Università degli studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Magistrale in

Scienze Statistiche

Relazione Finale

Statistica Bayesiana e Big Data

Relatore Prof. Bruno Scarpa
Dipartimento di Scienze Statistiche

Laureando Edoardo Vignotto
Matricola N 1134581

Anno Accademico 2016/2017

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Oggigiorno accade sempre più di frequente di avere a che fare con enormi quantità di dati quando si vuole analizzare uno specifico fenomeno di interesse. Ciò è possibile soprattutto grazie alla presenza di tecnologie che permettono la raccolta di dati sui più svariati problemi in modo semplice, automatizzato e poco dispendioso. Certamente, tale abbondanza di informazione, se sfruttata in modo opportuno, può essere molto positiva al fine di ottenere risultati affidabili sul fenomeno che si vuole studiare. Allo stesso tempo però, l'avere a che fare con grandi quantità di dati, o con dati dalla struttura particolarmente complessa, porta con sé degli ovvi problemi di natura computazionale. Per questo, nelle situazioni in cui si ha a che fare con Big Data, può essere necessario adottare opportuni accorgimenti per rendere possibile lo studio inferenziale del fenomeno studiato. L'obiettivo di questa tesi è quello di studiare quali accorgimenti di questo tipo è necessario applicare ai metodi che compongono la statistica bayesiana quando li si applica a Big Data.

L'inferenza bayesiana è una metodologia il cui uso si è molto intensificato negli ultimi anni, grazie, anche, allo sviluppo di algoritmi, soprattutto di carattere simulativo, che rendono possibile utilizzarla in modo semplice nei contesti più diversi. Tali algoritmi soffrono spesso, però, di problemi di natura computazionale quando applicati a una grande mole di dati. Per questo motivo, al fine di rendere agevole l'inferenza bayesiana in contesti di Big Data, è necessario apportare a essi opportune modifiche, che verranno trattate nel corso di questa tesi.

La presente tesi è strutturata nel seguente modo. Nel Capitolo 1 si descriverà il contesto in cui ci si muoverà, spiegando cosa si intende per Big Data

ed enunciando le componenti principali che formano il processo inferenziale in ambito bayesiano. Nel Capitolo 2 verranno, invece, descritti alcuni degli algoritmi che permettono l'utilizzo della statistica bayesiana, con particolare attenzione agli algoritmi MCMC. Si spiegherà, inoltre, perché tali algoritmi non sono direttamente applicabili in presenza di Big Data. Nei Capitoli 3 e 4 verranno poi trattati alcuni dei possibili accorgimenti che permettono di usare tali tecniche anche in questo contesto. Infine, nel Capitolo 5, verranno effettuati alcuni studi di simulazione e un'applicazione a dati reali per testare empiricamente il comportamento del Divide et Impera, descritto al Capitolo 4, all'interno di modelli di regressione, ovvero dell'approccio più completo, tra quelli studiati, per rendere agevole l'utilizzo dell'inferenza bayesiana in un contesto di Big Data.

Capitolo 1

Big Data e inferenza bayesiana

1.1 Cosa si intende per Big Data

Nell'era tecnologica in cui viviamo capita sempre più di frequente di avere a che fare con enormi quantità di dati per i più svariati problemi. Ciò è dovuto principalmente alla facilità con cui essi possono essere raccolti in modo rapido, veloce e spesso automatizzato (Azzalini e Scarpa (2012)). Un esempio, a tal proposito, può essere quello dell'analisi dei dati provenienti dai *social network* dove, con l'obiettivo di intercettare le opinioni e le tendenze degli utenti su un particolare argomento, si può avere a disposizione, almeno virtualmente, tutto ciò che è stato pubblicato nella rete internet da ognuno di essi e a un costo quasi nullo. L'analisi dei testi per comprendere opinioni e sentimenti è il campo in cui si muove la *sentimental analysis*. Un altro esempio, sempre più rilevante, è quello della bioinformatica. In questo senso, si pensi, ad esempio, ai casi nei quali si vogliono studiare le possibili relazioni tra il corredo genetico proprio di ogni individuo e la probabilità di insorgenza di una data malattia e, per fare ciò, si usino come unità statistiche, sequenze di DNA prese da diverse persone di cui è noto lo stato di salute. La dimensionalità dei dati a disposizione, operando in questo modo, è chiaramente molto elevata, in quanto ogni sequenza prelevata può essere lunga anche miliardi di basi, ovvero le unità che, concatenate, compongono il DNA.

Ovviamente, volendo applicare metodi statistici in questo contesto, i primi problemi che incorrono sono di natura computazionale, oltre che metodologica. Prima di occuparsi di ciò, occorre però definire in modo preciso cosa si intende per Big Data. In essi è presente almeno uno dei seguenti tre aspetti:

- un'alta numerosità campionaria n
- un alto numero di variabili p in gioco
- un'alta complessità interna.

I primi due aspetti sono probabilmente quelli in cui si incorre più spesso. Per esempio, all'interno della *sentimental analysis*, è comune dover operare sia con un alto numero di variabili (dell'ordine del numero di parole differenti presenti nei testi che si vanno a confrontare), che con un'alta numerosità campionaria (data dal numero di testi analizzati). Le stesse problematiche si riscontrano anche in ambito bioinformatico. Qui, inoltre, spesso il numero di variabili rilevate può essere molto maggiore del numero di soggetti presi in esame, ovvero si può facilmente avere $p \gg n$. Questo avviene perché su ogni soggetto vengono rilevate un altissimo numero di variabili e non è possibile replicare tale rilevazione su un numero altrettanto elevato di individui per ragioni economiche o di altro tipo. Il terzo caso, strettamente legato ai precedenti, racchiude tutte le casistiche in cui si devono analizzare dati caratterizzati da una struttura particolarmente complessa e legati tra loro da relazioni eterogenee e articolate. Dati di questo tipo richiedono, per essere trattati, modelli di rappresentazione altrettanto complessi, come possono essere, ad esempio, quelli per dati organizzati su reti, che impongono soluzioni metodologiche specifiche.

I problemi computazionali che emergono da tutto ciò riguardano principalmente due aspetti (Sagiroglu e Sinanc (2013)):

- la dimensione in memoria. I dati possono essere in quantità tale da non poter essere contenuti nella memoria di in un solo calcolatore
- il tempo di esecuzione. I metodi standard possono essere troppo lenti per essere applicati concretamente in questo contesto.

Un'altra possibile definizione di Big Data è, infatti, quella di dati la cui analisi su un unico calcolatore è intrattabile per una, o entrambe, delle questioni appena citate. In questa tesi verranno presentate alcune tecniche recenti che si occupano, in ambito bayesiano, sia delle problematiche legate alla dimensione in memoria che a quelle dovute al tempo di esecuzione e altre che, invece, si occupano solo di ridurre il tempo di esecuzione. In alcuni casi tali tecniche riescono a garantire l'esattezza dei risultati proposti, mentre in altri casi ricorrono ad opportune approssimazioni.

1.2 L'inferenza bayesiana

L'inferenza bayesiana è un approccio all'analisi dei dati che, negli ultimi anni, ha acquistato un peso sempre maggiore come alternativa al classico approccio frequentista (Damien et al. (2013)). L'approccio frequentista afferma che il parametro di interesse, che regola il meccanismo generatore dei dati che descrive il fenomeno oggetto di studio, è una costante non osservabile. Il processo inferenziale e la quantificazione dell'errore a esso legato è svolto mediante il principio del campionamento ripetuto, il quale sottintende che l'unica incertezza all'interno del modello usato sia quella relativa ai dati osservati. Per la metodologia bayesiana, invece, l'incertezza relativa all'inferenza viene codificata direttamente descrivendo il parametro di interesse e le conoscenze che si hanno su di esso con una distribuzione di probabilità, che viene poi aggiornata in base ai dati osservati.

Si denoti con y il campione delle osservazioni a disposizione di numerosità n, con θ il parametro di interesse, con $\pi(\theta)$ una qualsiasi funzione di densità, detta distribuzione *a priori*, che racchiude le conoscenze disponibili su θ prima di considerare i dati e con $L(y|\theta)$ l'usuale funzione di verosimiglianza, che rappresenta l'informazione portata da essi. Il risultato fondamentale su cui si basa tutto il processo inferenziale da un punto di vista bayesiano è il teorema di Bayes, per il quale:

$$\pi(\theta|y) = \frac{\pi(\theta)L(y|\theta)}{\int_\Theta \pi(\theta)L(y|\theta)\,d\theta} = \frac{\gamma(\theta|y)}{\int_\Theta \pi(\theta)L(y|\theta)\,d\theta}.$$

La funzione di densità $\pi(\theta|y)$ è la cosiddetta distribuzione *a posteriori*, ovvero il risultato a cui si giunge dopo aver unito le due funzioni sopra ci-

tate e che è il centro dell'inferenza in questo contesto. Spesso non si lavora direttamente con $\pi(\theta|y)$, ma con la sua versione non normalizzata $\gamma(\theta|y)$, in quanto l'informazione portata dalle due formulazioni è la stessa. L'espressione al denominatore è una costante di normalizzazione nella maggior parte dei casi intrattabile a causa dell'alta dimensionalità dell'integrale coinvolto. Per questo, non è quasi mai possibile disporre di una forma chiusa per $\pi(\theta|y)$ e bisogna ricorrere a tecniche computazionali apposite per ricavarne un surrogato più o meno approssimato. Un caso particolare in cui è possibile disporre di una formula analitica per la distribuzione a posteriori è quello delle famiglie coniugate, in cui $\pi(\theta|y)$ ha la stessa forma della distribuzione a priori, scegliendo opportunamente $\pi(\theta)$ e $L(y|\theta)$. Ad esempio, operando con una distribuzione a priori Beta di parametri (a, b) e una verosimiglianza Binomiale di parametro p per un campione di osservazioni y di numerosità n, si ha che la distribuzione a posteriori $\pi(\theta|y)$ è ancora Beta di parametro $(a + x, b + n - x)$, con x numero di successi in y (Lee (2012)).

I vantaggi di un approccio basato sull'inferenza bayesiana sono molteplici. Tra gli altri (Smith (1997)):

- permette di utilizzare distribuzioni a priori per introdurre nelle analisi svolte ciò che si sa sul parametro di interesse prima di considerare i dati in y
- ogni aspetto relativo al parametro di interesse è qui probabilizzato in modo diretto
- modelli di questo tipo sono solitamente semplici da comprendere e formulare, almeno nel loro disegno generale, e si prestano ad essere aggiornati in modo sequenziale all'arrivo di nuovi dati
- anche da un punto di vista formale, la statistica bayesiana si è dimostrata adatta ad affrontare diversi tipi di problemi in vari ambiti, dal machine learning alla biologia.

Malgrado ciò, l'utilizzo dell'inferenza bayesiana si è intensificato solo di recente, soprattutto a causa delle difficoltà computazionali che si devono affrontare per ottenere la distribuzione a posteriori $\pi(\theta|y)$ per la maggior parte

dei modelli, ovvero il risultato stesso del processo inferenziale in questo contesto. Tali difficoltà facevano sì che nella maggior parte dei casi la modellazione in ambito bayesiano si basasse quasi esclusivamente sulle famiglie coniugate, costringendo a eccessive semplificazioni. A partire dagli anni novanta sono state, però, sviluppate tecniche apposite, di carattere simulativo, per ottenere la distribuzione a posteriori a partire da qualsiasi distribuzione a priori e verosimiglianza. Ciò è stato possibile soprattutto grazie alla maggior potenza di calcolo raggiunta dai moderni calcolatori. Tuttavia, anche considerando la potenza dei computer odierni, tali tecniche si sono spesso rivelate computazionalmente troppo onerose se usate in un contesto di Big data. Nei prossimi Capitoli si mostreranno, quindi, alcune delle tecniche esistenti per ottenere l'oggetto di interesse $\pi(\theta|y)$ e si spiegheranno, inoltre, i motivi per cui la loro applicazione diretta a grandi masse di dati è proibitiva e quali modifiche sono necessarie in questi casi.

Capitolo 2

Metodi per ottenere la a posteriori

Come già accennato nel Capitolo precedente, esistono tecniche computazionali specifiche per ottenere l'oggetto di interesse $\pi(\theta|y)$ in ambito bayesiano a partire dalla distribuzione a priori $\pi(\theta)$ e dalla verosimiglianza $L(y|\theta)$ costituenti il modello usato. Nelle prossime Sezioni si descriveranno due metodologie di questo tipo, molto usate all'interno dell'inferenza bayesiana.

2.1 Algoritmi MCMC

Gli algoritmi Markov Chain Monte Carlo sono probabilmente il metodo oggi più utilizzato per ottenere la distribuzione a posteriori (Brooks et al. 2011). La logica sottostante a essi è quella di costruire una catena markoviana di primo ordine (ovvero una successione di variabili casuali $X^{(0)}, X^{(1)}, X^{(2)}, \ldots$ in cui $X^{(t+1)}$ è indipendente da $X^{(0)}, \ldots, X^{(t-1)}$ condizionatamente a $X^{(t)}$) con distribuzione limite $\pi(\theta|y)$. Agendo in questo modo, la simulazione della catena stessa fornisce un algoritmo per generare valori dalla distribuzione a posteriori. Ciò avviene perché la distribuzione di $X^{(N)}$, con N grande, è approssimativamente $\pi(\theta|y)$, ovvero la distribuzione limite della catena, e quindi la serie $x^{(N+1)}, x^{(N+2)}, \ldots$ può essere considerata come generata dalla distribuzione a posteriori. A tal proposito, vengono fatte le seguenti osservazioni (Robert 2004):

- la convergenza della catena alla distribuzione limite $\pi(\theta|y)$ può avvenire più o meno lentamente ed è solo asintotica. In questo senso, se $N+1$ può essere considerato il primo istante in cui tale convergenza è stata raggiunta, i valori generati $x^{(0)}, \ldots, x^{(N)}$ dovrebbero essere scartati da quelli usati nelle analisi successive. Il periodo in cui la catena converge è detto di *burn-in* o *warm up*

- determinare N non è sempre semplice

- i valori generati dalla catena markoviana sono tra loro dipendenti e, affinché l'approssimazione di $\pi(\theta|y)$ sia buona, si vorrebbe che tale dipendenza fosse il meno forte possibile. Una soluzione usata è quella di "filtrare" la serie, tenendo solo un sottoinsieme di x_i equispaziati. All'aumentare della loro correlazione, inoltre, aumenta anche il numero di generazioni richieste per mantenere un certo livello prefissato di accuratezza.

L'obiettivo è quindi quello di costruire una catena di Markov la cui simulazione sia semplice e che abbia $\pi(\theta|y)$ come distribuzione limite. I valori così generati saranno usati come sostituti della rappresentazione analitica della distribuzione a posteriori nelle analisi successive. Un metodo semplice e molto generale per ottenere ciò è l'algoritmo Metropolis-Hastings (Chib e Greenberg 1995). Tale algoritmo, di tipo sequenziale, genera, a ogni passo, un valore proposto da una distribuzione, detta *distribuzione proposta*, da cui sia semplice simulare e accetta, o rifiuta, tale valore con probabilità scelte in modo che la distribuzione limite sia quella voluta. Nel contesto dell'inferenza bayesiana, l'algoritmo Metropolis-Hastings può essere descritto come segue (Brooks et al. 2011):

1. si ponga $t = 0$ e si scelga un valore iniziale $x^{(0)}$

2. si simuli $x^{(*)}$ da una funzione di densità $q(\cdot|x^{(t)})$

3. si calcoli
$$\alpha = \min(1, \frac{\gamma(x^{(*)}|y)q(x^{(t)}|x^{(*)})}{\gamma(x^{(t)}|y)q(x^{(*)}|x^{(t)})}),$$
con $\gamma(x|y) = \pi(x)L(y|x)$

4. si definisca
$$x^{(t+1)} = \begin{cases} x^{(*)} & \text{con probabilità } \alpha \\ x^t & \text{con probabilità } 1-\alpha \end{cases}$$

5. si ponga $t = t + 1$ e si torni al passo 2.

I valori cosi generati definiscono una catena markoviana con distribuzione limite $\pi(\theta|y)$ e possono essere quindi usati per approssimare la distribuzione a posteriori stessa. Inoltre, si vede facilmente che l'inserimento al passo 3 della costante di normalizzazione $\int_\Theta \pi(\theta)L(y|\theta)\,d\theta$ non giocherebbe alcun ruolo e che, quindi, l'algoritmo può lavorare con qualsiasi funzione proporzionale a $\pi(\theta|y)$.

