Politecnico di Milano

Scuola di Ingegneria Industriale e dell'Informazione

Tesi di Laurea Magistrale in Ingegneria Matematica

A MCMC method for the analysis of large datasets via Generalized Linear Model

Candidato:
Emanuele Battistello
matr. 786403

Relatore:
Prof.ssa Alessandra Guglielmi

Correlatore:
PhD Anders Sjögren

Anno Accademico 2013/2014

ABSTRACT

Markov chain Monte Carlo are a class of computational statistics algorithms, used to sample from a known probability distribution. Because of their potentialities, they are widely used in the scientific literature. They are based on Markov chains, deeply studied random processes, and their properties. The goal of our work is to introduce, describe and implement a particular type of Markov chain Monte Carlo: the Hamiltonian Monte Carlo. Firstly introduced by Duane et al. in 1987, despite its potentiality, it is not part of the Monte Carlo methods usually studied. The main characteristic of Hamiltonian Monte Carlo is that, at each step of the sampler scheme, a simulation of an Hamiltonian dynamics is included. This peculiarity improves several aspects of the sampler, analyzed in this work.

Finally we apply the algorithm to a simulated dataset using a generalized linear model through the Bayesian approach. The posterior estimates that we get are in complete agreement with the true values of the regression parameters used to generate the data, showing the good performance of the Hamiltonian Monte Carlo in this case.

All the algorithms developed for the applications in this thesis are implemented in **R** language.

This thesis has been written at Chalmers Tekniska Högskola (Göteborg, Sweden), at the Mathematics Department, under the supervision of doctor Anders Sjögren.

SOMMARIO

I metodi Monte Carlo per catene di Markov sono uno strumento di statistica computazionale molte potente e diffuso, utilizzato per campionare variabili aleatorie data la loro distribuzione di probabilità. Essi si basano sulle proprietà delle catene di Markov, un particolare tipo di processo aleatorio le cui caratteristiche sono state ampiamente studiate. Lo scopo di questo lavoro è introdurre, descrivere ed implementare uno di questi metodi, particolarmente interessante per le sue proprietà: l'Hamiltonian Monte Carlo. Introdotto per la prima volta da Duane et al. nel 1987, nonostante le sue notevoli potenzialità non è parte dei metodi Monte Carlo solitamente studiati. La sua principale differenza con i metodi classici è che esso include nel proprio algoritmo la risoluzione di un sistema Hamiltoniano, peculiarità che ne migliora diversi aspetti.

Infine applichiamo l'algoritmo ad un dataset simulato, utilizzando un modello di regressione lineare generalizzata tramite un approccio Bayesiano. La stima delle distribuzioni a posteriori che otteniamo corriponde perfettamente ai valori reali dei parametri di regressione utilizzati per generare i dati, mostrando la bontà del metodo descritto.

Tutti gli algoritmi presentati in questo lavoro sono stati implementati in linguaggio **R**.

Questa tesi è stata scritta all'Università di Chalmers (Göteborg, Svezia), nel Dipartimento di Matematica, sotto la supervione del dottor Anders Sjögren.

"Per la ragione, che non cesserà di sognare
un qualche disegno del labirinto"
J.L. Borges

CONTENTS

INTRODUCTION

A common problem in statistics is sampling from a given probability density function. For this purpose Metropolis et al. in 1953 introduced the *Markov chain Monte Carlo*, with the landmark paper [Met+53]. This computational method, now widely used in different contexts of applied probability and statistics, was initially designed to reproduce the state of a system of molecules.
Another approach to molecular simulation, presented by Alder and Wainwright in [AW59], was deterministic and followed the laws of motion, formalized as *Hamiltonian Dynamics*.
Finally, in 1987, Duane et al. combined these two different schemes, creating an "hybrid Monte Carlo", which used both a random component, like Metropolis, and a deterministic Hamiltonian dynamics simulation [Dua+87].
In this work we present this method, called *Hamiltonian Monte Carlo*, studying its theoretical motivations and practical implementations.
Finally, we apply the method to a Bayesian problem: we compute the posterior distribution of parameters in a regression model, so that we obtain not only pointwise estimates, but also credibility intervals.

In **Chapter 1** we shortly describe all the mathematical tools we need. We begin with an overview about the physics concepts of Hamiltonian system and canonical distribution. Then the Markov chains on a general state space are introduced, followed by a resume on their Monte Carlo applications: the Markov chain Monte Carlo methods.

Chapter 2 presents the Hamiltonian Monte Carlo (HMC) scheme. We describe it, its properties and theoretical bases. The sampler is a Metropolis-Hastings-like scheme, in which the proposal state is chosen in the following way. First we resample a support variable, called *momentum*, which provides the randomness of the procedure. Then, starting from it, we simulate an Hamiltonian dynamics to obtain a new proposed variable, which can be accepted or rejected as new state of the chain.
Afterward an important improvement is introduced, increasing the method performance. Using a property of the Hamiltonian system we precondition, at each step, the variable of interest to better explore the target density. This modification requires the computing of a particular Hessian matrix and increases the computational cost. A complete description of the two algorithms

and their pseudocodes is included. The full codes in **R** language are reported in the appendix.
We conclude the chapter presenting an optimization technique, called *quasi-Newton method* and adapt it to our HMC sampler, increasing the efficiency from a time and memory consuming point of view.

Chapter 3 is dedicated to the description of a wide class of models, called *Generalized Linear Models* (GLMs). After an overview, we illustrate the application of HMC to GLMs: estimate the posterior distributions of regression parameters, given covariates and outcomes.
Finally we sketch a real problem that can be modeled and solved with the techniques previously presented: the *ad optimization*. The goal is, observing several web pages, infer which factors and how affect the popularity of on-line advertisements.

Final conclusions end this work with **Chapter 4**.

1

HAMILTONIAN SYSTEMS AND MCMC

We start introducing some concepts usefull for the understanding of the next chapters, like hamiltonian systems and canonical distributions. Then we will shortly describe the important statistical tool we will use to implement our methods: Markov chain Monte Carlo. For a more complete study see [RC99], [RC+10], [GL06], [Jac09].

1.1 HAMILTONIAN DYNAMICS

1.1.1 *Hamiltonian System*

Let us consider a two dimensions system, consisting in a puck moving on frictionless surface. The state of this system is formed by two variables, that describe the situation of the puck in a given instant: its *position* q and its *momentum* p (i.e. mass times velocity). To each one of these physical quantities we can associate an energy: the *potential energy* $U(q)$, proportional to the height of the current position of the puck and the *kinetic energy* $K(p) = |p|^2/2m$. In this case the evolution with respect to time of the system, technically its *dynamics*, describes how the puck moves on the frictionless surface. Thanks to the momentum the mass can move on a rising slope and increase its height, until its momentum (and so its kinetic energy) becomes 0. Then it moves back, decreasing the potential energy and increasing the kinetic one.
Formally the physics system is fully described by the *Hamiltonian function*

$$H(q,p) = U(q) + K(p) \tag{1.1}$$

that is the sum of the two energies, while its dynamics (i.e. evolution of its state (q,p) over time) is the solution of the *Hamilton's equations*

$$\begin{aligned} \frac{dq}{dt} &= \frac{\partial H}{\partial p} \\ \frac{dp}{dt} &= -\frac{\partial H}{\partial q} \end{aligned} \tag{1.2}$$

We now generalize this example to a $2d$-dimension system: the position and momentum variables are d-vectors, the potential energy is a function

$$U : \mathbb{R}^d \to \mathbb{R} \tag{1.3}$$

while the kinetic one is

$$K(p) = \frac{1}{2} p^T M^{-1} p, \tag{1.4}$$

basically a multidimensional generalization of

$$K(p) = |p|^2/2m.$$

M is a $d \times d$ matrix, consistently called *mass matrix*.
In the multidimensional case

$$q = [q_1, \ldots, q_d] \quad p = [p_1, \ldots, p_d] \tag{1.5}$$

and equations (1.2) become

$$\begin{aligned} \frac{dq_i}{dt} &= \frac{\partial H}{\partial p_i} \\ \frac{dp_i}{dt} &= -\frac{\partial H}{\partial q_i}. \end{aligned} \tag{1.6}$$

for $i = 1, 2 \ldots, d$.

Since the quadratic form of the kinetic energy and considering that the variables are separated, we can rewrite equations (1.6) as

$$\begin{aligned} \frac{dq_i}{dt} &= [M^{-1} p]_i \\ \frac{dp_i}{dt} &= -\frac{\partial U}{\partial q_i}. \end{aligned} \tag{1.7}$$

Let us note that, perfectly consistently with its physical interpretation, the potential energy is defined up to an additive constant. That is because the dynamics equations concern only its derivative.

1.1.2 *Properties of Hamiltonian Dynamics*

The solution of (1.7) is the dynamics, i.e. two functions $q(t)$ and $p(t)$ that describe the evolution of the system. This dynamics has some important properties:

- **Reversibility** If the system evolves from a state (q_0, p_0) to a new state (q_1, p_1) in a time s, it is always possible moving

in the opposite direction. So the system can evolve from (q_1, p_1) to (q_0, p_0). This is equivalent to the assumption that the map

$$T_s : (q(t), p(t)) \to (q(t+s), p(t+s))$$

describing the dynamics is one-to-one and admits inverse T_{-s}

- **Conservation of the Hamiltonian** Moving according the Hamiltonian dynamics keeps the Hamiltonian function constant, i.e.

$$H(q, p) = H(T_s(q), T_s(p)) \; \forall s \in \mathbb{R}^+$$

- **Symplecticness** It is a generalization of volume conservation. Intuitively, given a set of points A in the $2d$-phase space, its evolution according to Hamiltonian dynamics lets the volume of the set invariant

$$V(A) = V(T_s(A)) \; \forall s \in \mathbb{R}^+$$

The first two properties can be easily understood if we think to the initial example. Since the surface is frictionless the total energy is constant and, changing the direction of velocity the system can go back to its previous states. The third one instead is harder to imagine and it means, basically, that the system states cannot implode in a small area or explode to a big one. These properties are very important in the next chapter.

1.1.3 *Numerical solution of Hamiltonian Dynamics*

In our simulations we need to implement a method that replicates the dynamics, without analytically solving the equations (1.7). The *Leapfrog method* is a modification of Forward Euler method, in which we partially update the momentum variables, then we do a full step for the position variables and finally we complete the momentum update, i.e.

$$\begin{aligned} p(t+\varepsilon/2) &= p(t) - (\varepsilon/2)\nabla U(q(t)) \\ q(t+\varepsilon) &= q(t) + \varepsilon M^{-1} p(t+\varepsilon/2) \\ p(t+\varepsilon) &= p(t+\varepsilon/2) - (\varepsilon/2)\nabla U(q(t+\varepsilon/2)) \end{aligned} \tag{1.8}$$

Using L times equations (1.8) we compute the dynamics from position $(q(t), p(t))$ to $(q(t + L\varepsilon), p(t + L\varepsilon))$. The choice of suitable values for L and ε is part of the tuning process.

Algorithm 1: LeapfrogSim

Data: $(q(t), p(t)), U(q), M, \varepsilon, L$
Result: $(q(t + L\varepsilon), p(t + L\varepsilon))$
$(q, p) \leftarrow (q(t), p(t))$;
for $i \leftarrow 1$ **to** L **do**
  $p \leftarrow p - (\varepsilon/2)\nabla U(q)$;
  $q \leftarrow q + \varepsilon M^{-1} p$;
  $p \leftarrow p - (\varepsilon/2)\nabla U(q)$;
$(q(t + L\varepsilon), p(t + L\varepsilon)) \leftarrow (q, p)$;

Obviously Algorithm 1 is just an approximation of the real dynamics. Anyway, even if approximated, the simulated trajectory has reversibility and symplecticness properties.
On the contrary, conservation of the Hamiltonian property is not valid anymore, i.e. due to numeric approximation, simulating the dynamic with leapfrog method does not leave the Hamiltonian exactly invariant.

1.1.4 *Linear change of variables*

Considering an Hamiltonian System, with state (q, p), choosing a non singular square matrix A, we define new variables $(\hat{q}, \hat{p})$, in the following way

$$\begin{cases} \hat{q} = Aq \\ \hat{p} = A^{-T} p \end{cases} . \tag{1.9}$$

Then we define new energy functions $(\hat{U}(\hat{q}), \hat{K}(\hat{p}))$ as

$$\begin{cases} \hat{U}(\hat{q}) = U(A^{-1}\hat{q}) \\ \hat{K}(\hat{p}) = K(A^T \hat{p}) \end{cases} . \tag{1.10}$$

Let us note that, if the original system kinetic energy has quadratic form (1.4), then the new one will be

$$\hat{K}(\hat{p}) = K(A^T \hat{p}) = (A^T \hat{p})^T M^{-1} (A^T \hat{p})/2 = \hat{p}^T (A M^{-1} A^T) \hat{p}/2. \tag{1.11}$$

In other words only the mass matrix is changed

$$\begin{aligned} \hat{K}(\hat{p}) &= \hat{p}^T \hat{M}^{-1} \hat{p}/2 \\ \hat{M} &= A^{-T} M A^{-1}. \end{aligned} \tag{1.12}$$

So we have two different systems, related by (1.9) and (1.10).

