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

INFERENZA DISTRIBUITA IN MODELLI DI REGRESSIONE PER DATI BINARI

Relatore Prof. Nicola Sartori
Dipartimento di Scienze Statistiche

Laureando Mattia Compagno
Matricola 2004211

Anno Accademico 2021/2022

Indice

Introduzione

Al giorno d'oggi è sempre più comune avere a disposizione grandi quantità di dati e questo porta alla nascita di nuove problematiche da affrontare, dovute soprattutto alle limitate risorse computazionali. Ci sono due principali limitazioni per l'analisi dei *big data*: da un lato, la memoria di un singolo computer spesso non è in grado di contenere interamente dati di queste dimensioni; dall'altro, invece, la potenza di calcolo non è sufficiente per ottenere dei risultati in tempo utile. Nella pratica queste situazioni problematiche, a livello di collocazione e di memorizzazione, possono essere gestite in diversi modi in base alla quantità di dati: (i) tutti i dati possono essere contenuti in un'unica macchina; (ii) utilizzo di architetture distribuite sui cui memorizzare le informazioni, spesso anche per motivi di privacy.

In questo lavoro si considereranno entrambi gli scenari descritti ma un maggiore interesse sarà rivolto al contesto distribuito, dove il costo di comunicazione tra macchine assume una notevole importanza. Proprio in quest'ultimo contesto risulta naturale l'utilizzo di un approccio *divide et impera* il quale, se si assume che i dati siano già divisi in gruppi, consiste in (i) ottenere delle statistiche in ciascun gruppo e (ii) combinare le statistiche per ottenere una stima finale. Se queste due fasi vengono effettuate un'unica volta allora il metodo viene definito *one-shot*, se invece vengono ripetute più volte allora l'approccio viene definito iterativo. Sebbene i metodi *one-shot* comportino un minor costo di comunicazione tra le macchine, allo stesso tempo presentano alcuni svantaggi. In particolare, nelle macchine locali deve esserci una numerosità campionaria sufficiente, altrimenti lo stimatore finale non può raggiungere lo stesso livello di precisione dello stimatore globale. Questo impedisce di utilizzare più macchine per velocizzare il tempo di calcolo (Wang et al., 2017). Per ulteriori aspetti negativi degli stimatori *one-shot* si rimanda a Gao et al. (2022). Proprio la presenza di questi svantaggi motivano l'utilizzo di approcci iterativi. A tal proposito, Jordan et al. (2019) definiscono un *framework* per le architetture distribuite denominato *communication-efficient surrogate likelihood* (CSL), la cui implementazione porta alla formulazione di un algoritmo di natura iterativa.

Una possibile alternativa, ideata per un contesto generico di *big-data*, riguarda i metodi di sottocampionamento ottimali, nei quali l'analisi è svolta su un sottoinsieme dei dati, opportunamente scelto. È da tenere in considerazione che tale approccio può venire utilizzato anche quando i dati sono distribuiti ma, comportando necessariamente il trasferimento di intere osservazioni e non di statistiche tra le macchine a disposizione, non può venire utilizzato in contesti con limitazioni dovute alla privacy ed inoltre implica un costo di comunicazione non indifferente qualora la dimensione del sottocampione diventi grande. In particolare Yu et al. (2020) definiscono dei metodi di sottocampionamento che presentano degli aspetti che ne rendono agevole l'implementazione in contesti distribuiti.

In questo lavoro di tesi si propone di confrontare le metodologie proposte da Jordan et al. (2019) e Yu et al. (2020) utilizzandole nella regressione logistica. Ci si focalizzerà su situazioni in cui il numero di variabili d non è sufficientemente elevato da rendere problematica l'inferenza globale ma lo è abbastanza per complicare l'inferenza locale. Sarà quindi d'interesse capire come la relazione tra numerosità campionaria locale e il numero di variabili possa complicare il problema di inferenza. Nel dettaglio si analizzeranno le problematiche che ne derivano nell'approccio CSL e se ne vedrà un'estensione, nota come CEASE (*Communication-Efficient Accurate Statistical Estimation*), proposta da Fan et al. (2021).

Nel primo capitolo vengono presentati i metodi CSL e CEASE, definiti metodi *gradient-enhanced*. Verranno analizzate le problematiche di convergenza dell'approccio CSL e si riporteranno delle analisi, svolte in Fan et al. (2021), che permetteranno di definire l'apporto positivo dell'estensione CEASE nella risoluzione di tali problemi.

Nel secondo capitolo vengono descritti i metodi di sottocampionamento ottimale proposti in Yu et al. (2020). Si presenteranno due metodologie basate su criteri di ottimalità differenti e verrà illustrato anche un possibile algoritmo distribuito, sempre formulato in Yu et al. (2020).

In seguito, nel terzo capitolo vengono riportati i risultati di alcuni studi di simulazione, in cui si confrontano i metodi presentati, con alcune modifiche, sia a livello di stime puntuali che a livello di intervalli di confidenza.

Come già accennato, l'interesse principale in questo lavoro di tesi sarà quello di applicare le diverse metodologie in esame in un modello di regressione logistica per dati binari. In tale modello si assume che ogni componente del vettore $y = (y_1, \ldots, y_n)$ sia una realizzazione di variabili casuali binomiali con numero di prove $m_1, \ldots, m_n$ e probabilità

di successo $\pi_1, \ldots, \pi_n$. Inoltre, si ipotizza che ogni risposta y_i sia associata a un vettore d-dimensionale di covariate $x_i = (x_{i1}, \ldots, x_{id})^T$, per $i = 1, \ldots, n$. Nello svolgimento di questo lavoro le risposte verranno considerate come realizzazioni di variabili casuali bernoulliane e quindi $m_i = 1$, per $i = 1, \ldots, n$. Dunque, un modello di regressione logistica per dati binari ha

$$(y_i|\pi_i) \sim \text{Bin}(1, \pi_i), \quad \pi_i(\theta) = \frac{\exp(x_i^T\theta)}{1 + \exp(x_i^T\theta)}, \quad (i = 1, \ldots, n), \tag{1}$$

dove $\theta = (\theta_1, \ldots, \theta_d)^T$ è un vettore d-dimensionale di coefficienti di regressione ignoti. Si assume che $\mathbf{X}$ sia la matrice di disegno $n \times p$, le cui righe sono $x_1^T, \ldots, x_n^T$, sia di rango pieno. La funzione di log-verosimiglianza è

$$\ell(\theta) = \ell(\theta; y) = \sum_{i=1}^{n} \big\{ y_i(x_i^T\theta) - \log\left[1 + \exp(x_i^T\theta)\right] \big\}, \tag{2}$$

e il suo punto di massimo è la stima di massima verosimiglianza $\hat{\theta}$ di θ. Il gradiente di tale funzione, anche detto *score*, corrisponde a

$$\ell_*(\theta) = \ell_*(\theta; y) = \sum_{i=1}^{n} x_i[y_i - \pi_i(\theta)], \tag{3}$$

mentre la matrice $d \times d$ delle derivate parziali cambiate di segno, detta *informazione osservata di Fisher*, è

$$j(\theta) = j(\theta; y) = \mathbf{X}^T\mathbf{W}\mathbf{X}, \tag{4}$$

dove $\mathbf{W}$ è la matrice diagonale $n \times n$ avente elementi diagonali pari a $\pi_i(\theta)(1 - \pi_i(\theta))$. L'*informazione attesa di Fisher*, $i(\theta)$, corrisponde a

$$i(\theta) = \mathbb{E}_\theta[j(\theta; Y)] = j(\theta), \tag{5}$$

ed è quindi uguale all'informazione osservata, essendo quest'ultima una funzione solo delle variabili esplicative e del parametro θ. Se la stima di massima verosimiglianza esiste finita, essendo essa soluzione consistente dell'equazione di verosimiglianza $\ell_*(\theta) = 0$ ed essendo il campione composto da osservazioni indipendenti, al divergere della numerosità campionaria si ha il seguente risulato asintotico (si veda ad esempio Pace & Salvan, 1997, Paragrafo 3.4.1)

$$i(\theta^*)^{1/2}(\hat{\theta} - \theta^*) \xrightarrow{d} N_d(0, \mathrm{I}_d), \tag{6}$$

dove $\xrightarrow{d}$ indica convergenza in distribuzione, θ^* è il vero valore del parametro e $i(\theta^*)^{1/2}$

è una matrice $d \times d$ tale che

$$i(\theta^*)^{1/2} i(\theta^*)^{1/2} = i(\theta^*).$$

Siccome il contesto di riferimento è quello distribuito, è essenziale tenere in considerazione il comportamento delle stime di massima verosimiglianza quando il rapporto tra il numero di variabili e il numero di osservazioni cresce, fatto che può accadere quando i dati vengono suddivisi in un numero elevato di sottoinsiemi o se il numero di osservazioni a disposizione non è molto alto. Nei modelli di regressione logistica infatti, sfortunatamente, le stime di massima verosimiglianza potrebbero non esistere, comportando che almeno una delle componenti del vettore $\hat{\theta}$ sia infinita. È noto che questo fenomeno accade in situazioni di perfetta o quasi-perfetta separazione, ossia quando esiste un iperpiano che permette di individuare due regioni in $\mathbb{R}^d$, tali che in ognuna di esse sono situate osservazioni appartenenti a una sola delle classi definite dalla variabile risposta o al massimo alcune osservazioni sono situate esattamente sull'iperpiano (si veda Pace & Salvan, 1997, Paragrafo 6.5). In caso di presenza di una situazione di separazione, le procedure inferenziali standard potrebbero non convergere, producendo delle previsioni degeneri per le probabilità di successo e delle conclusioni inferenziali inaffidabili. In Candès & Sur (2020) viene caratterizzata l'esistenza di stime finite, assumendo che d cresca con il numero di osservazioni n, relazionandola a delle quantità essenziali per la sua determinazione. Ciò che ne risulta è quanto questo fenomeno possa essere frequente nel contesto di architetture distribuite. Per questo motivo si rivela di cruciale importanza la scelta di un metodo di stima che garantisca sempre l'esistenza di stime finite o che permetta di ottenerle più frequentemente della stima di massima verosimiglianza. A tal proposito, in Firth (1993) viene mostrato come il punto di massimo di un'adeguata verosimiglianza penalizzata, ottenuta considerando la penalità derivante dall'utilizzo della priori di Jeffrey, abbia una distorsione asintotica minore rispetto allo stimatore di massima verosimiglianza. Inoltre Kosmidis & Firth (2021) dimostrano che la correzione di Firth produce stime sempre finite permettendo di risolvere il problema della separabilità. La log-verosimiglianza penalizzata di Firth è così definita

$$\ell^F(\theta) = \ell^F(\theta; y) = \ell(\theta) + \frac{1}{2} \log |i(\theta)|, \tag{7}$$

dove $|A|$ corrisponde al determinante della matrice A. Lo *score* invece presenta la seguente modifica

$$\ell^F_*(\theta) = \ell^F_*(\theta; y) = \ell_*(\theta) - \mathbf{X}^T \mathbf{W} \xi, \tag{8}$$

con componente r-esima

$$\begin{aligned} \ell^F_{*,r}(\theta) &= \ell_{*,r}(\theta) + \frac{1}{2}\frac{\partial}{\partial\theta_r}\log|i(\theta)| \\ &= \sum_{i=1}^{n}\{(y_i + h_i/2) - (1+h_i)\pi_i\}x_{i,r}, \end{aligned} \tag{9}$$

dove $\mathbf{W}\xi$ è un vettore n-dimensionale con i-esimo elemento $h_i(\pi_i - \frac{1}{2})$ e h_i è l'i-esimo elemento diagonale della matrice

$$\mathbf{H} = \mathbf{W}^{1/2}\mathbf{X}(\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}^{1/2}. \tag{10}$$

Per quanto riguarda la consistenza e la normalità dello stimatore $\hat{\theta}^F = \arg\max_{\theta\in\mathbb{R}^d}\ell^F_*(\theta)$, se valgono tali proprietà per lo stimatore di massima verosimiglianza, allora si può affermare che (si veda Kosmidis, 2007, Paragrafi 6.3, 6.4 e 6.5)

$$i(\theta^*)^{1/2}(\hat{\theta}^F - \theta^*) \xrightarrow{d} N_d(0, \mathbf{I}_d), \tag{11}$$

e per questo motivo non si considererà tale correzione nella definizione dell'informazione attesa, la quale verrà presa uguale a quella ottenuta con la verosimiglianza non penalizzata.