Per quanto riguarda l'implementazione dell'algoritmo Metropolis-Hastings, le componenti $x^{(0)}$ e $q(\cdot|\cdot)$ devono essere specificate. Per quanto riguarda la scelta del valore iniziale $x^{(0)}$ da cui far partire la catena, essa non influenza in modo particolare l'efficacia dell'algoritmo. L'unico requisito fondamentale è che $x^{(0)}$ appartenga al supporto di $\pi(\theta|y)$. Per quanto riguarda, invece, la distribuzione strumentale $q(\cdot|\cdot)$, che ha il compito di proporre i valori che verranno poi accettati, o rifiutati, secondo la condizione specificata al punto 3, la sua scelta è di fondamentale importanza per l'efficienza dell'algoritmo. A tal proposito è infatti desiderabile che $q(\cdot|\cdot)$ sia di facile simulazione e garantisca un buon rimescolamento della catena, nel senso che il tempo per raggiungere la distribuzione limite sia basso. Infatti, seppur è garantito che tale distribuzione limite sia la voluta $\pi(\theta|y)$, una scelta non adeguata di $q(\cdot|\cdot)$ potrebbe causare una convergenza troppo lenta a fini pratici.

Una scelta comune, in questo senso, è quella di generare $x^{(*)}$ da una distribuzione Uniforme con supporto $[x_t - \varepsilon, x_t + \varepsilon]$, per un dato ε, oppure da una qualche $q(x^{(*)}) = q(x^{(*)}|x^{(t)})$. In questo ultimo caso, l'algoritmo prende il nome di *independence sampler*. Nel caso di un algoritmo MCMC a passeggiata casuale, com'è il caso dato da $q(x^{(*)}|x^{(t)}) \sim U[x_t - \varepsilon, x_t + \varepsilon]$, con $U[a, b]$ distribuzione Uniforme con supporto $[a, b]$, una regola empirica è di scegliere il parametro ε in modo che il tasso di accettazione dei valori proposti sia compreso tra il 20% e il 50% (Robert 2004). Un requisito necessario per la correttezza dell'algoritmo è, invece, che il supporto della distribuzione strumentale contenga quello della distribuzione a posteriori, ovvero che non

ci siano valori ai quali $q(\cdot|\cdot)$ assegni densità nulla, mentre $\pi(\theta|y)$ assegni densità positiva.

L'algoritmo Metropolis-Hastings può essere facilmente generalizzato al caso multivariato. Un modo per fare ciò è il seguente:

1. si ponga $t = 0$ e si scelga un valore iniziale $x^{(0)}$

2. si ponga $x^{(t+1)} = x^{(t)}$

3. per $j = 1, \ldots, d$:

   (a) si simuli x_j^* da una distribuzione strumentale unidimensionale $q_j(\cdot|x^{(t+1)})$ e si ponga $x^* = x^{(t)}$ con $x_j^{(t)}$ sostituito da x_j^*

   (b) si calcoli $\alpha_j = \min(1, \frac{\gamma(x^*|y)q_j(x^{(t+1)}|x_j^{(*)})}{\gamma(x^{(t+1)}|y)q_j(x_j^*|x^{(t+1)})})$

   (c) con probabilità α_j si definisca $x_j^{(t+1)} = x_j^*$, altrimenti $x_j^{(t+1)} = x_j^{(t)}$

4. si ponga $t = t + 1$ e si torni al passo 2,

dove d è la dimensionalità di $x^{(t)}$ (Brooks et al. 2011). Come si può notare, qui le componenti di $x^{(t)}$ vengono aggiornate singolarmente in modo sequenziale. Tale algoritmo viene detto *ibrido*, in quanto ogni distribuzione strumentale $q_j(\cdot|\cdot)$ può essere di tipo diverso. Spesso, inoltre, si ha che $q_j(\cdot|x) = q_j(\cdot|x_j)$, ovvero che il valore proposto per la componente x_j dipende solamente dal valore attuale di tale componente. Un caso particolare di scelta della distribuzione ausiliaria per le singole componenti x_j^* è quella di usare la distribuzione a posteriori condizionata alle altre componenti di $x^{(t+1)}$. In questo caso l'algoritmo è detto *Gibbs sampler* e le proposte vengono sempre accettate: ciò è possibile solo quando tale distribuzione condizionata è facilmente ottenibile, ma migliora di molto l'efficienza dell'algoritmo. Si sottolinea, infine, che il risultato fornito da un algoritmo MCMC è esatto per $N \to \infty$, in quanto l'unico errore presente nell'approssimazione fornita per la distribuzione a posteriori risiede nel tempo di convergenza della catena markoviana costruita.

2.1.1 Problematiche degli algoritmi MCMC in un contesto di Big Data

Gli algoritmi MCMC costituiscono uno strumento molto potente e versatile per ottenere la distribuzione a posteriori in ambito bayesiano, in quanto applicabili in modo semplice e diretto alla maggior parte dei modelli. Negli anni ne sono state sviluppate molte varianti in base allo specifico campo di applicazione, che ne migliorano ulteriormente le performance. Inoltre, l'approssimazione di $\pi(\theta|y)$ da essi fornita è asintoticamente esatta rispetto al numero di iterazioni effettuate, seppur non analitica. Soffrono, però, di alcune problematiche computazionali specifiche quando vengono applicati in un contesto di Big Data, dovute alla loro natura sequenziale e alle operazioni richieste a ogni iterazione. Il più grosso limite è dovuto al fatto che essi devono calcolare la funzione di verosimiglianza $L(y|\theta)$ a ogni passo della catena. In quanto essa è una funzione globale di tutti le osservazioni presenti nel modello, ciò causa difficoltà principalmente per due motivi:

- valutare la verosimiglianza quando la mole di dati è molto grande è, spesso, particolarmente oneroso
- i dati possono essere in quantità tale da non poter essere contenuti in un solo calcolatore e quindi valutare la verosimiglianza nella sua interezza può essere problematico.

Come si può notare, essi soffrono quindi dei due tipici problemi che caratterizzano l'analisi dei dati in presenza di Big Data, ovvero quello riguardante il tempo computazionale necessario e quello riguardante la quantità di spazio di archiviazione necessario. Occorrerà, quindi, prendere adeguati accorgimenti per rendere utilizzabile l'inferenza bayesiana in queste condizioni. Alcune strategie possibili in questo senso, che prevedono opportune modifiche degli algoritmi MCMC, verranno presentate nei prossimi Capitoli.

2.2 Variational Bayes

Le procedure che rientrano all'interno del Variational Bayes sono un'alternativa agli algoritmi MCMC per ottenere la distribuzione a posteriori in

ambito bayesiano. Si tratta di algoritmi deterministici che si pongono l'obbiettivo di approssimare $\pi(\theta|y)$ con un'altra distribuzione $q(\theta)$, scelta opportunamente in modo da essere il più simile possibile a quella di interesse. Nello specifico, vengono fatte alcune assunzione sulla forma di $q(\theta)$ e, all'interno della famiglia di distribuzioni aventi tali caratteristiche, viene scelta quella che minimizza la distanza di Kullback–Leibler con $\pi(\theta|y)$ (Bishop 2006). Si sottolinea, fin da subito, che contrariamente a quanto avviene con gli algoritmi MCMC, dove l'approssimazione della distribuzione a posteriori può essere considerata esatta per $N \to \infty$, il risultato ottenuto in questo caso non può considerarsi esatto.

Il problema che si vuole risolvere è quindi quello di trovare la distribuzione $q(\theta)$ che minimizza la seguente espressione

$$KL(q(\theta), \pi(\theta|y)) = KL(q, \pi) = -\int q(\theta) \log(\frac{\pi(\theta|y)}{q(\theta)}) d\theta.$$

Ovviamente, senza porre nessuna condizione su $q(\theta)$, il minimo lo si ha quando $q(\theta) = \pi(\theta|y)$, ovvero proprio l'oggetto di interesse di cui si sarebbe voluto semplificare la ricerca. La potenza del Variational Bayes rientra, però, nel fatto che è possibile introdurre assunzioni sulla forma di $q(\theta)$, con l'obiettivo di semplificare la risoluzione del problema di minimizzazione precedente. In questo modo, si può ottenere una rappresentazione, seppur approssimata, della distribuzione a posteriori $\pi(\theta|y)$, che non è direttamente trattabile.

L'assunzione più spesso fatta in questo senso è quella di indipendenza tra le componenti (o sottoinsiemi delle stesse) del parametro θ, senza fare ulteriori assunzioni sulla forma parametrica della distribuzione approssimante $q(\theta)$ (Grimmer 2010). In questo modo, il parametro θ che indicizza la distribuzione a posteriori viene fattorizzato in K blocchi, nel senso che $\theta = (\theta_1, \theta_2, \ldots, \theta_K)$ e, per quanto riguarda $q(\theta)$, si ha che

$$q(\theta) = \prod_{k=1}^{K} q_k(\theta_k).$$

Si noti che a ogni sottoinsieme di componenti del parametro è associata una sua propria distribuzione $q_k(\cdot)$. Tali distribuzioni possono avere anche forme differenti, che andranno a loro volta stimate.

Per trovare la distribuzione approssimante $q(\theta)$, sottoposta alle condizioni appena enunciate, che minimizza la distanza di Kullback–Leibler $KL(q(\theta), \pi(\theta|y))$ tra essa e la distribuzione a posteriori $\pi(\theta|y)$ si può usare un algoritmo di tipo iterativo (Bishop 2006). Si supponga di avere a disposizione una stima provvisoria di $q(\theta)$ nella forma

$$q_1(\theta_1)^{old}, q_2(\theta_2)^{old}, \ldots, q_K(\theta_K)^{old}$$

e che si voglia aggiornare il $k-$esimo fattore. Per fare ciò, si definisce

$$\mathbb{E}_{j\neq k}[\log \pi(\theta, y)] = \int \prod_{j\neq k} \log \pi(\theta, y) q_j(\theta_j)^{old} d\theta_j,$$

ovvero il logaritmo della distribuzione a posteriori mediato tramite la stima corrente di quella approssimante. Usando tale espressione, si aggiorna $q_k(\theta_k)^{old}$ a $q(\theta_k)^{new}$ ponendo $q_k(\theta_k)^{new} = \frac{\exp(\mathbb{E}_{j\neq k}[\log \pi(\theta,y)])}{\int \exp(\mathbb{E}_{j\neq k}[\log \pi(\theta,y)])d\theta_k}$. A ogni iterazione dell'algoritmo ciò viene fatto per tutte le componenti di $q(\theta)$, usando la sua stima attuale. Così facendo, a ogni iterazione si aggiornano tutte le $q_k(\theta_k)$ nel seguente modo

$$\begin{aligned}
q_1(\theta_1)^{new} &= \frac{\exp(\mathbb{E}_{j\neq 1}[\log \pi(\theta, y)])}{\int \exp(\mathbb{E}_{j\neq 1}[\log \pi(\theta, y)])d\theta_1} \\
q_2(\theta_k)^{new} &= \frac{\exp(\mathbb{E}_{j\neq 2}[\log \pi(\theta, y)])}{\int \exp(\mathbb{E}_{j\neq 2}[\log \pi(\theta, y)])d\theta_2} \\
&\ldots \\
&\ldots \\
&\ldots \\
q_K(\theta_K)^{new} &= \frac{\exp(\mathbb{E}_{j\neq K}[\log \pi(\theta, y)])}{\int \exp(\mathbb{E}_{j\neq K}[\log \pi(\theta, y)])d\theta_K}.
\end{aligned}$$

L'algoritmo cosi definito offre un metodo semplice ed efficace per ottenere una buona approssimazione della distribuzione a posteriori $\pi(\theta|y)$ usando il Variational Bayes. É importante notare, inoltre, che le singole $q_k(\theta_k)$ possono essere anche molto distanti dalle rispettive distribuzioni marginali $\pi_k(\theta_k|y)$ e che tale approssimazione dovrebbe essere considerata sulla congiunta $q(\cdot)$ (Andrieu e Vihola (2011)).

Anche il Variational Bayes, usato in un contesto di Big Data, soffre degli stessi problemi computazionali descritti nella Sezione 1.1. Infatti, poiché esso

si riduce essenzialmente alla risoluzione di un problema di minimizzazione, tale processo può essere troppo lento, a fini pratici, se non si apportano opportune modifiche in presenza di una grande mole di dati. Inoltre, questo potrebbe essere reso ancor più difficoltoso qualora tali dati siano distribuiti su più calcolatori, in quanto non è immediato generalizzare l'algoritmo di minimizzazione descritto qualora i dati non risiedessero nella memoria di un solo calcolatore.

Capitolo 3

Tecniche di sotto-campionamento

In questo Capitolo e in quello seguente si descriveranno due diversi approcci per adattare le tecniche descritte al Capitolo 2 per ottenere $\pi(\theta|y)$ nei contesti in cui si dispone di una grande mole di dati, concentrandosi principalmente sugli algoritmi MCMC. Come già accennato nella Sezione 2.1.1, il più grosso problema che devono affrontare tali algoritmi in questo contesto è dovuto al fatto di dover valutare la verosimiglianza complessiva del modello a ogni iterazione. La prima tipologia di metodi che si va a presentare per affrontare tale limite quando si ha a che fare con Big Data prevede di usare a ogni iterazione solo una parte delle osservazioni a disposizione, composta da un campione casuale delle stesse con numerosità minore a quella di y. Verranno descritte due tecniche che rientrano in questo approccio. La prima sfrutta l'uso di uno stimatore di $L(y|\theta)$, basato su un sottoinsieme delle osservazioni, in luogo della verosimiglianza stessa. La seconda sfrutta, invece, un'opportuna strategia di *data augmentation*. Seppure in entrambi i casi è possibile mantenere l'esattezza teorica del risultato finale fornito, bisogna tener conto del fatto che usare i dati in modo parziale produce alcuni problemi di primaria importanza (Andrieu et al. (2015)), ovvero:

- la velocità di convergenza della catena markoviana è solitamente inferiore a quella che si otterrebbe usando sempre i dati in modo completo
- la varianza delle stime prodotte è solitamente più alta di quella che si avrebbe usando a ogni passo dell'algoritmo tutti i dati disponibili.

A causa di ciò, il loro utilizzo può risultare problematico in alcune situazioni, anche se, come si vedrà, esistono diversi accorgimenti che permettono di controllare tali aspetti. Inoltre, occorre sottolineare che con questo tipo di approccio si risolve solo uno dei due problemi enunciati nel Capitolo 1, ovvero quello legato al tempo di esecuzione. Se, infatti, diventa possibile, da un punto di vista computazionale, valutare in modo rapido la verosimiglianza a ogni iterazione dell'algoritmo MCMC usato, in quanto ciò avviene solo su una parte delle osservazioni, queste metodologie non si prestano a essere usate quando i dati non stanno all'interno della memoria di un unico calcolatore, in quanto a ogni passo è richiesto un loro campionamento casuale.

3.1 Pseudo-Marginal Metropolis-Hastings

In questa sottosezione si presenterà il Pseudo-Marginal Metropolis-Hastings applicato ai Big Data (Andrieu et al. (2009)), che prevede di usare uno stimatore di $L(y|\theta)$ in luogo della verosimiglianza stessa a ogni iterazione dell'algoritmo MCMC usato. Il primo approccio descritto è esatto, nel senso che la distribuzione limite della catena fornita dall'algoritmo MCMC è esattamente $\pi(\theta|y)$, mentre il secondo è approssimato. Come già esposto, entrambi gli approcci si basano sulla sostituzione, all'interno dell'algoritmo MCMC usato, della verosimiglianza calcolata su tutte le osservazioni con un suo stimatore basato su un sottoinsieme delle stesse. A seconda del fatto che tale stimatore sia o meno non distorto, si mantiene, o meno, la correttezza teorica del risultato fornito dall'algoritmo MCMC.

3.1.1 Un approccio esatto

L'idea che sta alla base di questo approccio è quella di non utilizzare, a ogni iterazione dell'algoritmo Metropolis-Hastings, la verosimiglianza calcolata su tutte le osservazioni disponibili, ma un suo stimatore non negativo e non distorto basato su un sottoinsieme delle stesse. Qui è interessante notare come, per fare inferenza in ambito bayesiano, si usi un concetto puramente frequentista, ovvero quello di non distorsione. In questo caso viene naturale comportarsi in tal modo in quanto il valore di $L(y|\theta)$, per un dato θ, è una

costante fissa, che si va ad approssimare usando un sottoinsieme di y. Appare chiaro come il paradigma frequentista sia perfettamente adatto a questa situazione, in quanto un "vero valore" per la quantità di interesse esiste e non è casuale, ma non è calcolabile in modo esatto in quanto non si può disporre (per motivi computazionali) dell'intera popolazione (ovvero di tutto y).

