim N_p(\boldsymbol{\theta}^{(r)}, s_p^2 I_p)$ for large values of p, with

$$s_p^2 = 2.38^2 (p\mathcal{I})^{-1},$$

  where $\mathcal{I}$ must be estimated/guessed somehow.

- If f is a Gaussian density with variance σ^2, then we obtain $s_p^2 = 2.38^2 \sigma^2 p^2$.

Getting practical

- These results are **asymptotic** (large p) and require that the posterior distribution has **independent components**.
- In practice, when $p \approx 5$, the optimal acceptance rate is close to 0.234 based on simulation studies (Gelman et al., 1996).
- When $p = 1$ the **optimal acceptance rate is higher** and about 0.44.
- If the posterior distribution is Gaussian with $p \times p$ covariance matrix Σ, it suffices to translate the components and **rotate the axes** according to Σ to make the components iid, leading to

$$\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(r)} \sim N_p(\boldsymbol{\theta}^{(r)}, S), \qquad S = 2.38^2 \Sigma / p.$$

- This procedure is optimal for large p, although it requires the knowledge of Σ.

Binary regression

- Let $\boldsymbol{y}^\intercal = (y_1, \ldots, y_n)$ be a vector of the observed binary responses.
- Let X be the corresponding **design matrix** whose generic row is $x_i = (x_{i1}, \ldots, x_{ip})$, for $i = 1, \ldots, n$. All predictors have been **standardized**.
- We consider a generalized linear model such that

$$y_i \mid \lambda_i \overset{ind}{\sim} Bern(\lambda_i), \qquad \lambda_i = g(\eta_i), \qquad \eta_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip},$$

  where $g(\cdot) : \mathbb{R} \to [0, 1]$. Here, we focus here on the **logistic regression case**, i.e. with

$$g(\eta_i) = \frac{e^{\eta_i}}{1 + e^{\eta_i}}.$$

- We aim at estimating the parameter vector $\boldsymbol{\beta}^\intercal = (\beta_1, \ldots, \beta_p)$ using a RWMH.
- We will employ a relatively vague prior centered at $\mathbf{0}$, namely

$$\boldsymbol{\beta} \sim N_p(\mathbf{0}, 100 I_p).$$

The Pima Indian dataset

- During the course, we will test several algorithms on the "famous" **Pima Indian** dataset, with $n = 532$ and $p = 8$.
- The purpose of this exercise is mainly to present the implementation of the various mcmc algorithms and show their performance in this specific example.
- **The following results should not be generalized** to any statistical models nor even to any logistic regression model.
- Depending on the sample size n, the dimension of the parameter space p, as well as the dependence structure of the predictor, the results may vary significantly.
- Refer to the nice paper by Chopin & Ridgway (2017) for a more comprehensive discussion on this aspect.

Computational details

- Recall that at each step of the algorithm, we need a sample from a multivariate Gaussian distribution $N_p(\mathbf{0}, S)$.
- Albeit tempting, using built-in R functions such as rmvnorm and mvrnorm leads to a sensible **waste of computing time**.
- Indeed, in order to get a sample from $\boldsymbol{V} \sim N_p(\boldsymbol{\mu}, S)$, one needs to compute

$$\boldsymbol{V} = \boldsymbol{\mu} + A\boldsymbol{Z},$$

  where $Z \sim N_p(\mathbf{0}, I_p)$ is standard Gaussian and A is a $p \times p$ matrix such that $AA^\intercal = S$.
- Hence, there is no need to compute A at every step, as this can be done **before running the MCMC**.

```
A <- chol(S) # Cholesky decomposition of S (outside the MCMC)
V <- mu + crossprod(A, rnorm(2)) # Sample from V (inside the MCMC)
```

Naive covariance matrix

- Let us start with a naive choice for the proposal covariance $S = 10^{-3} I_p$.
- Albeit being sub-optimal, this "random" choice of S works decently.

```
# Covariance matrix of the proposal
S <- diag(1e-3, ncol(X))

# Running the MCMC (R = 30000, burn_in = 30000)
fit_MCMC <- as.mcmc(RMH(R, burn_in, y, X, S)) # Convert the matrix into a "coda" object

summary(effectiveSize(fit_MCMC)) # Effective sample size (beta)
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 174.9 205.0 258.5 259.6 320.7 333.1

summary(R / effectiveSize(fit_MCMC)) # Integrated autocorrelation time (beta)
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 90.06 93.56 119.31 122.76 146.43 171.52

print(1 - rejectionRate(fit_MCMC)) # Acceptance rate
# 0.7191
```

Approximating the posterior covariance matrix

- We know that a **sensible choice** for S would be based on **posterior covariance matrix** Σ.
- The true Σ is unknown and therefore we need to rely on some (fast) approximation
- A possibility is based on a **quadratic approximation** of the likelihood function, evaluated at the **maximum likelihood estimate**.
- This is particularly simple in the logistic regression case (do it as an exercise!) since we can set

$$\hat{\Sigma} = (\mathrm{X}^\intercal \hat{H} \mathrm{X})^{-1}, \qquad \hat{H} = \mathrm{diag}\left(\hat{\lambda}_1(1-\hat{\lambda}_1), \ldots, \hat{\lambda}_n(1-\hat{\lambda}_n)\right),$$

  with $\hat{\lambda}_i = (1 + e^{\boldsymbol{x}_i^\intercal \boldsymbol{\beta}_{ML}})^{-1}$.
- This estimate $\hat{\Sigma}$ corresponds to the Fisher information, evaluated at the MLE.
- This is a variant of the **Laplace approximation** that ignores the prior contribution. For a more general and detailed explanation, refer to Chopin & Ridgway (2017).

Laplace covariance matrix

- Let us use a covariance matrix based on the Laplace approximation $S = 2.38^2\hat{\Sigma}/p$.
- This choice for S is almost optimal, and the effective sample size is much higher.

```
# Covariance matrix is selected using a Laplace approximation
fit_logit <- glm(type ~ X - 1, family = binomial(link = "logit"), data = Pima)
S <- 2.38^2 * vcov(fit_logit) / ncol(X) # The desired matrix is extracted using vcov

# Running the MCMC (R = 30000, burn_in = 30000)
fit_MCMC <- as.mcmc(RMH(R, burn_in, y, X, S))

summary(effectiveSize(fit_MCMC)) # Effective sample size
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 1107 1174 1206 1194 1228 1245

summary(R / effectiveSize(fit_MCMC)) # Integrated autocorrelation time (beta)
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 24.10 24.43 24.87 25.15 25.56 27.10

print(1 - rejectionRate(fit_MCMC)) # Acceptance rate
# 0.2746
```

Adaptive MCMC

- In several cases, it is not possible to come up with a fast and reasonable estimate $\hat{\Sigma}$.
- Hence, a possibility is **tuning the covariance matrix S on the fly**, namely using the previously obtained samples.
- This is **no longer guaranteed to be a MH algorithm**; therefore, the chain is not necessarily converging to the correct stationary distribution or converging
- However, in many cases ergodicity of the chain is preserved as long as we adaptively tune S in a reasonable manner.
- The key condition is called **diminishing adaptation**, which essentially means that the changes in S are negligible as $R \to \infty$. See Roberts & Rosenthal (2009).