Visto l'utilità della correzione di Firth, un ulteriore obiettivo di questo lavoro sarà quello di proporne un suo utilizzo all'interno degli approcci presentati nel tentativo di migliorarne le performance.

Capitolo 1

Metodi *gradient-enhanced*

In questo capitolo verranno presentati dei metodi iterativi basati su funzioni di perdita *gradient enhanced*. Tali approcci sono pensati per contesti distribuiti ma talvolta possono essere sfruttati e portano benifici in termini di tempo di calcolo e di memoria occupata anche se i dati, seppur di grandi dimensioni, sono presenti in un'unica macchina, in cui è possibile ricorrere al calcolo parallelo.

Considerando quindi che i dati siano distribuiti in più macchine, è necessario formulare il problema definendo una notazione opportuna (si veda Jordan et al., 2019, Paragrafo 2.1). Siano k e n rispettivamente il numero di macchine a disposizione e il numero di osservazioni contenute in ciascuna di esse. Dunque, per semplicità si considera che i dati siano suddivisi in sottogruppi di uguale dimensione. Si consideri $Z_1^N = \{Z_{ij} : i = 1, \dots, n,\ j = 1, \dots, k\}$ come l'insieme rappresentante la totalità di dati a disposizione, contenente $N = nk$ variabili indipendenti e identicamente distribuite con distribuzione marginale $\mathbb{P}_{\theta^*}$, dove $\{\mathbb{P}_\theta : \theta \in \Theta\}$ è una famiglia di modelli statistici parametrici con parametro $\theta \in \Theta \subset \mathbb{R}^d$, con Θ spazio parametrico e θ^* vero valore del parametro del modello da cui provengono i dati. Si definisce inoltre con $Z_j = \{Z_{ij} : i = 1, \dots, n\}$ il sottocampione contenuto nella macchina j-esima $\mathcal{M}_j$, con $j = 1, \dots, k$.

Sia $\mathcal{L} : \Theta \times \mathcal{Z} \mapsto \mathbb{R}$ una funzione di perdità differenziabile due volte, anche detta funzione di rischio empirico, e tale che il vero valore del parametro sia il punto di minimo della funzione di rischio della popolazione $\mathcal{L}^*(\theta) = \mathbb{E}_{\theta^*}[\mathcal{L}(\theta; Z)]$, ossia

$$\theta^* \in \arg\min_{\theta \in \Theta} \mathbb{E}_{\theta^*}[\mathcal{L}(\theta; Z)]. \tag{1.1}$$

Si definiscono rispettivamente le funzioni di rischio empirico locale e globale come

$$\mathcal{L}_j(\theta) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(\theta; z_{ij}), \quad \text{per } j = 1, \ldots, k \tag{1.2}$$

$$\mathcal{L}_N(\theta) = \frac{1}{N}\sum_{j=1}^{k}\sum_{i=1}^{n}\mathcal{L}(\theta; z_{ij}) = \frac{1}{k}\sum_{j=1}^{k}\mathcal{L}_j(\theta). \tag{1.3}$$

È da notare che le due funzioni di perdita appena specificate, hanno il codominio sulla stessa scala dato che vengono divise per il numero di osservazioni che le compongono. Inoltre, si osservi che $\mathcal{L}_j(\theta)$ viene valutata in θ usando solo i dati contenuti nella macchina $\mathcal{M}_j$. Un esempio standard di funzione di rischio empirico $\mathcal{L}$ non è altro che l'opposto della funzione di log-verosimiglianza divisa per il numero di osservazioni.

Si ricordi che l'obiettivo è quello di fare inferenza sul parametro θ, fornendo uno stimatore puntuale ed eventualmente degli intervalli di confidenza associati alle sue componenti, tenendo conto del contesto distribuito di riferimento.

1.1 *Communication-efficient surrogate likelihood*

La metodologia presentata in questo paragrafo rientra nella classe dei metodi iterativi proposti per l'inferenza statistica distribuita e in particolare si colloca all'interno degli approcci che mirano ad ottenere un basso costo di comunicazione tra le macchine. Shamir et al. (2014) hanno proposto un approccio Newton approssimato che non necessitasse di trasferire le matrici hessiane. Seguendo proprio questa strategia, Jordan et al. (2019) hanno sviluppato un approccio basato su una verosimiglianza approssimata.

Gran parte della trattazione che segue verte sull'articolo di Jordan et al. (2019), al quale si rimanda per ulteriori dettagli ed approfondimenti.

1.1.1 Definizione

Per risolvere problemi di inferenza statistica distribuita, Jordan et al. (2019) propongono un *framework* nominato *communication-efficient surrogate likelihood* (CSL), il quale si basa sulla definizione di un surrogato della funzione di log-verosimiglianza globale, o in generale della funzione di perdita globale, la cui implementazione risulta essere efficiente dal punto di vista del costo di comunicazione tra le macchine a disposizione.

La loro idea parte dallo sviluppo in serie di Taylor di ordine infinito della funzione di rischio empirico globale $\mathcal{L}_N$:

$$\mathcal{L}_N(\theta) = \mathcal{L}_N(\bar{\theta}) + \langle \nabla \mathcal{L}_N(\bar{\theta}), \theta - \bar{\theta} \rangle + \sum_{j=2}^{\infty} \frac{1}{j!} \nabla^j \mathcal{L}_N(\bar{\theta})(\theta - \bar{\theta})^{\otimes j}. \tag{1.4}$$

Il valore $\bar{\theta}$ in cui si effettua lo sviluppo è uno stimatore iniziale di θ, come ad esempio lo stimatore ottenuto minimizzando la funzione di rischio empirico locale $\mathcal{L}_1$ definita nella prima macchina $\mathcal{M}_1$. Essendo i dati distribuiti in più macchine, ogni derivata $\nabla^j \mathcal{L}_N(\bar{\theta})$ di ordine $j > 1$ può essere calcolata con un solo ciclo di comunicazione tra le macchine. È da notare però che, tralasciando il gradiente $\nabla \mathcal{L}_N(\bar{\theta})$, per ogni macchina trasmettere le derivate di ordine superiore comporta un costo di comunicazione maggiore di $O(d^2)$. Per questo motivo si suggerisce di sostituire tali derivate di ordine $j \geq 2$ globali con le corrispondenti derivate calcolate a livello locale nella prima macchina. Così facendo e utilizzando lo sviluppo in serie di Taylor di $\mathcal{L}_1(\theta)$ in $\bar{\theta}$ è possibile ottenere la formulazione

$$\tilde{\mathcal{L}}(\theta) = \mathcal{L}_1(\theta) - \langle \nabla \mathcal{L}_1(\bar{\theta}) - \nabla \mathcal{L}_{\mathcal{N}}(\bar{\theta}), \theta \rangle \,. \tag{1.5}$$

La funzione appena ottenuta $\tilde{\mathcal{L}}(\theta)$ viene quindi considerata come un surrogato della funzione di perdita globale. Essa può anche essere vista come una funzione di perdita *gradient-enhanced*, in cui il gradiente in $\bar{\theta}$ calcolato con i dati locali è sostituito da quello globale. Definendo $\hat{\theta}$ come il punto di minimo della funzione di perdita globale,

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \mathcal{L}_N(\theta),$$

si può vedere facilmente che il punto di minimo di $\tilde{\mathcal{L}}$, con $\bar{\theta} = \hat{\theta}$, corrisponde proprio a $\hat{\theta}$, il che chiaramente è una proprietà auspicabile per tale surrogato.

Nel *framework* definito, una scelta naturale per definire un sostituto dello stimatore globale $\hat{\theta}$ è quello di minimizzare $\tilde{\mathcal{L}}$ anzichè $\mathcal{L}_N(\theta)$, ottenendo così lo stimatore

$$\tilde{\theta} = \arg\min_{\theta \in \Theta} \tilde{\mathcal{L}}(\theta). \tag{1.6}$$

1.1.2 *Iterative local estimation algorithm*

Il problema di minimo riportato in (1.6) può essere risolto esattamente tramite algoritmi di minimizzazione numerica, come ad esempio *Newton-Raphson*, oppure è possibile ridurre la complessità computazionale considerando un'approssimazione quadratica locale di $\mathcal{L}$. Ragionando in modo analogo a quanto fatto per la definizione di $\tilde{\mathcal{L}}$, si riesce

ad ottenere il seguente surrogato quadratico:

$$\tilde{\mathcal{L}}^H(\theta) = \langle \nabla \mathcal{L}_N(\bar{\theta}), \theta - \bar{\theta} \rangle + \frac{1}{2}(\theta - \bar{\theta})^T \nabla^2 \mathcal{L}_1(\bar{\theta}) (\theta - \bar{\theta}). \tag{1.7}$$

L'utilizzo di $\tilde{\mathcal{L}}^H$ come funzione obiettivo, porta allo stimatore in forma chiusa

$$\tilde{\theta}^H = \arg\min_{\theta \in \Theta} \tilde{\mathcal{L}}^H(\theta) = \bar{\theta} - \nabla^2 \mathcal{L}_1(\bar{\theta})^{-1} \nabla \mathcal{L}_N(\bar{\theta}), \tag{1.8}$$

anche detto stimatore *one-step* basato sull'approssimazione quadratica.

Come verrà analizzato nel seguito di questo lavoro di tesi, l'utilizzo di un approccio iterativo permette di ridurre l'errore di approssimazione $\|\tilde{\theta} - \hat{\theta}\|_2$. L'algoritmo che ne deriva viene detto *iterative local estimation algorithm* (ILEA). Per una descrizione dettagliata di questa procedura si veda l'Algoritmo 1.1 riportato di seguito.