Since

$$\begin{aligned}\frac{dq}{dt} &= A^{-1}\frac{d\hat{q}}{dt} = A^{-1}\hat{M}^{-1}\hat{p} = A^{-1}(AM^{-1}A^T)A^{-T}p = M^{-1}p \\ \frac{dp}{dt} &= A^T\frac{d\hat{p}}{dt} = -A^T\nabla\hat{U}(\hat{q}) = -A^TA^{-T}\nabla U(A^{-1}\hat{q}) = -\nabla U(q)\end{aligned} \tag{1.13}$$

these two systems have the same dynamics.
That is, using the relations (1.9, 1.10), the variables $(\hat{q}, \hat{p})$ represent the same physical system state of (q, p) using another coordinate system, that is a linear transformation of the original one. It means that the evolution of the variables $(\hat{q}, \hat{p})$ follows exactly (q, p).

1.2 CANONICAL DISTRIBUTION OF AN HAMILTONIAN SYSTEM

1.2.1 *Canonical Distribution*

We now introduce the notion of *canonical distribution*, borrowed from statistical mechanics. Let us consider a physics system, its state x and the energy function $E(x)$. It is possible to prove that the state variable x is a random variable, following a distribution called canonical with a density function

$$P(x) = \frac{1}{Z}exp(-E(x)) \tag{1.14}$$

1.2.2 *Building the target joint distribution*

If we consider an Hamiltonian system, identified by the function $H(q, p) = U(q) + K(p)$, and we compute its canonical distribution we obtain

$$P(q, p) = \frac{1}{Z}exp(-U(q))exp(-K(p)) \tag{1.15}$$

Since in H the variables p and q are separated, $P(q, p)$ is the product of two marginal distributions of *independent* random variables. This is the key property of $P(q, p)$ and the reason we use it.

Now given a target distribution $\pi(q)$, we define

$$\begin{aligned}U(q) &= -log[\pi(q)] \\ K(p) &= \frac{1}{2}p^TM^{-1}p\end{aligned} \tag{1.16}$$

and we obtain

$$\begin{aligned} P(q,p) &= \frac{1}{Z} exp(-U(q)) exp(-K(p)) = \\ &= \frac{1}{Z}\pi(q) exp(-\frac{1}{2}p^T M^{-1} p) \end{aligned} \tag{1.17}$$

and so

$$P(q,p) \propto \pi(q) f_{\mathcal{N}_d(0,M)}(p) \tag{1.18}$$

where $f_{\mathcal{N}_d(0,M)}(p)$ is the density function of a d-dimension multivariate normal random variable, with expected value $[0,\dots,0]^T$ and covariance matrix M.

1.3 MARKOV CHAIN AND MCMC

1.3.1 *Basic notions*

Let us consider a random process, i.e. a sequence of random variables that can be imagined as evolving over time

$$X_0, X_1, X_2, \dots, X_i, \dots$$

These variables assume value in a set χ. The description of the process and its properties are different if χ and the time index i are discrete or continue. In this work we just consider variables with *discrete time index on a continuous space* $\chi \subseteq \mathbb{R}^d$.

A *Markov chain* is a random process that satisfies the Markov condition:

$$\begin{aligned} &\mathbb{P}(X_i \in A | X_0 = x_0, X_1 = x_1, \dots, X_{i-1} = x_{i-1}) = \\ &= \mathbb{P}(X_i \in A | X_{i-1} = x_{i-1}) \\ &\forall A \in \mathcal{B}(\chi) \end{aligned}$$

where $\mathcal{B}(\chi)$ is the Borel σ-algebra on χ.

The condition above is a conditional independence restriction: given an instant i, the present state X_i of the process depends on the past only through the previous state X_{i-1}.

We consider the time homogeneous case, in which

$$\mathbb{P}(X_i \in A | X_{i-1} = x_{i-1}) = \mathbb{P}(X_1 \in A | X_0 = x_0) \;\; \forall i$$

Because of its property, a Markov chain is fully determined by its kernel K, that is a function defined on $\chi \times \mathcal{B}(\chi)$ such that

- $\forall x \in \chi,\ K(x, \cdot)$ is a probability measure
- $\forall A \in \mathcal{B}(\chi),\ K(\cdot, A)$ is measurable

and

$$\mathbb{P}(X_i \in A | X_{i-1} = x_{i-1}) = \int_A K(x_{i-1}, dx)$$

So the kernel K describes the conditional probability distribution of X_i given X_{i-1}.
On the same way we denote with $K^n(\cdot,\cdot)$ the kernel for n transitions, indicating the conditional probability distribution of X_i given X_{i-n}.

A key notion related with Markov chain is

Definition 1. *Let $\{X_i\}$ be a Markov chain on state space χ, with a transition kernel $K(\cdot,\cdot)$. A probability measure π is said to be an* invariant probability measure *with respect to $K(\cdot,\cdot)$ and hence with respect to the Markov chain if*

$$\pi(A) = \int_\chi K(x, A)\pi(dx) \; \forall A \in \mathcal{B}(\chi)$$

It follows from the definition that an additional iteration of the chain with kernel K leaves the distribution π unchanged, i.e.

$$X_i \sim \pi \implies X_{i+1} \sim \pi \;\; \forall i.$$

For this reason π is also called *stationary distribution*

1.3.2 *Irriducibility and Harris recurrence*

We now introduce some definitions and theorems with a considerable practical importance

Definition 2. *Let φ be a measure on $(\chi, \mathcal{B}(\chi))$. A Markov chain $\{X_i\}$ on state space χ, with a transition kernel $K(\cdot,\cdot)$, is said to be* $\boldsymbol{\varphi}$ - irreducible *if $\forall A \in \mathcal{B}(\chi)$ with $\varphi(A) > 0$, $\exists n$ s.t. $K^n(x, A) > 0 \; \forall x \in \chi$*

The key idea of irreducibility is that, regardless of where the chain starts, it can reach any states.

The following theorem can be found in [RC99, Pb. 6.60]

Theorem 3. *If a chain is φ-irreducible and allows for an invariant probability measure π, then this measure is unique.*

Now let us define

Definition 4. *A Markov chain that is φ- irreducible and admits unique invariant probability measure is said to be* positive.

The next definitions formalize the idea of a chain that visits all the states infinitely many times.

Definition 5. *Let $\{X_i\}$ be a Markov chain on state space χ and a subset $A \in \mathcal{B}(\chi)$. The* number of passages *of the chain through A in a infinitely long run is*

$$v(A) = \sum_{i=1}^{\infty} \mathbb{I}(X_i \in A)$$

Definition 6 (Harris recurrent set)**.** *Let $\{X_i\}$ be a Markov chain on state space χ. A set A is said* Harris recurrent *if $\mathbb{P}(v(A) = \infty | X_0 = x) = 1 \; \forall x \in A$*

Definition 7 (Harris recurrent chain)**.** *Let $\{X_i\}$ be a Markov chain on state space χ. If there exists a measure φ s.t. $\{X_i\}$ is φ-irreducible and $\forall A \in \mathcal{B}(\chi)$ s.t. $\varphi(A) > 0$, A is Harris recurrent the chain is said* Harris recurrent

1.3.3 *Reversibility*

Another important property of a chain dynamics is the independence on the direction of time. Let us define:

Definition 8. *A Markov chain $\{X_i\}$ is said to be* reversible *if the distribution of X_{n+1} conditionally on $X_{n+2} = x$ is the same as the distribution of X_{n+1} conditionally on $X_n = x$.*

Reversibility is very important cause is strongly related with the existence of a stationary distribution.

Definition 9 (Detailed balance condition)**.** *A Markov chain $\{X_i\}$ on state space χ, with a transition kernel $K(\cdot, \cdot)$, satisfies the* detailed balance condition *if there exists a probability density function π s.t.*

$$K(y, x)\pi(y) = K(x, y)\pi(x)$$

for every (x, y).

It can be prove the following

Theorem 10. *If a Markov chain $\{X_i\}$ with a transition kernel $K(\cdot, \cdot)$ satisfies the detailed balance condition with a probability density function π, then*

1. *π is the invariant density of the chain*
2. *the chain is reversible*

So we have a sufficient condition, often easy to check, for π to be a stationary distribution.

Moreover we do not want that the chain has some deterministic paths, i.e. its kernel has to provide random behavior of the

walk in the χ space.
Since in the continuous case the proper definition is quite difficult, we give it considering χ a discrete space. That should give an idea of the property we are describing.

Definition 11. *Let a Markov chain $\{X_i\}$ on a discrete space with transition function $K(\cdot,\cdot)$. The period of a state A is*

$$g.c.d.\{m \geq 1 | K^m(A,A) > 0\}.$$

An irreducible Markov chain with all the states having period 1 is said to be aperiodic.

1.3.4 *Convergence of Markov chains*

Following the notation of [Jaco9, p. 187], we introduce the subsequent definitions:

Definition 12. *A Markov chain $\{X_i\}$ is said to be* ergodic *if it is*

1. *positive*
2. *Harris recurrent*
3. *aperiodic*

Definition 13. *Let μ and γ be measures over $(\chi, \mathcal{B}(\chi))$. Then the* total variation norm *is*

$$\|\mu - \gamma\|_{TV} = \sup_{A \in \mathcal{B}(\chi)} |\mu(A) - \gamma(A)|$$

We can now enunciate this two important theorems, that allow us to use Monte Carlo simulations:

Theorem 14. *If a Markov chain $\{X_i\}$, with a transition kernel $K(\cdot,\cdot)$, is ergodic with invariant distribution π then, for every initial distribution π_0*

$$\lim_{n\to\infty} \| \int_\chi K^n(x,\cdot)\pi_0(dx) - \pi\|_{TV} = 0$$

Theorem 15. *Let $\{X_i\}$ an Harris recurrent Markov chain on state space χ, with invariant probability measure π. Let h a π-measurable function s.t. $\int_\chi |h(x)| d\pi(x) < \infty$. Then*

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} h(X_i) = \int_\chi |h(x)| d\pi(x) = \mathbb{E}_\pi[h(X)]$$

The last theorem is a sort of Law of Large Numbers. It is important to note that is valid even if the numbers generated X_i's are not independent.

1.3.5 *McMC*

We summarized a large piece of theory in order to be able to describe the meaning of Markov chain Monte Carlo. The basic idea is the following: given a target distribution π, build a Markov chain with a kernel K satisfying the assumptions of the limiting theorem Th. 15. So we have to build an chain such that it has stationary and limiting distribution π.
Then, starting from an arbitrary point X_0, generate the chain $\{X_i\}_{i=0}^N$ using the kernel K. After a high number of chain steps, X_i (the i-th random variable of the process) has *approximately distribution* $\boldsymbol{\pi}$.

1.3.6 *Metropolis-Hasting algorithm*

Given the target density function $\pi(x) : \mathbb{R}^d \to \mathbb{R}$, the Metropolis-Hastings approach uses a conditional density $q(y|x)$ to produce a Markov chain $\{X_i\}_{i=0}^N$ in the following way

Algorithm 2: Metropolis-Hastings

Data: $\pi(x), x_0, N$
Result: $\{x_0, x_1, x_2, \ldots, x_N\}$
for $i \leftarrow 1$ **to** N **do**

1. Sample $Y \sim q(y|x_{i-1})$;
2. Compute $\rho = \min\left[1, \frac{\pi(Y)}{\pi(x_{i-1})} \frac{q(x_{i-1}|Y)}{q(Y|x_{i-1})}\right]$
3. $X_i = \begin{cases} Y & \text{with prob.} \quad \rho \\ x_{i-1} & \text{otherwise} \end{cases}$

It is possible to prove that the chain $\{X_i\}$ generated by Algorithm 2 has a transition kernel K that respects the detailed balance equation with the target density π. So it has π as stationary distribution. As reference see [Jaco9, ch. 5.1.1].

To verify the consistency of Metropolis-Hasting *we have to prove also the ergodicity* of $\{X_i\}$, and so the convergence of the chain variables distribution to the π, independently on the initial value or distribution.