Più in generale, si vuole quindi trovare uno stimatore non distorto e non negativo $\hat{\gamma}(\theta|y)$, per ogni $\theta \in \Theta$, di $\gamma(\theta|y)$, ovvero della distribuzione a posteriori non normalizzata, costruito usando solo una parte dei dati a disposizione. Tale stimatore sarà poi usato all'interno degli algoritmi Metropolis-Hastings, descritti nella Sezione 2.1, al posto di $\gamma(\theta|y)$. La distribuzione limite della catena costruita in questo modo è ancora $\pi(\theta|y)$, ovvero la distribuzione a posteriori di interesse. Come già accennato in precedenza, però, nonostante si mantenga la correttezza asintotica dei risultati ottenuti, si presentano al contempo alcuni svantaggi. Questo è dato in particolare dal fatto che lo stimatore $\hat{\gamma}(\theta|y)$ ha spesso una varianza elevata e ciò si traduce in probabilità α di accettazione del valore proposto a ogni passo dell'algoritmo anche molto lontane da quelle che si avrebbero usando un algoritmo Metropolis-Hastings classico. Per queste ragioni, è di fondamentale importanza tenere sotto controllo la varianza dello stimatore $\hat{\gamma}(\theta|y)$. Alcune strategie per fare ciò sono presenti in Quiroz et al. (2016).

Viene ora descritto come ottenere lo stimatore $\hat{\gamma}(\theta|y)$, ovvero uno stimatore non distorto e non negativo di $\pi(\theta)L(y|\theta)$ per ogni θ, della forma $\hat{\gamma}(\theta|y) = \pi(\theta)\hat{L}(y|\theta)$. Per prima cosa, è importante notare che uno stimatore non distorto e non negativo della log-verosimiglianza $l(y|\theta) = \log L(y|\theta)$ è

$$\hat{l}(y|\theta) = \frac{n}{t}\sum_{i=1}^{t} \log L(y_i^*|\theta),$$

con $y_1^*, \ldots, y_t^*$ osservazioni estratte casualmente con reinserimento dalle n osservazioni presenti in y e $L(y_i^*|\theta)$ funzione di verosimiglianza per la singola osservazione y_i^*. Purtroppo, $\exp[\hat{l}(y|\theta)]$ è, per il teorema di Jensen, uno stimatore distorto di $L(y|\theta)$ e non può essere quindi qui utilizzato. È però utile chiedersi se sia possibile costruire $\hat{\gamma}(\theta|y)$ basandosi su $\hat{l}(y|\theta)$. Ciò è fattibile se è presente una funzione $a(\theta)$ tale che $a(\theta) < \log p(y_i^*|\theta) = l_i(\theta)$ per ogni $i = 1, \ldots, n$, ovvero un *lower-bound* per la log-verosimiglianza, mentre non è

possibile in assenza di assunzioni aggiuntive (Jacob et al. (2015)). Introducendo la funzione $a(\theta)$, si arriva allo stimatore $\hat{\gamma}(\theta|y)$ di $\gamma(\theta|y)$ descritto nella seguente equazione:

$$\begin{aligned}\hat{\gamma}(\theta|y) =& \pi(\theta)\hat{L}(y|\theta) \\ =& \pi(\theta)\exp[na(\theta)](1+\sum_{k=1}^{G}\frac{1}{Pr(N\geq k)k!}\prod_{j=1}^{k}D_j^*)\end{aligned}$$

con G estratto casualmente da una variabile geometrica di parametro fissato e

$$D_j^* = \frac{n}{t}\sum_{i=1}^{t}\log L(y_{i,j}^*|\theta) - na(\theta),$$

dove le $y_{i,j}^*$ sono estratte casualmente sia rispetto a i, che rispetto a j, con reinserimento da y. Lo stimatore $\hat{\gamma}(\theta|y)$ ottenuto è uno stimatore non distorto e non negativo di $\gamma(\theta|y)$ e può essere usato al suo posto all'interno di un normale algoritmo Metropolis-Hastings. Una possibile alternativa a $\hat{\gamma}(\theta|y)$ è quella proposta di seguito, che gode delle medesime proprietà:

$$\hat{\gamma}_{Poiss}(\theta|y) = \exp[\lambda + na(\theta)]\prod_{j=1}^{J}\frac{D_j^* - na(\theta)}{\lambda},$$

con J valore estratto da una distribuzione di Poisson di parametro λ (Fearnhead et al. (2010)). Anche $\hat{\gamma}_{Poiss}(\theta|y)$ può essere usato in sostituzione della distribuzione a posteriori non normalizzata all'interno di un qualsiasi algoritmo MCMC e gode della fondamentale proprietà di non distorsione.

I valori prodotti dalla catena markoviana avranno, in questo modo, ancora come distribuzione limite la distribuzione a posteriori $\pi(\theta|y)$ di interesse, ma a ogni passo dell'algoritmo è richiesto di valutare la verosimiglianza solo su un sottoinsieme dei dati a disposizioni, diminuendo quindi l'onere computazionale (Bardenet et al. (2015)). Si sottolinea, infine, che l'esattezza teorica del risultato ottenuto è data essenzialmente dalla non distorsione dello stimatore per $\gamma(\theta|y)$ usato. Nella prossima sottosezione si rilasserà questa assunzione, perdendo tale correttezza teorica. Così facendo, però, non sarà più necessario disporre del lower-bound $a(\theta)$, di cui è difficile disporre al di fuori di contesti specifici.

3.1.2 Un approccio approssimato

L'approccio appena descritto deve la sua esattezza essenzialmente al fatto di utilizzare uno stimatore non distorto di $L(y|\theta)$ in luogo della verosimiglianza stessa, che sarà poi moltiplicato per la distribuzione a posteriori $\pi(\theta)$ per ottenere un'approssimazione della distribuzione a posteriori non normalizzata $\gamma(\theta|y)$. Per fare ciò, però, occorre disporre di un lower-bound $a(\theta)$ tale che $a(\theta) < l_i(\theta)$ per ogni $i = 1, \dots, n$. Trovare tale lower-bound può essere problematico in applicazioni reali ed è quindi un grosso limite all'applicabilità di tale tecnica. Inoltre, un lower-bound $a(\theta)$ troppo piccolo può portare a una grande varianza per lo stimatore per $\gamma(\theta|y)$ (Quiroz et al. (2016)). Una possibile alternativa è quella di utilizzare uno stimatore la cui distorsione sia opportunamente corretta, seppur non nulla. Inoltre, come già sottolineato, è importante cercare di tenere sotto controllo la varianza dello stimatore usato. In questa sottosezione si descriverà, quindi, come ottenere tale stimatore e ridurre la sua varianza.

Il primo obbiettivo è quello di ottenere uno stimatore altamente efficiente $\hat{l}^*(y|\theta)$ della log-verosimiglianza $l(y|\theta) = \log \sum_{i=1}^{n} L(y|\theta)$, che sarà poi utilizzato per costruire uno stimatore $\hat{L}^*(y|\theta)$ per la verosimiglianza $L(y|\theta)$ con distorsione minore rispetto alla semplice trasformazione $\exp[\hat{l}^*(y|\theta)]$. Poiché alcune unità contribuiranno, verosimilmente, alla sommatoria che compone $l(y|\theta)$ in modo maggiore rispetto ad altre, usare uno stimatore basato sul campionamento casuale semplice delle osservazioni si traduce, nella maggior parte dei casi, in una varianza elevata. Un modo per risolvere questo problema è quello di introdurre delle variabili di controllo $q_i(\theta) \approx l_i(\theta)$ che rendano gli elementi che compongono la sommatoria che si vuole stimare circa equivalenti. Per semplicità, da qui in poi spesso la dipendenza da y delle quantità in gioco sarà omessa. In questo senso, si definiscono le differenze $d_i(\theta) = l_i(\theta) - q_i(\theta)$, per $i = 1, \dots, n$ e le quantità

$$\mu(\theta) = \frac{1}{n} \sum_{i=1}^{n} d_i(\theta)$$

$$\sigma^2(\theta) = \frac{1}{n} \sum_{i=1}^{n} [d_i(\theta) - \mu(\theta)]^2.$$

Uno stimatore efficiente della log-verosimiglianza può essere allora ottenuto costruendo uno stimatore basato sulle differenze (Sarndal et al. (2003)) del tipo

$$\hat{l}^*(\theta) = q(\theta) + n\hat{\mu}(\theta)$$
$$\hat{\mu}(\theta) = \frac{1}{m}\sum_{i=1}^{t} d_{u_i}(\theta)$$
$$q(\theta) = \sum_{i=1}^{n} q_i(\theta),$$

con u_i sequenza di indici estratti casualmente da $1, \ldots, n$. Per quanto riguarda le variabili di controllo $q_i(\theta)$ esse possono essere ricavate in modo efficiente attraverso delle espansioni di Taylor al secondo ordine dopo aver suddiviso i dati in gruppi mediante una qualche tecnica di *clustering* (Quiroz et al. (2017)).

Come già esposto nella sottosezione precedente, anche se $\hat{l}^*(\theta)$ è uno stimatore non distorto per la log-verosimiglianza $l(\theta)$, non è immediato ottenere una sua trasformazione che sia uno stimatore non distorto per $L(y|\theta)$. Senza introdurre lower-bound, come fatto per la variante esatta del Pseudo-Marginal Metropolis-Hastings, si può considerare come stimatore per la verosimiglianza $\exp \hat{l}^*(\theta)$ e correggerne la distorsione (Ceperley e Dewing (1999)), ottenendo

$$\hat{L}^*(\theta) = \exp[\hat{l}^*(\theta) - \frac{n^2}{2m}\hat{\sigma}^2(\theta)]$$
$$\hat{\sigma}^2(\theta) = \frac{1}{m}\sum_{i=1}^{m}[d_{u_i}(\theta) - \hat{\mu}(\theta)]^2.$$

Ora che si dispone di uno stimatore di $L(y|\theta)$ efficiente e la cui distorsione è stata opportunamente corretta, esso può essere utilizzato al posto della verosimiglianza stessa all'interno di un qualsiasi algoritmo MCMC. In questo modo, il risultato ottenuto non sarà più esatto come si ha, invece, con l'approccio precedente, ma l'impostazione rende in questo caso l'implementazione più generale.

3.2 Firefly Monte Carlo

Un'altra tecnica che unisce l'algoritmo Metropolis-Hastings al campionamento dei dati è quella del Firefly Monte Carlo, presentata in Maclaurin e Adams (2014). Come col Pseudo-Marginal Metropolis-Hastings, è richiesto di valutare la log-verosimiglianza solo su un sottoinsieme dei dati a disposizione a ogni iterazione. In questo caso, però, non è richiesta la costruzione di nessun stimatore, ma ci si affida, invece, a un'opportuna procedura di *data augmentation*. Si supponga, innanzitutto, di disporre di un insieme di funzioni $b_i(\theta)$ tali che $b_i(\theta) \leq l_i(\theta)$. Per semplicità nella notazione, è stata omessa la dipendenza da y_i. Si noti che, anche in questo caso, come per il Pseudo-Marginal Metropolis-Hastings, è richiesto di avere a disposizione dei lower-bound per la log-verosimiglianza. Nella sezione precedente, però, era richiesto di avere a disposizione un solo lower-bound $a(\theta)$, dominato da ogni singola $l_i(\theta)$, mentre in questo caso è presente una flessibilità maggiore, in quanto si può usare un diverso lower-bound $b_i(\theta)$ per ogni $l_i(\theta)$. Viene poi definita la seguente distribuzione $w(\theta, z)$, che estende quella di interesse $\pi(\theta|y)$ e ha come supporto $\Theta \times \{0,1\}^n$:

$$w(\theta, z) \propto \pi(\theta) \prod_{i=1}^{n} [\exp(l_i(\theta)) - \exp(b_i(\theta))]^{z_i} \exp(b_i(\theta))^{1-z_i} =$$
$$= \pi(\theta) \prod_{i=1}^{n} \exp(b_i(\theta)) \prod_{i=1}^{n} [\exp(l_i(\theta) - b_i(\theta)) - 1]^{z_i},$$

dove le z_i sono variabili casuali dicotomiche con distribuzione bernoulliana condizionatamente a θ. Tale distribuzione ha due importanti caratteristiche:

- ammette $\pi(\theta|y)$ come distribuzione marginale
- richiede, a ogni iterazione, di valutare la log-verosimiglianza $l_i(\theta)$ solo per quegli i tali per cui $z_i = 1$ quando si applica un algoritmo Metropolis-Hastings per generare valori (θ^*, z^*) da essa.

Bisogna però sottolineare che, allo stesso tempo, è richiesto di calcolare $\prod_{i=1}^{n} \exp(b_i(\theta))$ a ogni iterazione. Per questo motivo, i lower-bound $b_i(\theta)$ devono essere scelti in modo che tale calcolo sia semplice ed efficiente.

In pratica, a ogni iterazione, l'algoritmo seleziona in modo dinamico le osservazioni sulle quali calcolare la verosimiglianza, basandosi sugli indicatori casuali forniti dalle z_i (Angelino et al. (2016)). A ogni passo, vengono generati prima i nuovi valori per le z_i e poi il valore per θ. Per quanto riguarda il costo computazionale dato dal numero di osservazioni sulle quali occorre calcolare la log-verosimiglianza a ogni iterazione, esso può essere controllato in modo esplicito specificando il numero di cambiamenti (da $z_i = 1$ a $z_i = 0$, o viceversa) atteso, o implicitamente specificandone la frazione. In questo modo, applicando un qualsiasi algoritmo MCMC a $w(\theta, z)$, la catena formata dai valori generati per θ avrà come distribuzione limite esattamente $\pi(\theta|y)$. Nonostante tale esattezza, questo approccio soffre degli svantaggi menzionati nella parte introduttiva di questa sezione e la sua efficienza è strettamente legata a quanto i lower-bound per le singole osservazioni sono vicini alle rispettive $l_i(\theta)$. Poiché non è sempre possibile trovare facilmente funzioni $b_i(\theta)$ che soddisfino tale caratteristica e per le quali sia contemporaneamente veloce calcolare la quantità $\prod_{i=1}^{n} \exp(b_i(\theta))$, tale metodo non è immediato da generalizzare al di fuori di contesti specifici.

3.3 Commenti

In questo Capitolo sono state esposte diverse tecniche per adattare gli algoritmi MCMC in contesti di Big Data basate, essenzialmente, sull'uso di un campione delle osservazioni disponibili, piuttosto che su tutto y, a ogni iterazione. Questo viene fatto per ridurre l'onere computazionale dovuto alla valutazione della verosimiglianza a ogni passo dell'algoritmo MCMC usato, principale motivo per cui una loro applicazione diretta a grandi insiemi di dati è inagevole. Come già sottolineato, seppure tali metodologie possano garantire l'esattezza teorica dell'approssimazione per la distribuzione a posteriori $\pi(\theta|y)$ fornita, un loro impiego al di fuori di contesti specifici non è immediato. Necessitano, inoltre, di un'implementazione efficiente e attenta al fine di ottenere buoni risultati empirici.

Inoltre, poiché affrontano solamente le problematiche degli algoritmi MCMC dovute al tempo di esecuzione in presenza di una grande mole di dati, ma non quelle dovute a una possibile distribuzione dei dati sulla memoria di più

calcolatori, risolvono solo parzialmente i problemi esposti nei Capitoli 1 e 2. Nel prossimo Capitolo si esporrà, invece, un approccio più completo, che affronta sia la questione dell'onere computazionale, che quella dello spazio in memoria in presenza di Big Data.

Capitolo 4

Divide et impera

Il secondo approccio per adattare le tecniche descritte al Capitolo 2 a un utilizzo in presenza di una grande mole di dati è, per certi versi, più radicale di quello basato sul campionamento dei dati a ogni passo dell'algoritmo MCMC appena presentato. In questo ultimo caso, infatti, si può, per certi versi, ancora pensare che tutti i dati, nel loro insieme, siano usati a ogni iterazione dell'algoritmo. I vantaggi computazionali derivano dal fatto che, seppur nella fase di campionamento vengono prese in considerazione tutte le osservazioni, solo una parte delle stesse contribuisce alla valutazione della verosimiglianza.