Algorithm 1.1: *Iterative local estimation*

Input: valore iniziale $\theta^{(0)} = \bar{\theta}$, numero di iterazioni T

for $t = 0, 1, \ldots, T-1$ **do**
- Trasmissione di $\theta^{(t)}$ alle macchine locali $\{\mathcal{M}_j\}_{j=1}^k$;
- **for** $j = 1, \ldots, k$ **do**
  - Calcolo del gradiente locale $\nabla \mathcal{L}_j(\theta^{(t)})$ nella macchina $\mathcal{M}_j$;
  - Trasmissione del gradiente locale $\nabla \mathcal{L}_j(\theta^{(t)})$ alla macchina $\mathcal{M}_1$;
- Calcolo del gradiente globale $\nabla \mathcal{L}_N(\theta^{(t)}) = \frac{1}{k} \sum_{j=1}^k \nabla \mathcal{L}_j(\theta^{(t)})$ nella macchina $\mathcal{M}_1$;
- Costruzione del surrogato $\tilde{\mathcal{L}}^t(\theta) = \mathcal{L}_1(\theta) - \langle \nabla \mathcal{L}_1(\theta^{(t)}) - \nabla \mathcal{L}_N(\theta^{(t)}), \theta \rangle$;
- Svolgere uno dei seguenti aggiornamenti nella macchina $\mathcal{M}_1$

  (1) $\theta^{(t+1)} \in \arg\min_{\theta \in \Theta} \tilde{\mathcal{L}}^t(\theta)$; `// Minimizzazione esatta di` $\tilde{\mathcal{L}}$

  (2) $\theta^{(t+1)} = \theta^{(t)} - \nabla^2 \mathcal{L}_1(\theta^{(t)})^{-1} \nabla \mathcal{L}_N(\theta^{(t)})$; ; `// Approssimazione quadratica one-step`

Output: $\theta^{(T)}$

In ogni iterazione dell'algoritmo ILEA viene costruito il surrogato della funzione di perdita $\tilde{\mathcal{L}}^{(t)}(\theta)$ nella stima corrente $\theta^{(t)}$, il quale, come detto in precedenza, può essere minimizzato esattamente o tramite (1.8) per ottenere la stima successiva $\theta^{(t+1)}$. Le due tipologie di aggiornamento, esatto o approssimato, verranno confrontate e approfondite empiricamente tramite uno studio di simulazione nel Capitolo 3.

Focalizzandosi sui diversi passaggi dell'Algoritmo 1.1, è possibile stabilire che il costo di comunicazione di una singola iterazione risulta essere pari a $2kd$: trasmissione del valore corrente $\theta^{(t)}$ e dei k gradienti locali d-dimensionali attraverso le k macchine. Tale costo, dell'ordine di $O(dk)$, è sicuramente contenuto ed è chiaramente uno degli

obiettivi a cui tale metodologia mira. Allo stesso tempo però si può osservare che il metodo in analisi è asimmetrico, in quanto il surrogato della funzione di perdita viene costruito in una sola macchina e l'utilizzo dei dati immagazzinati nelle macchine non è uguale. Si potrebbe quindi pensare di condurre la stessa procedura in ognuna delle macchine e successivamente effettuare una media delle stime trovate localmente. Questa alternativa simmetrica è riportata dettagliatamente nell'Algoritmo 1.2, in cui $\tilde{\mathcal{L}}_j^t(\theta)$ è il surrogato della funzione di perdita ottenuto nella stima $\theta^{(t)}$ con i dati contenuti nella macchina $\mathcal{M}_j$, mentre $\theta_j^{(t+1)}$ non è altro che la stima ottenuta minimizzando quest'ultima funzione. Un aspetto da considerare dell'algoritmo appena presentato, riguarda sicuramente l'aumento del costo di comunicazione rispetto alla precedente proposta. Infatti in questo caso risulta essere uguale a $4kd$: oltre alle due fasi di comunicazione presenti nell'Algoritmo 1.1, è necessaria anche la trasmissione del gradiente globale alle macchine locali, le quali calcoleranno le loro stime locali che a loro volta dovranno essere comunicate a una macchina (da noi sempre considerata la prima per convenzione) che ne andrà a fare una media. Il costo di comunicazione risulta quindi essere il doppio che in precedenza, ma a tal proposito è opportuno considerare che l'Algoritmo 1.2 permette di sfruttare pienamente le risorse computazionali a disposizione, dato che ogni macchina ha un problema di ottimizzazione da risolvere. Altro aspetto da tenere a mente è che l'onere computazionale del calcolo della media delle stime locali è decisamente minore rispetto a quello di minimizzare la funzione surrogata. Questo comporta che il tempo di calcolo di un'iterazione non sia necessariamente il doppio e che quindi la convenienza nell'utilizzare questa proposta simmetrica dipenda dal miglioramento nell'accuratezza apportato dall'operazione di media. Ci si soffermerà su questo aspetto nei Paragrafi 1.2.2 e 1.2.3.

L'utilizzo dell'Algoritmo 1.1 in ogni caso presenta diversi aspetti positivi, i quali hanno orientato Jordan et al. (2019) ad approfondire principalmente tale metodologia all'interno del loro articolo. In primo luogo, avere a disposizione un solo surrogato della funzione di perdita consente facilmente di formulare e adattare problemi di stima e di inferenza in un contesto distribuito. Risulta infatti naturale il suo impiego all'interno di un contesto bayesiano e allo stesso tempo, come vedremo nel paragrafo successivo, è altrettanto intuitivo pensare a una possibile definizione degli intervalli di confidenza nell'ambito frequentista. Un ultimo aspetto riguarda l'utilizzo di tale metodo in un contesto non distribuito, in cui la dimensione campionaria è così elevata che la valutazione della funzione di perdita globale risulterebbe essere eccessivamente onerosa. Dopo la costruzione di $\tilde{\mathcal{L}}(\theta)$, quest'ultima può essere valutata utilizzando una piccola porzione dei dati, prospettiva che anche nel caso simmetrico comporterebbe la scelta di uno dei k

Algorithm 1.2: *Symmetric iterative local estimation*

Input: valore iniziale $\theta^{(0)} = \bar{\theta}$, numero di iterazioni T

for $t = 0, 1, \ldots, T-1$ **do**
- Trasmissione di $\theta^{(t)}$ alle macchine locali $\{\mathcal{M}_j\}_{j=1}^k$;
- **for** $j = 1, \ldots, k$ **do**
  - Calcolo del gradiente locale $\nabla\mathcal{L}_j(\theta^{(t)})$ nella macchina $\mathcal{M}_j$;
  - Trasmissione del gradiente locale $\nabla\mathcal{L}_j(\theta^{(t)})$ alla macchina $\mathcal{M}_1$;
- Calcolo del gradiente globale $\nabla\mathcal{L}_N(\theta^{(t)}) = \frac{1}{k}\sum_{j=1}^k \nabla\mathcal{L}_j(\theta^{(t)})$ nella macchina $\mathcal{M}_1$;
- Trasmissione del gradiente globale $\nabla\mathcal{L}_N(\theta^{(t)})$ alle altre macchine $\{\mathcal{M}_j\}_{j=2}^k$;
- **for** $j = 1, \ldots, k$ **do**
  - Costruzione del surrogato $\tilde{\mathcal{L}}_j^t(\theta) = \mathcal{L}_j(\theta) - \langle \nabla\mathcal{L}_j(\theta^{(t)}) - \nabla\mathcal{L}_{\mathcal{N}}(\theta^{(t)}), \theta\rangle$;
  - Svolgere uno dei seguenti aggiornamenti nella macchina $\mathcal{M}_j$

    (1) $\theta_j^{(t+1)} \in \arg\min_{\theta\in\Theta} \tilde{\mathcal{L}}_j^t(\theta)$; `// Minimizzazione esatta di` $\tilde{\mathcal{L}}_j$

    (2) $\theta_j^{(t+1)} = \theta^{(t)} - \nabla^2\mathcal{L}_j(\theta^{(t)})^{-1}\nabla\mathcal{L}_N(\theta^{(t)})$; `// Approssimazione quadratica` *`one-step`*

  - Trasmissione della stima $\theta_j^{(t+1)}$ alla macchina $\mathcal{M}_1$;
- Calcolo della media $\theta^{(t+1)} = \frac{1}{k}\sum_{j=1}^k \theta_j^{(t+1)}$ nella macchina $\mathcal{M}_1$;
- Trasmissione di $\theta^{(t+1)}$ alle altre macchine locali $\{\mathcal{M}_j\}_{j=2}^k$;

Output: $\theta^{(T)}$

surrogati a disposizione. L'alternativa simmetrica potrebbe quindi essere utilizzata per ottenere una stima puntuale ma per le rimanenti procedure inferenziali sarebbe necessario considerare un unico sottogruppo di riferimento, in modo tale da evitare eccessivi costi di comunicazione.

1.1.3 Intervalli di confidenza

L'interesse di questo paragrafo è quello di definire delle proprietà e delle quantità utili ai fini dello svolgimento delle usuali procedure inferenziali, quali gli intervalli di confidenza e test d'ipotesi. Lo scopo naturalmente è quello di poter utilizzare questi strumenti basandosi sulla funzione surrogata $\tilde{\mathcal{L}}(\theta)$, la quale utilizza solo il sottocampione $\{z_{i1}\}_{i=1}^n$ nella macchina $\mathcal{M}_1$.

Per i risultati che verranno presentati nel seguito, sono necessarie alcune assunzioni (si veda Jordan et al., 2019, Assunzioni PA-PD):

1. una prima assunzione relativa allo spazio parametrico Θ e alla sua relazione con il vero valore del parametro θ^*;

2. una seconda riguardante la convessità locale della funzione di rischio della popolazione $\mathcal{L}^*$, ossia una condizione di identificabilità locale;

3. una terza inerente l'identificabilità a livello globale;

4. un'ultima che permette di controllare i momenti delle derivate di ordine elevato della funzione di perdita.

Sotto le condizioni appena elencate, lo stimatore globale $\hat{\theta}$, trovato minimizzando $\mathcal{L}(\theta)$, soddisfa

$$\sqrt{N}(\hat{\theta} - \theta^*) \to \mathcal{N}_d(0, \Sigma) \quad \text{in distribuzione con } N \to \infty, \tag{1.9}$$

dove $\Sigma = I(\theta^*)^{-1}\mathbb{E}[\nabla\mathcal{L}(\theta^*; Z)\nabla\mathcal{L}(\theta^*; Z)^T]I(\theta^*)$ è la nota matrice di varianza e covarianza *sandwich* e $I(\theta) = \nabla^2\mathcal{L}^*(\theta)$. Si noti che nel caso in cui la funzione obiettivo sia l'opposto della log-verosimiglianza, $I(\theta)$ corrisponderebbe all'informazione attesa di Fisher $i(\theta)$ divisa per il numero di osservazioni ed inoltre $\Sigma = I(\theta^*)^{-1}$.

Sempre assumendo valide le assunzioni riportate in precedenza, se lo stimatore iniziale $\bar{\theta}$ soddisfa $\|\bar{\theta} - \theta^*\|_2 = O_p(n^{-1/2})$, allora la distribuzione asintotica dello stimatore $\tilde{\theta}$, ottenuto o tramite minimizzazione esatta o tramite approssimazione quadratica, corrisponde con quella dello stimatore globale $\hat{\theta}$, ossia

$$\sqrt{N}(\tilde{\theta} - \theta^*) \to \mathcal{N}_d(0, \Sigma) \quad \text{in distribuzione con } N \to \infty.$$

Risulta quindi d'interesse ottenere una stima di Σ e a tal proposito in Jordan et al. (2019) vengono proposti due possibili stimatori:

- uno stimatore *plug-in* locale,

$$\tilde{\Sigma} = \nabla^2\tilde{\mathcal{L}}(\tilde{\theta})^{-1}\left(\frac{1}{n}\sum_{i=1}^{n}\nabla\tilde{\mathcal{L}}(\tilde{\theta}; z_{i1})\nabla\tilde{\mathcal{L}}(\tilde{\theta}; z_{i1})^T\right)\nabla^2\tilde{\mathcal{L}}(\tilde{\theta})^{-1}; \tag{1.10}$$

- uno stimatore *plug-in* globale,

$$\tilde{\Sigma}' = \nabla^2\tilde{\mathcal{L}}(\tilde{\theta})^{-1}\left(\frac{n}{k}\sum_{j=1}^{k}\nabla\mathcal{L}_j(\tilde{\theta})\nabla\mathcal{L}_j(\tilde{\theta})^T\right)\nabla^2\tilde{\mathcal{L}}(\tilde{\theta})^{-1}. \tag{1.11}$$

Entrambi gli stimatori appena presentati sono consistenti. Si noti che quando $\mathcal{L}(\theta)$ è l'opposto della funzione di log-verosimiglianza, si può utilizzare $\nabla^2\tilde{\mathcal{L}}(\tilde{\theta})^{-1}$ come stimatore *plug-in* di Σ.