Since we have already prove the existence of the stationary distribution, to prove the positivity we need that $\{X_i\}$ is irreducible.
A sufficient condition for *irreducibility* is that

$$q(y|x) > 0 \ \forall x, y \in \chi.$$

Moreover, irreducibility implies Harris recurrence for chains generated by the Algorithm 2.

Finally aperiodicity is a consequence of $\mathbb{P}(X_{i-1} = X_i) > 0$, that is

$$\mathbb{P}\left(q(Y|x_{i-1})\pi(x_{i-1}) \leq \pi(Y)q(x_{i-1}|Y)\right) < 1$$

In general these are not restrictive conditions on $q(y|x)$.
For instance the Random Walk Metropolis algorithm uses as proposal distribution for the new state $q(y|x)$ a random walk centered in the current state then

$$Y \sim \mathbb{N}(x, \sigma^2)$$

2

HAMILTONIAN MONTE CARLO

In this chapter we will introduce a new type of Markov chain Monte Carlo based on the Metropolis-Hasting scheme. After the description of the algorithm we will sketch its theoretical motivation and some useful changes. Finally we will use a optimization technique to improve the performance of the algorithm from a time and memory consuming point of view. As reference see [Nea11] and [ZS11].

2.1 THE HAMILTONIAN MONTE CARLO ALGORITHM

Our goal is to sample from a given *continuous target density function* $\pi(q)$ in $\mathbb{R}^d$, supposing we are able to compute its logarithm and the partial derivatives of its logarithm.
Hamiltonian Monte Carlo accomplishes this achievement sampling from a joint density $P(q,p)$ of the form 1.18:

$$P(q,p) = \frac{1}{Z} exp(-U(q)) exp(-K(p)) =$$

$$= \frac{1}{Z} \pi(q) exp(-\frac{1}{2} p^T M^{-1} p)$$

with marginal distribution the target density π. So, knowing π, we define the proper functions $U(q)$ and $K(p)$ as specified in 1.16:

$$U(q) = -log[\pi(q)]$$

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

After the definition of the energy the iterative part can start. At each iteration there are two different steps, the first changes only the momentum variable while the second one concerns both position and momentum.

1. In the first step we resample the momentum variable p from its marginal distribution, that is a gaussian distribution independent from the current value of the position q. So we move from the state (q,p) to (q,p') and, since the drawing from the correct conditional (due to independence), the joint distribution does not change.

2. The second step has the basic structure of the classical Metropolis-Hasting sampler, with a new proposed variable accepted using a proper acceptance rate. To propose a new state, instead of using a conditional distribution $q(\cdot|\cdot)$, we evolve the states according to the Hamiltonian dynamics. So, starting from the current point (q, p') we simulate the Hamiltonian dynamics using the leapfrog method, computing the trajectory of L steps of length ε and obtaining the state (q^*, p^*). This proposed state is accepted as the new state of the Markov chain with probability

$$\rho = min[1, exp(-H(q^*, p^*) + H(q, p'))].$$

If it is refused the chain does not move in this iteration and the new state is the same of the current one: $q_i = q_{i-1}$.

The chain explores the joint distribution $P(q, p)$, but we just care of the position variable q while the momentum p is resampled at each iteration. So, on practical implementation, we save only the sequence of the position states $\{q_1, q_2, \ldots, q_N\}$.

Since there are no reason to have correlation between the momentum components the easiest choice of the mass matrix M is the identity matrix: $M = \mathbb{I}_d$.

Hamiltonian Monte Carlo implementation scheme is reported in Algorithm 3. Let us note we use, at each iteration, Algorithm 1 to simulate the dynamics of the system.

Algorithm 3: HMC

Data: $\pi(q), M, q_1, \varepsilon, L, N$
Result: $\{q_1, q_2, \ldots, q_N\}$
Define $U(q) = -log[\pi(q)]$;
for $i \leftarrow 2$ **to** N **do**

1. Sample $p' \sim \mathcal{N}_d(0, M)$;
2. $(q^*, p^*) \leftarrow \text{LeapfrogSim}((q_{i-1}, p'), U(q), M, \varepsilon, L)$;
3. Compute $\rho = min[1, exp(-H(q^*, p^*) + H(q, p'))]$;
4. $q_i = \begin{cases} q^* & \text{with prob. } \rho \\ q_{i-1} & \text{otherwise} \end{cases}$

As pointed up in the description of equation 1.6, the potential energy $U(q)$ can be defined up to an *additive* constant. Since $P(q, p) \propto exp(-U(q))exp(-K(p))$, that matches perfectly with

the property of Metropolis-Hasting scheme in which the target distribution is known up to a *multiplicative* constant.

2.1.1 *Ergodicity of HMC*

Hamiltonian Monte Carlo generates a chain with unique limiting invariant distribution the canonical distribution $P(q, p)$. We now try to explain why.
As we said, the first passage leaves the canonical joint distribution invariant because it samples from the proper marginal, that we choose easy to sample from.
The second step respects the Metropolis-Hasting scheme and has the same invariance property. Indeed, substituting the conditional density $q(\cdot|\cdot)$ with a deterministic evolution of the Hamiltonian system keeps the detailed balance condition respected:

$$P(q_0, p_0)T(q_1, p_1|q_0, p_0) = P(q_1, p_1)T(q_0, p_0|q_1, p_1) \tag{2.1}$$

where $P(q, p)$ is as usual the canonical distribution and T represents the transition kernel of the chain created with the Algorithm 3.
That happens because the properties of reversibility and volume preservation of the Hamiltonian dynamics.
As consequence the chain is reversible and has unique invariant distribution $P(q, p)$.
To understand the limiting property we can imagine it as the capacity of not be stacked in some subset of the variable space. It is achieved by the resampling of the momentum, performed at each step, that can highly change it. So the chain widely explores, independently of choice of the starting point q_1, the space and converges to its invariant distribution.

2.1.2 *Properties of HMC*

Let us compare HMC method with the classic Metropolis-Hasting scheme, illustrated in the previous chapter (Algorithm 2).
The main difference is that in Algorithm 3 the mixing is provided by the first step: it can change the probability $P(q, p)$ for a large amount because the independent modification of the momentum.
In the subsequently step, since the invariance property of the Hamiltonian function:

$$H(q, p') = H(q^*, p^*)$$

and the formula of the canonical distribution:

$$P(q,p) = \frac{1}{Z} exp(-H(q,p))$$

the dynamics moves on a equiprobability surface in the space $\mathbb{R}^{2d}$, affecting the position variable in arbitrary way. As consequence, if simulated exactly, the dynamics leads to an acceptance rate

$$min[1, exp(-H(q^*,p^*) + H(q,p')] = min[1, exp(0)] = 1.$$

As we said describing the leapfrog method, since it is a numeric solution, it is affected by numeric approximation and it does not conserve exactly the Hamiltonian function H. So:

$$H(q,p') \simeq H(q^*,p^*)$$

and

$$min[1, exp(-H(q^*,p^*) + H(q,p')] = min[1, exp(0)] \simeq 1.$$

Then, on practical implementation, the acceptance rate is very high, even not always 1, that is still a good result.

So we provide, at each step, both a significant mixing in the chain and a high acceptance ratio, making the Hamiltonian Monte Carlo a performing Markov chain Monte Carlo method.

The two functions, written in **R**, implementing Algorithm 3 are shown in the code section:

- Listing 4.1 is the code for a single iteration of the chain
- Listing 4.2 implements the chain loop.

2.2 A CHANGE OF VARIABLES

The position variable takes place in $\mathbb{R}^d$ and the chain could have some "favorite moving directions", i.e. it could better explore some components of q, while some others are less considered. That mainly depends on the shape of the function $U(q)$.

Since our goal is to have an algorithm that explores equally every component, we want to mitigate this behavior and have a better description of the density function $\pi(q)$.
To achieve that and improve the performance of the chain, we use at each step i, to compute q_i, an estimate of the Hessian matrix of $U(q)$ for q_{i-1}, in the following way.
Suppose we know, for a given point q_x, the Hessian matrix B of

$U(q)$ and **suppose B is positive-definite**, so we can consider its Cholesky decomposition $B = LL^T$

$$H_q(U(q))\big|_{q=q_x} = B = LL^T. \tag{2.2}$$

If we define a new variable $\check{q}$ and a new potential energy as

$$\begin{cases} \check{q} = L^T q \\ \check{U}(\check{q}) = U(L^{-T}\check{q}) \end{cases}, \tag{2.3}$$

it follows, from the Hessian properties, that

$$\begin{aligned} & H_{\check{q}}(\check{U}(\check{q}))\big|_{\check{q}=\check{q}_x} = \\ & = L^{-1}\, H_q(U(q))\big|_{q=q_x}\, L^{-T} \\ & = L^{-1}BL^{-T} = \\ & = L^{-1}LL^TL^{-T} = \mathbb{I}_d. \end{aligned} \tag{2.4}$$

So, with respect the new variable $\check{q}$, the new function $\check{U}$ is at least locally better conditioned. I.e. the components of $\check{q}$ are less correlated and have similar scale.
We now consider 2.3 as the position and potential energy of our system. Since the dynamics moves according to it, choosing as starting point $\check{q}_x$ the first step explores equally every component.

We now apply the variable change idea exposed above in two simple examples. Let us note how, applying Equations 2.3, the shape of functions changes.

Example 1. *Let us consider a 2-dimension multivariate normal distribution*

$$q = [q_1, q_2]^T \sim \mathcal{N}_2(\mu, \Sigma)$$

with

$$\mu = [3\ 5]^T \quad \Sigma = \begin{bmatrix} 50 & 3 \\ 3 & 2 \end{bmatrix}.$$

Its density function is

$$f(q) = \frac{1}{2\pi\sqrt{|\Sigma|}} exp\left[-\frac{1}{2}p^T\Sigma^{-1}p\right].$$

and its contour lines are shown in figure 2.1. Let us note the big difference in scale between the two components q_1 and q_2.

Consistently with the notation above, we choose $q_x = \mu$. So, the Hessian of f evaluated in μ is

$$H_q(f(q))\big|_{q=\mu} = \Sigma^{-1} = \begin{bmatrix} 50 & 3 \\ 3 & 2 \end{bmatrix}^{-1} = \begin{bmatrix} 0.022 & -0.033 \\ -0.033 & 0.55 \end{bmatrix}.$$

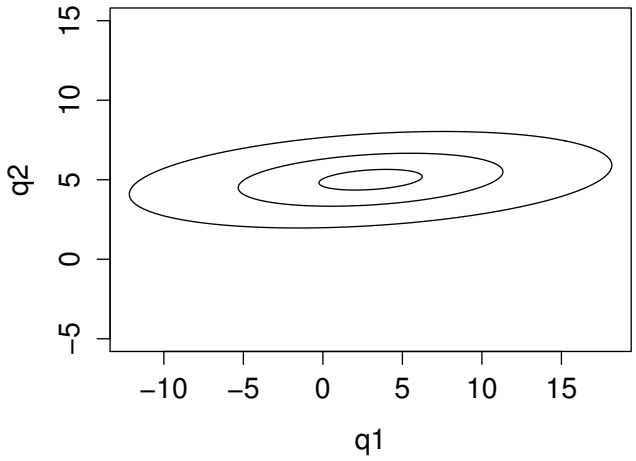

Figure 2.1: q

We obtain the Cholesky decomposition

$$L = \begin{bmatrix} 0.15 & 0 \\ -0.22 & 0.71 \end{bmatrix}.$$

Using Equations 2.2 and 2.3 we define

$$\begin{cases} \check{q} = L^T q \\ \check{f}(\check{q}) = f(L^{-T}\check{q}) \end{cases},$$

The contour lines of the new system are shown in figure 2.2, with

$$\check{q}_x = [-0.67\ 3.54]^T$$

Example 2. *Let us consider the function*

$$U(q) = q_1^2 + q_1 q_2 - q_2^3.$$

Using the same notation as before we define

$$q_x = [-1\ -1]^T$$

$$B = \begin{bmatrix} 2 & 1 \\ 1 & 6 \end{bmatrix}$$

$$L = \begin{bmatrix} 1.4 & 0 \\ 0.71 & 2.3 \end{bmatrix}$$

and so

$$\check{q}_x = [-2.1\ \ -2.3]^T.$$

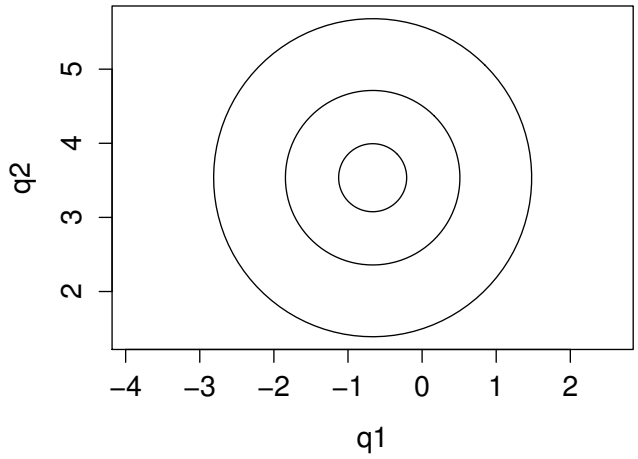

Figure 2.2: $\check{q} = L^T q$

Figure 2.3 shows the contour lines for the original system $(q, U(q)$*, while Figure 2.4 is referred to the new one* $(\check{q}, \check{U}(\check{q}))$.