  $\to$ Ideally, S tends to stabilize to certain values.

Adaptive Metropolis

- An example of adaptive MCMC is the so-called adaptive Metropolis (AM) algorithm.
- We implement here a version of the AM which makes use of the following proposal distribution

$$q_r(\boldsymbol{\beta}^* \mid \boldsymbol{\beta}) \stackrel{d}{=} N_p(\boldsymbol{\beta}, 2.38^2/p\boldsymbol{\Sigma}_r + \epsilon I_p)$$

  where $\boldsymbol{\Sigma}_r$ is the (unbiased) covariance matrix estimate of the previously r sampled values $\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(r)}$.
- The constant $\epsilon > 0$ is some small value that avoid degeneracies. We will use $\epsilon = 10^{-6}$.
- Moreover, note that the following **recursive formula** holds true:

$$\begin{aligned}\boldsymbol{\Sigma}_r &= \frac{1}{r-1}\sum_{j=1}^{r}\left(\boldsymbol{\beta}^{(j)} - \overline{\boldsymbol{\beta}}^{(r)}\right)\left(\boldsymbol{\beta}^{(j)} - \overline{\boldsymbol{\beta}}^{(r)}\right)^{\intercal} \\ &= \frac{r-2}{r-1}\boldsymbol{\Sigma}_{r-1} + \frac{1}{r}\left(\boldsymbol{\beta}^{(j)} - \overline{\boldsymbol{\beta}}^{(r-1)}\right)\left(\boldsymbol{\beta}^{(j)} - \overline{\boldsymbol{\beta}}^{(r-1)}\right)^{\intercal}\end{aligned}$$

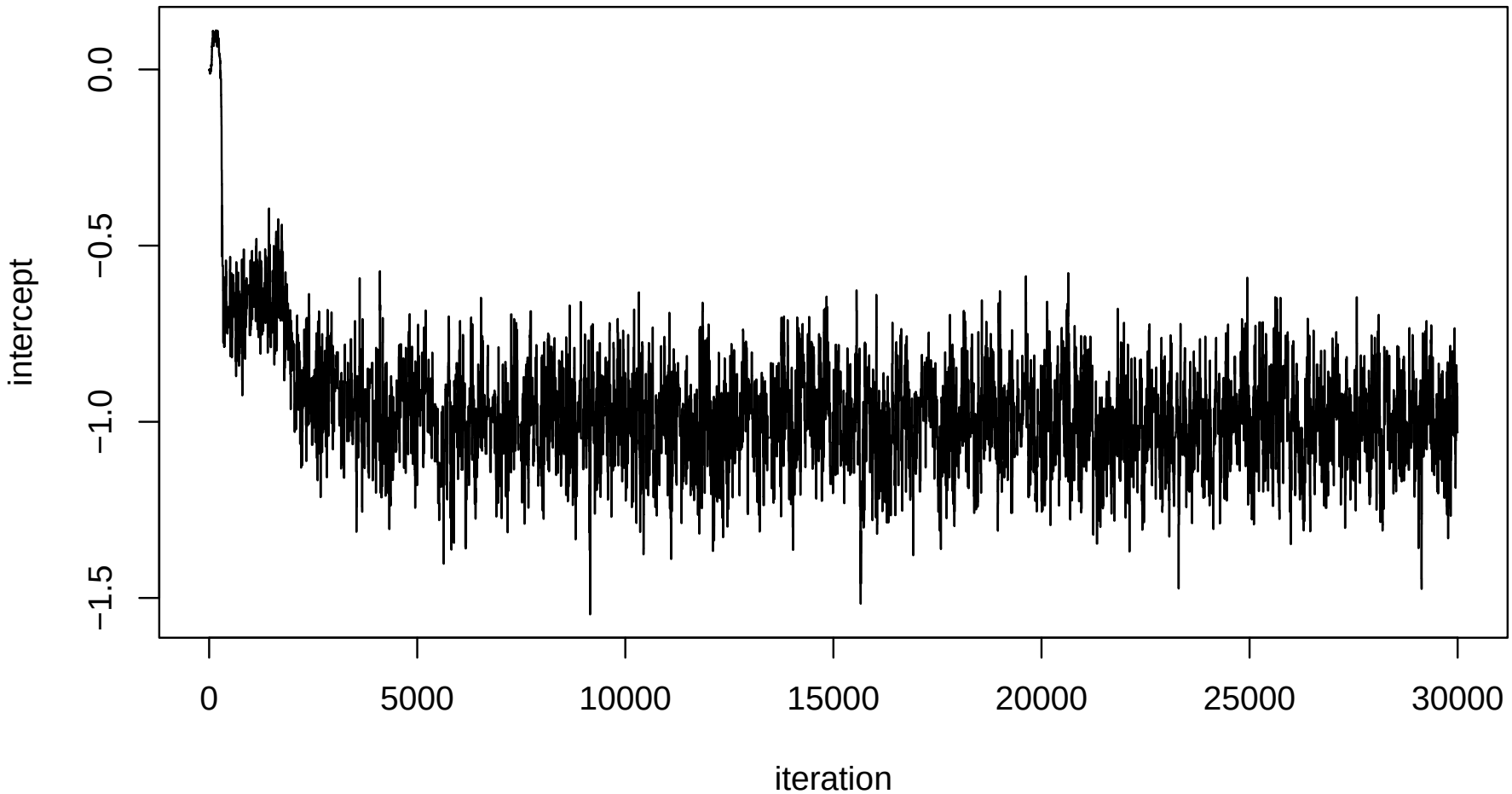
  where $\overline{\boldsymbol{\beta}}^{(r)} = (r-1)/r\overline{\boldsymbol{\beta}}^{(r-1)} + \boldsymbol{\beta}^{(r)}/r$ is the arithmetic mean of the first r values.
- **Several variants** of this scheme exist, but the core idea is trying to estimate $\boldsymbol{\Sigma}$.