A questo punto la specificazione di intervalli di confidenza diventa immediata. Un intervallo di confidenza di livello $1-\alpha$ per l'i-esimo parametro θ_i è dato da

$$\left(\tilde{\theta}_i - Z_{1-\alpha/2}\sqrt{\tilde{\sigma}^2_{i,i}/N},\ \tilde{\theta}_i + Z_{1-\alpha/2}\sqrt{\tilde{\sigma}^2_{i,i}/N}\right), \tag{1.12}$$

dove $Z_{1-\alpha/2}$ è il quantile di ordine $1-\alpha/2$ di una variabile normale standard e $\sigma^2_{i,i}$ è l'elemento di posizione (i,i) di una delle due possibili stime di Σ, (1.10) o (1.11).

1.1.4 Problemi di convergenza dell'algoritmo

Per come è definito il metodo CSL, è necessario che i dati siano distribuiti in modo omogeneo tra le macchine. Infatti, se ci si focalizza sulla costruzione del surrogato $\tilde{\mathcal{L}}$, è naturale richiedere che i dati presenti nella macchina centrale, da noi sempre considerata la prima, siano un campione estratto uniformemente dall'intero insieme di dati a disposizione. Stando a ciò, i dati nelle altre macchine possono anche essere suddivisi in modo arbitrario e non necessariamente uniforme se si considera l'Algoritmo 1.1, se invece si vuole utilizzare l'alternativa simmetrica descritta nell'Algoritmo 1.2, allora è preferibile una distribuzione omogenea dei dati in tutte le macchine.

L'omogeneità dei dati viene quindi richiesta affinché la funzione di perdita locale assomigli il più possibile a quella globale. Tale condizione però non è sufficiente per garantire questa somiglianza. Infatti un altro fattore cruciale da considerare è senz'altro la numerosità campionaria presente in ogni macchina, ossia n. Come si vedrà in seguito nell'Esempio 1.1.1, se n non è sufficientemente elevato rispetto al numero d di parametri da stimare, l'Algoritmo 1.1 potrebbe riscontrare dei problemi di convergenza e portare la stima ad oscillare verso valori estremi dello spazio parametrico.

Purtroppo questo tipo di problematiche si riscontrano anche quando la numerosità campionaria non è poi così esigua ed inoltre non è possibile proporre per quest'ultima una soglia necessaria di riferimento, poiché essa dipenderebbe da parametri strutturali relativi alla funzione obiettivo globale, che nel contesto considerato è impossibile conoscere o definire. Tali parametri verranno presentati nel Paragrafo 1.2 e si potrà notare come l'aspetto più cruciale della somiglianza delle funzioni obiettivo, riguarda il loro hessiano piuttosto che le altre derivate di ordine superiore. Proprio per questo motivo, la non convergenza si presenta sia quando si ottimizza esattamente $\tilde{\mathcal{L}}(\theta)$ sia quando la si minimizza tramite una sua approssimazione quadratica $\tilde{\mathcal{L}}^H(\theta)$.

L'Esempio 1.1.1 riportato di seguito, mira proprio a mettere in evidenza tali problematiche prendendo in considerazione un problema di regressione per dati binari, essendo

il caso di maggiore interesse in questo lavoro di tesi, con solo due variabili esplicative, in modo tale da rendere possibili delle semplici rappresentazioni grafiche.

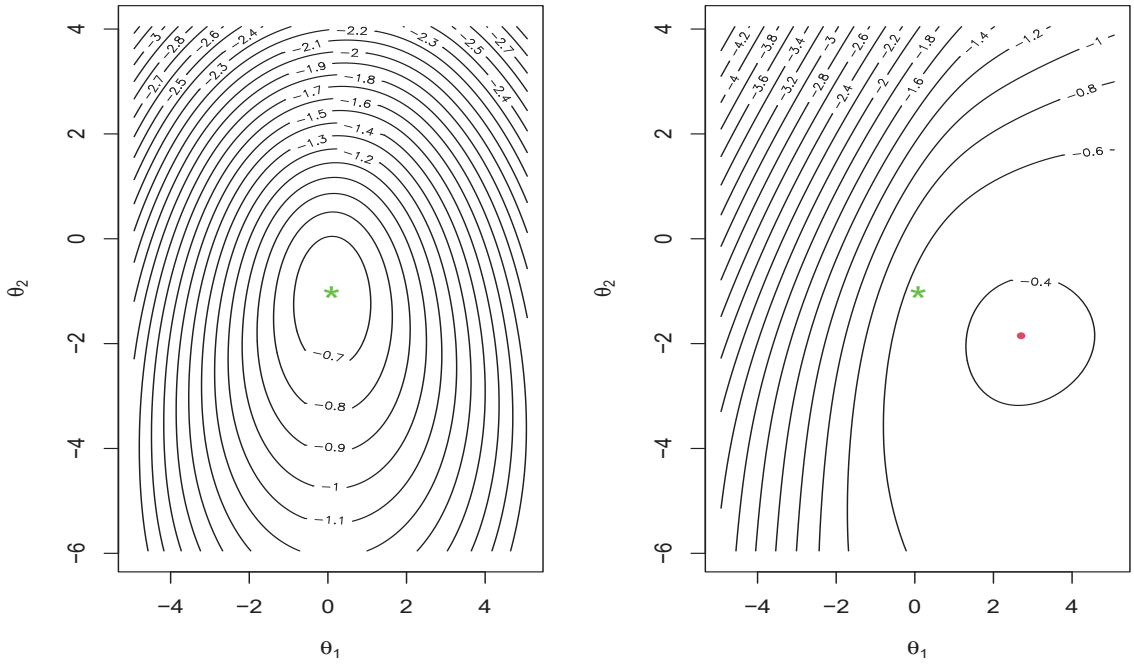
Esempio 1.1.1. Si consideri un modello di regressione logistica (si veda la formula (1) per la sua specificazione) con osservazioni indipendenti $z_1^N = \{z_{ij} = (x_{ij}, y_{ij}) : i = 1, \dots, n,\ j = 1, \dots, k\}$, dove $n = 100$ e $k = 10$ e y_{ij} corrisponde alla realizzazione di una variabile casuale bernoulliana Y_{ij}. Il vero valore del parametro θ^* è un vettore bidimensionale generato uniformemente nel cubo di lato di lunghezza due $[-1, 1]^2$, mentre il vettore delle covariate x_{ij} coincide con la realizzazione di una variabile causale $\mathcal{N}_2(0, \mathbf{I}_2)$. In particolare, nella simulazione presa in considerazione $\theta^* = (0.072, -0.954)$. Come specificato all'inizio di questo paragrafo, la suddivisione in k gruppi è stata effettuata in modo uniforme, cercando di avere una distribuzione omogenea dei dati. Il primo gruppo Z_1 nella macchina $\mathcal{M}_1$ è stato preso come riferimento per l'implementazione della metodologia CSL.

Un primo controllo grafico è riportato in Figura 1.1, in cui si rappresentano $-\mathcal{L}_N(\theta)$ e $-\mathcal{L}_1(\theta)$ che, in questo caso, corrispondono rispettivamente alle log-verosimiglianze globale e locale, entrambe divise per il numero di osservazioni utilizzate e quindi con valori del codominio nella stessa scala. La stima globale $\hat{\theta}$ è segnata in verde mentre la stima locale $\tilde{\theta}$ in rosso. Risulta evidente come queste due funzioni non siano del tutto simili tra di loro, in particolare in corrispondenza dei loro massimi. La versione locale infatti sembra essere molto più piatta vicino al suo massimo e con una forma decisamente meno regolare rispetto alla sua controparte globale.

A questo punto si è proceduto ad utilizzare l'Algoritmo 1.1, considerando inizialmente l'aggiornamento *one-step* (1.8). Come valore iniziale si è deciso di utilizzare la media degli stimatori locali $\hat{\theta}_j = \arg\min_{\theta \in \Theta} \mathcal{L}_j(\theta)$ con $j = 1, \dots, k$, cioè

$$\bar{\theta} = 1/k \sum_{j=1}^{k} \hat{\theta}_j,$$

in modo tale da proporre una buona inizializzazione dell'algoritmo e favorirne la convergenza. In Figura 1.2 è rappresentato il surrogato quadratico della log-verosimiglianza $-\tilde{\mathcal{L}}^H(\theta)$ nelle prime quattro iterazioni dell'algoritmo. Si noti come, iterazione dopo iterazione, la stima corrente $\theta^{(t)}$ (in rosso) si allontana dalla stima globale (in verde), oscillando sempre di più verso valori estremi. Sembra quasi che la derivata seconda calcolata localmente costringa il gradiente globale ad annullarsi troppo lentamente, causando tale comportamento oscillatorio.



(a) Log-verosimiglianza globale $-\mathcal{L}_N$

(b) Log-verosimiglianza locale $-\mathcal{L}_1$

FIGURA 1.1: Confronto, tramite curve di livello, tra log-verosimiglianza globale e locale, entrambe suddivise per il numero di osservazioni utilizzate, in linea con le Formule (1.2) e (1.3).

Per completezza si è implementato l'Algoritmo 1.1 considerando in questo caso l'aggiornamento esatto (1.6) (si veda la Figura 1.3). La performance dell'algoritmo non sembra migliorare e il comportamento risultante è lo stesso individuato in precedenza. Infatti, come già accennato, il problema è principalmente dovuto alla diversità tra l'hessiano locale e quello globale.