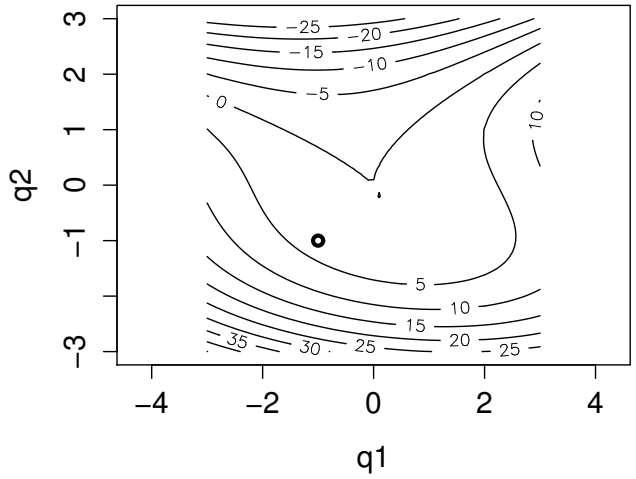

Figure 2.3: q

Example 1 is a very simple case, because the function $f(q)$ we are considering has Hessian $B = \Sigma^{-1}$ constant for each q. As consequence the linear change of variable better rescale the components in each point $\check{q}$, indeed

$$H_{\check{q}}(\check{f}(\check{q})) = \mathbb{I}_2 \quad \forall \check{q} \in \mathbb{R}^2$$

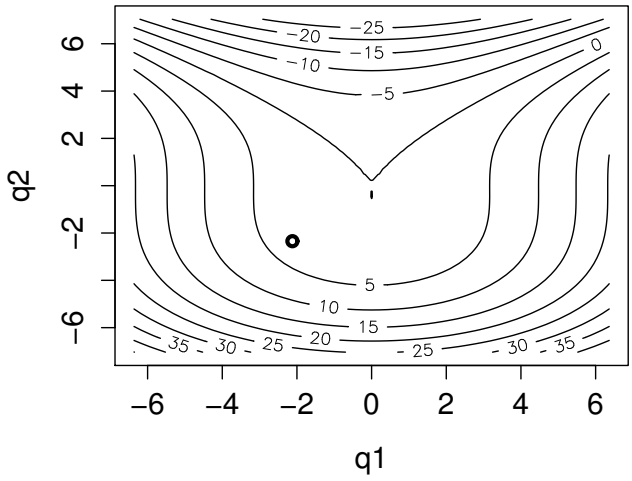

Figure 2.4: $\check{q} = L^T q$

Moreover let us note we can obtain the same result thanks to the multivariate normal distribution property:

$$x \sim \mathcal{N}(\mu, \Sigma) \implies Sx \sim \mathcal{N}(S\mu, S\Sigma S^T)$$

choosing $S = L^T$.

On the contrary, in Example 2 the Hessian matrix of the function is not constant. In Figure 2.4 we note the scale similarity between the two components is in a bound of $\check{q}_x$ (marked with a black dot).

We now return to the original problem. How can we use the change of variable property, described above, to regularize the shape of the U function?
Obviously we are not interested in the variable $\check{q}$, because we want to sample q. Using the property described in subsection 1.1.4 and choosing $A = L^{-T}$ we deduce that the system

$$(\check{q}, p) = (L^T q, p)$$

has the same dynamics of

$$(L^{-T}\check{q}, (L^{-T})^{-T} p) = (q, Lp).$$

So we can improve each step of the algorithm modifying only the momentum variable and consistently its kinetic energy. I.e.,

as explained by equation 1.12, we have just to change the mass matrix $M = \mathbb{I}_d$ with a new one:

$$\hat{M} = A^{-T} M A^{-1} = (L^{-T})^{-T} \mathbb{I}_d (L^{-T})^{-1} = LL^T = B.$$

So, to improve the performance and eliminate the existence of chain drifting favorite directions, determined by the different scale and correlation between the components, at each step we use the Hessian matrix of $U(q)$ evaluated in the current state as the mass matrix of the kinetic energy.

The new improved algorithm is Algorithm 4.

Algorithm 4: HMC_mass

Data: $\pi(q), q_1, \varepsilon, L, N$
Result: $\{q_1, q_2, \ldots, q_N\}$
Define $U(q) = -log[\pi(q)]$;
for $i \leftarrow 2$ **to** N **do**

1. Compute $M = H_q(U(q))\big|_{q=q_{i-1}}$;
2. Sample $p' \sim \mathcal{N}_d(0, M)$;
3. $(q^*, p^*) \leftarrow \text{LeapfrogSim}((q_{i-1}, p'), U(q), M, \varepsilon, L)$;
4. Compute $\rho = min[1, exp(-H(q^*, p^*) + H(q, p')]$;
5. $q_i = \begin{cases} q^* & \text{with prob. } \rho \\ q_{i-1} & \text{otherwise} \end{cases}$

The mass matrix M is different at each step and it is used several times: as covariance of the sampled momentum p', to solve the differential equation in the leapfrog simulation and to compute the kinetic energy.

The code implementation of the new Algorithm 4 is reported in Listing 4.3.
Let us note how the only difference with the previous code 4.2 is the definition of the variable v, while the single iteration script does not change. Because Algorithm 4 evaluates the Hessian matrix of $U(q)$ at each step, on practical implementation we need to write a function (in our case `hess_U()`) to do that.

2.2.1 *Effective sample size*

To show the improvement, we now introduce the concept of *effective sample size* of a Markov chain. For a more detailed description see [RC99] and [RC+10].

Let us consider N random variables X_i, independent and identically distributed. The mean, defined as

$$\bar{X} = \frac{\sum_{i=1}^{N} X_i}{N}, \tag{2.5}$$

is a random variable with variance

$$\frac{Var(X_i)}{N}. \tag{2.6}$$

On the contrary, for a Markov chain of length N, $\{X_i\}$, we have to count the loss in efficiency due to the fact that each variable is not independent to the previous ones. So, using the same formula as the independent case 2.6 underestimates the true variance of the mean. Given a Markov chain, the effective sample size (ESS) is **the size of an iid sample with the same variance of the mean as the current chain**. So the variance of the mean of the Markov chain is

$$\frac{Var(X)}{N_{ESS}}.$$

The ESS can be computed in several ways. For instance, the **R** language, with the command `effectiveSize{CODA}` computes it estimating the spectral density function.
In our work it is particularly useful checking the effective sample size for each component of the chain variable q, to diagnose any wrong behaviors of algorithm in the exploration of every direction of π support.

Example 3. *Let us consider a* 100*-dimension multivariate normal distribution, with all the components independent each others, expected value equal to* 1 *and standard deviation respectively* $0.01, 0.02, \ldots, 0.99, 1$. *So*

$$X \sim \mathcal{N}_{100}(\mu, \Sigma)$$

with

$$\mu = [1 \ \ldots 1 \ 1]^T \quad \Sigma = \begin{bmatrix} 0.01^2 & 0 & \cdots & 0 \\ 0 & 0.02^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1^2 \end{bmatrix}$$

We want a sample $x_1, x_2, \ldots, x_N$ of X, obtained using the two different algorithm. So we run two Markov chains, the first one using the variable change (implemented in Algorithm 4), while the second one without (Algorithm 3).
For each one of the 100 *components of the variable X we compute the mean, the standard deviation and the effective sample size. Figure 2.5 shows the different results using Algorithm 4 (left column) and Algorithm 3 (right column).*

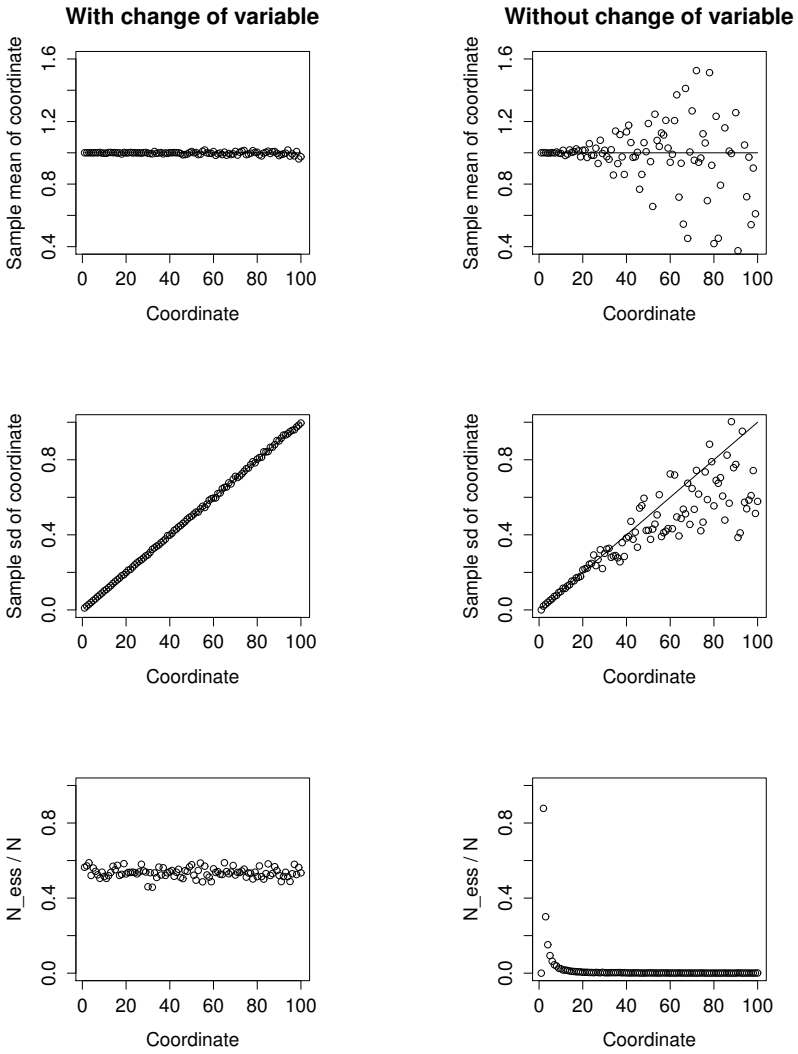

Figure 2.5: Mean, standard deviation and effective sample size of each component, using the two different algorithm. The continuous lines represent the real means and standard deviations

The continuous lines represent the true values of mean and standard deviation, while dots are the approximated ones. The last graphs represent the effective sample size for each components.
Let us note how in the improved method each direction is explored equally (the ESS is more or less constant at 50% for each components). Meanwhile without variable change, the chain explores significantly only the first components and, as consequence, means and standard deviations are worse approximated in the last components, as displayed in the right column.
Finally we remark that using HMC to sample multivariate normal distribution does not have any sense, because at each step of the sampler we have to be able to generate exactly a multivariate normal distribution (obviously different from the target one). Indeed this example has to be considered a simple case without any real application.

2.3 A QUASI-NEWTON METHOD

In this subsection we want to improve our algorithm on the implementation side. In each step of Algorithm 4 we evaluate and save the $d \times d$ Hessian matrix of $U(q)$. This passage can be, especially for high value of d, very expensive from a time and memory consuming point of view.
So we introduce a technique, borrowed from optimization literature, to avoid forming the full Hessian matrix. More precisely, we obtain a second order information (the Hessian) using some first order informations, i.e. the gradient vectors of the previous k states of the chain. The schemes written for this purpose are part of a family called *quasi-Newton methods*. In our work we use one of them called *BFGS*. It can be found and deeply described in [WN99].

2.3.1 *Quasi-Newton in general*

In the context of optimization, quasi-Newton methods work in the following way: given a function $f : \mathbb{R}^d \to \mathbb{R}$ they search for the minimum by generating a sequences $q_{k+2}, q_{k+1}, q_k, \dots$ that converges to it. The q values are iteratively computed using the formula

$$q_{k+1} = q_k - \alpha_k D_k \nabla f(q_k)$$

where D_k is an approximation of the inverse of the Hessian matrix of f evaluated in q_k and α_k another parameter.