Adaptive Metropolis

- We obtain results that are comparable to the MH based on the Laplace approximation in terms of effective sample size.
- However, the computing time is much higher, because we need to decompose S at each iteration.

```
# Running the MCMC (R = 30000, burn_in = 30000)
fit_MCMC <- as.mcmc(RMH_Adaptive(R = R, burn_in = burn_in, y, X))

summary(effectiveSize(fit_MCMC)) # Effective sample size (beta)
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 856.7 905.9 1124.5 1110.9 1269.2 1412.6

summary(R / effectiveSize(fit_MCMC)) # Integrated autocorrelation time
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 21.24 23.65 26.69 27.89 33.12 35.02

print(1 - rejectionRate(fit_MCMC)) # Acceptance rate
# 0.1907
```

Adaptive Metropolis



Metropolis-within-Gibbs (recap)

- The **Metropolis-within-Gibbs algorithm** is an MCMC algorithm that combines the MH and the Gibbs sampling algorithms.
- Let $\pi(\theta_j \mid -)$ be the so-called **full-conditional** of θ_j, that is

$$\pi(\theta_j \mid -) = \pi(\theta_j \mid \boldsymbol{X}, \theta_1, \ldots, \theta_{j-1}, \theta_{j+1}, \ldots, \theta_p), \qquad j = 1, \ldots, p,$$

  namely the conditional distribution of θ_j given the data and the other parameters.
- In the Metropolis-within-Gibbs, we proceed as in a standard Gibbs sampling, but instead of drawing from the full conditional $\pi(\theta_j|-)$, we conduct a Metropolis step.

  ,
- We propose a value from $q(\theta_j^* \mid \theta_j)$, typically performing an univariate Gaussian random walk, that we accept/reject in the usual manner.
- This means that at each step of the chain, some parameters are updated, others are not. This produces **local moves** rather than **global moves**.

Metropolis-within-Gibbs

- We use random walk proposals $\theta_j^* \mid \theta_j \sim N_1(\theta_j, s_j^2)$, for $j = 1, \ldots, p$.
- In this first experiment we set $s_1^2 = \cdots = s_p^2 = 10^{-4}$
- These results are **unacceptable**. We need a better specification for the variances s_j^2.

```
p <- ncol(X) # Dimension of the parameter space
se <- sqrt(rep(1e-4, p)) # Standard deviations of the proposal distributions

# Running the MCMC (R = 30000, burn_in = 30000)
fit_MCMC <- as.mcmc(RMH_Gibbs(R = R, burn_in = burn_in, y, X, se))

summary(effectiveSize(fit_MCMC)) # Effective sample size (beta)
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 27.02 36.43 37.37 37.57 40.58 44.21

summary(R / effectiveSize(fit_MCMC)) # Integrated autocorrelation time (beta)
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 678.6 740.1 802.8 814.8 824.1 1110.1

summary(1 - rejectionRate(fit_MCMC)) # Acceptance rate (beta)
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 0.9682 0.9685 0.9697 0.9698 0.9710 0.9719
```

Adaptive Metropolis-within-Gibbs

- In order to get a better mixing, we could **adaptively choose** the variances s_j^2 as in Roberts & Rosenthal (2009).
- Since the updates are univariate, we can rely on a more direct adaptive approach targeting the optimal acceptance rate, which is 0.44.
- Every 50 iterations (a batch), the algorithm increases or decreases the standard errors $s_j j$ according to the fraction of accepted values among the 50 batch values.
- It is convenient to work in the **logarithmic scale**, to facilitate the exploration of the space of suitable values.
- If the fraction of accepted values for the jth component is **higher/lower** than 0.44, then we **increase/decrease** the corresponding $\log s_j$ by the quantity $\min\{0.01, 1/\sqrt{r}\}$.
- Note that the **diminishing adaptation** condition is satisfied, as the correction is vanishing as r (the number of iterations) increases.

Adaptive Metropolis-within-Gibbs

- These results are comparable with the other suitably tuned MH approaches.
- However, the computing time is much higher.
- Note that the overall acceptance rates are close to 0.44.

```
fit_MCMC <- as.mcmc(RMH_Gibbs_Adaptive(R = R, burn_in = burn_in, y, X)) #

summary(effectiveSize(fit_MCMC)) # Effective sample size (beta)
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 653.2 733.1 1021.5 1009.3 1293.6 1373.3

summary(R / effectiveSize(fit_MCMC)) # Integrated autocorrelation time (beta)
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 21.84 23.19 31.43 32.76 41.07 45.93

summary(1 - rejectionRate(fit_MCMC)) # Acceptance rate (beta)
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 0.4451 0.4472 0.4479 0.4483 0.4494 0.4517
```

Adaptive Metropolis-within-Gibbs

- The following table compare the average results. Here, ESS represents the estimated and average effective sample size.
- Among these competitors, the Laplace MH seems preferable.
- Note that we could sensibly speed up these results by using Rcpp (try at home).

	Seconds	ESS	ESS / Sec.	Acceptance rate
Vanilla MH	1.89	259.58	137.60	0.72
Laplace MH	1.77	1194.42	676.49	0.27
AM	4.88	1110.90	227.45	0.19
Metropolis-within-Gibbs	11.95	37.57	3.14	0.97
Ad. Metropolis-within-Gibbs	11.95	1009.32	84.48	0.45

Dynamic-based methods

Limitations of the Metropolis algorithm

- The random walk Metropolis (RWM) algorithm is very popular among practitioners because it is **general** and **easy to implement**.
- In addition, the RWM is **quite robust** to the choice of the tuning (scaling) parameters.
- Unfortunately, this seductive simplicity leads to performance that scales poorly with **increasing dimension** and **increasing complexity** of the target density.
- Even when the proposal is optimally chosen, the RWM relies on **local moves** that lead to slow mixing, especially in high dimensions.
- The proposal distribution of a RWM is indeed **randomly exploring** the interesting parts of the posterior density without considering its structure.

Gradient-based methods

- Intuitively, we are seeking better proposal distributions that incorporate the **structure** of the target density, leading to **faster mixing**.
- Let $\pi(\boldsymbol{\theta} \mid \boldsymbol{X})$ be a continuous and differentiable posterior density in $\mathbb{R}^p$. We will exploit the gradient of the logarithm of the target density, written

$$\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta} \mid \boldsymbol{X}) = \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta} \mid \boldsymbol{X}) + \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta}).$$

- The gradient is often available in **closed form**, and it does not require the knowledge of the normalizing constant.
- The gradient informs about the **direction** and the **rate of increase** of a given function.
- For instance, for a given value $\boldsymbol{\theta}^{(r)}$ and $\epsilon > 0$, the update

$$\boldsymbol{\theta}^{(r+1)} \leftarrow \boldsymbol{\theta}^{(r)} + \epsilon \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta}^{(r)} \mid \boldsymbol{X})$$

leads to an increase of $pi(\boldsymbol{\theta} \mid \boldsymbol{X})$, for ϵ small enough. This corresponds to the well-known gradient ascent method.

Gradient-based methods

- Incorporating the gradient in the mcmc procedure is an intuitive and appealing idea: this will push the Markov chain towards values with **higher density**.
- Besides, a strong theoretical justification exists for gradient adjusted MH proposals, based on **Langevin diffusion**.
- Let $\boldsymbol{B}^{(t)}$ be a p-dimensional standard Brownian motion.
- We consider a continuous-time stochastic process $\boldsymbol{\theta}^{(t)}$ satisfying the following stochastic differential equation.

$$\mathrm{d}\boldsymbol{\theta}^{(t)} = \frac{1}{2}\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{X})\mathrm{d}t + \mathrm{d}\boldsymbol{B}^{(t)}.$$

- The stationary distribution of the above Langevin diffusion is the **posterior density** $\pi(\boldsymbol{\theta} \mid \boldsymbol{X})$. Hence, we can get use of the previous diffusion to inform our sampling strategy.