Nel *Divide et Impera* applicato in questo contesto e di seguito descritto, invece, la visione è in un certo senso ribaltata, in quanto i dati vengono suddivisi casualmente in gruppi all'inizio dell'algoritmo e poi su ognuno di essi viene applicato un opportuno algoritmo MCMC che a ogni iterazione valuta la verosimiglianza per le osservazioni presenti in quel dato gruppo. Agire in questo modo presenta indubbi vantaggi, primo fra tutti il fatto che non è necessario che i dati risiedano nella memoria di un solo calcolatore. Inoltre, la convergenza degli algoritmi MCMC usati su ogni gruppo di osservazioni avviene, in genere, piuttosto velocemente, in quanto ogni algoritmo viene applicato su un insieme di dati dalla numerosità molto inferiore a quella originaria. La fase critica di questo approccio è, però, quella data dalla combinazione dei risultati ottenuti dagli algoritmi MCMC applicati ai diversi gruppi, che è fondamentale per la buona riuscita della sua applicazione.

Nel seguito ne verranno proposte due varianti, una molto semplice e com-

putazionalmente poco onerosa, ma che non garantisce l'esattezza teorica del risultato finale, l'altra che, al contrario, garantisce che, sotto opportune condizioni, il campione finale abbia come distribuzione limite $\pi(\theta|y)$, al costo, però, di un onere computazionale maggiore. Inoltre, nel prossimo Capitolo si confronteranno tali tecniche di combinazione attraverso alcuni studi di simulazione e un'applicazione a dati reali.

4.1 Algoritmo generale

Il metodo, qui presentato, per rendere utilizzabili le tecniche descritte al Capitolo 2 in un contesto di Big Data, si basa essenzialmente sul calcolo parallelo e affronta entrambe le problematiche esposte nel Capitolo 1. Nel seguito, si farà riferimento all'adattamento degli algoritmi MCMC ma, come sarà meglio precisato più avanti, la stessa impostazione si applica anche al Variational Bayes. L'idea che sta alla base di questo approccio è, come già accennato, quella dividere i dati a disposizione in sottoinsiemi disgiunti, ottenere diverse stime della distribuzione a posteriori da ognuno di essi con un qualsiasi algoritmo MCMC e, infine, combinare tali stime in una finale. Come appare chiaro fin da subito, una strategia di questo tipo si presta in modo semplice ad essere parallelizzata su più calcolatori, ognuno dei quali avrà il compito di effettuare le operazioni su un singolo sottoinsieme dei dati.

Un secondo aspetto importante da sottolineare è che, operando in questo modo, la comunicazione fra i diversi calcolatori è mantenuta al minimo, dal momento che essa è richiesta solo all'inizio della procedura, quando i dati vengono distribuiti tra i vari calcolatori, e nella parte finale della stessa, quando le stime parziali provenienti da ognuno di essi sono unite in quella finale. La parte computazionalmente più onerosa, ovvero quella relativa alle simulazioni Monte Carlo legate a ogni sottoinsieme di dati, è invece gestita da ogni macchina in modo completamente indipendente dalle altre. Questo è particolarmente importante in quanto in un ambiente di calcolo distribuito la comunicazione è uno degli elementi più dispendiosi ed è quindi importante tenerla sotto controllo. In questo senso, è buona norma, in un algoritmo pensato per il calcolo parallelo, che la maggior parte dell'onere computazionale sia dovuto alle operazioni effettuate da ogni macchina, piuttosto che a

quello relativo alla comunicazione tra le stesse, cosa che con questo approccio avviene (Scott et al. (2016)).

Si descrive ora nello specifico l'algoritmo alla base di questo metodo, cosi come proposto in Scott et al. (2016). Nel seguito, si indicherà con y l'insieme completo di tutti i dati, con y_s quelli che compongono l' $s-$esimo sottoinsieme degli stessi e con S il numero totale di tali sottoinsiemi. Così facendo la distribuzione a posteriori può essere riscritta nella seguente espressione:

$$\pi(\theta|y) = \prod_{s=1}^{S} L(y_s|\theta)\pi(\theta)^{1/S}. \tag{4.1}$$

L'unica assunzione necessaria affinché tale equazione sia corretta è che le osservazioni dei diversi gruppi siano condizionatamente indipendenti tra loro dati i parametri, mentre è ammessa qualsiasi tipo di dipendenza all'interno di ciascun sottoinsieme. Si noti, inoltre, che è stato necessario scomporre la distribuzione a priori $\pi(\theta)$ in S componenti pari a $\pi(\theta)^{1/S}$ in modo da preservare l'ammontare complessivo di informazione a priori. Tale scelta è arbitraria, dal momento che l'uso di una qualsiasi famiglia di funzioni $g_s(\cdot)$ per $s = 1, \ldots, S$, una per ogni sottoinsieme, tali che $\pi(\theta) = \prod_{s=1}^{S} g_s(\pi(\theta))$, porterebbe allo stesso risultato. Tuttavia, la scelta di porre $g_s(\pi(\theta)) = \pi(\theta)^{1/S}$ per ogni s appare la più naturale ed è stata quindi preferita. Dalla equazione 4.1 deriva in modo diretto la seguente procedura:

1. per prima cosa, i dati vengono suddivisi in S sottoinsiemi disgiunti
2. per ogni sottoinsieme si simula un campione di valori da $\pi(\theta|y_s) \propto L(y_s|\theta)\pi(\theta)^{1/S}$; per fare ciò si può utilizzare un qualsiasi algoritmo MCMC
3. i campioni provenienti dalle varie $\pi(\theta|y_s)$ sono combinati in un campione finale rappresentativo della distribuzione a posteriori $\pi(\theta|y)$ di interesse.

Per quanto riguarda il primo passo di tale algoritmo, in assenza di motivazioni aggiuntive, i dati vengono suddivisi tra gli S gruppi casualmente in modo che ogni gruppo abbia la stessa dimensionalità. Per quanto riguarda il secondo passo dell'algoritmo, esso è quello che può essere parallelizzato e

che rende quindi questo approccio computazionalmente adatto in un contesto di Big Data. Appare chiaro, inoltre, come questa tecnica si possa applicare anche ad insiemi di dati tanto vasti da non poter essere memorizzati in un solo calcolatore, in quanto ogni macchina necessita di disporre solo di una parte degli stessi. Il terzo passo dell'algoritmo, ovvero quello che riguarda la combinazione dei diversi campioni provenienti da ogni calcolatore, è quello più critico per quanto riguarda la bontà della stima per la distribuzione a posteriori $\pi(\theta|y)$ prodotta e verrà affrontato nello specifico nella prossima sezione, descrivendo due metodologie dalla filosofia molto differente.

In ultimo, si sottolinea, come già accentato, il fatto che l'approccio descritto in questo capitolo può essere applicato anche al Variational Bayes, piuttosto che agli algoritmi MCMC (si veda, a tal proposito, Rabinovich et al. (2015) e Tran et al. (2016)). Anche in questo caso, i dati vengono divisi in sottoinsiemi disgiunti e vengono stimate diverse distribuzioni a posteriori parziali da ognuno di essi, questa volta affidandosi al Variational Bayes, che poi vengono combinate per ottenere quella finale.

4.2 Metodi di combinazione

Vengono nel seguito presentate due diverse tecniche per effettuare la combinazione dei campioni generati dalle varie $\pi(\theta|y_s)$, che costituisce il passo 3 dell'algoritmo descritto nella precedente sezione.

La prima procedura, proposta in Scott et al. (2016), prevede di ottenere il campione finale rappresentativo della distribuzione a posteriori $\pi(\theta|y)$ di interesse come media pesata dei campioni intermedi ottenuti al passo 2 dell'algoritmo sopra proposto. In particolare, si consideri di disporre di S campioni θ^s per $s = 1, \ldots, S$, ognuno proveniente da $\pi(\theta|y_s)$ e composto dai valori $\theta_1^s, \ldots, \theta_T^s$, dove T è la numerosità dei valori simulati da ogni sottoinsieme. Si supponga, inoltre, di assegnare a ogni gruppo una matrice di pesi $W_s = \Sigma_s^{-1}$, con $\Sigma_s = Var(\theta|y_s)$. Le componenti del campione finale $\hat{\theta}$ sono quindi ottenute con la seguente formula:

$$\hat{\theta}_g = (\sum_s W_s)^{-1} \sum_s W_s \theta_g^s$$

per $g = 1, \ldots, T$.

Agendo in questo modo, le componenti $\hat{\theta}_1, \ldots, \hat{\theta}_T$ di $\hat{\theta}$ hanno distribuzione limite esatta $\pi(\theta|y)$ nel caso in cui la distribuzione di tutte le $\pi(\theta|y_s)$ sia gaussiana. Se non si è in questa situazione, però, non c'è alcuna garanzia teorica sulla distribuzione limite della catena costituita dai valori $\hat{\theta}_1, \ldots, \hat{\theta}_T$. Occorre sottolineare, però, che questa tecnica di combinazione ha fornito buoni risultati in studi di simulazioni anche in condizioni generali (Scott et al. (2016)). Inoltre, la distribuzione delle varie $\pi(\theta|y_s)$ tende ad essere gaussiana al crescere del numero di osservazioni disponibili (Le Cam e Yang (2012)) quindi, in molti casi riguardanti Big Data, è probabile che tale assunzione valga almeno in modo approssimato. Un ovvio vantaggio di questo approccio è il bassissimo costo computazionale richiesto dalla fase di combinazione, composta da semplici operazioni algebriche, praticamente ininfluente se confrontato con le altre fasi dell'algoritmo. La tecnica di combinazione appena descritta è, probabilmente, la più semplice tra quelle proposte e per questo è interessante confrontarla con quella che viene enunciata di seguito, che è computazionalmente molto più onerosa, ma è anche l'unica, tra le tecniche di combinazione finora proposte, che garantisce l'esattezza teorica del risultato finale fornito.

La seconda procedura per combinare i campioni provenienti dalle diverse $\pi(\theta|y_s)$ in quello finale descritta in questo capitolo si basa su un approccio semiparametrico ed è presentata in Neiswanger et al. (2013). Si noti che, da questo punto di vista, la tecnica basata sulla media pesata dei campioni intermedi θ^s provenienti dalle diverse $\pi(\theta|y_s)$, può essere considerata di tipo parametrico, in quanto esatta solo nel caso in cui le varie $\pi(\theta|y_s)$, e quindi anche $\pi(\theta|y)$, abbiano distribuzione gaussiana. Tale metodo è computazionalmente molto più oneroso del precedente, ma produce un campione finale la cui distribuzione converge alla vera distribuzione a posteriori per $T \to \infty$. In particolare, si consideri uno stimatore $\hat{p}_s$ semiparametrico di $p_s = \pi(\theta|y_s)$, che consiste nel prodotto di una componente parametrica $\hat{f}_s(\theta) \equiv \mathcal{N}_d(\theta|\hat{\mu}_s, \hat{\Sigma}_s)$ di p_s, dove $\hat{\mu}_s$ e $\hat{\Sigma}_s$ sono, rispettivamente, media e varianza dei valori simulati per θ nel sottoinsieme s, e di una componente non parametrica $\hat{r}(\theta)$ di tipo kernel della funzione di correzione $r(\theta) = p_s(\theta)/\hat{f}_s(\theta)$. In questo modo si ha

che

$$\begin{aligned}\hat{p}_s &= \hat{f}_s(\theta)\hat{r}(\theta) \\ &= \frac{1}{T}\sum_{t_s=1}^{T}\frac{1}{h^d}K(||\theta-\theta^s_{t_s}||/h)\frac{\hat{f}_s(\theta)}{\hat{f}_s(\theta^s_{t_s})} \\ &= \frac{1}{T}\sum_{t_s=1}^{T}\frac{\mathcal{N}_d(\theta|\theta^s_{t_s},h^2\mathcal{I}_d)\mathcal{N}_d(\theta|\hat{\mu}_s,\hat{\Sigma}_s)}{\mathcal{N}_d(\theta^s_{t_s}|\hat{\mu}_s,\hat{\Sigma}_s)},\end{aligned}$$

dove con $\theta^s_{t_s}$ è stato indicato il $t-$esimo valore del campione dei valori simulati dall'$s-$esimo gruppo di osservazioni. Per la parte non parametrica $\hat{r}(\theta)$ di $\hat{p}_s$ è stato usato uno stimatore kernel gaussiano con valore h per il parametro della ampiezza di banda. In questo modo, lo stimatore complessivo fornisce una stima pressoché gaussiana per T piccolo e converge alla vera distribuzione a posteriori al crescere di T. Lo stimatore $\hat{p}$ di $\pi(\theta|y)$ è quindi ottenuto moltiplicando le stime $\hat{p}_s$ delle varie $\pi(\theta|y_s)$, ottenendo la seguente espressione

$$\begin{aligned}\hat{p}(\theta) &= \hat{p}_1(\theta)\dots\hat{p}_S(\theta) \\ &= \frac{1}{T^S}\prod_{s=1}^{S}\sum_{t_s=1}^{T}\frac{\mathcal{N}_d(\theta|\theta^s_{t_s},h\mathcal{I}_d)\mathcal{N}_d(\theta|\hat{\mu}_s,\hat{\Sigma}_s)}{h^d\mathcal{N}_d(\theta^s_{t_s}|\hat{\mu}_s,\hat{\Sigma}_s)} \\ &\propto \sum_{t_1=1}^{T}\cdots\sum_{t_S=1}^{T}W_{t\cdot}\mathcal{N}_d(\theta|\mu_{t\cdot},\Sigma_{t\cdot}).\end{aligned}$$

Tale formula è proporzionale alla funzione di densità di una mistura di T^S normali con pesi non normalizzati

$$\begin{aligned}&W_{t\cdot} = \frac{w_{t\cdot}\mathcal{N}_d(\tilde{\theta}_{t\cdot}|\hat{\mu},\hat{\Sigma}+\frac{h}{S}\mathcal{I}_d)}{\prod_{s=1}^{S}\mathcal{N}_d(\theta^s_{t_s}|\hat{\mu}_s,\hat{\Sigma}_s)} \\ &\mu_{t\cdot} = \Sigma_{t\cdot}(\frac{S}{h}\mathcal{I}_d\tilde{\theta}t\cdot+\hat{\Sigma}^{-1}\hat{\mu}),\ \Sigma_{t\cdot} = (\frac{S}{h}\mathcal{I}_d+\hat{\Sigma}^{-1})^{-1} \\ &\tilde{\theta}_{t\cdot} = \frac{1}{S}\sum_{s=1}^{S}\theta^s_{t_s},\ w_{t\cdot} = \prod_{s=1}^{S}\mathcal{N}_d(\theta^s_{t_s}|\tilde{\theta}_{t\cdot},h^2\mathcal{I}_d) \\ &\hat{\Sigma} = (\sum_{s=1}^{S}\hat{\Sigma}_s^{-1})^{-1},\ \hat{\mu} = \hat{\Sigma}(\sum_{s=1}^{S}\hat{\Sigma}_s^{-1}\hat{\mu}_s).\end{aligned}$$

Per ottenere il campione finale rappresentativo di $\pi(\theta|y)$ si simulano quindi valori direttamente da tale mistura. Ciò può essere fatto agevol-

mente con una particolare implementazione dell'algoritmo MCMC, chiamata *Independent Metropolis within Gibbs* (IMG).

Il vantaggio di questo approccio è, come già accennato, il fatto che la stima finale di $\pi(\theta|y)$ converge alla distribuzione esatta di interesse per $T \to \infty$, al costo, però, di un maggiore onere computazionale rispetto alla procedura precedente. La scelta di usare uno stimatore semiparametrico per $\pi(\theta|y)$ date le $\pi(\theta|y_s)$ è qui stata fatta per unire i vantaggi dati dall'usare un approccio completamente parametrico, che assume una forma gaussiana per la distribuzione a posteriori e gode di una veloce convergenza, ma è distorto, e di uno completamente non parametrico, che è consistente per $T \to \infty$, ma può convergere lentamente se la dimensionalità di θ è elevata.

Sono state proposte altre tecniche per effettuare la combinazione dei campioni provenienti dai vari sottoinsiemi dei dati in quello finale. Una rassegna è presente in Scott (2017).

4.3 Un semplice esempio

Per mostrare le potenzialità dell'approccio proposto in questo capitolo, oltre che per mostrare in pratica il suo funzionamento, si mostra come esso si comporta in un semplice, ma significativo, esempio. I dati considerati sono 1000 realizzazioni indipendenti di una variabile di Bernoulli di parametro $1/2$. Sono stati ottenuti, in questo modo, 507 successi e 493 insuccessi. Come distribuzione a priori è stata considerata una distribuzione Beta di parametro $(1, 1)$, ovvero una Uniforme sull'intervallo $[0, 1]$. La distribuzione a posteriori $\pi(\theta|y)$ è, in questo caso, disponibile in forma esplicita in quanto la distribuzione Beta è la distribuzione a priori coniugata per la verosimiglianza bernoulliana. Tale distribuzione a posteriori riportata nella figura 4.1.