The important point is that D_k is computed from the previous values

$$q_k, q_{k-1}, q_{k-2}, \dots$$

and their gradients

$$\nabla f(q_k), \nabla f(q_{k-1}), \nabla f(q_{k-2}), \dots$$

There are several quasi-Newton methods, with different approximation of D_k.
The one we are interested in is *limited-memory BFGS* (L-BFGS). It works iteratively in the following way:

$$D_{k+1} = (\mathbb{I} - \frac{y_k s_k^T}{s_k^T y_k}) D_k (\mathbb{I} - \frac{s_k y_k^T}{s_k^T y_k}) + s_k s_k^T \tag{2.7}$$

with

$$s_k = q_{k+1} - q_k \tag{2.8}$$

$$y_k = \nabla f(q_{k+1}) - \nabla f(q_k) \tag{2.9}$$

and as base case of the recursion

$$D_{k-m} = \mathbb{I}. \tag{2.10}$$

This method is called *limited-memory* because it does not use *all* the previous iterates q, but only the last ones.

Since we are interested in the Hessian of the function $U(q)$ we could use this method to approximate it. It is particularly suitable for our purpose because, since we are saving all the chain path, it is not a problem to keep in memory the last states. Moreover, because the leapfrog method requires the computation of the gradient of $U(q)$, we have already implement it.

2.3.2 *Quasi-Newton in samplers*

So we want to approximate the Hessian matrix, at each step of the chain, exploiting the quasi-Newton technique above. For this purpose we need to introduce some modifications, essential to include the algorithm in the Markov chain Monte Carlo sampler. These changes are sketched in [ZS11], for a more complete description see [BGG73] and [DJS96].

At each step Algorithm 4 needs the Hessian B and its inverse B^{-1} to, respectively, sample the new momentum variable $p' \sim \mathcal{N}_d(0, B)$ and compute the kinetic energy

$$K(p) \;=\; \frac{1}{2} p^T B^{-1} p.$$

We now introduce a variant of the method above. Instead of the Hessian B and its inverse $D = B^{-1}$, it computes, as before iteratively, their decompositions:

$$B_{k+1} = L_{k+1}L_{k+1}^T \quad D_{k+1} = S_{k+1}S_{k+1}^T \tag{2.11}$$

using the equations:

$$L_{k+1} = (\mathbb{I} - u_k t_k^T)L_k \tag{2.12}$$

$$u_k = \frac{s_k}{s_k^T B_k s_k} \quad t_k = \sqrt{\frac{s_k^T B_k s_k}{s_k^T y_k}} y_k + B_k s_k \tag{2.13}$$

$$S_{k+1} = (\mathbb{I} - v_k w_k^T)S_k \tag{2.14}$$

$$v_k = \frac{s_k}{s_k^T y_k} \quad w_k = \sqrt{\frac{s_k^T y_k}{s_k^T B_k s_k}} B_k s_k - y_k \tag{2.15}$$

with the base cases

$$L_{k-m} = S_{k-m} = \mathbb{I}. \tag{2.16}$$

Because of the matrices forms, the matrix-vector product can be computed very fast, as a sequence of inner products

$$S_{k+1}z = \prod_{i=k-m-1}^{k} (\mathbb{I} - v_i w_i^T)S_{k-m}z,$$

with z a general vector.
So we avoid the high memory cost of storing the full $d \times d$ matrix.
Furthermore, we can sample $p' \sim \mathcal{N}_d(0, B)$ more easily, because we know the decomposition of $B = LL^T$:

$$p' = Lw \text{ with } w \sim \mathcal{N}(0, \mathbb{I})$$

2.3.3 *Positive definition of the Hessian matrix*

Finally we need to guarantee that B_k is positive-define, since it is used as a covariance matrix.
It can be proved that, for a convex function f, an optimizer using a quasi-Newton method, always forms a positive-definite Hessian matrix (see [WN99, ch. 8]).

The case of a sampler is totally different. Indeed, we are not defining a sequence q_i with a complete deterministic algorithm, since there is a random component in the choice of the next state

(however we do not want to converge to some point, we want widely explore an area).
So we have to ensure the positive definition of the matrix B_k in the some way. First we sort the previous position states $\{q_i\}$ in ascending order with respect $U(q)$. Then we remove any repeated state $q_i = q_{i-1}$ of the chain, that can be present due to some rejections of the proposed state. Finally we apply the formulae 2.12- 2.15.
The proof of the positive-definition of the matrix obtained is displayed in [JWE92, ch. 9.2].

However if the Hessian of potential energy function U is not positive-definite we can not use it as covariance matrix. Even in this case, sorting the states and using equations 2.12- 2.15 we produce a positive-definite Hessian approximation. Indeed, that happens because the method introduced computes **the best positive-definite approximation of the Hessian matrix**. What "best" means is described in [DJS96] and it is not part of this work. We just say the formulae above are written to form the positive-definite matrix nearest to the Hessian, where nearest is respect to a specific norm.

So we do not have any requirement on the function U, because using equations 2.12- 2.15 we always have an exploitable Hessian approximation.

2.3.4 *Convergence*

To complete the description of the chain obtained with the quasi-Newton Hessian approximation we have to formalize its convergence to the target distribution. We now describe the general idea, for a complete picture see [ZS11].

Since each step uses the previous k states to compute the approximated Hessian, the chain will be then a K order Markov chain. Moreover can be easier analyzed as a first order Markov chain over an enlarged space. So, the position state of the new chain is an ordered set of the last k positions q. At each step only one of its components is updated, using the others $k-1$ to approximate the Hessian and perform a single step of the Hamiltonian Monte Carlo with the quasi-Newton scheme, described above.

That chain is over a $\mathbb{R}^{d\times k}$ space and can be proved it has a stationary limiting distribution:

$$\prod_{i=1}^{k} \pi(q_i).$$

I.e. reached the stationarity, every component is distributed according to the target distribution π. The proof of this property needs the concept of *ensemble-chain adaptation*, see Gilks, Roberts, and George in [GRG94].

3

APPLICATION TO GENERALIZED LINEAR MODELS

In this chapter we will apply the Hamiltonian Monte Carlo, illustrated and implemented in the previous chapter, to a real problem. At first we will describe a wide class of models called Generalized Linear Models (GLMs) and its importance (a full description is available in [Wooo6, ch. 2]).
Then we will introduce some practical situations in which GLMs can be very useful to model the outcomes.

3.1 GENERALIZED LINEAR MODELS

Linear models are statistical models in which a response y_i is represented as the sum of two terms:

1. a linear predictor μ_i
2. a random error ε_i

The linear predictor depends on some predictor variables called *covariates*

$$x_1^{(i)}, x_2^{(i)}, \dots, x_{d-1}^{(i)}$$

and measured with the response variable y_i, and some unknown parameters called *regressors*

$$\beta_0, \beta_1, \dots, \beta_{d-1}.$$

They are related according to the formula:

$$\mu_i = \beta_0 + x_1^{(i)}\beta_1 + x_2^{(i)}\beta_2 + \cdots + x_{d-1}^{(i)}\beta_{d-1}. \tag{3.1}$$

The error is modeled as a zero-mean Gaussian variable

$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2). \tag{3.2}$$

So, the outcome variable y_i is

$$y_i \sim \mathcal{N}(\boldsymbol{X_i\beta}, \sigma^2) \tag{3.3}$$

with expected value

$$\mathbb{E}[Y_i] = \boldsymbol{X_i\beta} \tag{3.4}$$

where $\boldsymbol{X_i}$ is the i^{th} row of the model matrix $\boldsymbol{X}$

$$\boldsymbol{X_i} = [1,\ x_1^{(i)},\ x_2^{(i)}, \ldots, x_{d-1}^{(i)}] \tag{3.5}$$

and $\boldsymbol{\beta}$ is the vector of unknown parameters we want estimate

$$\boldsymbol{\beta} = [\beta_0, \beta_1, \ldots, \beta_{d-1}]^T. \tag{3.6}$$

Generalized Linear Models (GLMs) relax the hypothesis of normal distribution of y_i (Equation 3.3) and the linear dependence of the mean with respect the predictors and the regressors (Equation 3.4).
Indeed, the new assumption on which GLMs are based is

$$g(\mu_i) = \boldsymbol{X_i}\boldsymbol{\beta} \tag{3.7}$$

where

$$\mathbb{E}[Y_i] = \mu_i \tag{3.8}$$

and g is a smooth monotonic *link function*, that represents the non-linearity of the model.

Moreover, we assume:

$$Y_i \sim \text{some exponential family distribution.}$$

As shown in the next subsection, the exponential family of distributions includes Normal distribution and many others, very used in practical modeling, such as Poisson, Binomial and Gamma distributions.

To explain the adjective *generalized* let us note that, if the link function is chosen as the identity $f(x) = x$, and the normal distribution of Y_i is assumed, then linear models are recovered as a particular case of GLM.

3.1.1 *The exponential family of distributions*

In a GLM we assume that the response variable Y_i can have any distribution from the exponential family. A distribution belongs to the exponential family if its probability density function, can be written in the following form:

$$f_\theta(y) = exp\left[\frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)\right] \tag{3.9}$$

where

- $a(\phi), b(\theta)$ and $c(y, \phi)$ are arbitrary functions

- ϕ an arbitrary "scale" parameter
- θ another parameter, called *canonical parameter*.

We choose the notation $f_\theta(y)$ to point out that the density distribution is a function on the variable y, while θ is a parameter.

It is easy to see that Normal and Poisson distributions belong to the exponential family, as shown in the following examples:

Example 4. *The normal distribution density can be written as*

$$f(y) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left[-\frac{(y-\mu)^2}{2\sigma^2}\right] =$$

$$= exp\left[\frac{-y^2+2y\mu-\mu^2}{2\sigma^2} - log(\sqrt{2\pi\sigma^2})\right] =$$

$$= exp\left[\frac{y\mu-\mu^2/2}{\sigma^2} - \frac{y^2}{2\sigma^2} - log(\sqrt{2\pi\sigma^2})\right]$$

So, referring to Equation 3.9, we define:

$$\begin{array}{c||c} \theta & \mu \\ \phi & \sigma^2 \\ a(\phi) & \phi \\ b(\theta) & \frac{\theta^2}{2} \\ c(y,\phi) & -\frac{y^2}{2\phi} - log(\sqrt{2\pi\phi}) \end{array}$$

Example 5. *Also a discrete random variable can belong to the exponential family. Indeed, we can rewrite the Poisson mass function in the following way:*

$$f(y) = \frac{\mu^y e^{-\mu}}{y!} =$$

$$= exp\left[ylog(\mu) - \mu - log(y!)\right] =$$

$$= exp\left[\frac{ylog(\mu) - e^{log(\mu)}}{1} - log(y!)\right]$$

As before, referring to Equation 3.9:

$$\begin{array}{c||c} \theta & log(\mu) \\ \phi & 1 \\ a(\phi) & \phi \\ b(\theta) & e^\theta \\ c(y,\phi) & -log(y!) \end{array}$$

3.1.2 *Likelihood function*

Let us consider a random variable Y with some exponential family distribution.

Given a realization y of Y, we define the *likelihood function* $L(\theta)$ simply as $f_\theta(y)$ as a function of θ.

$$L(\theta) = exp\left[\frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)\right]. \tag{3.10}$$

It can be easily prove (see [Wooo6, ch. 2.14] or [JWE92]) the relation between the canonical parameter θ and the expected value of Y is expressed by the formula:

$$\mathbb{E}[Y] = b'(\theta). \tag{3.11}$$

We skip the proof of this result because describing the theory of likelihood function and its property is not a goal of this work. We just note that, for the two examples above, the relation is immediately verified, as shown below.