The MALA algorithm

- In practice, we need to consider **discrete approximations** of the Langevin diffusion, for example using the so-called **Euler method**.
- This leads to the following discrete-time stochastic process

$$\mathrm{d}\boldsymbol{\theta}^{(r+1)} = \boldsymbol{\theta}^{(r)} + \frac{\epsilon^2}{2}\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{X})\mathrm{d}t + \epsilon \boldsymbol{z}^{(r)},$$

  for any chosen step size $\epsilon > 0$¡ and with iid $\boldsymbol{z}^{(r)} \sim N_p(\boldsymbol{0}, I_p)$.
- This discrete approximation is no longer guaranteed to converge to $\pi(\boldsymbol{\theta} \mid \boldsymbol{X})$.
- There is a delicate trade-off between the accuracy of this approximation ($\epsilon \to 0$) and the sampling efficiency, increasing as ϵ grows.
- This issue is solved by treating the above distribution as a **proposal density** of a Metropolis-Hastings algorithm.

The MALA algorithm

- The Metropolis adjusted Langevin algorithm (MALA) therefore can be seen as a specific MH algorithm with proposal distribution

$$\boldsymbol{\theta}^* \mid \boldsymbol{\theta} \sim N_p\left(\boldsymbol{\theta} + \frac{s_p^2}{2}\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta} \mid \boldsymbol{X}), s_p^2 I_p\right),$$

  where $s_p^2 > 0$ is some **tuning parameter** that must be carefully chosen.

- This proposal distribution is **not symmetric** as in the RWM case therefore the acceptance probability takes into account also the proposal densities, namely

$$\alpha = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^* \mid \boldsymbol{X}) q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta} \mid \boldsymbol{X}) q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})}\right\}.$$

- There is strong theoretical and empirical evidence showing that a much faster mixing compensates the price paid for computing the gradient.

Asymptotics and optimal scaling

- As for the rwm, some insights about the optimal choice of s_p^2 can be gained by looking at the **asymptotic behaviour** of the mala, for large values of p.

- Let us assume again that the posterior distribution has the following form

$$\pi(\boldsymbol{\theta} \mid \boldsymbol{X}) = \prod_{j=1}^{p} f(\theta_j), \qquad \mathrm{Var}(\boldsymbol{\theta} \mid \boldsymbol{X}) = \sigma^2 I_p.$$

  Namely, the components of $\boldsymbol{\theta}$ are iid from some density f, satisfying the same **smoothness conditions** of the previous adaptive case.

- It turns out that to get sensible asymptotic results, we need to set

$$s_p^2 = \ell^2 / p^{1/3},$$

  as opposed to the p^{-1} term we have in the RWM case.

Diffusion processes (again)

- As for the rwm case, let us define a speeded-up continuous-time stochastic process,

$$Z^{(t)} = \theta_1^{([tp^{1/3}])},$$

making jumps every $p^{1/3}$ units.

Theorem (Roberts and Rosenthal, 1997)

Let $B^{(t)}$ be the standard Brownian motion and let $\Phi(\cdot)$ be the cdf of a standard Gaussian. The continuous-time stochastic process Z weakly converges to

$$Z \xrightarrow{d} W, \qquad p \to \infty,$$

where W is a diffusion process satisfying the stochastic differential equation

$$\mathrm{d}W^{(t)} = h(\ell)^{1/2}\mathrm{d}B^{(t)} + \frac{h(\ell)\nabla \log f(W^{(t)})}{2}\mathrm{d}t,$$

where the speed of the diffusion is

$$h(\ell) = \ell^2 2\Phi\left(-\mathcal{J}\ell^3\right),$$

for some constant $\mathcal{J}$ that depends solely on f.

Speed of diffusion $h(\ell)$

- As in the RWM, the speed of diffusion $h(\ell)$ is strictly related to the asymptotic variance. Hence, we will look for the **optimal** ℓ that maximizes the diffusion $h(\ell)$.

- Let $\tau_g(\ell)$ be the **integrated autocorrelation** of the MALA. Then, for large p, it holds that

$$\tau_g(\ell)^{-1} \approx h(\ell) e_g p^{-1/3},$$

  where $e_g > 0$ is some constant depending only on g.

- These findings imply that the MALA algorithm has complexity $\mathcal{O}(p^{1/3})$, which is considerably more efficient than the $\mathcal{O}(p)$ complexity of the RWM.

- In practice, these theoretical results suggest that the mala algorithm should perform better, especially in high-dimensional problems (large p).

Speed of diffusion and acceptance rate

- Let us recall that the acceptance rate of a mh algorithm is informally defined as

$$A_p(\ell) = \lim_{R\to\infty} \frac{\text{"\# of accepted moves"}}{R}.$$

- Then, it can be shown that in the MALA case, we have

$$\lim_{p\to\infty} A_p(\ell) = A(\ell) = 2\Phi(-\mathcal{J}\ell^2),$$

  implying (again) that the **speed of diffusion** relates to the **acceptance rate**.

- The optimal value ℓ_{opt} maximizing $h(\ell)$ does not require the knowledge of $\mathcal{J}$.

- Indeed, the **asymptotic acceptance rate** evaluated at the optimum is such that

$$A(\ell_{opt}) \approx 0.574,$$

  so ℓ can be chosen by trial and error or using adaptive method.

- This means that the optimally scaled MALA mixes faster than the RWM.

Getting practical

- Let us consider the logistic regression problem under iid Gaussian priors and variance 100 again, using the Pima Indian dataset.

- In this example, we **do not standardize the predictors** to make the problem more challenging.

- Recall that the **gradient** of the log-posterior in this case is easily obtained as follows

$$\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta} \mid \boldsymbol{X}) = \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{X} \mid \boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta}) = \boldsymbol{X}^{\intercal}(\boldsymbol{y} - \boldsymbol{\lambda}) - \boldsymbol{\beta}/100,$$

  where each entry of $\lambda_i = (1 + e^{-\boldsymbol{x}_i^{\intercal}\boldsymbol{\beta}})$, for $i = 1, \ldots, n$.