Dopo aver individuato tali problemi di convergenza, si è pensato di risolverli apportando alcune modifiche alla definizione del surrogato della log-verosimiglianza, focalizzandosi sul caso della regressione logistica. L'idea principale delle modifiche proposte è stata quella di cercare di introdurre un certo grado di regolarizzazione nella funzione di log-verosimiglianza locale, in modo tale da permettere a quest'ultima di avere una forma più simile a quella della sua versione globale risentendo meno della bassa numerosità campionaria locale. Si mostrano nel seguito le modifiche formulate, dove per convenienza si considera come funzione obiettivo standard di riferimento, indicata con $\mathcal{L}(\theta)$, l'opposto della log-verosimiglianza divisa per la numerosità delle osservazioni utilizzate per definirla:

1. Una prima proposta è stata quella di introdurre la correzione di Firth (1993) nella definzione delle log-verosimiglianze locali. In questo modo, facendo riferimento alla

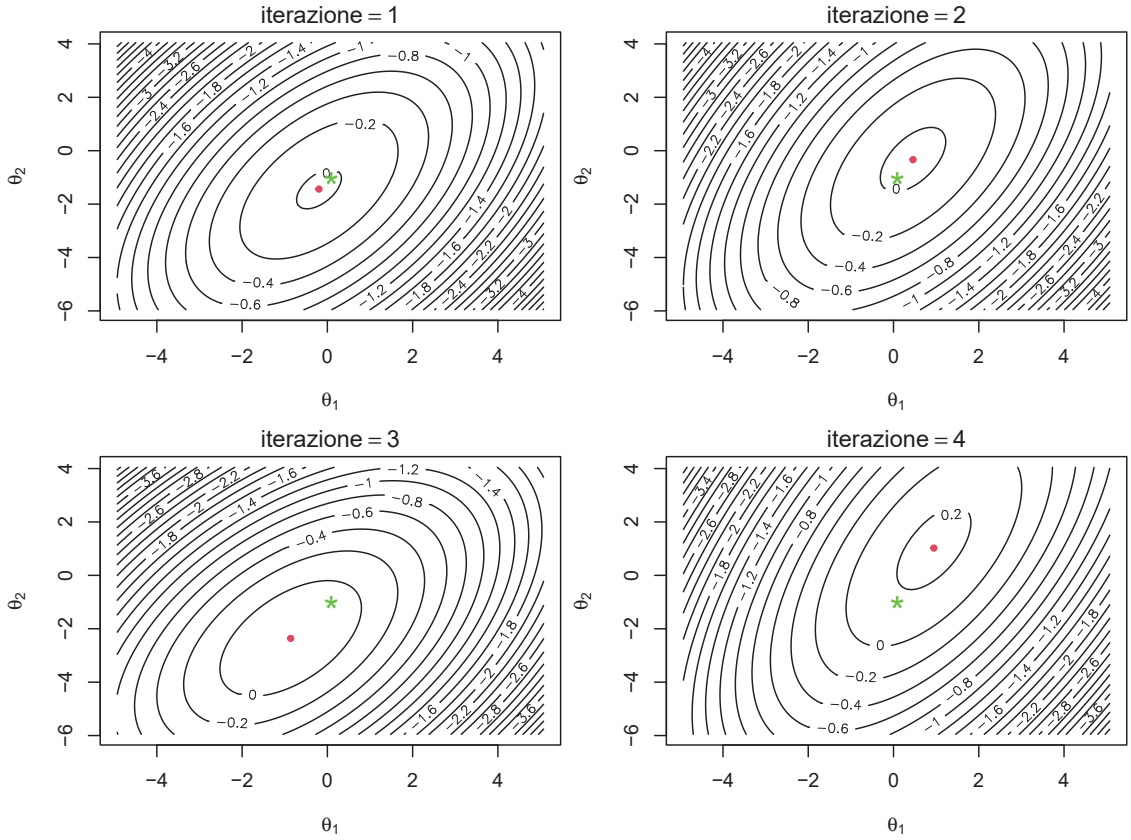


FIGURA 1.2: Rappresentazione del surrogato quadratico della log-verosimiglianza $\tilde{\mathcal{L}}^H(\theta)$ nelle prime quattro iterazioni dell'Algoritmo 1.1, in cui il valore iniziale $\bar{\theta}$ è preso pari alla media degli stimatori locali e l'aggiornamento viene effettuato tramite (1.8).

formula (7) per la log-verosimiglianza penalizzata che ne risulta, si definiscono i singoli termini di penalità in ogni macchina utilizzando solo i dati presenti in essa. Naturalmente tale termine aggiuntivo è stato diviso per il numero di osservazioni, in quanto esso è definito su una scala dipendente dalla numerosità campionaria. In questo modo la nuova funzione obiettivo locale della macchina j-esima si è considerata definita come

$$\mathcal{L}_j^F(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_{ij}(x_{ij}^T\theta) - \log\{1+\exp(x_{ij}^T\theta)\} - \frac{1}{2n}\log|i_j(\theta)|,$$

dove $i_j(\theta)$ è l'informazione attesa determinata nella macchina $\mathcal{M}_j$. Nella pratica però la correzione di Firth è stata utilizzata solo nella definizione dei gradienti locali, dato che il suo contributo nell'informazione attesa, e quindi nelle derivate seconde, può essere trascurato. Riferendosi alla formula (9), la componente r-esima del gradiente locale corretto della macchina j-esima risulta definita come

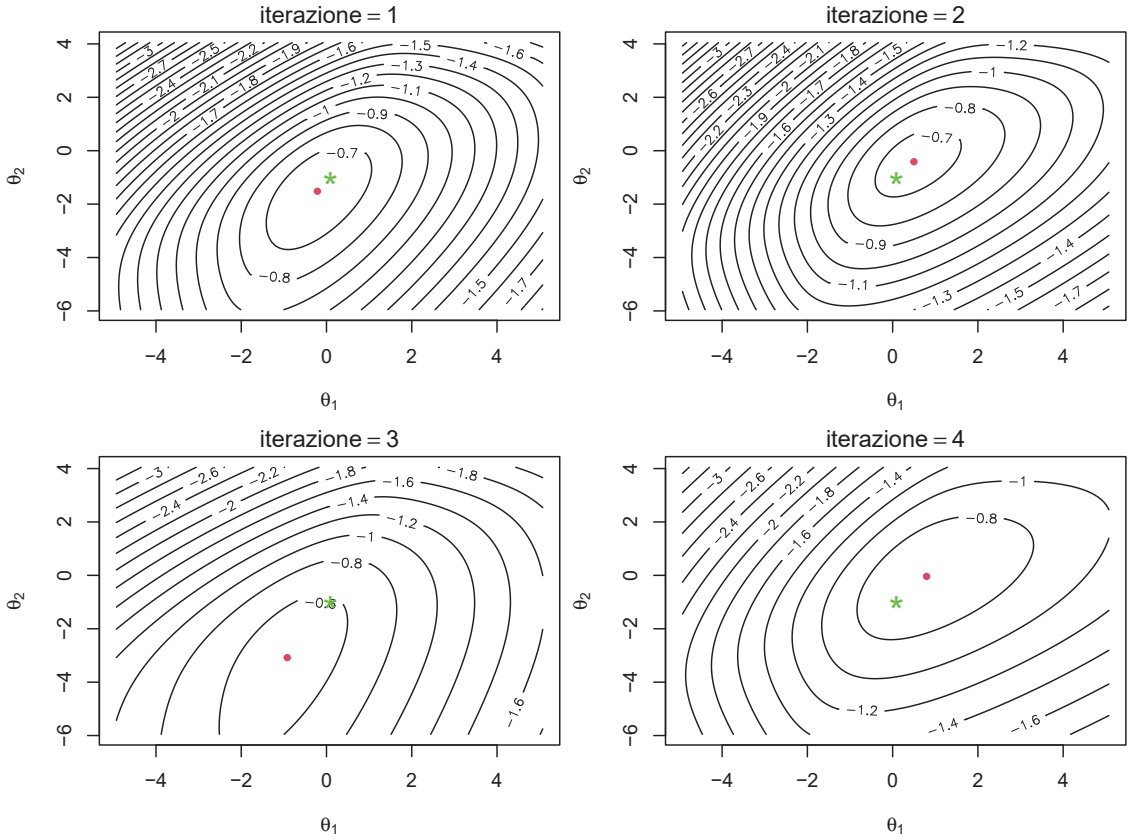


FIGURA 1.3: Rappresentazione del surrogato della log-verosimiglianza $\tilde{\mathcal{L}}(\theta)$ nelle prime quattro iterazioni dell'Algoritmo 1.1, in cui il valore iniziale $\bar{\theta}$ è preso pari alla media degli stimatori locali e l'aggiornamento viene effettuato tramite (1.6).

segue:

$$\begin{aligned}\nabla\mathcal{L}^F_{j,r}(\theta) &= \nabla\mathcal{L}_{j,r}(\theta) - \frac{1}{2}\frac{\partial}{\partial\theta_r}\log|I_j(\theta)| \\ &= -\frac{1}{n}\sum_{i=1}^{n}\{(y_{ij} + h_{ij}/2) - (1+h_{ij})\pi_{ij}\}x_{ij,r},\end{aligned} \tag{1.13}$$

dove h_{ij} non è altro che l'i-esimo elemento diagonale della matrice $\mathbf{H}_j$, definita nella formula (10), ottenuta con i dati nella macchina j-esima. Il gradiente globale è quindi stato ottenuto tramite la media dei gradienti locali corretti appena definiti,

$$\nabla\mathcal{L}^F_N(\theta) = \frac{1}{k}\sum_{j=1}^{k}\nabla\mathcal{L}^F_j(\theta),$$

e quest'ultimo è stato poi utilizzato al posto di $\nabla\mathcal{L}_N(\theta)$ nelle definizioni dei surrogati (1.5) e (1.7) e considerando anche $\nabla\mathcal{L}^F_1(\theta)$ al posto di $\nabla\mathcal{L}_1(\theta)$, ogni qualvolta quest'ultima quantità venisse richiesta all'interno degli algoritmi. Si noti

che $\nabla\mathcal{L}_N^F(\theta)$ non corrisponde al gradiente della funzione obiettivo globale ottenuta con l'aggiunta della correzione di Firth definita su tutti i dati. Tale modifica verrà nuovamente trattata nel Paragrafo 1.2.7.

2. Un'ulteriore modifica è stata quella di inserire la penalizzazione proposta da Rigon & Aliverti (2022) a livello di log-verosimiglianza globale. Infatti a differenza della correzione di Firth, tale penalità può essere suddivisa nelle sue componenti locali. Così facendo si è considerata come funzione obiettivo la log-verosimiglianza penalizzata cambiata di segno e divisa per il numero di osservazioni e si è proceduto con la metodologia CSL. In particolare, Rigon & Aliverti (2022) hanno dimostrato che definendo delle pseudo-osservazioni

$$\dot{y}_{ij} = \frac{d}{d+N}\frac{1}{2} + \frac{N}{d+N}y_{ij} \quad (j = 1, \dots, k,\ i = 1, \dots, n),$$

la funzione obiettivo globale basata sulla log-verosimiglianza penalizzata, da noi denotata con $\dot{\mathcal{L}}_N(\theta; y)$, può essere scritta in termini di una log-verosimiglianza genuina, a meno di costanti moltiplicative,

$$\dot{\mathcal{L}}_N(\theta; y) = (d/N + 1)\mathcal{L}_N(\theta; \dot{y}) = (d/N + 1)(-1/N)\ell(\theta; \dot{y}), \tag{1.14}$$

in cui $y = \{y_{ij} : j = 1, \dots, k,\ i = 1, \dots, n\}$ e $\dot{y} = \{\dot{y}_{ij} : j = 1, \dots, k,\ i = 1, \dots, n\}$. Operativamente ci si è quindi focalizzati su $\mathcal{L}_N(\theta; \dot{y})$, utilizzandola come funzione obiettivo di cui definire il surrogato. Tuttavia è opportuno osservare che se la penalizzazione viene valutata usando tutti i dati, per come è definita, non ha un contributo rilevante nel modificare la log-verosimiglianza. Proprio per questo motivo non ha aiutato a risolvere i problemi di convergenza.

3. Basandosi sull'idea precedente, si è dunque pensato di utilizzare la correzione di Rigon & Aliverti (2022) solo nella macchina di riferimento, in modo tale che l'entità di questa penalizzazione risultasse maggiore che in precedenza. Nel dettaglio si è deciso di avvalersi di essa solo nella definizione dell'hessiano della prima macchina che, considerando la formula (1.14), è pari a

$$\nabla^2\dot{\mathcal{L}}_1(\theta) = (d/n + 1)\nabla^2\mathcal{L}_1(\theta),$$

il quale corrisponde semplicemente a moltiplicare l'hessiano per una costante.

Tutte e tre le modifiche non hanno permesso di risolvere i problemi di convergenza dell'algoritmo. La modifica basata sulla correzione di Firth, oltre a non risolvere tali

problematiche, quando la numerosità locale n non era molto elevata, introduceva una distorsione eccessiva nei parametri con valore maggiore.