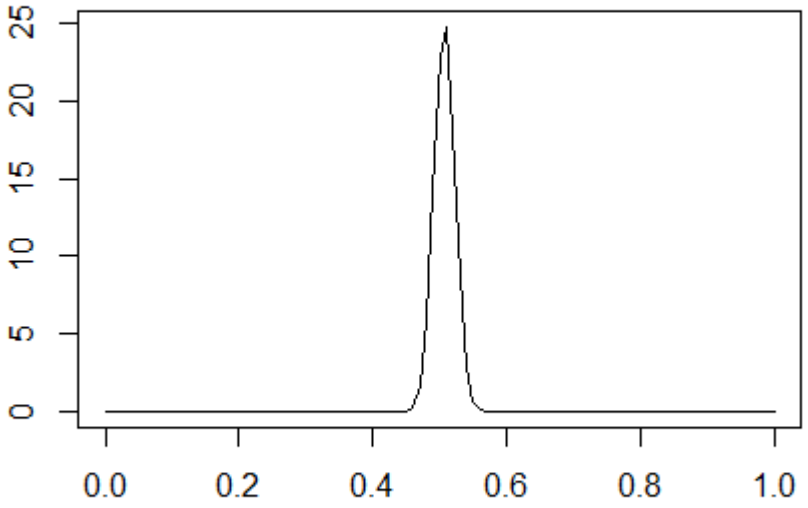

Figura 4.1: Vera distribuzione a posteriori per θ.

I dati sono poi stati suddivisi casualmente in 10 sottoinsiemi disgiunti di pari numerosità. Su ognuno di essi è stato applicato un algoritmo MCMC con distribuzione uniforme in $[x - 0.2, x + 0.2]$ come distribuzione proposta, dove x è il valore della catena per l'iterazione precedente. In questo modo sono state ottenute le stime per $\pi(\theta|y_s)$ per i 10 gruppi di osservazioni. Il tasso di accettazione complessivo di tutti gli algoritmi MCMC costruiti varia dal 37% al 40%, in linea con quanto desiderabile per gli algoritmi MCMC a passeggiata casuale, come ricordato al Capitolo 2. Per ogni algoritmo MCMC utilizzato sono state effettuate 10000 iterazioni e sono stati scartati i primi 5000 valori per il parametro in quanto considerati come burn-in. Come si può notare dalla figura 4.2, che riporta sia la densità di $\pi(\theta|y)$ che delle varie $\pi(\theta|y_s)$, tali stime parziali differiscono in modo significativo dalla vera distribuzione a posteriori.

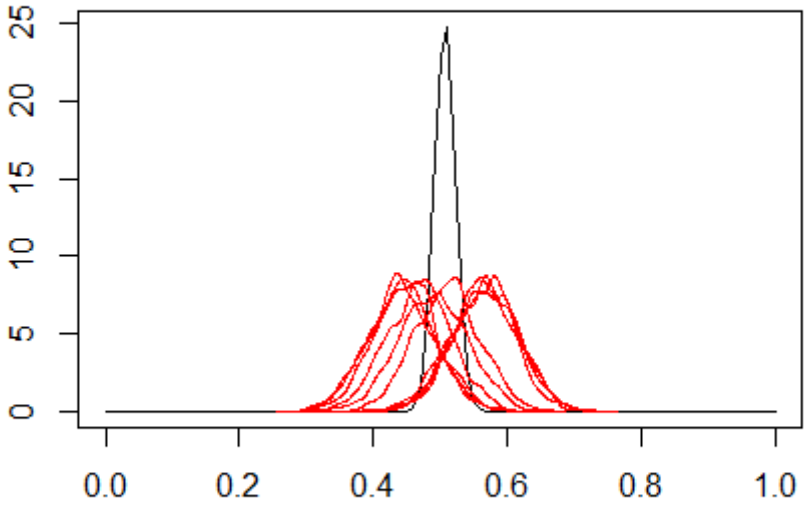

Figura 4.2: Vera distribuzione a posteriori per θ (in nero) confrontata con le distribuzioni a posteriori ottenute da ogni sottoinsieme (in rosso).

Si è poi proceduto nel combinare le 10 stime parziali $\pi(\theta|y_s)$, per $s = 1, \ldots, S$, con le due tecniche descritte nella sezione 3.3 di questo capitolo. Per fare ciò, si sono utilizzate le funzioni disponibili nel pacchetto `parallelMCMCcombine` (Miroshnikov et al. (2014)) del software statistico `R`. Nella figura 4.3, sono riportate le stime finali per $\pi(\theta|y)$ ottenute con entrambe le tecniche di combinazioni usate.

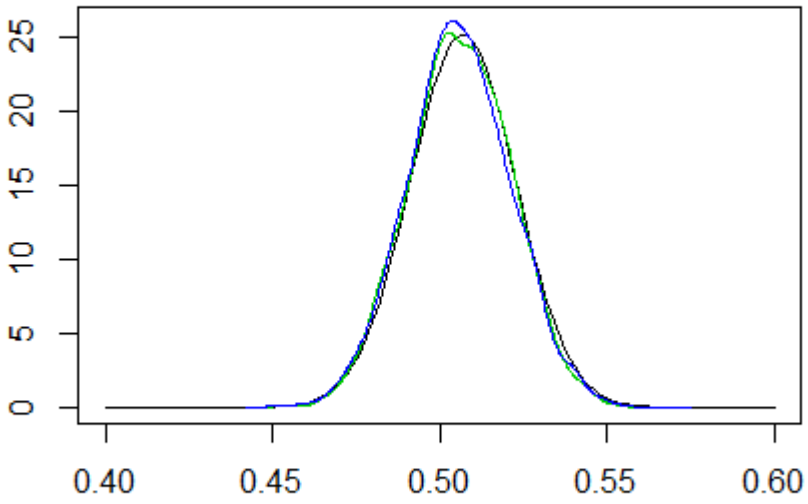

Figura 4.3: Vera distribuzione a posteriori per θ (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico).

Come si può notare, in tutti e due i casi tali stime sono vicine alla vera distribuzioni a posteriori, malgrado le stime parziali $\pi(\theta|y_s)$ fossero molto diverse da essa. Questo mette in luce, inoltre, come, anche in un esempio semplice, non sia una strada percorribile quella di utilizzare un approccio *naive* in cui si considera solo un campione casuale dei dati a disposizione, tale che la sua gestione sia agevole in termini computazionali. I risultati ottenuti in tale modo, come mostrano le distribuzioni in rosso della figura 4.2 sarebbero molto distanti da $\pi(\theta|y)$, cosa che invece non avviene considerando tutti i dati in y nel modo descritto in questo Capitolo.

4.4 Commenti e confronto tra tecniche basate sul campionamento e Divide et Impera

In questo Capitolo e nel Capitolo 3 sono state presentati due approcci molto differenti per adattare gli algoritmi MCMC, essenziali in ambito bayesiano, nei casi in cui la mole di dati che si vuole analizzare sia tale da rendere una scelta computazionalmente inadatta una loro applicazione diretta. Come già sottolineato più volte, il principale ostacolo è, in questo senso, la valutazione, a ogni iterazione, della verosimiglianza $L(y|\theta)$ che, coinvolgendo tutto y, spesso diventa un'operazione particolarmente lenta in presenza di Big Data, oltre a essere inagevole nel caso in cui i dati non risiedano nella memoria di un solo calcolatore.

Per prime, sono state presentate una serie di tecniche basate sull'uso di un campione delle osservazioni, in numero inferiore ad n, a ogni iterazione dell'algoritmo MCMC usato. In questo modo è possibile risolvere la questione relativa all'onere computazionale, mentre tale metodologia non risolve le problematiche che insorgono nel caso in cui i dati, per via della loro quantità, non possano risiedere nella memoria di un solo calcolatore. L'approccio presentato in questo Capitolo, ovvero il Divide et Impera, risolve entrambe le questioni ed è, quindi, da considerarsi più completo nell'affrontare l'adattamento degli algoritmi MCMC in contesti di Big Data.

Per questo motivo, nel prossimo Capitolo, si è voluto analizzare il suo comportamento empirico attraverso alcuni studi di simulazione e un'applicazione a dati reali. In particolare, ci si è concentrati, nell'effettuare queste operazioni, su modelli di regressione, molto usati in svariati campi della statistica.

Capitolo 5

Divide et Impera nei modelli di regressione

Nel Capitolo 4 sono stati spiegati i motivi per cui, all'interno dell'approccio Divide et Impera, l'ultima fase, ovvero quella che si occupa di combinare i campioni intermedi provenienti dai diversi gruppi in cui sono stati divisi i dati nel risultato finale, sia quella di maggiore importanza. In questo Capitolo verranno, quindi, messe a confronto le due tecniche precedentemente esposte, la prima caratterizzata dall'essere computazionalmente poco onerosa e di semplice impostazione e l'altra, più complessa, ma che garantisce l'esattezza asintotica del risultato finale proposto. A tal proposito, vengono presentate alcune simulazioni e un'applicazione a dati reali delle due tecniche di combinazione descritte al Capitolo 4. Ci si è concentrati sui modelli di regressione e l'obiettivo è quello di confrontare il comportamento di tali tecniche in questo contesto. Per fare ciò, si è utilizzato il software statistico `R` e, in particolar modo, alcune funzioni facenti parte delle librerie `MHadaptive` (Chivers (2012)) e `parallelMCMCcombine` (Miroshnikov et al. (2014)).

5.1 Dati simulati

Gli studi di simulazioni sono molto usati in statistica per valutare le proprietà delle procedure e delle metodologie in essa usate, sopratutto quando

sono coinvolte approssimazioni di un qualche tipo, che sarebbe difficoltoso comprendere appieno attraverso uno studio solo analitico. In questa Sezione si effettueranno alcuni studi di simulazione per valutare la bontà delle due metodologie di combinazione, presentate al Capitolo 4, che costituiscono la fase finale dell'approccio Divide et Impera. Le simulazioni che seguono riguardano il modello di regressione lineare, quello di regressione logistica e quello di regressione di Poisson.

Per ogni modello sono state generate 50000 osservazioni dalla specificazione descritta. Sui dati così ottenuti, è stato applicato un algoritmo MCMC adattivo, ovvero un algoritmo MCMC che regola in modo automatico i parametri della distribuzione proposta al fine di ottenere un buon tasso di accettazione, grazie alle funzioni della libreria `MHadaptive`. Sono state effettuate 20000 iterazioni, di cui le prime 10000 sono state scartate in quanto considerate come appartenenti al periodo di burn-in della catena markoviana costruita, in linea con quanto fatto nella Sezione riguardante l'applicazione a dati reali. I 10000 valori rimanenti per il parametro sono poi stati considerati come rappresentativi della vera distribuzione a posteriori $\pi(\theta|y)$. In seguito i dati sono stati suddivisi in modo casuale in 10 gruppi y_s per $s = 1, \ldots, 10$ e per ognuno di essi sono stati generati 20000 valori per θ da $\pi(\theta)^{1/10} L(y_s|\theta)$, di cui i primi 10000 sono stati scartati. I campioni intermedi così ottenuti sono stati quindi uniti nel campione finale per θ con le due tecniche di combinazione presentate al capitolo 4 grazie alle funzioni della libreria `parallelMCMCcombine`. Infine, si sono confrontate le due stime della distribuzione a posteriori cosi ottenute con la stima per $\pi(\theta|y)$ ottenuta considerando i dati in modo complessivo e considerata come vera distribuzione a posteriori per il modello in questione.

Seguendo quanto fatto nel presentare l'approccio Divide et Impera in Scott et al. (2016) e Neiswanger et al. (2013), tale confronto è stato basato sostanzialmente sul confronto delle distribuzioni marginali a posteriori per ogni singola componente di θ. In questo senso, è stata considerata anche una misura di distanza tra distribuzioni, ovvero la distanza di Hellinger, pari a $H(p(\theta), f(\theta)) = \frac{1}{2} \int (p(\theta)^{\frac{1}{2}} - f(\theta)^{\frac{1}{2}})^2$ (Nikulin (2001)), dove $p(\theta)$ e $f(\theta)$ sono due funzioni di densità di cui si vuole misurare la distanza. Per semplificare l'esposizione, nelle tabelle che seguono il valore stimato per tale distanza è stato moltiplicato per 1000. Inoltre, con $\pi(\theta|y)^{avg}$ si farà riferimento alla stima

della distribuzione a posteriori ottenuta mediante la tecnica di combinazione basata sulla media pesata dei campioni intermedi, mente con $\pi(\theta|y)^{semi}$ a quella ottenuta mediante l'approccio semiparametrico. Con $\pi(\theta|y)$ si indicherà la stima per la distribuzione a posteriori ottenuta applicando un algoritmo MCMC a tutti i dati e considerata come vera distribuzione a posteriori per il parametro.

5.1.1 Modello di regressione lineare

Il modello considerato è del seguente tipo:

$$\begin{aligned}
&\theta = (\beta_1, \sigma^2) \\
&y_i = \beta_1 x_i + \varepsilon_i,\ \varepsilon_i \sim N(0, \sigma_2) \\
&\pi(\beta_1) \sim N(b_1, b_2),\ \pi(\sigma^2) \sim Gamma(c_1, c_2) \\
&\pi(\theta) \sim \pi(\beta_1)\pi(\sigma^2)
\end{aligned}$$

per $i = 1, \ldots, 50000$, con $\beta_1 = 1$, $b_1 = 0$, $b_2 = 100$, $\sigma^2 = 5$, $c_1 = 1$ e $c_2 = 1/100$. La covariata x_i è formata realizzazioni indipendenti e identicamente distribuite di una variabile Uniforme in $[-5, 5]$.

Nelle figure 5.1 e 5.2 sono riportate le distribuzioni marginali a posteriori ottenute mediante le due tecniche di combinazione, oltre che a quelle ottenute considerando i dati in modo complessivo e considerate come corrette. Nella figura 5.3 sono confrontate le curve di livello sotto cui giace, rispettivamente il 75%, il 90% e il 99% della distribuzioni a posteriori per θ.

Nella tabella 5.1 sono, invece, riportate le stime di mediana, scarto interquartile e distanza di Hellinger per le distribuzioni marginali a posteriori ottenute mediante combinazione e quelle ottenute considerando i dati globalmente.

Come si può vedere da tali risultati, il comportamento delle due tecniche di combinazione è, in questo caso, simile. Inoltre, con entrambi gli approcci si nota un comportamento complessivo piuttosto buono del Divide et Impera nell'approssimare la distribuzione a posteriori del modello $\pi(\theta|y)$.

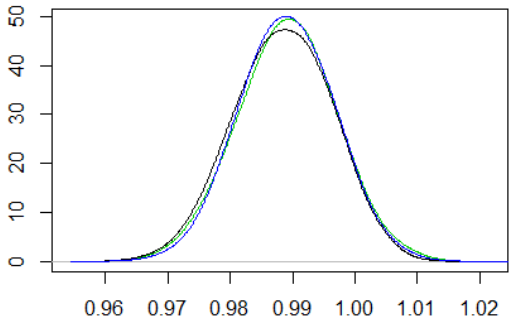

Figura 5.1: Distribuzione a posteriori per β_1 (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico) nel modello di regressione lineare.

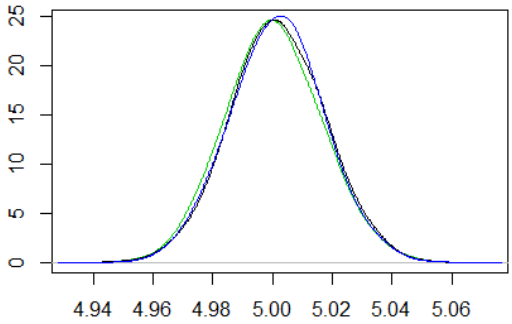

Figura 5.2: Distribuzione a posteriori per σ^2 (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico) nel modello di regressione lineare.