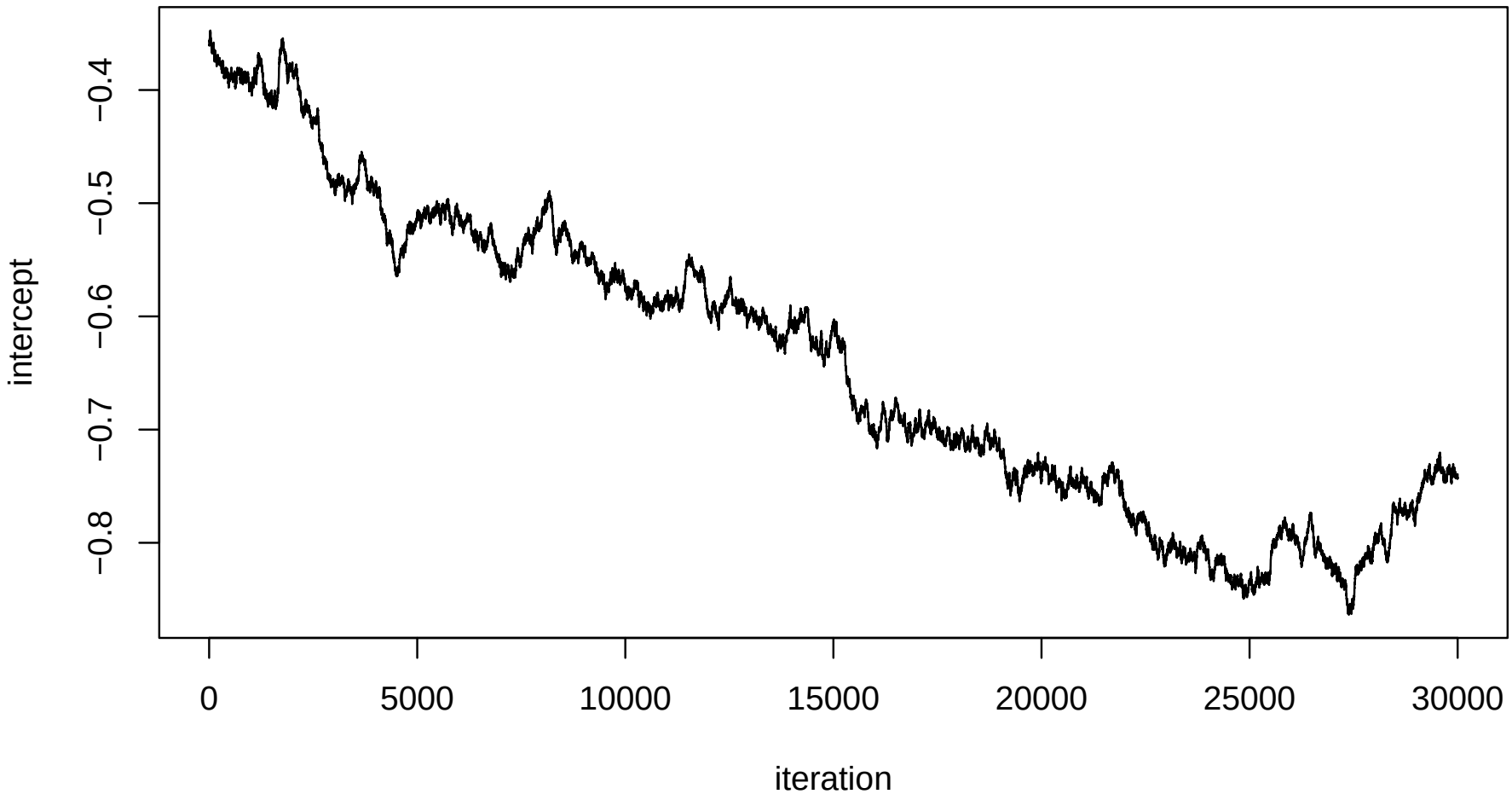
- The mathematical simplicity of the gradient follows from the fact that the logistic regression belongs to an exponential family.

MALA algorithm in practice

- After some trial and error, we set $s_p = 0.0017$, to get the optimal acceptance rate.
- However, the results are a complete disaster. The chain does not reach convergence, and the samples are garbage.

```
# Scaling parameter (after a few trials)
s <- 0.0017

# Running the MCMC (R = 30000, burn_in = 5000)
fit_MCMC <- as.mcmc(MALA(R = R, burn_in = burn_in, y, X, s, S = diag(ncol(X))))

summary(effectiveSize(fit_MCMC)) # Effective sample size (beta)
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 2.900 9.358 27.201 44.321 46.238 166.223

summary(R / effectiveSize(fit_MCMC)) # Integrated autocorrelation time (beta)
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 180.5 728.7 1208.5 2671.3 3283.2 10343.4

print(1 - rejectionRate(fit_MCMC)) # Acceptance rate (beta)
# 0.5638
```

MALA algorithm in practice



- After some trial and error, we set $s_p = 0.0017$, to get the optimal acceptance rate.

What went wrong?

- These performance issues **are not specific to MALA**. Indeed, a vanilla RWM with isotropic covariance matrix $S = s^2 I_p$ would also perform poorly.
- In both cases, the **theory assumes a posterior distribution** with iid components. We need the posterior variances of each component $\theta_1, \ldots, \theta_p$ to be similar.
- If we standardize the predictors, the results improve remarkably. However, this is a workaround that can not be applied in general.
- In the RWM case we solved this issue by considering a covariance matrix S depending on the posterior covariance matrix Σ.
- The very same strategy can be applied to the MALA algorithm, as well as in more elaborate contexts such as Hamiltonian Monte Carlo.

Rotating the diffusion

- Let Σ be the posterior covariance and let $\Sigma = AA^\intercal$ be its Cholesky decomposition.
- Let us consider a **rotation of the parameters** $\tilde{\boldsymbol{\theta}} = A^{-1}\boldsymbol{\theta}$ (reparametrization), implying that the new set of parameters are such that

$$\mathrm{Var}(\tilde{\boldsymbol{\theta}} \mid \boldsymbol{X}) = A^{-1}(AA^\intercal)(A^{-1})^\intercal = I_p,$$

  i.e. they are orthogonal. If the posterior of $\boldsymbol{\theta}$ is Gaussian, they are also independent.
- Therefore, the MALA based on the rotated Langevin diffusion is

$$\mathrm{d}\tilde{\boldsymbol{\theta}}^{(t)} = \frac{1}{2}\nabla_{\tilde{\boldsymbol{\theta}}} \log \pi(\tilde{\boldsymbol{\theta}}^{(t)} \mid \boldsymbol{X})\mathrm{d}t + \mathrm{d}\boldsymbol{B}^{(t)}.$$

- This leads to a MALA proposal distribution targeting the posterior law of $\tilde{\boldsymbol{\theta}}$, namely

$$\tilde{\boldsymbol{\theta}}^* \mid \tilde{\boldsymbol{\theta}} \sim N_p\left(\tilde{\boldsymbol{\theta}} + \frac{s_p^2}{2}\nabla_{\tilde{\boldsymbol{\theta}}} \log(\tilde{\boldsymbol{\theta}} \mid \boldsymbol{X}), s_p^2 I_p\right)$$

  which is expected to perform well due to the orthogonality of $\tilde{\boldsymbol{\theta}}$.