Example 6. *Recall that in the Gaussian case*

$$b(\theta) = \frac{\theta^2}{2} \qquad \theta = \mu$$

and so

$$b'(\theta) = \theta = \mu$$

Example 7. *Poisson case*

$$b(\theta) = e^\theta \qquad \theta = log(\mu)$$

and so

$$b'(\theta) = e^\theta = e^{log(\mu)} = \mu$$

3.1.3 *Fitting Generalized Linear Models*

We now consider a vector of n random variables Y_i:

$$\boldsymbol{Y} = [Y_1,\ Y_2,\ \ldots Y_n]^T$$

and the vector $\boldsymbol{y}$ of observed values, realization of $\boldsymbol{Y}$:

$$\boldsymbol{y} = [y_1,\ y_2, \ldots y_n]^T.$$

We assume the variables $Y_1,\ Y_2,\ \ldots Y_n$ are independent and each one Y_i is distributed according an exponential family distribution f, with canonical parameter θ_i:

$$Y_i \sim f_{\theta_i}(y_i).$$

We can also define the likelihood function L of $\boldsymbol{Y} = \boldsymbol{y}$. Due to independence of Y_i it is:

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} L_i(\theta_i) \tag{3.12}$$

where $L_i(\theta_i)$ is the likelihood function of θ_i given the observation y_i, as defined in 3.10.
Let us note that L is a function of the vector of canonical parameters

$$\boldsymbol{\theta} = [\theta_1, \theta_2, \dots \theta_n]^T.$$

Considering the data $\boldsymbol{y}$ coming from a GLM, we know, recalling Equation 3.7:

$$g(\mu_i) = \boldsymbol{X_i}\boldsymbol{\beta} \tag{3.13}$$

and, because the property of likelihood 3.11:

$$\mu_i = \mathbb{E}[Y_i] = b'(\theta_i). \tag{3.14}$$

we can express the likelihood function L as a function of $\boldsymbol{\beta}$

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} L_i(\theta_i(\boldsymbol{\beta})) \tag{3.15}$$

We now give a simple but complete example of the setting illustrated above.

Example 8. *Let us consider*

$$Y_i \sim Poi(\mu)$$

and the link function 3.7

$$log(\mu_i) = \boldsymbol{X_i}\boldsymbol{\beta}.$$

The observed values, with $d = 2$ and $n = 3$ are

$$\boldsymbol{X_1} = \left[1,\ \frac{1}{3}\right],\ y_1 = 12$$

$$\boldsymbol{X_2} = \left[1,\ \frac{1}{2}\right],\ y_2 = 26$$

$$\boldsymbol{X_3} = [1,\ 1],\ y_3 = 52$$

We can represent Equation 3.7 using the model matrix $\boldsymbol{X}$:

$$g(\boldsymbol{\mu}) = \boldsymbol{X}\boldsymbol{\beta}$$

$$g\left(\begin{bmatrix}\mu_1\\ \mu_2\\ \mu_3\end{bmatrix}\right) = \begin{bmatrix}1 & \frac{1}{3}\\ 1 & \frac{1}{2}\\ 1 & 1\end{bmatrix}\begin{bmatrix}\beta_0\\ \beta_1\end{bmatrix}$$

So, the likelihood function L as a function of $\boldsymbol{\beta}$, introduced above as Equation 3.15, is:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} \frac{\mu_i(\boldsymbol{\beta})^{y_i} e^{-\mu_i(\boldsymbol{\beta})}}{y_i!} = \prod_{i=1}^{n} \frac{e^{\boldsymbol{X_i}\boldsymbol{\beta} y_i} e^{-e^{\boldsymbol{X_i}\boldsymbol{\beta}}}}{y_i!} =$$

$$= \frac{e^{\beta_0+\frac{1}{3}\beta_1} e^{-e^{\beta_0+\frac{1}{3}\beta_1}}}{12!} \cdot \frac{e^{\beta_0+\frac{1}{2}\beta_1} e^{-e^{\beta_0+\frac{1}{2}\beta_1}}}{26!} \cdot \frac{e^{\beta_0+1\beta_1} e^{-e^{\beta_0+1\beta_1}}}{52!}$$

In this example the link function g is chosen equal to the function $\theta = \theta(\mu)$, so we obtain:

$$\theta_i = \boldsymbol{X_i}\boldsymbol{\beta}. \tag{3.16}$$

In a GLM that is a common choice of g for several reasons we will not consider in this work. We just note that, one of the consequences of 3.16 is that Equation 3.15 is a convex function in $\boldsymbol{\beta}$. I.e. its Hessian matrix is positive-definite. For the proof of that fact and a complete exposition of the other effects see [BS77, ch. 2].

We now return on the original GLM problem: given the observed values

$$(\boldsymbol{X_i},\ y_i) \ \text{ for } \ i = 1, 2, \ldots, n$$

we want to estimate the regressors vector

$$\boldsymbol{\beta} = [\beta_0,\ \beta_1, \ldots, \beta_{d-1}]^T.$$

The classical approach is **to compute the vector $\hat{\boldsymbol{\beta}}$ that maximize the likelihood function 3.15**

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} L(\boldsymbol{\beta}) \tag{3.17}$$

$\hat{\boldsymbol{\beta}}$ is pointwise estimate of the unknown parameter $\boldsymbol{\beta}$ called *maximum likelihood estimate*

3.2 HMC FOR GENERALIZED LINEAR MODELS

We introduce another strategy to solve the GLM problem, requiring some basic knowledge of Bayesian statistics. Thus we sketch here the central ideas on the topic, useful to describe the method.
Nevertheless Bayesian statistics is deeply studied in literature and the reader can easily find many texts on the argument, for example see [Chr+11].

As before, given

$$(\boldsymbol{X_i},\ y_i) \ \text{ for } \ i = 1, 2, \ldots, n$$

we build the likelihood function $L(\boldsymbol{\beta})$, as defined in 3.15 .

According to the Bayesian approach, the parameters are random variable. A distribution is assigned on the vector of parameters, according to information available "prior to see" the data. If such information is difficult to elicit, standard "non-informative" priors can be used instead. Let us denote the *prior distribution* of the regression parameters $\boldsymbol{\beta}$ with:

$$f(\boldsymbol{\beta}). \tag{3.18}$$

Then we define the *posterior distribution*:

$$f(\boldsymbol{\beta}|\boldsymbol{X}, \boldsymbol{Y} = \boldsymbol{y}) \propto L(\boldsymbol{\beta}) f(\boldsymbol{\beta}) \tag{3.19}$$

Prior distribution 3.18 represents a belief before the observation of the outcomes. Meanwhile the posterior one 3.19 expresses the knowledge on $\boldsymbol{\beta}$ after the evidences have been observed.
If we do not have any information on $\boldsymbol{\beta}$ before the collection of data, the prior, called *non-informative*, is a sort of uniform distribution all over the $\boldsymbol{\beta}$ space and its density is proportional to a constant. Because its nature $f(\boldsymbol{\beta})$ is not a distribution anymore, for that reason it is also called *improper prior*.
As consequence 3.19 is proportional to the likelihood:

$$f(\boldsymbol{\beta}|\boldsymbol{X}, \boldsymbol{Y} = \boldsymbol{y}) \propto L(\boldsymbol{\beta}). \tag{3.20}$$

So, we are interested in $\boldsymbol{\beta}$, a random variable with the distribution 3.20. We explore it using the Hamiltonian Monte Carlo introduced in the previous chapter.

Let us note, as pointed up before, we just need to know the distribution of interest up to a multiplicative constant. Since we concern 3.20, consistently with the former notation, the target function $\pi(q)$ is now $L(\boldsymbol{\beta})$.

Consequently the correct potential energy function is, as explained in 1.16:

$$\begin{aligned} U(\boldsymbol{\beta}) &= -\log[\pi(q)] = \\ &= -\log\left[\prod_{i=1}^{n} L_i(\theta_i(\boldsymbol{\beta}))\right] = \\ &= -\sum_{i=1}^{n} \log\left[L_i(\theta_i(\boldsymbol{\beta}))\right] = \\ &= -\sum_{i=1}^{n} l_i(\theta_i(\boldsymbol{\beta})) \end{aligned} \tag{3.21}$$

Let us note that the function $l(x) = \log[L(x)]$, called *log-likelihood*, is widely used in statistics.

While the classic approach gives us a pointwise estimate for regressors, we now are able, knowing the empirical distribution, to build a *credibility interval* for $\boldsymbol{\beta}$ and, as consequence, for its components $\beta_0,\ \beta_1, \ldots, \beta_{d-1}$.

We now apply this approach continuing Example 8.

Example 9. *Remanding the likelihood function:*

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} \frac{\mu_i(\boldsymbol{\beta})^{y_i} e^{-\mu_i(\boldsymbol{\beta})}}{y_i!}$$

we define the potential energy U, function of $\boldsymbol{\beta}$

$$U(\boldsymbol{\beta}) = -\log L(\boldsymbol{\beta}) =$$

$$= \sum_{i=1}^{n} \left[-y_i \log[\mu_i(\boldsymbol{\beta})] + \mu_i(\boldsymbol{\beta}) + \log[y_i!]\right] =$$

$$= \sum_{i=1}^{n} \left[-y_i \log[\mu_i(\boldsymbol{\beta})] + \mu_i(\boldsymbol{\beta})\right] + \sum_{i=1}^{n} \left[\log[y_i!]\right]$$

Since $\sum_{i=1}^{n} [\log[y_i!]]$ does not depend on $\boldsymbol{\beta}$ and $U(\boldsymbol{\beta})$ is defined up to an additive constant, we can remove it and re-define:

$$U(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[-y_i \log[\mu_i(\boldsymbol{\beta})] + \mu_i(\boldsymbol{\beta})\right]$$

Finally, reminding

$$log(\mu_i) = \boldsymbol{X_i}\boldsymbol{\beta}$$

we obtain

$$U(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[-y_i \boldsymbol{X_i}\boldsymbol{\beta} + e^{X_i\beta}\right]$$

So, substituting the data

$$U(\boldsymbol{\beta}) = -12\left(\beta_0 + \frac{1}{3}\beta_1\right) + e^{\beta_0+\frac{1}{3}\beta_1} - 26\left(\beta_0 + \frac{1}{2}\beta_1\right) + e^{\beta_0+\frac{1}{2}\beta_1} - \\ - 52\left(\beta_0 + 1\beta_1\right) + e^{\beta_0+1\beta_1}$$

We explore that function using Algorithm 4. The functions $U, \nabla U$ and $H(U)$, required to run the chain, are implemented in the code displayed in Listing 4.4.

We run 10 chains with length $N = 10^3$, parameters $\varepsilon = 0.5,\ L = 1$ and starting points randomly chosen as uniformly distributed variables over $[0,\ 10] \times [0,\ 10]$.
Traceplot of the first variable is shown in Figure 3.1 and of the second one in 3.2. The black continuous lines represent the true values of β_0 and β_1 used to generate the data, i.e. $\beta_0 = \beta_1 = 2$. The red lines are their maximum likelihood estimates (computed by **R** *built-in function* `glm{stats}`*).*

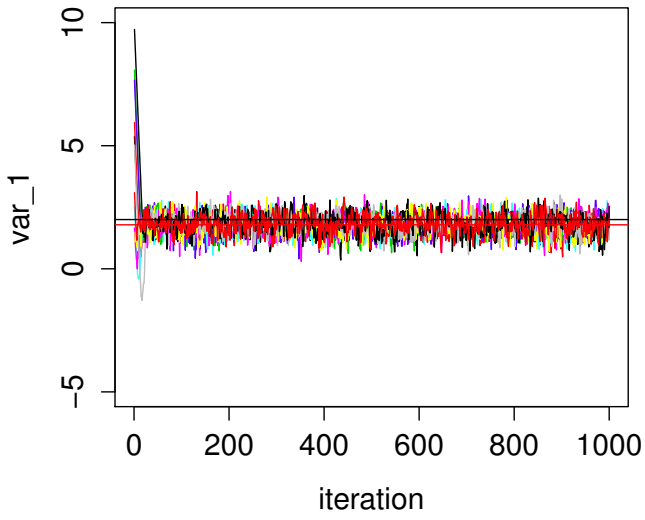

Figure 3.1: β_0: traceplots of the 10 chains, true value (black) and MLE (red)

We can be easily note that the choice of the starting point does not affect the convergence. Indeed zooming the graphs we see the convergence is reached, more or less, after 40 iterations for both the components.

Since $\boldsymbol{\beta}$ is just a 2-dimension vector, we can plot, in Figure 3.5, the (position part of the) chains in $\mathbb{R}^2$, over the contour of U. Let us note again the convergence of all the chains, independently of the starting points.