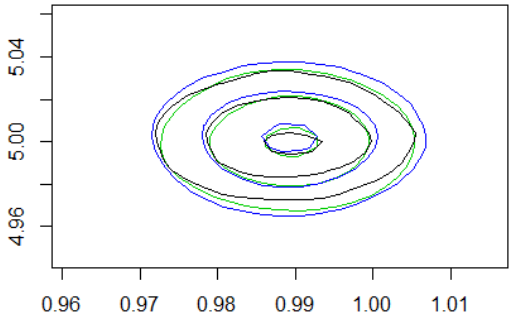

Figura 5.3: Curve di livello della distribuzione a posteriori per θ ottenuta nel modello di regressione lineare (in verde tramite media pesata e in blu tramite l'approccio semiparametrico).

	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_1\|y))$
$\pi^{avg}(\beta_1\|y)$	0.989	0.010	1.816
$\pi^{semi}(\beta_1\|y)$	0.989	0.010	1.878
$\pi(\beta_1\|y)$	0.989	0.011	0.000
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\sigma^2\|y))$
$\pi^{avg}(\sigma^2\|y)$	5.000	0.021	1.342
$\pi^{semi}(\sigma^2\|y)$	5.002	0.021	0.747
$\pi(\sigma^2\|y)$	5.002	0.021	0

Tabella 5.1: Mediana, scarto interquartile e distanza di Hellinger per le distribuzioni marginali a posteriori ottenute nel modello di regressione lineare

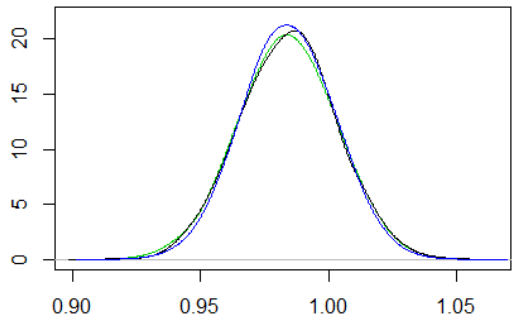

Figura 5.4: Distribuzione a posteriori per β_1 (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico) nel modello di regressione logistica.

5.1.2 Modello di regressione logistica

Il modello considerato è del seguente tipo:

$$\begin{aligned}
&\theta = (\beta_1, \beta_2) \\
&y_i \sim Bernoulli(p_i),\ p_i = \frac{\exp(\beta_1 x_{1,i} + \beta_2 x_{2,i})}{1 + \exp(\beta_1 x_{1,i} + \beta_2 x_{2,i})} \\
&\pi(\beta_1) \sim N(b_1, b_2),\ \pi(\beta_2) \sim N(b_3, b_4) \\
&\pi(\theta) \sim \pi(\beta_1)\pi(\beta_2)
\end{aligned}$$

per $i = 1, \ldots, 50000$, con $\beta_1 = \beta_2 = 1$, $b_1 = b_3 = 0$ e $b_2 = b_4 = 100$. Le covariate $x_{1,i}$ sono realizzazioni indipendenti e identicamente distribuite di una variabile Uniforme in $[-1, 1]$, mentre le covariate $x_{2,i}$ sono realizzazioni indipendenti e identicamente distribuite di una variabile Binomiale di parametro pari a $1/2$ e supporto $\{-1, 1\}$.

Le figure 5.4, 5.5, 5.6 e la tabella 5.2 riportano i risultati descritti precedentemente per il modello di regressione logistica.

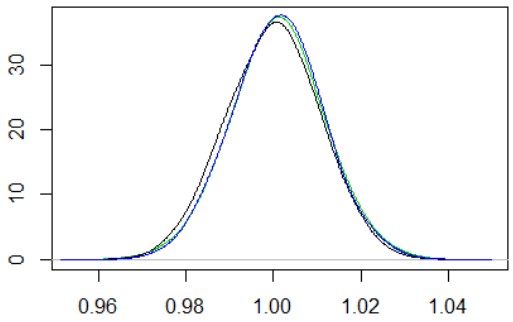

Figura 5.5: Distribuzione a posteriori per β_2 (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico) nel modello di regressione logistica.

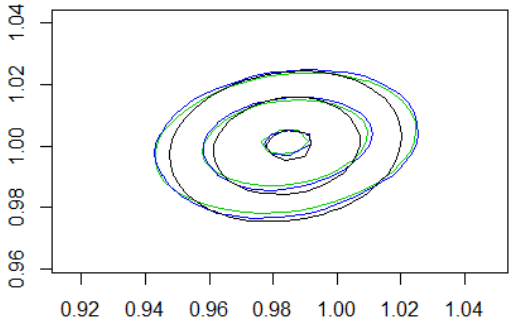

Figura 5.6: Curve di livello della distribuzione a posteriori per θ ottenuta nel modello di regressione logistica (in verde tramite media pesata e in blu tramite l'approccio semiparametrico).

	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_1\|y))$
$\pi^{avg}(\beta_1\|y)$	0.984	0.026	1.095
$\pi^{semi}(\beta_1\|y)$	0.984	0.024	1.372
$\pi(\beta_1\|y)$	0.985	0.025	0
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_2\|y))$
$\pi^{avg}(\beta_2\|y)$	1.001	0.014	1.758
$\pi^{semi}(\beta_2\|y)$	1.001	0.014	1.905
$\pi(\beta_2\|y)$	1	0.015	0

Tabella 5.2: Mediana, scarto interquartile e distanza di Hellinger per le distribuzioni marginali a posteriori ottenute nel modello di regressione logistica.

Anche in questo caso non ci sembrano essere grosse differenze tra le due tecniche di combinazione dei campioni intermedi prodotti con il Divide et Impera e l'approssimazione ottenuta per $\pi(\theta|y)$ si mostra, in entrambi i casi, buona.

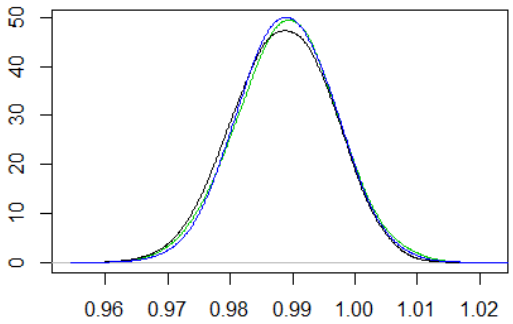

Figura 5.7: Distribuzione a posteriori per β_1 (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico) nel modello di regressione di Poisson.

5.1.3 Modello di regressione di Poisson

Il modello considerato è del seguente tipo:

$$\begin{aligned}
&\theta = (\beta_1, \beta_2) \\
&y_i \sim Poisson(\mu_i),\ \mu_i = \exp(\beta_1 x_{1,i} + \beta_2 x_{2,i}) \\
&\pi(\beta_1) \sim N(b_1, b_2),\ \pi(\beta_2) \sim N(b_3, b_4) \\
&\pi(\theta) \sim \pi(\beta_1)\pi(\beta_2)
\end{aligned}$$

per $i = 1, \ldots, 50000$, con $\beta_1 = \beta_2 = 1$, $b_1 = b_3 = 0$ e $b_2 = b_4 = 100$.

Di seguito sono riportati i risultati ottenuti con questo modello. Le covariate $x_{1,i}$ sono realizzazioni indipendenti e identicamente distribuite di una variabile Uniforme in $[0, 2]$, mentre le covariate $x_{2,i}$ sono realizzazioni indipendenti e identicamente distribuite di una variabile binomiale di parametro pari a $1/2$ e supporto $\{-1, 1\}$.

Anche in questo ultimo caso le due tecniche di combinazione forniscono risultati simili, approssimando bene la vera distribuzione a posteriori.

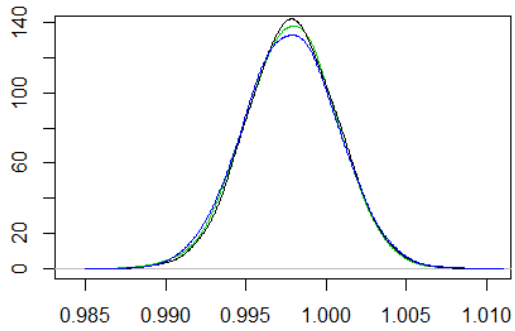

Figura 5.8: Distribuzione a posteriori per β_2 (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico) nel modello di regressione di Poisson.

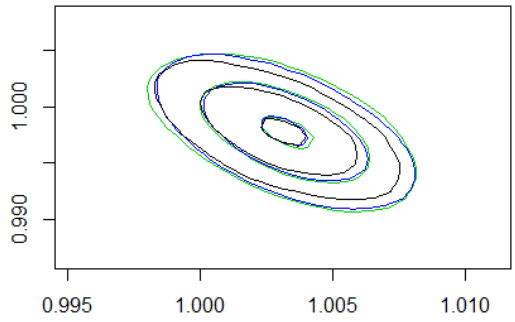

Figura 5.9: Curve di livello della distribuzione a posteriori per θ ottenuta nel modello di regressione di Poisson (in verde tramite media pesata e in blu tramite l'approccio semiparametrico).

	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_1\|y))$
$\pi^{avg}(\beta_1\|y)$	1.003	0.003	2.998
$\pi^{semi}(\beta_1\|y)$	1.003	0.003	3.076
$\pi(\beta_1\|y)$	1.003	0.003	0
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_2\|y))$
$\pi^{avg}(\beta_2\|y)$	0.998	0.004	0.783
$\pi^{semi}(\beta_2\|y)$	0.998	0.004	1.114
$\pi(\beta_2\|y)$	0.998	0.004	0

Tabella 5.3: Mediana, scarto interquartile e distanza di Hellinger per le distribuzioni marginali a posteriori ottenute nel modello di regressione di Poisson.

5.1.4 Riassunto dei risultati ottenuti sui dati simulati

La tabella 5.4 riporta un riassunto dei risultati ottenuti tramite gli studi di simulazione effettuati, concernenti il modello di regressione lineare, quello di regressione logistica e quello di regressione di Poisson. Come si può vedere, in tutti i casi analizzati, i risultati forniti dalle due diverse tecniche di combinazione studiate, ovvero dai due metodi di aggregazione dei risultati nel Divide et Impera trattati in questa tesi, sono molto simili e l'approssimazione della vera distribuzioni a posteriori risulta sempre molto buona.

Modello di regressione lineare			
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_1\|y))$
$\pi^{avg}(\beta_1\|y)$	0.989	0.010	1.816
$\pi^{semi}(\beta_1\|y)$	0.989	0.010	1.878
$\pi(\beta_1\|y)$	0.989	0.011	0
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\sigma^2\|y))$
$\pi^{avg}(\sigma^2\|y)$	5.000	0.021	1.342
$\pi^{semi}(\sigma^2\|y)$	5.002	0.021	0.747
$\pi(\sigma^2\|y)$	5.002	0.021	0
Modello di regressione logistica			
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_1\|y))$
$\pi^{avg}(\beta_1\|y)$	0.984	0.026	1.095
$\pi^{semi}(\beta_1\|y)$	0.984	0.024	1.372
$\pi(\beta_1\|y)$	0.985	0.025	0
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_2\|y))$
$\pi^{avg}(\beta_2\|y)$	1.001	0.014	1.758
$\pi^{semi}(\beta_2\|y)$	1.001	0.014	1.905
$\pi(\beta_2\|y)$	1	0.015	0
Modello di regressione di Poisson			
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_1\|y))$
$\pi^{avg}(\beta_1\|y)$	1.003	0.003	2.998
$\pi^{semi}(\beta_1\|y)$	1.003	0.003	3.076
$\pi(\beta_1\|y)$	1.003	0.003	0
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_2\|y))$
$\pi^{avg}(\beta_2\|y)$	0.998	0.004	0.783
$\pi^{semi}(\beta_2\|y)$	0.998	0.004	1.114
$\pi(\beta_2\|y)$	0.998	0.004	0

Tabella 5.4: Riassunto dei risultati ottenuti con i dati simulati

5.2 Dati reali

In questa Sezione le due tecniche di combinazione studiate, ovvero i due metodi di aggregazione dei risultati nel Divide et Impera trattati in questa tesi, sono state usate in un modello di regressione logistica applicato a dati reali. In particolare, si è preso in esame lo *Skin Segmentation Dataset* (Bhatt e Dhall (2010)). Tale dataset prende in considerazione campioni del colore della pelle provenienti da facce di persone di diversi gruppi di età, etnie e sesso, oltre che campioni di colore non provenienti da facce di individui. Ogni unità è composta da tre covariate, di seguito denominate con B, G e R, che codificano il colore rilevato (Hunt (1967)). La variabile risposta y indica se il campione di colore proviene da una faccia (in tal caso è codificata con 0), o meno (in tal caso è codificata con 1). Sono state prese in considerazione 100000 unità di tale dataset, la metà delle quali provenienti da individui. La struttura dei dati in questione è visibile nella tabella 5.5.

B	G	R	y
44	51	18	1
181	178	133	1
177	174	129	1
199	198	160	1
53	105	151	0
158	162	121	1
198	209	253	0
170	184	237	0
179	175	134	1
127	171	234	0

Tabella 5.5: Struttura dei dati presenti nello *Skin Segmentation Dataset*.

L'obbiettivo è quello di prevedere quali osservazioni non provengono da facce di individui e per fare ciò è stato applicato ai dati appena descritti un modello di regressione logistica. Tutte le covariate sono state standardizzate in modo che i loro valori fossero compresi in $[0, 1]$ e sono state prese in considerazione tutte le interazioni di primo e secondo livello. Il modello così

costruito è del seguente tipo:

$$\theta = (\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8)$$
$$y_i \sim Bernoulli(p_i)$$
$$p_i = \frac{\exp(\beta_1 + \beta_2 B_i + \beta_3 G_i + \beta_4 R_i + \beta_5 B_i G_i + \beta_6 B_i R_i + \beta_7 G_i R_i + \beta_8 B_i G_i R_i)}{1 + \exp(\beta_1 + \beta_2 B_i + \beta_3 G_i + \beta_4 R_i + \beta_5 B_i G_i + \beta_6 B_i R_i + \beta_7 G_i R_i + \beta_8 B_i G_i R_i)}$$
$$\pi(\beta_1) \sim N(0, 100),\ \pi(\beta_2) \sim N(0, 100),\ \pi(\beta_3) \sim N(0, 100),\ \pi(\beta_4) \sim N(0, 100)$$
$$\pi(\beta_5) \sim N(0, 100),\ \pi(\beta_6) \sim N(0, 100),\ \pi(\beta_7) \sim N(0, 100),\ \pi(\beta_8) \sim N(0, 100)$$
$$\pi(\theta) \sim \pi(\beta_1)\pi(\beta_2)\pi(\beta_3)\pi(\beta_4)\pi(\beta_5)\pi(\beta_6)\pi(\beta_7)\pi(\beta_8)$$

per $i = 1, \ldots, 100000$.

Su tale modello è stato quindi per prima cosa applicato un algoritmo MCMC adattivo, allo stesso modo di quanto fatto nella Sezione riguardante i Dati Simulati, considerando i dati in modo complessivo. Sono state effettuate 60000 iterazioni e la prima metà dei valori della catena markoviana costruita sono stati scartati in quanto considerati facenti parte del periodo di burn-in, seguendo quanto fatto, in un contesto simile, in Neiswanger et al. (2013). I valori rimanenti sono quindi stati usati come rappresentativi della vera distribuzione a posteriori $\pi(\theta|y)$, obiettivo dell'analisi. L'avvenuta convergenza della catena markoviana così ottenuta è stata verificata attraverso la procedura diagnostica di Heidelberger e Welch (Heidelberger e Welch (1981)) implementata nella libreria `coda` (Plummer et al. (2006)). In seguito i dati sono stati suddivisi casualmente in 20 gruppi disgiunti e, su ognuno di essi, è stato usato il medesimo algoritmo MCMC, ottenendo 60000 valori per θ e scartandone la prima metà. I campioni intermedi così ottenuti sono stati poi uniti a ottenere le stime finali per la distribuzione a posteriori $\pi^{avg}(\theta|y)$ e $\pi^{semi}(\theta|y)$ con le due tecniche descritte al Capitolo 4. I grafici e la tabella di questa sezione riportano i confronti tra $\pi(\theta|y)$, $\pi^{avg}(\theta|y)$ e $\pi^{semi}(\theta|y)$ come descritto nella Sezione riguardante le simulazioni svolte. Per completezza, nei grafici in cui si confrontano le distribuzioni a posteriori marginali per il modello dato, sono state riportate le stime che si ottengono con il corrispettivo modello di regressione logistica in ottica frequentista, ognuna rappresentata da una linea tratteggiata.