Rotating the diffusion

- By pre-multiplying by A the diffusion for $\tilde{\boldsymbol{\theta}}^{(t)}$, one obtain that

$$\mathrm{d}\boldsymbol{\theta}^{(t)} = \frac{1}{2}\Sigma\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{X})\mathrm{d}t + A\mathrm{d}\boldsymbol{B}^{(t)}.$$

- This leads to a **pre-conditioned MALA**, targeting the posterior law of $\boldsymbol{\theta}$, namely

$$\boldsymbol{\theta}^* \mid \boldsymbol{\theta} \sim N_p\left(\boldsymbol{\theta} + \frac{s_p^2}{2}\Sigma\nabla_{\boldsymbol{\theta}} \log(\boldsymbol{\theta} \mid \boldsymbol{X}), s_p^2\Sigma\right)$$

  with $s_p^2 = \ell^2/p^{1/3}$.

- In other words, this simple modification of the original MALA proposal is equivalent to running the MALA algorithm on the orthogonal parametrization.

- Σ is unknown, but fast approximations and adaptive strategies can be used.

Pre-conditioned MALA algorithm

- After some trial and error, we set $s_p = 1.68$, to get the optimal acceptance rate.
- The pre-conditioned MALA, based on the Laplace approximation, leads to a very high effective sample size.

```
# Covariance matrix is selected via Laplace approximation
fit_logit <- glm(y ~ X - 1, family = binomial(link = "logit"))
S <- vcov(fit_logit)
s_p <- 1.68 # After some trial and error

# Running the MCMC (R = 30000, burn_in = 5000)
fit_MCMC <- as.mcmc(MALA(R = R, burn_in = burn_in, y, X, s_p, S))

summary(effectiveSize(fit_MCMC)) # Effective sample size (beta)
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 8583 8762 9196 9063 9312 9409

summary(R / effectiveSize(fit_MCMC)) # Integrated autocorrelation time (beta)
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 3.189 3.222 3.263 3.314 3.424 3.495

print(1 - rejectionRate(fit_MCMC)) # Acceptance rate (beta)
# 0.5686
```

Hamiltonian Monte Carlo

- In complex scenarios, such as multimodality and heavy-tailed distributions, pushing the Markov chain towards the mode (as with the MALA), may lead to an inefficient exploration of the parameter space.
- **Hamiltonian Monte Carlo** (HMC) is essentially a Metropolis-Hastings algorithm with a "smart" proposal distribution based on the gradient.
- HMC performs **several steps** in the parameter space before accepting/rejecting the move, favoring **bigger jumps** and **better mixing**.
- The proposed value is far from the previous one, but the proposal accommodates also for level set information. This is in contrast with RWM, in which big jumps often lead to values with low density.
- HMC, also known as Hybrid Monte Carlo, has been known for some time in physics, but it seems to have be considered only recently in statistics.

Hamiltonian Monte Carlo

- Recall that $\boldsymbol{\theta} \in \mathbb{R}^p$ is the parameter of interest, and that we want to sample from its posterior distribution $\pi(\boldsymbol{\theta} \mid \boldsymbol{X})$.
- Here, the possible chain status play the role of **position** of a dynamical system.
- We then augment the problem with another quantity, called **momentum**, which describes the force applied to the system during its dynamical evolution.
- Hence, we rely on an auxiliary set of parameters $\boldsymbol{\psi} \in \mathbb{R}^p$, independent on $\boldsymbol{\theta}$, so that the joint density of $(\boldsymbol{\theta}, \boldsymbol{\psi})$ is given by

$$\pi(\boldsymbol{\theta}, \boldsymbol{\psi} \mid \boldsymbol{X}) = \pi(\boldsymbol{\theta} \mid \boldsymbol{X})\pi(\boldsymbol{\psi}),$$

  i.e. the momentum is independent of the position.
- The term $\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{\psi}) = -\log \pi(\boldsymbol{\theta}, \boldsymbol{\psi} \mid \boldsymbol{X})$ is called the Hamiltonian, and describes the evolution of the system.
- Assuming $\boldsymbol{\psi} \sim N_p(\mathbf{0}, M)$, the Hamiltonian equals

$$\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{\psi}) = -\log \pi(\boldsymbol{\theta} \mid \boldsymbol{X}) + \frac{1}{2}\boldsymbol{\psi}^\intercal M^{-1}\boldsymbol{\psi},$$

  up to an irrelevant additive constant not depending on $(\boldsymbol{\theta}, \boldsymbol{\psi})$.

Hamiltonian dinamics

- In HMC we will sample values from the joint distribution $\pi(\boldsymbol{\theta}, \boldsymbol{\psi} \mid \boldsymbol{X})$ and then we will marginalize (ignore) $\boldsymbol{\psi}$.

- The gradient of $\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{\psi})$ with respect to $(\boldsymbol{\theta}, \boldsymbol{\psi})$ admits a physical interpretation as the **time evolution**, with respect to a fictitious time t, of an Hamiltonian dynamic system.

- Let us assume that $(\boldsymbol{\theta}(t), \boldsymbol{\psi}(t))$ is a deterministic evolution flowing according to the following set of differential equations

$$\begin{cases} \dfrac{\mathrm{d}\boldsymbol{\theta}(t)}{\mathrm{d}t} & = \dfrac{\partial \mathcal{H}(\boldsymbol{\theta}, \boldsymbol{\psi})}{\partial \boldsymbol{\psi}} = M^{-1}\psi(t), \\ \dfrac{\mathrm{d}\boldsymbol{\psi}(t)}{\mathrm{d}t} & = -\dfrac{\partial \mathcal{H}(\boldsymbol{\theta}, \boldsymbol{\psi})}{\partial \boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta}(t) \mid \boldsymbol{X}). \end{cases}$$

  We can use the previous dynamical evolution to explore the support of interest.

- By assuming that $\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{\psi})$ does not depend on time t, for any $t, s \in \mathbb{R}$, we have that

$$(\boldsymbol{\theta}(t+s), \boldsymbol{\psi}(t+s)) = \mathcal{T}_s(\boldsymbol{\theta}(t), \boldsymbol{\psi}(t)),$$

  for some mapping $\mathcal{T}_s$ depending only on s.

Properties of Hamiltonian dynamics

- These differential equations preserve the value of the Hamiltonian, namely

$$\mathcal{H}(\boldsymbol{\theta}(t), \boldsymbol{\psi}(t)) = \mathcal{H}(\boldsymbol{\theta}(t+s), \boldsymbol{\psi}(t+s))$$

- Hence, they also preserve the value of the joint density

$$\pi(\boldsymbol{\theta}(t) \mid \boldsymbol{X})\pi(\boldsymbol{\psi}(t)) = \pi(\boldsymbol{\theta}(t+s) \mid \boldsymbol{X})\pi(\boldsymbol{\psi}(t+s))$$

- Any move according to Hamiltonian dynamics **preserves the joint level set** of **position** and **momentum**.