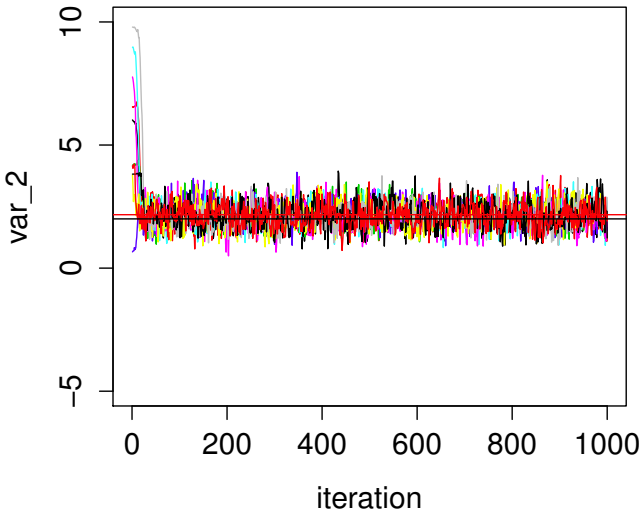

Figure 3.2: β_1: traceplots of the 10 chains, true value (black) and MLE (red)

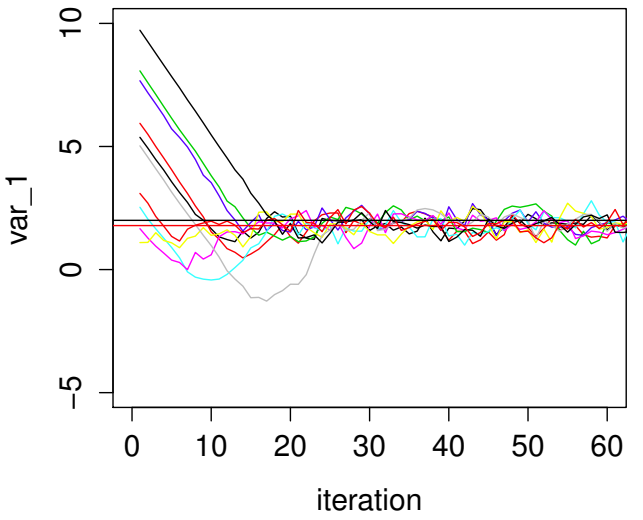

Figure 3.3: β_0: first 60 iterations of the chains

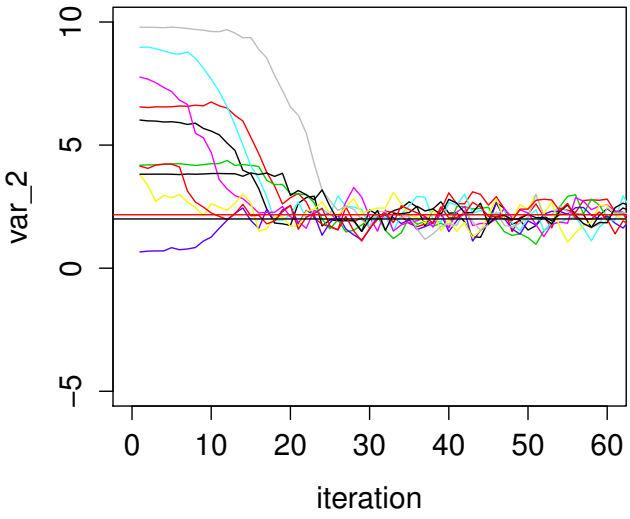

Figure 3.4: β_1: first 60 iterations of the chains

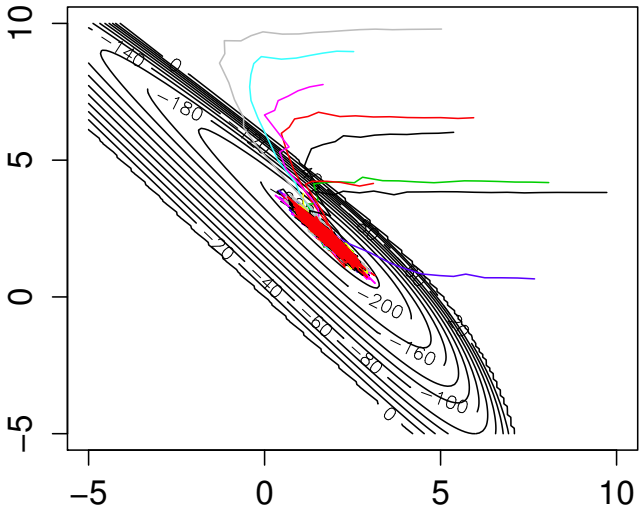

Figure 3.5: Walks of the chains in $\mathbb{R}^2$, over the contour lines of U

Finally, we show, in Figure 3.6, the output of the `summary{base}` *command on the McMC list. Using empirical quantile, we can build credibility intervals for the* β *components: let us note how they include the real values of* β.

```
Iterations = 1:1000
Thinning interval = 1
Number of chains = 10
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

      Mean     SD Naive SE Time-series SE
[1,] 1.747 0.5073 0.005073        0.01182
[2,] 1.966 0.8395 0.008395        0.02018

2. Quantiles for each variable:

        2.5%   25%   50%   75% 97.5%
[1,] 1.04415 1.446 1.761 2.036 2.528
[2,] 0.04426 1.726 2.090 2.418 3.052
```

Figure 3.6: `summary` of the chains list

3.3 A PRACTICAL EXAMPLE: AD OPTIMIZATION

We now describe a practical use of HMC: the *ad optimization*, we refer to [Wäs12] for more extended studies on the topic.
On-line advertisements are promotional market messages displayed on web pages, they consist in pictures, texts or animations. As ordinary advertisements, they are created to promote products among consumers.
The success of an on-line advertisement is measured with the number of clicks it receives: more clicks means more profits.
Our goal is to increase the earnings for on-line publishers. To achieve that, given an ad, we want to infer its number of received clicks basing on some available informations, such as ad content or hosting website. So we will be able to optimize an ad, knowing which parameters are involved and how, in the revenues.

Thus we build a Generalized Linear Model predicting the expected number of clicks via the covariates:

- *Placement id (p):* the position of the ad in the web page
- *Material id (m):* the name of the advertisement
- *Advertiser id (a):* the advertiser who has designed the ad

We know about n ads displayed on various website, let consider one of them. We define the *i-th observation* as:

- the number of clicks received Y_i

- the covariates $(p_i,\ m_i,\ a_i)$

We can now introduce the model hypothesis.

The basic idea is that each covariates possible value affects the model in a different way. So we define a parameter β_x for each one. For example, let P the ordered set of all possible *placements* p:

$$P = \{top, bottom, left, right\},$$

as consequence we have the regressors

$$\beta_{top}, \beta_{bottom}, \beta_{left}, \beta_{right}.$$

Consistently with the previous chapter, we now define $\boldsymbol{X}_i$, the i-th row of the model matrix $\boldsymbol{X}$. Thus, if corresponding to the i-th outcome Y_i we observe the covariate $p_i = top$, $\boldsymbol{X}_i$ is defined as

$$\boldsymbol{X}_i = [1,\ 1,\ 0,\ 0,\ 0, \dots]$$

Where, the first element equal to 1 is due, as usual, at the presence of the intercept β_0. Whereas the second, third, fourth and fifth elements represent the covariate $p_i = top$. More precisely the components have boolean values, 1 in correspondence to the observed one and 0 the others.
The dots mean that there are other 0 and 1 values, corresponding to the other covariates: material, frequency,...

I.e. supposing the observed covariates

$$(p_i,\ m_i,\ a_i),$$

because the $\boldsymbol{X}_i$ defined above, we obtain

$$\boldsymbol{X_i\beta} = \beta_0 + \beta_{p_i} + \beta_{m_i} + \beta_{a_i}.$$

As in the previous easier example, we suppose the outcomes Y_i Poisson distributed

$$Y_i \sim Poi(\mu)$$

and the link function

$$log(\mu_i) = \boldsymbol{X_i\beta}.$$

We immediately note that, as consequence of this kind of modelization, the dimension of the GLM is very high. Indeed the number of regressors is $c = 1 + |P| + |M| + |A|$, where $P,\ M,\ A$, are the ordered set of placement, material, frequency and advertiser respectively.

Moreover, the model matrix $\boldsymbol{X}$ is sparse. On the implementation side that can be exploited to write more performing code. However the analysis of this aspect is not included in our work.

Finally we give to all regressors a prior distribution. The assumption is that parameters β are independent, zero mean normally distributed, with different variant according to the set they are associated to (P, M or A).
Normal distribution is chosen because its simple analytical form and its light tails, avoiding extreme values results.

$$\beta_i \sim \mathcal{N}(0, \sigma^2_{x_i}) \ \forall i \tag{3.22}$$

where

$$x_i \in \{P,\ M,\ A\} \quad s.t. \quad i \in x_i \tag{3.23}$$

So the prior is:

$$f(\boldsymbol{\beta}) = \prod_{s=0}^{c-1} f_{\mathcal{N}(0,\sigma^2_{x_s})}(\beta_s) \tag{3.24}$$

The prior parameters σ_x, called *hyperparameters*, are fixed and chosen basing on the previous knowledge we have on $\boldsymbol{\beta}$.

The posterior distribution 3.19 is not simply proportional to the likelihood anymore, we have to take in account a "belief before the outcomes" about the regressors.

$$f(\boldsymbol{\beta}|\boldsymbol{X}, \boldsymbol{Y} = \boldsymbol{y}) \propto L(\boldsymbol{\beta}) f(\boldsymbol{\beta}) \tag{3.25}$$

$$L(\boldsymbol{\beta}) f(\boldsymbol{\beta}) = \prod_{i=1}^{n} \frac{\mu_i(\boldsymbol{\beta})^{y_i} e^{-\mu_i(\boldsymbol{\beta})}}{y_i!} \prod_{s=0}^{c-1} f_{\mathcal{N}(0,\sigma^2_{x_s})}(\beta_s) \tag{3.26}$$

The potentially energy function is

$$\begin{aligned} U(\boldsymbol{\beta}) &= -\log\left[\prod_{i=1}^{n} \frac{\mu_i(\boldsymbol{\beta})^{y_i} e^{-\mu_i(\boldsymbol{\beta})}}{y_i!} \prod_{s=0}^{c-1} f_{\mathcal{N}(0,\sigma^2_{x_s})}(\beta_s)\right] = \\ &= \sum_{i=1}^{n} \left[-y_i \log[\mu_i(\boldsymbol{\beta})] + \mu_i(\boldsymbol{\beta}) + \log[y_i!]\right] + \\ &- \sum_{s=0}^{c-1} \left[\frac{1}{2}\log[2\pi\sigma^2_{x_s}] + \frac{\beta_s^2}{2\sigma^2_{x_s}}\right] \end{aligned} \tag{3.27}$$

We can eliminate the additive parts constant with respect to β and redefine U as

$$U(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[-y_i \log[\mu_i(\boldsymbol{\beta})] + \mu_i(\boldsymbol{\beta})\right] - \sum_{s=0}^{c-1} \left[\frac{\beta_s^2}{2\sigma^2_{x_s}}\right]. \tag{3.28}$$

According with the model descripted above, we now show an example of ad optimization. Chosen the true $\boldsymbol{\beta}$ parameters, we simulate, using **R** bult-in functions, the outcomes $\boldsymbol{Y}$.

Example 10. *Consistently with the above notation, we define:*

- *Number of observations:* $n = 200$
- *Number of covariates:* $|P| = 2,\ |M| = 2,\ |A| = 3$
- *Hyperparameters:* $\sigma^2_{x_i} = 10^4\ \forall i$

Then the number of regressors is $c = 1 + 2 + 2 + 3 = 8$.
Using Algorithm 4, we run an McMC of length $N = 10^4$. *We remove, as burn-in period, the first* 1000 *iterations. Figure 3.7 shows traceplots, one for each regressor. The black continuous lines represent the true values of* β_i, *fixed and used to generate the data. The red lines are their maximum a-posteriori estimates. Green lines are* 2.5%, 50% *and* 97.5% *empirical quantiles of the chain. They create credibility intervals, which include the true values of* $\boldsymbol{\beta}$.
The traceplots show clearly the convergence of the chain to its stationary distribution.

The code generating data and required functions is displayed in Listing 4.5

The analysis of non-simulated datasets is not part of this work. Anyway we conclude analyzing the performance of the scheme with gradually larger datasets, to give an idea of the potentiality for future applications.
Indeed we run a short chain (10 iterations) for increasing dimensions, to explain the time consuming as function of the dimension.

As displayed in Figure 3.8, the time elapsed seems to be linear with respect to the dimension, so using HMC to solve bigger dimension GLM is a promising direction for future works.