Prendendo spunto dall'ultima idea delle tre elencate, si è tentato di correggere l'hessiano della macchina di riferimento moltiplicandolo per delle costanti più grandi e si è potuto riscontrare che, con un valore sufficientemente grande, le stime non tendono a divergere e l'algoritmo converge. Se però la costante moltiplicativa è troppo grande, la velocità di convergenza dell'algoritmo tende a diminuire drasticamente. È da tenere a mente che queste considerazioni sono state fatte basandosi su un numero limitato di prove empiriche. Inoltre non è immediato poter definire un valore ottimale per cui moltiplicare l'hessiano perché, come detto in precedenza, potrebbe dipendere dal determinato problema di ottimizzazione e da diversi parametri strutturali annessi.

Da questi tentativi di modifica si può ipotizzare che l'utilizzo di regolarizzazioni pensate per contrarre le stime verso situazioni di sparsità non consentono all'algoritmo di convergere nelle situazioni problematiche. A tal proposito, nel paragrafo successivo verrà presentata un'estensione dell'Algoritmo 1.1 e dell'Algoritmo 1.2 che introdurrà una penalizzazione nella definizione della funzione surrogata $\tilde{\mathcal{L}}$, atta a regolarizzare la stima verso il valore del parametro su cui si sta approssimando la funzione obiettivo globale nell'iterazione corrente.

1.2 *Communication-Efficient Accurate Statistical Estimation*

1.2.1 L'algoritmo CEASE

Nel Paragrafo 1.1.4 si è visto come, anche con numerosità campionarie n moderate per macchina, gli algoritmi basati sulla metodologia CSL possono non convergere ed essere instabili. Un metodo *naive* per rimediare è quello di aggiungere una regolarizzazione quadratica convessa. Sebbene questa soluzione permetta all'algoritmo di convergere rapidamente, il fatto che tale regolarizzazione non sia di natura adattiva porterebbe la stima verso valori target erronei. Per questo motivo Fan et al. (2021) propongono di usare una penalizzazione che venga definita in base al valore della stima nell'iterazione corrente. Questa loro idea deriva dall'algoritmo *proximal point* (Rockafellar, 1976).

Definizione 1.2.1. Per qualsiasi funzione convessa $h : \mathbb{R}^d \to \mathbb{R}$, definiamo la funzione *proximal* $\text{prox}_h : \mathbb{R}^d \to \mathbb{R}^d$, $\text{x} \mapsto \arg\min_{\text{y}\in\mathbb{R}^d}\{h(\text{y}) + \|\text{y} - \text{x}\|_2^2/2\}$.

Per un dato valore $\alpha > 0$, l'algoritmo *proximal point* per minimizzare h iterativamente calcola

$$\mathrm{x}^{(t+1)} = \mathrm{prox}_{\alpha^{-1}h}(\mathrm{x}^{(t)}) = \underset{\mathrm{x}\in\mathbb{R}^d}{\arg\min}\{h(\mathrm{x}) + (\alpha/2)\|\mathrm{x} - \mathrm{x}^{(t)}\|_2^2\}, \quad \forall t \geq 0,$$

partendo da un valore iniziale $\mathrm{x}^{(0)}$. È un'ottimizzazione fortemente convessa che contrae verso il valore corrente $\mathrm{x}^{(t)}$. Sotto condizioni deboli, $\{x^{(t)}\}_{t=0}^{\infty}$ converge linearmente verso $\hat{x} \in \arg\min_{\mathbb{R}^d} h(\mathrm{x})$ (Rockafellar, 1976).

Se ora consideriamo $h(\theta) = \mathcal{L}_N(\theta)$, l'iterazione *proximal point* per il problema di stima globale d'interesse diventa:

$$\theta^{(t+1)} = \mathrm{prox}_{\alpha^{-1}\mathcal{L}_N}(\theta^{(t)}) = \underset{\theta\in\Theta}{\arg\min}\{\mathcal{L}_N(\theta) + (\alpha/2)\|\theta - \theta^{(t)}\|_2^2\}. \tag{1.15}$$

Ogni iterazione (1.15) corrisponde a un problema di ottimizzazione distribuita la cui funzione obiettivo non può essere definita in una singola macchina. Come abbiamo visto nell'approccio CSL, questo tipo di problema può essere risolto tramite gli Algoritmi 1.1 o 1.2 ridefinendo la funzione obiettivo con l'aggiunta del termine *proximal point* e partendo da $\theta^{(0)} = \theta^{(t)}$. In ogni caso, non c'è alcun bisogno di risolvere (1.15) esattamente, dato che $\mathrm{prox}_{\alpha^{-1}\mathcal{L}_N}(\theta^{(t)})$ non è altro che uno strumento per poter calcolare $\hat{\theta}$. Per questo motivo è sufficiente eseguire solo un'iterazione degli Algoritmi 1.1 o 1.2 e usare tale soluzione approssimata come $\theta^{(t+1)}$. Tale tipo di aggiornamento viene definito aggiornamento *proximal* inesatto. Se si vuole minimizzare esattamente il surrogato della funzione obiettivo come in (1.6) l'aggiornamento è

$$\theta^{(t+1)} = \mathrm{prox}_{\alpha^{-1}\tilde{\mathcal{L}}^t}(\theta^{(t)}) = \underset{\theta\in\Theta}{\arg\min}\{\tilde{\mathcal{L}}^t(\theta) + (\alpha/2)\|\theta - \theta^{(t)}\|_2^2\}, \tag{1.16}$$

altrimenti se si vuole minimizzare tramite approssimazione quadratica del surrogato come nello stimatore *one-step* (1.8),si ha

$$\theta^{(t+1)} = \theta^{(t)} - \big(\nabla^2\mathcal{L}_1(\theta^{(t)}) + \alpha I\big)^{-1}\big[\nabla\mathcal{L}_N(\theta^{(t)})\big]. \tag{1.17}$$

Il ragionamento appena effettuato ha condotto Fan et al. (2021) a formulare gli Algoritmi 1.3 e 1.4, delle versioni regolarizzate degli Algoritmi 1.1 e 1.2 con un termine addizionale di prossimità nelle funzioni obiettivo. Questi due nuovi stimatori che si ottengono sono nominati *Communication-Efficient Accurate Statistical Estimators* (CEASE). Il termine aggiuntivo permette di ridurre le differenze tra le funzioni obiettivo locali

contenute nelle macchine individuali ed è cruciale per garantire la convergenza dell'algoritmo quando $\{\mathcal{L}_j\}_{j=1}^k$ non sono abbastanza simili. L'Algoritmo 1.4 può essere visto come l'algoritmo DANE descritto in Shamir et al. (2014), anche se è da sottolineare che l'algoritmo CEASE è stato pensato con una motivazione differente e può essere considerato come un'*implementazione distribuita dell'algoritmo proximal point.* Questo diverso punto di vista ha permesso di ottenere forti garanzie di convergenza sotto condizioni abbastanza generali e non focalizzandosi principalmente su funzioni di perdita quadratica a differenza di Shamir et al. (2014). Tali risultati, presentati in Fan et al. (2021), verranno riportati nei paragrafi seguenti.

Algorithm 1.3: CEASE

Input: valore iniziale $\theta^{(0)} = \bar{\theta}$, parametro di regolarizzazione $\alpha \geq 0$, numero di iterazioni T

for $t = 0, 1, \ldots, T-1$ **do**
- Trasmissione di $\theta^{(t)}$ alle macchine locali $\{\mathcal{M}_j\}_{j=1}^k$;
- **for** $j = 1, \ldots, k$ **do**
  - Calcolo del gradiente locale $\nabla\mathcal{L}_j(\theta^{(t)})$ nella macchina $\mathcal{M}_j$;
  - Trasmissione del gradiente locale $\nabla\mathcal{L}_j(\theta^{(t)})$ alla macchina $\mathcal{M}_1$;
- Calcolo del gradiente globale $\nabla\mathcal{L}_N(\theta^{(t)}) = \frac{1}{k}\sum_{j=1}^k \nabla\mathcal{L}_j(\theta^{(t)})$ nella macchina $\mathcal{M}_1$;
- Costruzione del surrogato $\tilde{\mathcal{L}}^t(\theta) = \mathcal{L}_1(\theta) - \langle \nabla\mathcal{L}_1(\theta^{(t)}) - \nabla\mathcal{L}_N(\theta^{(t)}), \theta\rangle$;
- Svolgere uno dei seguenti aggiornamenti nella macchina $\mathcal{M}_1$:

  (1) $\theta^{(t+1)} \in \arg\min_{\theta\in\Theta}\{\tilde{\mathcal{L}}^t(\theta) + (\alpha/2)\|\theta - \theta^{(t)}\|_2^2\}$; `// Minimizzazione esatta`

  (2) $\theta^{(t+1)} = \theta^{(t)} - \big(\nabla^2\mathcal{L}_1(\theta^{(t)}) + \alpha\mathbf{I}\big)^{-1}\big[\nabla\mathcal{L}_N(\theta^{(t)})\big]$; `// Approssimazione // quadratica`

Output: $\theta^{(T)}$

1.2.2 Analisi deterministica dell'algoritmo CEASE

Nel paragrafo corrente verranno riportati dei risultati deterministici degli Algoritmi 1.3 e 1.4 basandosi su assunzioni strutturali ad alto livello e considerando che in essi venga effettuata la minimizzazione esatta della funzione surrogata. Verranno analizzati anche gli Algoritmi 1.1 e 1.2, come casi particolari dell'algoritmo CEASE in cui $\alpha = 0$. Si veda Fan et al. (2021, Paragrafo 3) per una trattazione più approfondita di tali risultati, in cui venga considerata anche l'aggiunta di una funzione di penalizzazione.

Nel prosieguo di questa analisi verranno utilizzate alcune notazioni che sono elencate nel seguito. Si denota con $[k]$ l'insieme $\{1, 2, \ldots, k\}$. Si scrive $a_n \asymp b_n$ se $a_n = O(b_n)$ e $b_n = O(b_n)$. Dati $x \in \mathbb{R}^k$ e $r > 0$, si definisce $B(x, r) = \{z \in \mathbb{R}^k : \|z - x\|_2 \leq r\}$. Per

Algorithm 1.4: *Symmetric* CEASE

Input: valore iniziale $\theta^{(0)} = \bar{\theta}$, parametro di regolarizzazione $\alpha \geq 0$, numero di iterazioni T

for $t = 0, 1, \ldots, T-1$ **do**
- Trasmissione di $\theta^{(t)}$ alle macchine locali $\{\mathcal{M}_j\}_{j=1}^k$;
- **for** $j = 1, \ldots, k$ **do**
  - Calcolo del gradiente locale $\nabla\mathcal{L}_j(\theta^{(t)})$ nella macchina $\mathcal{M}_j$;
  - Trasmissione del gradiente locale $\nabla\mathcal{L}_j(\theta^{(t)})$ alla macchina $\mathcal{M}_1$;
- Calcolo del gradiente globale $\nabla\mathcal{L}_N(\theta^{(t)}) = \frac{1}{k}\sum_{j=1}^k \nabla\mathcal{L}_j(\theta^{(t)})$ nella macchina $\mathcal{M}_1$;
- Trasmissione del gradiente globale $\nabla\mathcal{L}_N(\theta^{(t)})$ alle altre macchine $\{\mathcal{M}_j\}_{j=2}^k$;
- **for** $j = 1, \ldots, k$ **do**
  - Costruzione del surrogato $\tilde{\mathcal{L}}_j^t(\theta) = \mathcal{L}_j(\theta) - \langle \nabla\mathcal{L}_j(\theta^{(t)}) - \nabla\mathcal{L}_{\mathcal{N}}(\theta^{(t)}), \theta\rangle$;
  - Svolgere uno dei seguenti aggiornamenti nella macchina $\mathcal{M}_j$

    (1) $\theta_j^{(t+1)} \in \arg\min_{\theta\in\Theta}\{\tilde{\mathcal{L}}_j^t(\theta) + (\alpha/2)\|\theta - \theta^{(t)}\|_2^2\}$; `// Minimizzazione esatta`

    (2) $\theta_j^{(t+1)} = \theta_j^{(t)} - \left(\nabla^2\mathcal{L}_j(\theta_j^{(t)}) + \alpha\mathbf{I}\right)^{-1}\left[\nabla\mathcal{L}_N(\theta_j^{(t)})\right]$; `// Approssimazione quadratica`

  - Trasmissione della stima $\theta_j^{(t+1)}$ alla macchina $\mathcal{M}_1$;
- Calcolo della media $\theta^{(t+1)} = \frac{1}{k}\sum_{j=1}^k \theta_j^{(t+1)}$ nella macchina $\mathcal{M}_1$;
- Trasmissione di $\theta^{(t+1)}$ alle altre macchine locali $\{\mathcal{M}_j\}_{j=2}^k$;

Output: $\theta^{(T)}$

una funzione convessa h in $\mathbb{R}^k$, si indicherà con $\partial h(x)$ il suo insieme sub-differenziale in $x \in \mathbb{R}^k$. L'operatore $\|\cdot\|_2$ utilizzato per le matrici ne indica la norma di Frobenius.