- The mapping $\mathcal{T}_s$ is also **time-reversible**, which is crucial for showing that MCMC updates that use the dynamics leave the desired distribution invariant.

- Moreover, the mapping preserves the volume, a property which significantly simplifies the computations of the MCMC algorithm

A Gaussian example

- Let us assume that $(\theta, \psi) \sim N_2(\mathbf{0}, I_2)$, so that the Hamiltonian is equal to

$$\mathcal{H}(\boldsymbol{\theta}(t), \boldsymbol{\psi}(t)) = \frac{\theta(t)^2}{2} + \frac{\psi(t)^2}{2}.$$

- In this special case, the differential equations simplify as follows

$$\begin{cases} \dfrac{\mathrm{d}\theta(t)}{\mathrm{d}t} & = \dfrac{\partial \mathcal{H}(\theta, \psi)}{\partial \psi} = \psi(t), \\ \dfrac{\mathrm{d}\psi(t)}{\mathrm{d}t} & = -\dfrac{\partial \mathcal{H}(\theta, \psi)}{\partial \theta} = -\theta(t). \end{cases}$$

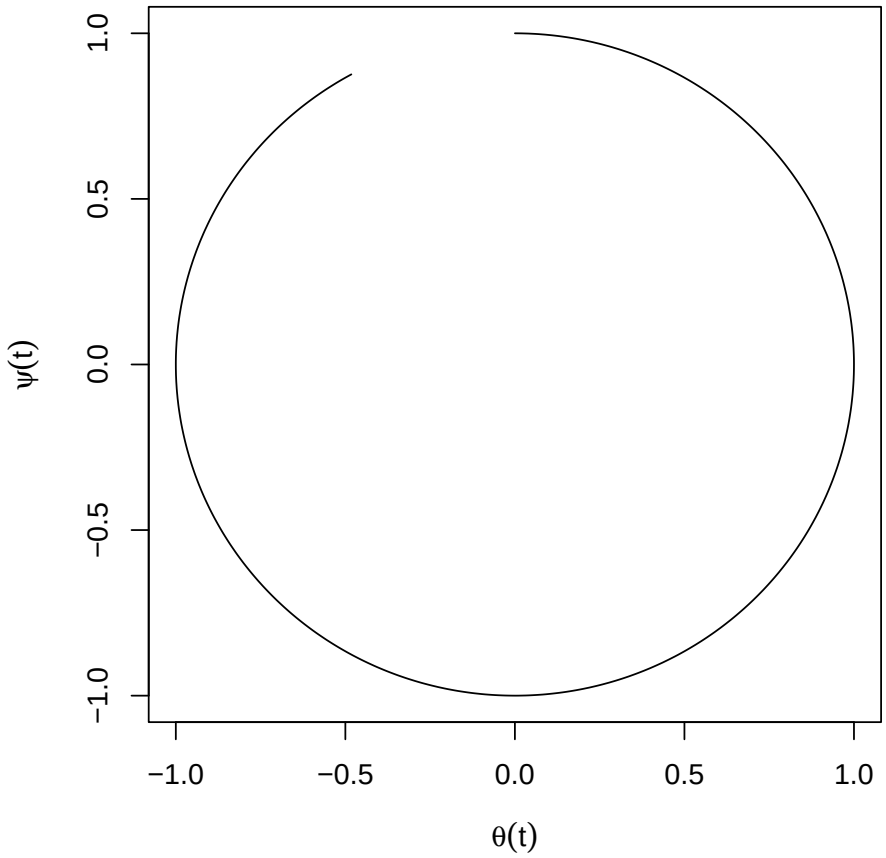
- It is easy to show that the solution is in the following form

$$\theta(t) = \rho \cos(\alpha + t), \qquad \psi(t) = -\rho \sin(\alpha + t),$$

  for some constants ρ and α.

- Hence, the mapping $\mathcal{T}_s$ is a rotation by s radians clockwise around the origin.

A Gaussian example



- Trajectory of the dynamics with $\rho = 1$, $\alpha = -\pi/2$ and for values of $t \in [0, 2\pi - 1/2]$.

The ideal HMC algorithm

- Recall that we aim at sampling values from the joint distribution $\pi(\boldsymbol{\theta}, \boldsymbol{\psi} \mid \boldsymbol{X})$ using an algorithm.

- If the solution of the Hamiltonian dynamics were available in closed form, the HMC would proceed as follows.

At the rth value of the chain

i) Let $\boldsymbol{\theta}^{(r)}$ be the current value of the chain. Draw a new value $\boldsymbol{\psi}^* \sim N_p(\mathbf{0}, M)$. This identifies a new level set.

ii) Obtain the proposed values $(\boldsymbol{\theta}^*, \boldsymbol{\psi}^*)$, remaining on the given level set by applying the mapping

$$(\boldsymbol{\theta}^*, \boldsymbol{\psi}^*) = \mathcal{T}(\boldsymbol{\theta}^{(r)}, \boldsymbol{\psi}),$$

for a certain value of time s, which is a tuning parameter.

iii) Thanks to the properties of Hamiltonian dynamics, the acceptance probability is always 1 therefore the next value of the chain coincides with the proposal $\boldsymbol{\theta}^{(r+1)} \leftarrow \boldsymbol{\theta}^*$.

Approximating Hamiltonian dynamics

- Unfortunately, most of the time the Hamiltonian differential equations do not admit a closed-form solution, thus, **approximations** are required.
- To maintain the main properties of the ideal hmc, we look for discretized Hamiltonian dynamics **preserving the volume** and also being **time-reversible**.
- Among several methods that aim at solving this issue, we focus on the **leapfrog method**, which unfortunately does not keep the Hamiltonian exactly constant over time.
- The leapfrog method is essentially a variation and more reliable version of the natural Euler's discretization method.
- The leapfrog method retains most of the properties of the ideal hmc, thus making it extremely appealing for sampling purposes.

The real HMC algorithm

- We want to obtain the proposed values $(\boldsymbol{\theta}^*, \boldsymbol{\psi}^*)$ by applying L times the leapfrog method with step-size ϵ and starting at time $t = 0$.