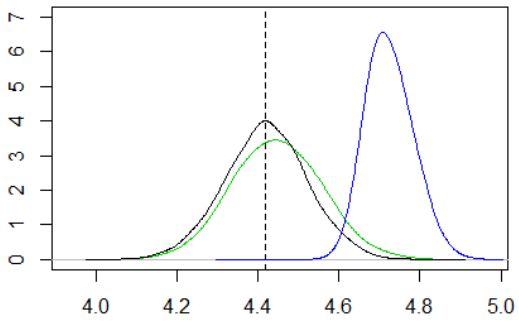

Figura 5.10: Distribuzione a posteriori per β_1 (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico) nel modello di regressione applicato ai dati reali.

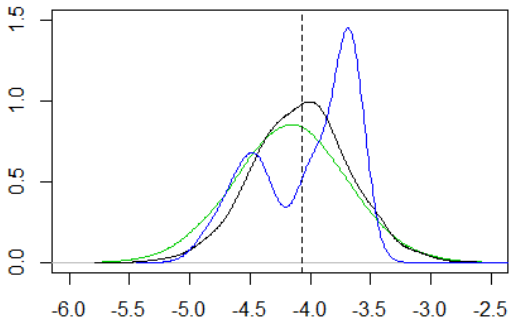

Figura 5.11: Distribuzione a posteriori per β_2 (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico) nel modello di regressione applicato ai dati reali.

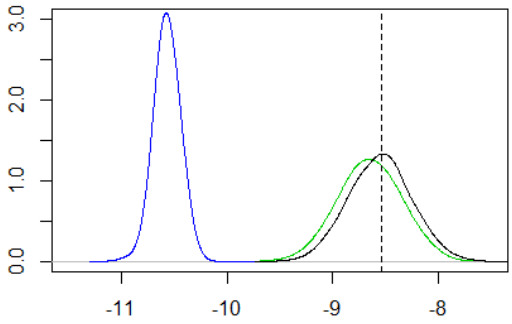

Figura 5.12: Distribuzione a posteriori per β_3 (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico) nel modello di regressione applicato ai dati reali.

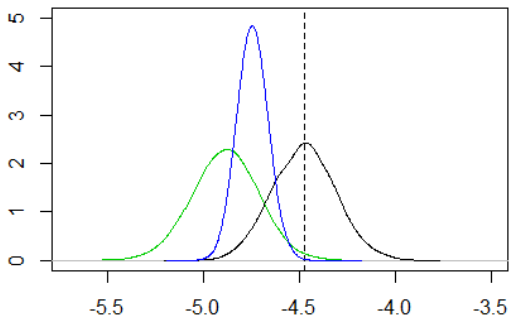

Figura 5.13: Distribuzione a posteriori per β_4 (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico) nel modello di regressione applicato ai dati reali.e

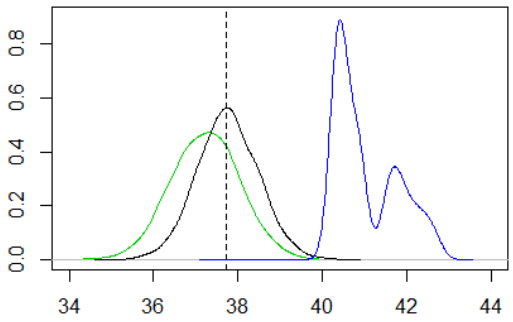

Figura 5.14: Distribuzione a posteriori per β_5 (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico) nel modello di regressione applicato ai dati reali.

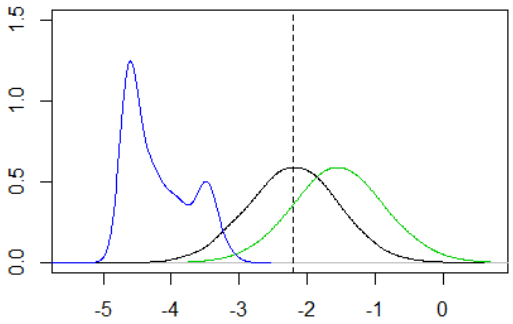

Figura 5.15: Distribuzione a posteriori per β_6 (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico) nel modello di regressione applicato ai dati reali.

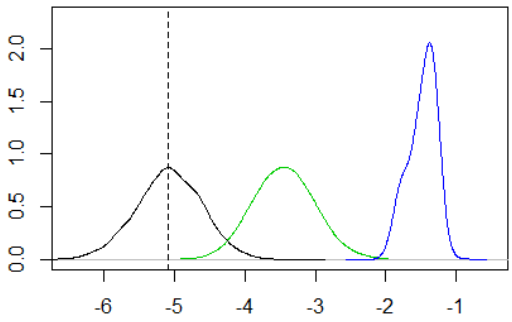

Figura 5.16: Distribuzione a posteriori per β_7 (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico) nel modello di regressione applicato ai dati reali.

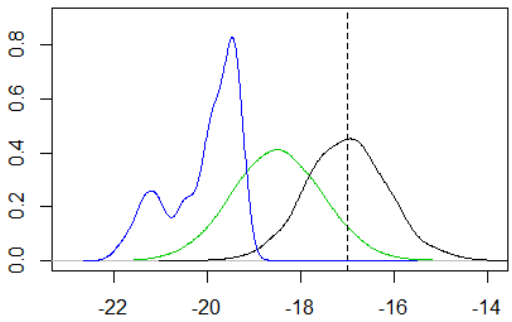

Figura 5.17: Distribuzione a posteriori per β_8 (in nero) confrontata con le distribuzioni ottenute dopo la combinazione delle diverse $\pi(\theta|y_s)$ (in verde tramite media pesata e in blu tramite l'approccio semiparametrico) nel modello di regressione applicato ai dati reali.

	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_1\|y))$
$\pi^{avg}(\beta_1\|y)$	4.446	0.152	10.781
$\pi^{semi}(\beta_1\|y)$	4.721	0.081	826.623
$\pi(\beta_1\|y)$	4.420	0.135	0
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_2\|y))$
$\pi^{avg}(\beta_2\|y)$	-4.161	0.616	11.158
$\pi^{semi}(\beta_2\|y)$	-3.919	0.732	75.981
$\pi(\beta_2\|y)$	-4.068	0.532	0
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_3\|y))$
$\pi^{avg}(\beta_3\|y)$	-8.65	0.413	13.687
$\pi^{semi}(\beta_3\|y)$	-10.573	0.172	1000.962
$\pi(\beta_3\|y)$	-8.55	0.396	0
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_4\|y))$
$\pi^{avg}(\beta_4\|y)$	-4.884	0.23	521.837
$\pi^{semi}(\beta_4\|y)$	-4.744	0.108	464.272
$\pi(\beta_4\|y)$	-4.475	0.226	0
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_5\|y))$
$\pi^{avg}(\beta_5\|y)$	37.239	1.098	62.29
$\pi^{semi}(\beta_5\|y)$	40.766	1.261	972.646
$\pi(\beta_5\|y)$	37.752	0.974	0
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_6\|y))$
$\pi^{avg}(\beta_6\|y)$	-1.544	0.891	117.522
$\pi^{semi}(\beta_6\|y)$	-4.327	0.78	800.682
$\pi(\beta_6\|y)$	-2.2	0.896	0
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_7\|y))$
$\pi^{avg}(\beta_7\|y)$	-3.449	0.596	815.564
$\pi^{semi}(\beta_7\|y)$	-1.446	0.289	1000.94
$\pi(\beta_7\|y)$	-5.077	0.626	0
	Mediana	Scarto Interquartile	$H(\cdot, \pi(\beta_8\|y))$
$\pi^{avg}(\beta_8\|y)$	-18.501	1.296	282.796
$\pi^{semi}(\beta_8\|y)$	-19.849	1.073	912.012
$\pi(\beta_8\|y)$	-17.005	1.172	0

Tabella 5.6: Mediana, scarto interquartile e distanza di Hellinger per le distribuzioni marginali a posteriori ottenute nel modello di regressione applicato a dati reali.

Come si può vedere, a parte per quanto avviene per i parametri β_4, β_7 e β_8, le stime delle distribuzioni a posteriori marginali ottenute con la tecnica di combinazione basata sulla media pesata dei campioni intermedi provenienti dai diversi gruppi in cui sono stati suddivisi i dati, sono, in genere, piuttosto vicine alle corrispettive $\pi(\cdot|y)$. Lo stesso non avviene per le stime delle distribuzioni a posteriori marginali ottenute con l'approccio semiparametrico, che in generale, si comporta in modo peggiore rispetto al precedente. Ciò potrebbe essere dovuto al fatto che, come sottolineato in Scott (2017), i metodi di combinazione basati su stime non parametriche di tipo kernel delle diverse distribuzioni a posteriori $\pi(\theta|y_s)$, ognuna ottenuta da uno degli S gruppi in cui sono stati divisi i dati, possono non essere affidabili quando la dimensionalità di θ è elevata. Il numero di componenti per θ oltre al quale potrebbe insorgere tale problema è, indicativamente, posto pari a 6, rifacendosi a quanto riportato in Scott (2015) sulle proprietà degli stimatori di tipo kernel per funzioni di densità.

5.3 Commenti

In questo Capitolo sono state confrontate le due tecniche di combinazione descritte al Capitolo precedente e il loro comportamento all'interno di modelli di regressione, mediante alcuni studi di simulazione e l'applicazione a un dataset reale. Per quanto riguarda gli studi di simulazione, i risultati forniti mostrano che entrambe le tecniche, ovvero quella basata sulla media pesata dei campioni intermedi provenienti dai diversi gruppi in cui sono stati suddivisi i dati e quella basata sull'approccio semiparametrico, riescono ad approssimare molto bene la vera distribuzione a posteriori, senza grandi differenze tra una e l'altra.

Per quanto riguarda, invece, l'applicazione allo *Skin Segmentation Dataset*, i risultati mostrano un comportamento migliore per la tecnica più semplice, ovvero quella basata sulla media pesata dei campioni intermedi, con la quale si ottengono buone approssimazioni delle distribuzioni marginali a posteriori per le componenti del parametro θ nella maggior parte dei casi. I risultati ottenuti, invece, con l'approccio semiparametrico sono, nella

maggior parte dei casi peggiori e l'approssimazione della vera distribuzione a posteriori si presenta distante dall'obbiettivo.

Dato tutto ciò, è interessante notare come l'approccio più semplice e con minori garanzie teoriche mostri un comportamento equivalente, o migliore, rispetto a quello più complesso e dotato di garanzie asintotiche in tutti i casi studiati. Il primo metodo richiede, inoltre, risorse computazionali molto minori, essendo quindi più rapido da eseguire. Per tutte queste ragioni, almeno all'interno dei modelli di regressione, sembra essere un ottimo candidato per la fase finale dell'approccio Divide et Impera.

Capitolo 6

5Conclusioni

Come sottolineato al Capitolo 2, le usuali tecniche che permettono di ottenere una stima della distribuzione a posteriori $\pi(\theta|y)$ in ambito bayesiano, ovvero l'oggetto di interesse del processo inferenziale in questo contesto, sono, spesso, computazionalmente troppo onerose per essere usate in presenza di Big Data. In questa tesi, facendo soprattutto riferimento agli algoritmi MCMC, sono stati descritti alcuni accorgimenti che permettono di utilizzare l'inferenza bayesiana anche in questo contesto.

I problemi che possono insorgere in questo senso, come è stato spiegato nei Capitoli 1 e 2, sono essenzialmente di due tipi:

- legati al tempo di esecuzione. Valutare la verosimiglianza $L(y|\theta)$ a ogni iterazione dell'algoritmo usato può essere troppo oneroso in presenza di Big Data

- legati al fatto che i dati a disposizione possono essere in quantità tale da non risiedere nella memoria di un solo calcolatore. Anche in questo caso, il problema è dovuto alla necessità di dover valutare la verosimiglianza $L(y|\theta)$ a ogni iterazione dell'algoritmo usato, cosa non agevole qualora le osservazioni disponibili fossero distribuite su più calcolatori.

Il primo approccio trattato e descritto al Capitolo 3, ovvero quello basato sul sotto-campionamento dei dati, per rendere utilizzabile l'inferenza bayesiana in un contesto di Big Data attraverso gli algoritmi MCMC affronta solo

il primo di tali problemi. Questo metodo, di cui sono state descritte tre diverse varianti, prevede di utilizzare, a ogni iterazione dell'algoritmo MCMC usato, un campione casuale semplice di numerosità inferiore a quella dei dati a disposizione, in luogo di tutte le osservazioni contenute in y. Tale metodologia, però, non affrontando la problematica dovuta alla possibilità che i dati non risiedano nella memoria di un solo calcolatore, non può essere considerata una soluzione del tutto soddisfacente ai problemi che insorgono nel voler utilizzare la statistica bayesiana in un contesto di Big Data.

Il secondo metodo, descritto al Capitolo 4, ovvero il Divide et Impera, affronta entrambe le problematiche descritte e può quindi essere considerato una soluzione più completa della precedente. Qui l'idea è quella dividere i dati a disposizione in sottoinsiemi disgiunti, ottenere diverse stime della distribuzione a posteriori da ognuno di essi con un qualsiasi algoritmo MCMC e, infine, combinare tali stime in una finale. Tale processo può essere, inoltre, facilmente parallelizzato su più calcolatori, rendendo il suo utilizzo ancora più appetibile in presenza di Big Data. Sono state descritte due diverse varianti di questo approccio, che differiscono per il modo in cui i risultati intermedi provenienti dai vari gruppi in cui sono stati suddivisi i dati vengono combinati in quello finale. La prima di esse è caratterizzata dall'essere computazionalmente poco onerosa e di semplice impostazione mentre l'altra è più complessa e computazionalmente onerosa, ma con maggiori garanzie teoriche. Si è, inoltre, mostrato il funzionamento del Divide et Impera tramite un esempio illustrativo.

Con il Capitolo conclusivo, ovvero il Capitolo 5, si è voluto testare il comportamento del Divide et Impera, ovvero dell'approccio più completo per rendere utilizzabile l'inferenza bayesiana in presenza di Big Data tra quelli descritti in questa tesi, all'interno di modelli di regressione, molto usati in svariati campi della statistica. Per fare ciò sono stati svolti alcuni studi di simulazione e un'applicazione a dati reali. I risultati hanno mostrato come il Divide et Impera possa essere considerato, in generale, una buona soluzione per condurre analisi inferenziali di tipo bayesiano in presenza di Big Data all'interno di modelli di regressione. É, inoltre, interessante notare, come, la tecnica di combinazione dei risultati provenienti dai diversi gruppi in cui sono stati suddivisi i dati più semplice e con minori garanzie teoriche mostri

un comportamento equivalente, o migliore, rispetto a quella più complessa e dotata di garanzie asintotiche in tutti i casi studiati, essendo, inoltre, computazionalmente molto meno onerosa. Come sottolineato nella parte finale del Capitolo 5, ciò è probabilmente dovuto al fatto che l'approccio di combinazione più complesso, ovvero quello semiparametrico, può essere poco affidabile nel caso in cui θ abbia una dimensionalità elevata. Poiché tale situazione si verifica spesso nelle applicazioni che riguardano Big Data e per tutte le altre ragioni enunciate, l'approccio più semplice, basato sulla media pesata dei risultati provenienti dai gruppi in cui sono state divise le osservazioni, sembra essere preferibile, almeno nei modelli di regressione, anche se sprovvisto delle garanzie teoriche di cui gode quello più complesso di tipo semiparametrico.

Bibliografia

Andrieu, C. e M. Vihola. «A Tutorial on Variational Bayesian Inference». In: *Artificial Intelligence Review* (2011).

Andrieu, Christophe e Gareth O Roberts. «The pseudo-marginal approach for efficient Monte Carlo computations». In: *The Annals of Statistics* (2009), pp. 697–725.

Andrieu, Christophe, Matti Vihola et al. «Convergence properties of pseudo-marginal Markov chain Monte Carlo algorithms». In: *The Annals of Applied Probability* 25.2 (2015), pp. 1030–1077.

Angelino, Elaine, Matthew James Johnson, Ryan P Adams et al. «Patterns of scalable Bayesian inference». In: *Foundations and Trends® in Machine Learning* 9.2-3 (2016), pp. 119–247.

Azzalini, A. e B. Scarpa. *Data analysis and data mining: an introduction*. Oxford University Press, 2012.

Bardenet, Rémi, Arnaud Doucet e Chris Holmes. «On Markov chain Monte Carlo methods for tall data». In: *arXiv preprint arXiv:1505.02827* (2015).

Bhatt, Rajen e Abhinav Dhall. «Skin segmentation dataset». In: *UCI Machine Learning Repository* (2010).

Bishop, Christopher M. *Pattern recognition and machine learning*. springer, 2006.