Definizione 1.2.2. Sia $h : \mathbb{R}^d \to \mathbb{R}$ una funzione convessa, $\Omega \subseteq \mathbb{R}^d$ un insieme convesso e $\rho \geq 0$. h è ρ-fortemente convessa in Ω se $h(y) \geq h(x) + \langle g, y - x\rangle + \frac{\rho}{2}\|y - x\|_2^2,\ \forall x, y \in \Omega$ e $g \in \partial h(x)$.

Assunzione 1.2.1 (Convessità forte)**.** $\mathcal{L}_N$ ha un unico punto di minimo $\hat{\theta} \in \mathbb{R}^d$, ed è ρ-fortemente convessa in $B(\hat{\theta}, R)$ per un certo $R > 0$ e $\rho > 0$.

Assunzione 1.2.2 (Omogeneità)**.** $\|\nabla^2\mathcal{L}_j(\theta) - \nabla^2\mathcal{L}_N(\theta)\|_2 \leq \delta,\ \forall j \in [k],\ \theta \in B(\hat{\theta}, R)$.

Le assunzioni appena presentate serviranno per la validità dei teoremi che verranno descritti successivamente. L'Assunzione 1.2.1 viene facilmente rispettata nella maggior parte dei problemi d'interesse, mentre si noti come l'Assunzione 1.2.2 sia in linea con quanto detto nel Paragrafo 1.1.4. Infatti la somiglianza tra le funzioni obiettivo locali e quella globale viene caratterizzata in termini di hessiano. Tale modo di definire la

similarità tra funzioni è una generalizzazione del concetto di "funzioni *delta-related*" per funzioni di perdita quadratica in Arjevani & Shamir (2015). Ci si riferirà a δ come un parametro di omogeneità. Basandoci su entrambe le assunzioni, si definisce

$$\rho_0 = \sup \Big\{ c \in [0, \rho] : \{\mathcal{L}_j\}_{j=1}^k \text{ sono } c\text{-fortemente convesse in } B(\hat{\theta}, R) \Big\}. \tag{1.18}$$

La seguente assunzione aggiuntiva riguarda la regolarità della matrice hessiana di $\mathcal{L}_N$ e non è necessaria per garantire la convergenza dello stimatore verso $\hat{\theta}$ ma aiuta ad ottenere un risultato più forte per il tasso di contrazione per l'Algoritmo 1.4, giustificando l'utilità dell'operazione di media degli stimatori locali.

Assunzione 1.2.3 (Regolarità dell'hessiano)**.** Esiste $M \geq 0$ tale che $\|\nabla^2 \mathcal{L}_N(\theta') - \nabla^2 \mathcal{L}_N(\theta'')\|_2 \leq M\|\theta' - \theta''\|_2, \ \forall \theta', \theta'' \in B(\hat{\theta}, R)$.

Il Teorema 1.2.1 (si veda Fan et al., 2021, Teorema 3.1) dà delle garanzie di convergenza per gli Algoritmi 1.3 e 1.4.

Teorema 1.2.1. Sotto le Assunzioni 1.2.1 e 1.2.2, si considerino gli stimatori iterativi $\{\theta^{(t)}\}_{t=0}^T$ definiti dagli Algoritmi 1.3 e 1.4. Si supponga che $\theta^{(0)} \in B(\hat{\theta}, R/2)$ e $[\delta/(\rho_0 + \alpha)]^2 < \rho/(\rho + \alpha)$.

(a) Per entrambi gli Algoritmi 1.3 e 1.4 abbiamo che

$$\|\theta^{(t+1)} - \hat{\theta}\|_2 \leq \|\theta^{(t)} - \hat{\theta}\|_2 \cdot \frac{\frac{\delta}{\rho_0 + \alpha}\sqrt{\rho^2 + 2\alpha\rho} + \alpha}{\rho + \alpha}, \quad 0 \leq t \leq T - 1; \tag{1.19}$$

(b) Se l'Assunzione 1.2.3 è valida, per l'Algoritmo 1.4 si ha che

$$\|\theta^{(t+1)} - \hat{\theta}\|_2 \leq \|\theta^{(t)} - \hat{\theta}\|_2 \cdot \frac{\gamma_t\sqrt{\rho^2 + 2\alpha\rho} + \alpha}{\rho + \alpha}, \quad 0 \leq t \leq T - 1, \tag{1.20}$$

dove si definisce $\gamma_t = \frac{\delta}{\rho_0 + \alpha} \cdot \min\{1, \frac{\delta}{\rho + \alpha}(1 + \frac{M}{\rho_0 + \alpha}\|\theta^{(t)} - \hat{\theta}\|_2)\}$;

(c) Entrambi i fattori moltiplicativi in (1.19) e (1.20) sono strettamente minori di 1.

La condizione $[\delta/(\rho_0 + \alpha)]^2 < \rho/(\rho + \alpha)$ assicura che entrambi i fattori di contrazione siano minori di 1. Si noti che (1.20) richiede l'Assunzione 1.2.3, la quale è facilmente rispettata, soprattutto se nella funzione obiettivo non si inseriscono termini aggiuntivi di regolarizzazione.

Definizione 1.2.3 (Convergenza *Q-lineare*)**.** Secondo Nocedal & Wright (2006), una sequenza $\{\mathtt{x}_n\}_{n=1}^\infty$ in $\mathbb{R}^d$ si dice che converge *Q-linearmente* a $\mathtt{x}^* \in \mathbb{R}^d$ se esiste $r \in (0, 1)$ tale che $\|\mathtt{x}_{n+1} - \mathtt{x}^*\|_2 \leq r\|\mathtt{x}_n - \mathtt{x}^*\|_2$ per un n sufficientemente grande.

Il Teorema 1.2.1 descrive la convergenza *Q-lineare* (si veda la Definizione 1.2.3) della sequenza di stime generate dagli Algoritmi 1.3 e 1.4 sotto condizioni abbastanza generali. Si noti come il tasso di contrazione dipenda direttamente dai parametri strutturali e dalla scelta di α. La convergenza inoltre è garantita scegliendo un valore di α sufficientemente grande. Invece, come si vedrà in seguito nel Teorema 1.2.2, la convergenza degli Algoritmi 1.1 e 1.2 dipende dall'assunzione di omogeneità $\rho_0 > \delta$, la quale richiede che le funzioni $\{\mathcal{L}_j\}_{j=1}^k$ siano abbastanza simili.

Dunque, imporre una regolarizzazione adeguata permette all'algoritmo CEASE di convergere in situazioni generali con una numerosità campionaria locale potenzialmente non sufficiente. Il Corollario 1.2.1 (si veda Fan et al., 2021, Corollario 3.1) qui sotto fornisce delle linee guida per la scelta di α in modo tale da garantire la convergenza degli Algoritmi 1.3 e 1.4 in generale.

Corollario 1.2.1. Si considerino valide la Assunzioni 1.2.1 e 1.2.2, $\theta^{(0)} \in B(\hat{\theta}, R/2)$, e $\{\theta^{(t)}\}_{t=0}^T$ le stime ottenute nelle iterazioni degli Algoritmi 1.3 e 1.4. Con qualsiasi $\alpha \geq 4\delta^2/\rho$, entrambi gli algoritmi convergono con i tassi di convergenza in (1.19) e (1.20) limitati da $\left(1 - \frac{\rho}{10(\alpha+\rho)}\right)$.

D'altra parte, nelle situazioni meno problematiche in cui le funzioni di perdita locali hanno una piccola differenza relativa δ/ρ si desidererebbe avere le stesse performance degli Algoritmi 1.1 e 1.2, i quali in questi casi hanno un tasso di convergenza rispettivamente dell'ordine di δ/ρ e $(\delta/\rho)^2$, come si vedrà nel Teorema 1.2.2. Il seguente corollario mira proprio a fornire un limite superiore per α in modo tale che il fattore di contrazione abbia questi ordini di grandezza (si veda Fan et al., 2021, Corollario 3.2).

Corollario 1.2.2. Si considerino valide la Assunzioni 1.2.1 e 1.2.2, $\theta^{(0)} \in B(\hat{\theta}, R/2)$, e si supponga $\alpha \leq C\delta^2/\rho$ per una certa costante C. Esistono delle costanti C_1 e C_2 tali che le successive proprietà siano valide quando δ/ρ è sufficientemente piccolo:

(a) Gli Algoritmi 1.3 e 1.4 hanno la proprietà di contrazione

$$\|\theta^{(t+1)} - \hat{\theta}\|_2 \leq C_1(\delta/\rho)\|\theta^{(t)} - \hat{\theta}\|_2, \quad 0 \leq t \leq T-1; \tag{1.21}$$

(b) Se è valida anche l'Assunzione 1.2.3 e $\|\theta^{(t)} - \hat{\theta}\|_2 \leq \rho/M$, allora per l'Algoritmo 1.4 si ha che

$$\|\theta^{(t+1)} - \hat{\theta}\|_2 \leq C_2(\delta/\rho)^2\|\theta^{(t)} - \hat{\theta}\|_2. \tag{1.22}$$

Il Corollario 1.2.2 rivela che una scelta di $\alpha \asymp \delta^2/\rho$ assicura che il fattore di contrazione sia dello stesso ordine di quello degli algoritmi senza regolarizzazione ($\alpha = 0$) quando δ/ρ è piccolo. Considerando quindi anche il Corollario 1.2.1 ne deriva che scegliendo

$\alpha \asymp \delta^2/\rho$ come default, gli Algoritmi 1.3 e 1.4 risultano essere sia veloci che robusti. In questo modo sono dunque affidabili e convergono in situazioni generali ed efficienti nei casi più favorevoli in cui le funzioni obiettivo locali si assomigliano abbastanza.

È da notare come gli algoritmi in questione traggano maggior efficienza dalla somiglianza delle funzioni obiettivo nelle macchine. Infatti il fattore di contrazione diminuisce se diminuisce δ/ρ, ossia la differenza relativa tra funzioni.