At the rth value of the chain

i) Let $\boldsymbol{\theta}^{(r)}$ be the current value of the chain. Draw a new value $\boldsymbol{\psi}^* \sim N_p(\mathbf{0}, M)$. This identifies a new level set.

ii) For $t = 1, \ldots, L$, repeat the following steps

$$\boldsymbol{\psi}^*(t + \epsilon/2) = \boldsymbol{\psi}^*(t) - \frac{\epsilon}{2} \nabla_{\boldsymbol{\theta}^*} \log \pi(\boldsymbol{\theta}^*(t) \mid \boldsymbol{X}),$$

$$\boldsymbol{\theta}^*(t + \epsilon) = \boldsymbol{\theta}^*(t) + \epsilon M^{-1} \boldsymbol{\psi}(t + \epsilon/2),$$

$$\boldsymbol{\psi}^*(t + \epsilon) = \boldsymbol{\psi}^*(t + \epsilon/2) - \frac{\epsilon}{2} \nabla_{\boldsymbol{\theta}^*} \log \pi(\boldsymbol{\theta}^*(t + \epsilon) \mid \boldsymbol{X}).$$

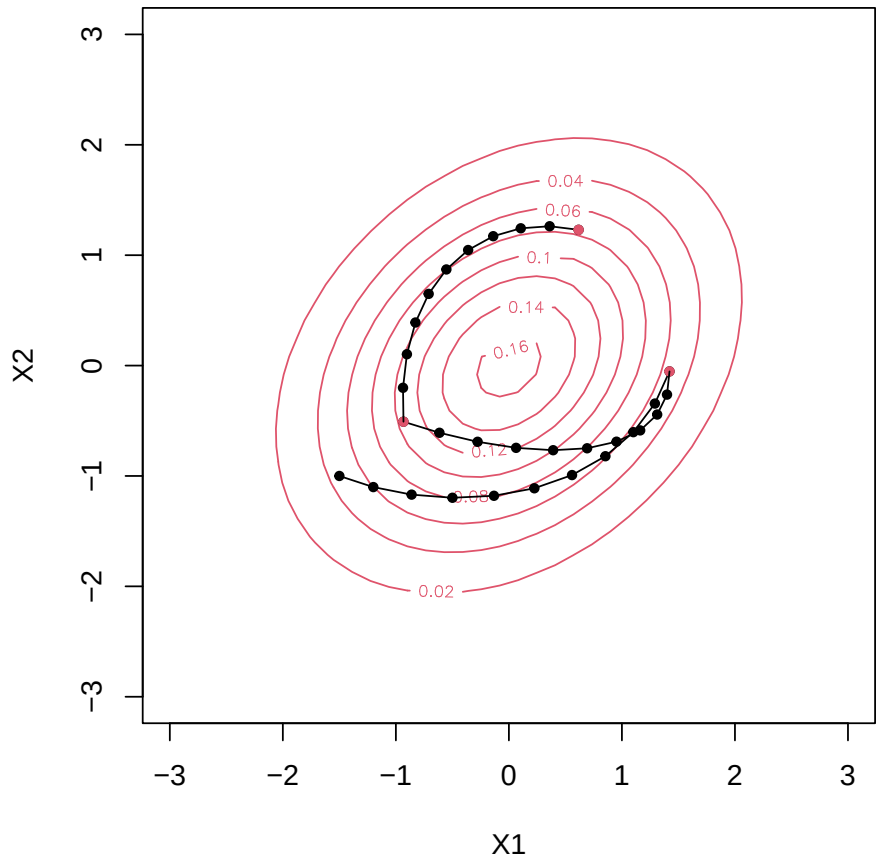
iii) Negate the momentum setting $\boldsymbol{\psi}^* = -\boldsymbol{\psi}^*$.

iv) accept or reject the proposed value with a probability depending only on the Hamiltonian, namely

$$\alpha = \min \left\{ 1, \exp[-\mathcal{H}(\boldsymbol{\theta}^*, \boldsymbol{\psi}^*) + \mathcal{H}(\boldsymbol{\theta}^{(r)}, \boldsymbol{\psi}^{(r)})] \right\}$$

- α is usually very close to 1 as the Hamiltonian is kept approximately constant by the leapfrog method.

HMC with leapfrog



- Three steps of HMC with leapfrog integrator. The plot shows the position, targeting a bivariate Gaussian distribution with correlation 0.3, marginally standardized components, $L = 10$ and $\epsilon = 0.2$.

Connection with the MALA algorithm

- There is a **strong connection** between MALA and HMC, even though these methods rely on very different notions.
- Indeed, it can be shown HMC using a single $L = 1$ leapfrog step coincides with the pre-conditioned MALA algorithm.
- More precisely, at each step, we propose from

$$\boldsymbol{\theta}^* \mid \boldsymbol{\theta} \sim N_p\left(\boldsymbol{\theta} + \frac{\epsilon}{2} M^{-1} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta} \mid \boldsymbol{X}), \epsilon^2 M^{-1}\right),$$

  and the acceptance probability coincides with that of MALA.
- This connection highlights that the covariance matrix of the Gaussian auxiliary variables is of great **practical importance**.
- In practice, it is advised to set $M = \Sigma^{-1}$, where Σ represents an estimate for the posterior covariance. See Neal (2010) for further details and a deeper perspective

How to choose ϵ and L?

- Beside the covariance matrix M, there are other 2 tuning parameters that must be chosen: the stepsize ϵ and number of leapfrog steps L.
- **The trajectory length** ϵL is often set equal to some constant, say ϵL or selected by trial and error.
- Small values for the **step-size** ϵ increase the goodness of the leapfrog approximation but require larger values for L, leading to higher computational costs.
- Large values for the **step-size** ϵ could lead to catastrophic results, as the approximated trajectory could diverge from the ideal dynamics.
- The no-u-turn algorithm implemented in Stan automatically selects L so that the trajectory completes a "loop".
- However, the no-u-turn requires a somewhat complex procedure that preserves the chain's reversibility.

HMC algorithm in practice

- After some trial and error, we set $\epsilon = 0.1$ and $L = 10$.
- We again relied on the Laplace approximation for $\hat{\Sigma} = M^{-1}$, which leads to an extremely high effective sample size.

```
epsilon <- 0.05 # Stepsize - After some trial and error
L <- 20 # Number of leapfrog steps

# Covariance matrix is selected via Laplace approximation
fit_logit <- glm(y ~ X - 1, family = binomial(link = "logit"))
S <- vcov(fit_logit)

# Running the MCMC
fit_MCMC <- as.mcmc(HMC(R = R, burn_in = burn_in, y, X, epsilon, S, L))

summary(effectiveSize(fit_MCMC)) # Effective sample size
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 23744 24846 25010 24943 25182 25851

summary(R / effectiveSize(fit_MCMC)) # Integrated autocorrelation time
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 1.160 1.191 1.200 1.203 1.207 1.263

print(1 - rejectionRate(fit_MCMC)) # Acceptance rate
# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 0.9993 0.9993 0.9993 0.9993 0.9993 0.9993
```

How to choose ϵ and L?

- The following table compare the average results. Here, ESS represents the estimated and average effective sample size.
- A suitably tuned hmc can be extremely effective and is the clear winner in this case.
- The implementation could even be improved by writing it in Rcpp (try at home).

	Seconds	ESS	ESS / Sec.	Acceptance rate
MH Laplace + Rcpp	0.230	1193.171	5187.926	0.273
MALA	1.808	3.444	1.904	1.000
MALA tuned	2.444	9206.215	3766.505	0.574
HMC	24.889	24800.657	996.448	0.999