Brooks, Steve et al. *Handbook of markov chain monte carlo*. CRC press, 2011.

Ceperley, DM e M Dewing. «The penalty method for random walks with uncertain energies». In: *The Journal of chemical physics* 110.20 (1999), pp. 9812–9820.

Chib, Siddhartha e Edward Greenberg. «Understanding the metropolis-hastings algorithm». In: *The american statistician* 49.4 (1995), pp. 327–335.

Chivers, C. *MHadaptive: general Markov Chain Monte Carlo for Bayesian inference using adaptive Metropolis-Hastings sampling*. 2012.

Damien, Paul et al. *Bayesian Theory and Applications*. OUP Oxford, 2013.

Fearnhead, Paul et al. «Random-weight particle filtering of continuous time processes». In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.4 (2010), pp. 497–512.

Grimmer, Justin. «An introduction to Bayesian inference via variational approximations». In: *Political Analysis* 19.1 (2010), pp. 32–47.

Heidelberger, Philip e Peter D Welch. «A spectral method for confidence interval generation and run length control in simulations». In: *Communications of the ACM* 24.4 (1981), pp. 233–245.

Hunt, Robert William Gainer. «The reproduction of colour». In: (1967).

Jacob, Pierre E, Alexandre H Thiery et al. «On nonnegative unbiased estimators». In: *The Annals of Statistics* 43.2 (2015), pp. 769–784.

Le Cam, Lucien e Grace Lo Yang. *Asymptotics in statistics: some basic concepts*. Springer Science & Business Media, 2012.

Lee, Peter M. *Bayesian statistics: an introduction*. John Wiley & Sons, 2012.

Maclaurin, Dougal e Ryan P Adams. «Firefly Monte Carlo: Exact MCMC with Subsets of Data.» In: *UAI*. 2014, pp. 543–552.

Miroshnikov, Alexey, Erin Conlon e Maintainer Alexey Miroshnikov. «Package 'parallelMCMCcombine'». In: (2014).

Neiswanger, Willie, Chong Wang e Eric Xing. «Asymptotically exact, embarrassingly parallel MCMC». In: *arXiv preprint arXiv:1311.4780* (2013).

Nikulin, Mikhail S. «Hellinger distance». In: *Encyclopedia of mathematics* 151 (2001).

Plummer, Martyn et al. «CODA: convergence diagnosis and output analysis for MCMC». In: *R news* 6.1 (2006), pp. 7–11.

Quiroz, Matias, Mattias Villani e Robert Kohn. «Exact subsampling MCMC». In: *arXiv preprint arXiv:1603.08232* (2016).

Quiroz, M. et al. «Speeding Up MCMC by Efficient Data Subsampling». In: *ArXiv e-prints* (apr. 2014). arXiv: `1404.4178 [stat.ME]`.

Rabinovich, Maxim, Elaine Angelino e Michael I Jordan. «Variational consensus monte carlo». In: *Advances in Neural Information Processing Systems*. 2015, pp. 1207–1215.

Robert, Christian P. *Monte carlo methods*. Wiley Online Library, 2004.

Sagiroglu, Seref e Duygu Sinanc. «Big data: A review». In: *Collaboration Technologies and Systems (CTS), 2013 International Conference on*. IEEE. 2013, pp. 42–47.

Särndal, Carl-Erik, Bengt Swensson e Jan Wretman. *Model assisted survey sampling*. Springer Science & Business Media, 2003.

Scott, David W. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

Scott, Steven L. «Comparing Consensus Monte Carlo Strategies for Distributed Bayesian Computation». In: *Brazilian Journal of Probability and Statistics* (2017).

Scott, Steven L et al. «Bayes and big data: The consensus Monte Carlo algorithm». In: *International Journal of Management Science and Engineering Management* 11.2 (2016), pp. 78–88.

Smith, Paul J. «Statistics: A Bayesian Perspective». In: *The American Statistician* 51.3 (1997), pp. 293–296.

Tran, Minh-Ngoc et al. «Parallel variational Bayes for large datasets with an application to generalized linear mixed models». In: *Journal of Computational and Graphical Statistics* 25.2 (2016), pp. 626–646.

Appendice A

Principale codice R utilizzato

Codice A.1: Codice utilizzato nel Capitolo 4

```
set.seed(111)
y=rbinom(1000,1,0.5)
L=function(x,b=y) sum(dbinom(b,1,x,log=T))
q_=function(a) runif(1,a-v,a+v)
P=function(x,a=1) (dbeta(x, 1,1,log=T))/a
PP=function(x,b=y,a=1) P(x,a)+L(x,b)
set.seed(11)
v=.2
curve(dbeta(x, 1+sum(y),1+1000-sum(y)),xlab="",ylab="")
h=1
x=matrix(NA,10,10001)
x[,1]=.5
ac=nacc=0
a=c()
for(i in 1:10) {
  ac=nacc=0
  y2=y[h:(h+99)]
  for(t in 1:(dim(x)[2]-1)) {
    y1=q_(x[i,t])
    if (runif(1)<exp(PP(y1,y2,1/10)-PP(x[i,t],y2,1/10))) {x[i,t+1]=y1;ac=
        ac+1}
```

```
    else {x[i,t+1]=x[i,t];nacc=nacc+1}
  }
  points(density(x[i,-c(1:5001)]),type="l",col=2)
  h=h+100
  a=c(a,ac/(ac+nacc))
}
a
x=x[,-c(1:5001)]
theta=array(NA,c(1,5000,10))
for(i in 1:10) theta[1,,i]=x[i,]
library(parallelMCMCcombine)
curve(dbeta(x, 1+sum(y),1+1000-sum(y)),xlim=c(0.4,0.6),ylim=c(0,26),
      ylab="",xlab="")
full.theta=consensusMCindep(theta)
points(density(full.theta[1,]),type="l",col=3)
full.theta=semiparamDPE(theta)
points(density(full.theta[1,]),type="l",col=4)
```

Codice A.2: Codice utilizzato nel Capitolo 5

```
set.seed(123)
library(MHadaptive)
li_reg<-function(pars,data)
{
  a<-pars[1]
  sd_e<-pars[2]
  if(sd_e<=0){return(NaN)}
  pred <- a* data[,1]
  log_likelihood<-sum(dnorm(data[,2],pred,sd_e, log=TRUE))
  prior<- prior_reg(pars)
  return(log_likelihood + prior)
}
prior_reg<-function(pars)
{
  a<-pars[1]
  epsilon<-pars[2]
  prior_a<-dnorm(a,0,100,log=TRUE)
```

```
  prior_epsilon<-dgamma(epsilon,1,1/100,log=TRUE)
  return(prior_a  + prior_epsilon)
}
x<-runif(50000,-5,5)
y<-x+rnorm(50000,0,5)
d<-cbind(x,y)
outf<-Metro_Hastings(li_func=li_reg,pars=c(1,5),
                     data=d,burn_in = 10001,it=20000)
li_reg<-function(pars,data){
  a<-pars[1]
  sd_e<-pars[2]
  if(sd_e<=0){return(NaN)}
  pred <- a* data[,1]
  log_likelihood<-sum( dnorm(data[,2],pred,sd_e, log=TRUE) )
  prior<- prior_reg(pars)/10
  return(log_likelihood + prior)
}
d_=d
ac=c()
h=1
z=s=matrix(NA,10,10000)
for(i in 1:10) {print(i)
  d=d_[h:(4999+h),]
  out <- Metro_Hastings(li_func=li_reg,pars=c(1,5),
                        data=d,burn_in = 10001,it=20000)
  ac=c(ac,out$acceptance_rate)
  z[i,]=out$trace[,1]
  s[i,]=out$trace[,2]
  h=h+5000
}
library(parallelMCMCcombine)
theta=array(NA,c(2,10000,10))
for(i in 1:10) {theta[1,,i]=z[i,];theta[2,,i]=s[i,]}
full.theta=consensusMCcov(theta)
full.theta2=semiparamDPE(theta)
```

```
set.seed(123)
library(MHadaptive)
li_reg<-function(pars,data)
{
  a<-pars[1]
  b<-pars[2]
  pred <- exp(a*data[,1]+b*data[,2])/(1+exp(a*data[,1]+b*data[,2]))
  log_likelihood<-sum(dbinom(data[,3],1,pred,log=TRUE))
  prior<- prior_reg(pars)
  return(log_likelihood + prior)
}
prior_reg<-function(pars)
{
  a<-pars[1]
  b<-pars[2]
  prior_a<-dnorm(a,0,100,log=TRUE)
  prior_b<-dnorm(b,0,100,log=TRUE)
  return(prior_a + prior_b)
}
a<-runif(50000,-1,1)
b=rbinom(50000,1,0.5)
b[b==0]=-1
pi<-exp(a+b)/(1+exp(a+b))
y=rbinom(50000,1,pi)
d<-cbind(a,b,y)
outf<-Metro_Hastings(li_func=li_reg,pars=c(1,1),
                     data=d,burn_in = 10001,it=20000)
li_reg<-function(pars,data){
  a<-pars[1]
  b<-pars[2]
  pred <- exp(a*data[,1]+b*data[,2])/(1+exp(a*data[,1]+b*data[,2]))
  log_likelihood<-sum(dbinom(data[,3],1,pred,log=TRUE))
  prior<- prior_reg(pars)/10
  return(log_likelihood + prior)
}
d_=d
```

```
ac=c()
h=1
z=s=matrix(NA,10,10000)
for(i in 1:10) {print(i)
  d=d_[h:(4999+h),]
  out <- Metro_Hastings(li_func=li_reg,pars=c(1,1),
                        data=d,burn_in = 10001,it=20000)
  ac=c(ac,out$acceptance_rate)
  z[i,]=out$trace[,1]
  s[i,]=out$trace[,2]
  h=h+5000
}
library(parallelMCMCcombine)
theta=array(NA,c(2,10000,10))
for(i in 1:10) {theta[1,,i]=z[i,];theta[2,,i]=s[i,]}
full.theta=consensusMCcov(theta)
full.theta2=semiparamDPE(theta)

set.seed(123)
library(MHadaptive)
li_reg<-function(pars,data)
{
  a<-pars[1]
  b<-pars[2]
  pred=exp(a*data[,1]+b*data[,2])
  log_likelihood<-sum(dpois(data[,3],pred,log=TRUE))
  prior<- prior_reg(pars)
  return(log_likelihood + prior)
}
prior_reg<-function(pars)
{
  a<-pars[1]
  b<-pars[2]
  prior_a<-dnorm(a,0,100,log=TRUE)
  prior_b<-dnorm(b,0,100,log=TRUE)
  return(prior_a + prior_b )
```

```
}
a<-runif(50000,0,2)
b=rbinom(50000,1,0.5)
b[b==0]=-1
pi<-exp(a+b)
y=rpois(50000,pi)
d<-cbind(a,b,y)
outf<-Metro_Hastings(li_func=li_reg,pars=c(1,1),
                     data=d,burn_in = 10001,it=20000)
li_reg<-function(pars,data)
{
  a<-pars[1]
  b<-pars[2]
  pred=exp(a*data[,1]+b*data[,2])
  log_likelihood<-sum(dpois(data[,3],pred,log=TRUE))
  prior<- prior_reg(pars)/10
  return(log_likelihood + prior)
}
d_=d
ac=c()
h=1
z=s=matrix(NA,10,10000)
for(i in 1:10) {print(i)
  d=d_[h:(4999+h),]
  out <- Metro_Hastings(li_func=li_reg,pars=c(1,5),
                       data=d,burn_in = 10001,it=20000)
  ac=c(ac,out$acceptance_rate)
  z[i,]=out$trace[,1]
  s[i,]=out$trace[,2]
  h=h+5000
}
library(parallelMCMCcombine)
theta=array(NA,c(2,10000,10))
for(i in 1:10) {theta[1,,i]=z[i,];theta[2,,i]=s[i,]}
full.theta=consensusMCcov(theta)
full.theta2=semiparamDPE(theta)
```

```
set.seed(123)
library(MHadaptive)
data=read.table("skin.txt")
skin=data[c(sample(which(data$V4==1),50000),
            sample(which(data$V4==2),50000)),]
skin$V4=skin$V4-1
names(skin)=c("B","G","R","y")
skin$B=(skin$B-min(skin$B))/max(skin$B)
skin$G=(skin$G-min(skin$G))/max(skin$G)
skin$R=(skin$R-min(skin$R))/max(skin$R)
skin$BG=skin$B*skin$G
skin$BR=skin$B*skin$R
skin$GR=skin$G*skin$R
skin$BRG=skin$B*skin$G*skin$R
library(MHadaptive)
li_reg<-function(pars,data)
{
  a<-pars[1]
  b<-pars[2]
  c=pars[3]
  d=pars[4]
  bg=pars[5]
  br=pars[6]
  gr=pars[7]
  bgr=pars[8]
  pred <- exp(a+b*data[,1]+c*data[,2]+d*data[,3]+
                bg*data[,4]+br*data[,5]
              +gr*data[,6]+bgr*data[,7])/
    (1+exp(a+b*data[,1]+c*data[,2]+d*data[,3]+
             bg*data[,4]+br*data[,5]+
             gr*data[,6]+bgr*data[,7]))
  log_likelihood<-sum(dbinom(data[,8],1,
                             pred,log=TRUE))
  prior<- prior_reg(pars)
  return(log_likelihood + prior)
```

```
}
prior_reg<-function(pars)
{
  a<-pars[1]
  b<-pars[2]
  c=pars[3]
  d=pars[4]
  bg=pars[5]
  br=pars[6]
  gr=pars[7]
  bgr=pars[8]
  prior_a<-dnorm(a,0,100,log=TRUE)
  prior_b<-dnorm(b,0,100,log=TRUE)
  prior_c<-dnorm(c,0,100,log=TRUE)
  prior_d<-dnorm(d,0,100,log=TRUE)
  prior_bg<-dnorm(bg,0,100,log=TRUE)
  prior_br<-dnorm(br,0,100,log=TRUE)
  prior_gr<-dnorm(gr,0,100,log=TRUE)
  prior_bgr<-dnorm(bgr,0,100,log=TRUE)
  return(prior_a + prior_b + prior_c+prior_d+
           prior_bg + prior_br +
           prior_gr+prior_bgr)
}
d<-cbind(skin[,-4],y=skin[,4])
outf<-Metro_Hastings(li_func=li_reg,pars=rep(0,8),
                     data=d,
                     burn_in = 30001,it=60000)
library(coda)
heidel.diag(mcmc.list(mcmc(outf$trace)))
li_reg<-function(pars,data)
{
  a<-pars[1]
  b<-pars[2]
  c=pars[3]
  d=pars[4]
  bg=pars[5]
```

```
  br=pars[6]
  gr=pars[7]
  bgr=pars[8]
  pred <- exp(a+b*data[,1]+c*data[,2]+d*data[,3]+
                bg*data[,4]+br*
                data[,5]+gr*data[,6]+bgr*data[,7])/
    (1+exp(a+b*data[,1]+c*data[,2]+d*data[,3]+
             bg*data[,4]+br*data[,5]
           +gr*data[,6]+bgr*data[,7]))
  log_likelihood<-sum(dbinom(data[,8],1,
                             pred,log=TRUE))
  prior<- prior_reg(pars)/20
  return(log_likelihood + prior)
}
d=cbind(skin[,-4],y=skin[,4])
d_=d[sample(1:100000,100000),]
ac=c()
h=1
z=s=g=u=bgr=gr=bg=br=matrix(NA,20,30000)
for(i in 1:20) {print(i)
  d=d_[h:(4999+h),]
  out <- Metro_Hastings(li_func=li_reg,pars=rep(0,8),
                        data=d,
                        burn_in = 30001,it=60000)
  ac=c(ac,out$acceptance_rate)
  z[i,]=out$trace[,1]
  s[i,]=out$trace[,2]
  g[i,]=out$trace[,3]
  u[i,]=out$trace[,4]
  bg[i,]=out$trace[,5]
  br[i,]=out$trace[,6]
  gr[i,]=out$trace[,7]
  bgr[i,]=out$trace[,8]
  h=h+5000
}
library(parallelMCMCcombine)
```

```
theta=array(NA,c(8,30000,20))
for(i in 1:20) {theta[1,,i]=z[i,];
theta[2,,i]=s[i,];theta[3,,i]=g[i,];theta[4,,i]=u[i,];
theta[5,,i]=bg[i,];theta[6,,i]=br[i,];
theta[7,,i]=gr[i,];theta[8,,i]=bgr[i,]}
full.theta=consensusMCcov(theta)
full.theta2=semiparamDPE(theta)
```