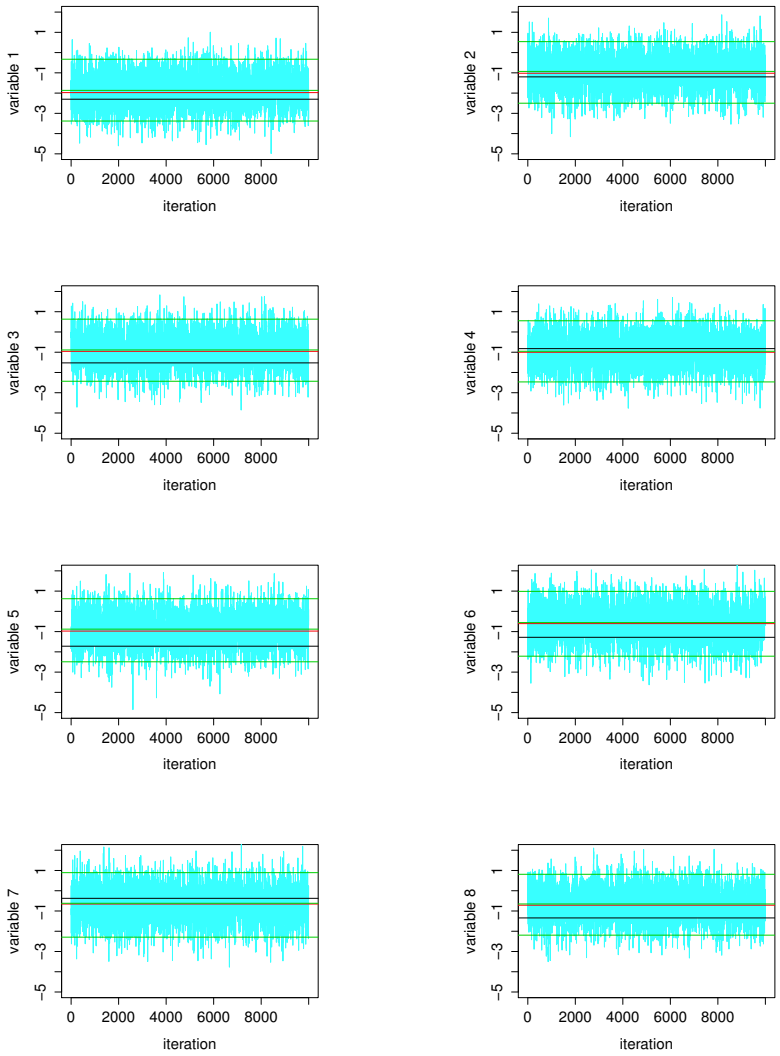

Figure 3.7: Traceplots of the components of the chain, corresponding to a component of β, true value (black), MAP (red) and quantiles (green)

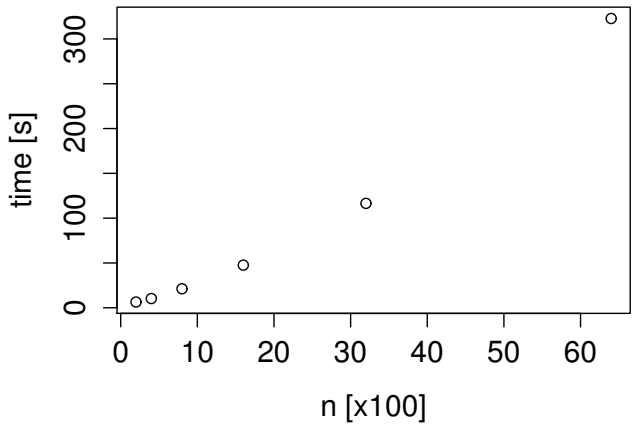

Figure 3.8: Total elapsed time for running the chain as function of the problem dimension

4

CONCLUSIONS AND FUTURE WORK

This work can be considered a simple introduction to Hamiltonian Monte Carlo and its application. The operation of the chain algorithm is completely depicted and theoretically motivated, pseudo-code algorithms are also implemented in **R** language. Application to a class of model has been tested on a small simulated dataset. The results are encouraging and seem to suggest good performance on real problems too.
The use of HMC in real applications is an interesting direction for future work and several modifications are required. The codes are written in order to clearly show the working procedure, rather than to maximize the efficiency. For example the memory usage optimization is not taken into account so far. Moreover implementing with a lower-level programming language, such as `C++`, is recommended, as it can speed up considerably the algorithm.

Generalized Linear Models, shortly described in the final chapter, are a family of models often involved in many current problems. Following the Bayesian approach, we compute the posterior distributions of the regression parameters. Because of the efficiency in the exploration of the target density function, Hamiltonian Monte Carlo is particularly promising: it can be used when the dataset dimension is large and classical McMC methods fail.
Indeed, several industry cases and machine learning challenges, modeled with GLM, could use HMC to obtain the posterior distributions.

Listing 4.1: Implementation of Alg. 3 iteration

```
HMC_mass = function (U, grad_U, M, epsilon, L,
  current_q)
{
  q = current_q
  d <- dim(M)[1]
  p = mvrnorm(1,mu=rep(0,d),M)
  current_p = p

  ##leapfrog
  p = p - epsilon * grad_U(q) / 2
  for(i in 1:L)
  {
    q = q + epsilon * solve(M,p)
    if (i!=L) p = p - epsilon * grad_U(q)
  }
  p = p - epsilon * grad_U(q) / 2

  # Evaluate potential and kinetic energies
  # at the beginning and
  # at the end of trajectory
  current_U = U(current_q)
  current_K = t(current_p) %*%
   solve(M, current_p) /2
  proposed_U = U(q)
  proposed_K = p %*% solve(M, p) /2

  # Accept or reject the state at
  # end of trajectory
  if ( runif(1) < exp(current_U-proposed_U+
  current_K-proposed_K) )
    return (q)  # accept
  else
    return (current_q)  # reject
}
```

Listing 4.2: Implementation of Alg. 3 loop

```
source('new_hmc_step.R', echo=F)

HMC_MC <- function(U, grad_U, epsilon, L, N,
  start)
{
  d <- length(start)
  chain<-matrix(data=rep(0,N*d), nrow=d)
  chain[,1]=start      #starting point
  v <- diag(d)
```

```
  for(i in 2:N)
    chain[,i]= HMC_mass(U, grad_U, v,
                  epsilon, L, chain[,i-1])

  return(t(chain))
}
```

Listing 4.3: Implementation of Alg. 4 loop

```
source('new_hmc_step.R', echo=F)

HMC_MC <- function(U, grad_U, hess_U, epsilon,
L, N,  start)
{
  d <- length(start)
  chain<-matrix(data=rep(0,N*d), nrow=d)
  chain[,1]=start       #starting point

  for(i in 2:N)
  {
    v<- hess_U(chain[,i-1])
    chain[,i]= HMC_mass(U, grad_U, v,
    epsilon, L, chain[,i-1])
  }


  return(t(chain))
}
```

Listing 4.4: Implementation of the functions in example 9

```
xx <- matrix( data=1:n/n, nrow = n, ncol=1)
x <- cbind( rep(1,n), xx)
y <- c(12,26,52)

#potential energy
U <- function(beta)
{
  return(sum(exp(x %*% beta) - y * (x %*% beta)))
}

#gradient
grad_U <- function(beta)
{
  temp <-    exp(x %*% beta)-y
  return( colSums( matrix( temp, nrow=length(temp),
   ncol= dim(x)[2]) * x) )
}

#hessian
hess_U <- function(beta)
{
      temp <- matrix(0, nrow=d, ncol=d)
```

```
    for(j in 1:n)
    {
        temp <- temp + exp(x %*% beta)[j] *
         (x[j,] %*% t(x[j,]))
    }
    return(temp)
}
```

Listing 4.5: Implementation of data generation and functions in example 10

```
# Placement
nPlac <- 2
allPlac = 1:nPlac
allPlacIds = factor(paste("P",allPlac,sep=""),
    levels=paste("P",sort(allPlac),sep=""))
betaPlac <- rnorm(nPlac, 0,1)
names(betaPlac)<- allPlacIds

# Materials
nMaterial <- 2
allMaterial = 1:nMaterial
allMaterialIds = factor(paste("M",allMaterial,sep=""),
    levels=paste("M",sort(allMaterial),sep=""))
betaMaterial <- rnorm(nMaterial, 0,1)
names(betaMaterial)<- allMaterialIds

# Advertisers
nAdver <- 3*A
allAdver = 1:nAdver
allAdverIds = factor(paste("A",allAdver,sep=""),
    levels=paste("A",sort(allAdver),sep=""))
betaAdver <- rnorm(nAdver, 0,1)
names(betaAdver)<- allAdverIds

# Intercepts
icept <- rnorm(1)

# True Beta
b_true <- c(icept,betaPlac,betaMaterial, betaAdver)

# Outcomes
n <- 200
placIx <- sample(1:nPlac, n, replace=T)
materialIx <- sample(1:nMaterial, n, replace=T)
adverIx <- sample(1:nAdver, n, replace=T)
true_mean <- exp( icept + betaPlac[placIx] +
betaMaterial[materialIx] + betaAdver[adverIx])
y <- rpois(n, true_mean)


Data <- data.frame(
    placment= allPlacIds[placIx],
    material= allMaterialIds[materialIx],
```

```
      advertiser = allAdverIds[adverIx],
      nrOfClicks=y
);


# m = nrOfClicks ~ placment + material + advertiser
xi <- model.Matrix(~1, Data, sparse=T)
xp <- model.Matrix(~placment-1, Data, sparse=T)
xm <- model.Matrix(~material-1, Data, sparse=T)
xa <- model.Matrix(~advertiser-1, Data, sparse=T)
x  <- cBind(xi,xp,xm,xa)

d <- ncol(x)

#potential energy
U <- function(beta)
{
      return(sum(exp(x %*% beta) - y * (x %*% beta))
            + sum(beta^2)/2)
}

#gradient
grad_U <- function(beta)
{
      temp <- exp(x %*% beta)-y
      return( colSums( matrix( temp, nrow=length(temp),
            ncol= dim(x)[2]) * x) + beta )
}

#hessian
hess_U <- function(beta)
{
      temp <- matrix(0, nrow=d, ncol=d)
      for(j in 1:n)
            temp <- temp + exp(x %*% beta)[j] *
                  (x[j,] %*% t(x[j,]))

      return(temp + diag(d))
}
```

LIST OF FIGURES

LISTINGS

BIBLIOGRAPHY

[AW59] Berni J Alder and TE Wainwright. "Studies in molecular dynamics. I. General method". In: *The Journal of Chemical Physics* 31.2 (1959), pp. 459–466.

[BGG73] Ken W Brodlie, AR Gourlay, and John Greenstadt. "Rank-one and rank-two corrections to positive definite matrices expressed in product form". In: *IMA Journal of Applied Mathematics* 11.1 (1973), pp. 73–82.

[BS77] Peter J Bickel and K Doksum Mathematical Statistics. *Basic Ideas and Selected Topics*. 1977.

[Chr+11] Ronald Christensen et al. *Bayesian ideas and data analysis: An introduction for scientists and statisticians*. CRC Press, 2011.

[DJS96] John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Vol. 16. Siam, 1996.

[Dua+87] Simon Duane et al. "Hybrid monte carlo". In: *Physics letters B* 195.2 (1987), pp. 216–222.

[GL06] Dani Gamerman and Hedibert F Lopes. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. CRC Press, 2006.

[GRG94] Walter R Gilks, Gareth O Roberts, and Edward I George. "Adaptive direction sampling". In: *The statistician* (1994), pp. 179–189.

[Jac09] Simon Jackman. *Bayesian analysis for the social sciences*. Vol. 846. John Wiley & Sons, 2009.

[JWE92] Richard Arnold Johnson, Dean W Wichern, and Pearson Education. *Applied multivariate statistical analysis*. Vol. 4. Prentice hall Englewood Cliffs, NJ, 1992.

[Met+53] Nicholas Metropolis et al. "Equation of state calculations by fast computing machines". In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[Nea11] Radford Neal. "MCMC using Hamiltonian dynamics". In: *Handbook of Markov Chain Monte Carlo* 2 (2011).

[RC+10] Christian P Robert, George Casella, et al. *Introducing Monte Carlo Methods with R*. Vol. 18. Springer, 2010.

[RC99] Christian P Robert and George Casella. *Monte Carlo statistical methods*. Springer, 1999.

[WN99] SJ Wright and J Nocedal. *Numerical optimization*. Vol. 2. Springer New York, 1999.

[Wood06] Simon Wood. *Generalized additive models: an introduction with R*. CRC press, 2006.

[Wäs12] Torbjörn Wästerlid. "Application of L-BFGS to a Large-Scale Poisson MAP Estimation". MA thesis. Chalmers University of Technology, 2012.

[ZS11] Yichuan Zhang and Charles A. Sutton. "Quasi-Newton Methods for Markov Chain Monte Carlo". In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor et al. Curran Associates, Inc., 2011, pp. 2393–2401. URL: http://papers.nips.cc/paper/4464-quasi-newton-methods-for-markov-chain-monte-carlo.pdf.