Inoltre, se $\mathcal{L}_N$ è liscia e $\theta^{(t)}$ è abbastanza vicino a $\hat{\theta}$, il Corollario 1.2.2 mostra che ogni iterazione dell'Algoritmo 1.4 è equivalente a due iterazioni dell'Algoritmo 1.3, sebbene il primo esegua solo un passo di ottimizzazione. Quindi l'operazione di media nell'Algoritmo 1.4 permette di ridurre l'errore quanto un'operazione di ottimizzazione, mettendoci meno tempo essendo la sue esecuzione meno onerosa. In questo caso l'Algoritmo 1.4 risulta preferibile e varrà lo stesso ragionamento per gli Algoritmi 1.1 e 1.2. Ricordiamo in ogni caso che, come detto alla fine del Paragrafo 1.1.2, le versioni asimmetriche degli algoritmi che si stanno analizzando, sebbene in alcuni casi non siano convenienti a livello di velocità di convergenza, presentano degli aspetti positivi tra cui la naturale definizione di un unico surrogato con cui procedere con eventuali aggiuntive procedure inferenziali.

Si noti come, a differenza di quanto presentato in Fan et al. (2021), nelle analisi riportate non si sta considerando l'eventuale presenza di funzioni di penalizzazione per problemi ad elevata dimensionalità e questo perché non è il principale interesse di questo lavoro di tesi. Come detto in precedenza, senza regolarizzazione aggiuntiva, l'Algoritmo 1.4 si riduce ad un caso utile dell'algoritmo DANE (Shamir et al., 2014) ma in questo caso, tramite il Teorema 1.2.1 e i suoi corollari si fornisce un'analisi più approfondita sotto diversi aspetti. Infatti vengono definiti dei risultati anche per funzioni di rischio empiriche non lisce ed inoltre si forniscono dei suggerimenti nella scelta del parametro di regolarizzazione α.

Analisi deterministica con n elevato

In questo paragrafo ci si focalizza sui casi in cui n è sufficientemente grande da avere $\rho_0 > \delta \geq 0$, per fornire dei risultati per gli Algoritmi 1.1 e 1.2. Il Teorema riportato di seguito elenca tali proprietà (si veda Fan et al., 2021, Teorema 3.2).

Teorema 1.2.2. Sotto le Assunzioni 1.2.1 e 1.2.2, e $\rho_0 > \delta \geq 0$, si considerino gli stimatori iterativi $\{\theta^{(t)}\}_{t=0}^{T}$ definiti dagli Algoritmi 1.1 e 1.2, con $\theta^{(0)} \in B(\hat{\theta}, R)$. Allora

$$\|\theta^{(t+1)} - \hat{\theta}\|_2 \leq (\delta/\rho_0)\|\theta^{(t)} - \hat{\theta}\|_2, \quad \forall t \geq 0.$$

In aggiunta, se l'Assunzione 1.2.3 è valida, allora per l'Algoritmo 1.2 si ha che

$$\|\theta^{(t+1)} - \hat{\theta}\|_2 \leq \frac{\delta}{\rho_0}\|\theta^{(t)} - \hat{\theta}\|_2 \cdot \min\{1, \frac{\delta}{\rho}(1 + \frac{M}{\rho_0}\|\theta^{(t)} - \hat{\theta}\|_2)\}, \quad \forall t \geq 0.$$

La prima parte del Teorema 1.2.2 perfeziona le analisi svolte in Jordan et al. (2019), dato che viene permesso allo stimatore iniziale di essere inaccurato e vengono dati dei tassi di convergenza più dettagliati.

La seconda parte non fa altro che mettere in evidenza ciò che si è già visto per gli Algoritmi 1.3 e 1.4. Infatti, sotto alcune condizioni riportate, l'Algoritmo 1.2 permette di raddoppiare la velocità di convergenza.

1.2.3 Analisi statistica dell'algoritmo CEASE

Le analisi statistiche che verranno esposte sono presenti anch'esse in Fan et al. (2021, Capitolo 4), articolo a cui si rimanda per una trattazione più dettagliata, e anche in questo caso sono relative agli algoritmi CEASE basati sull'aggiornamento esatto e non approssimato della funzione surrogata.

In questo paragrafo si considera il modello di regressione logistica per variabili risposta bernoulliane con funzione legame $\pi(\theta) = [1+\exp(-\mathtt{x}_i^T\theta)]^{-1}$, dove $\mathtt{x}_i = (1, \mathrm{u}_i^T)^T \in \mathbb{R}^d$ e $\{\mathrm{u}_i\}_{i=1}^N \subseteq \mathbb{R}^{d-1}$ sono variabili esplicative casuali *i.i.d.* con media zero e matrice di varianza e covarianza Σ. Si supponga che esistano tre costanti universali positive A_1, A_2 e A_3 tali che $A_1 \leq \|\Sigma\|_2 \leq A_2 d^{A_3}$. In questo caso la funzione obiettivo di riferimento $\mathcal{L}$ non è altro che l'opposto della funzione di log-verosimiglianza divisa per il numero di osservazioni. Di seguito vengono imposte alcune condizioni standard di regolarità.

Assunzione 1.2.4.

- $\{\Sigma^{-1/2}\mathrm{u}_i\}_{i=1}^N$ sono vettori di variabili *sub-gaussiane i.i.d.*
- Per ogni $x \in \mathbb{R}$, $|\pi''(x)|$ e $|\pi'''(x)|$ sono limitate da una costante.
- $\|\theta^*\|_2$ è limitato da una costante.

Per distribuzione *sub-gaussiana* si intende una distribuzione di probabilità le cui code sono dominate dalle code di una distribuzione gaussiana, cioè decadono almeno alla stessa velocità.

Un'ulteriore assunzione necessaria per le analisi è la seguente.

Assunzione 1.2.5. Esiste una costante universale $\rho > 0$ tale che $\mathcal{L}^*$ è ρ-fortemente convessa in $B(\theta^*, 2R)$ per una certa quantità positiva R.

La seguente assunzione, come visto nel paragrafo precedente, serve ad assicurare che l'operazione di media dell'Algoritmo 1.4 permetta di migliorare significativamente l'accuratezza.

Assunzione 1.2.6. Esiste una costante universale $M \geq 0$ tale che $\|\nabla^2\mathcal{L}^*(\theta') - \nabla^2\mathcal{L}^*(\theta'')\| \leq M\|\theta' - \theta''\|_2,\ \forall \theta', \theta'' \in B(\theta^*, 2R)$.

Sotto le suddette assunzioni del modello, si può determinare in modo esplicito il tasso di δ definito nell'Assunzione 1.2.2. Nel dettaglio (si veda ad esempio Fan et al., 2021, Lemma E.5) si ha che

$$\max_{j\in[k]} \max_{\theta\in B(\hat{\theta},R)} \|\nabla^2\mathcal{L}_j(\theta) - \nabla^2\mathcal{L}_N(\theta)\|_2 = O_p\left(\|\Sigma\|_2\sqrt{\frac{d(\log d + \log N)}{n}}\right), \tag{1.23}$$

a condizione che $n \geq cd$ per una costante arbitraria positiva c. Quindi, con alta probabilità, $\delta \asymp \|\Sigma\|_2\sqrt{d(\log d + \log N)/n}$. Se si omettono i termini logaritmici, basandosi sui risultati deterministici ottenuti nel capitolo precedente, si ha che il fattore di contrazione è approssimativamente $\kappa\sqrt{d/n}$, dove $\kappa = \|\Sigma\|_2/\rho$. Si noti come tale fattore dipenda in modo esplicito da p e κ, a differenza di quanto avviene in Jordan et al. (2019), dove tali quantità vengono considerate finite.

Come fatto in precedenza, ci si concentrerà nel descrivere l'errore di ottimizzazione $\|\theta^{(t)} - \hat{\theta}\|_2$ e nel seguente teorema verrà dimostrato che, ad ogni iterazione degli Algoritmi 1.3 e 1.4, $\theta^{(t)}$ si avvicina a $\hat{\theta}$ con un ordine che dipende dalla dimensione campionaria locale (si veda Fan et al., 2021, Teorema 4.1). Dunque, con un numero finito di iterazioni è possibile rendere l'errore di ottimizzazione di un ordine trascurabile rispetto all'errore statistico $\|\hat{\theta} - \theta^*\|_2$, rendendo così lo stimatore che si ottiene equivalente allo stimatore globale $\hat{\theta}$ definito avendo tutti i dati in un'unica macchina.

Teorema 1.2.3. Sotto le Assunzioni 1.2.4 e 1.2.5, si supponga che con alta probabilità $\theta^{(0)} \in B(\hat{\theta}, R/2)$. Si definisca $\eta = \kappa^2(\log N)d/n$ e $\kappa = \|\Sigma\|_2/\rho$. Per qualsiasi $c_1, c_2 > 0$, esiste $C > 0$ tale che le seguenti proprietà sono valide con elevata probabilità:

(a) Se $n \geq c_1 d$ e $\alpha \geq C\rho\eta$, allora entrambi gli Algoritmi 1.3 e 1.4 hanno convergenza lineare

$$\|\theta^{(t)} - \hat{\theta}\|_2 \leq \left[1 - \frac{\rho}{10(\alpha + \rho)}\right]^t \|\theta^{(0)} - \hat{\theta}\|_2, \quad \forall t \geq 0;$$

(b) Se η è sufficientemente piccolo e $\alpha \leq c_2\rho\eta$, allora per entrambi gli algoritmi

$$\|\theta^{(t)} - \hat{\theta}\|_2 = O_p(\eta^{t/2}\|\theta^{(0)} - \hat{\theta}\|_2), \quad \forall t \geq 0;$$

in aggiunta, se anche l'Assunzione 1.2.6 è valida, per l'Algoritmo 1.4 si ha che

$$\|\theta^{(t)} - \hat{\theta}\|_2 = O_p(\eta^{t-t_0}\|\theta^{(t_0)} - \hat{\theta}\|_2), \quad \forall t \geq t_0,$$

dove $t_0 = \lceil \frac{2\log(CMR/\rho)}{\log(1/\eta)} \rceil$.

Il Teorema 1.2.3 mostra come la dimensione campionaria locale influisca positivamente sulla velocità di convergenza degli Algoritmi 1.3 e 1.4. Infatti, si noti come l'errore di ottimizzazione viene contratto da un fattore che converge a zero all'aumentare di n.

Analogamente a quanto visto nei risultati deterministici, l'operazione di media ha all'incirca lo stesso effetto di un passo di ottimizzazione se la funzione di rischio $\mathcal{L}^*$ è liscia (fatto che avviene quasi sempre) e $\theta^{(t)}$ è sufficientemente vicino a $\hat{\theta}$ dopo un numero finito di iterazioni t_0.

Si noti che per quanto riguarda l'inizializzazione, la condizione $\theta^{(0)} \in B(\hat{\theta}, R/2)$ è debole e facile da rispettare, dato che $\hat{\theta}$ di solito è uno stimatore consistente e $\|\theta^*\|_2$ è limitato (Assunzione 1.2.4). A differenza di quanto definito in Jordan et al. (2019), Fan et al. (2021) permettono di inizializzare gli algoritmi proposti con valori inaccurati, come $\theta^{(0)} = 0$, garantendo ugualmente la convergenza, anche se chiaramente l'accuratezza di tale stimatore iniziale $\theta^{(0)}$ aiuta a ridurre il numero di iterazioni necessarie.

Sebbene nella pratica sia difficile capire se n è sufficientemente elevato, una scelta adeguata di α permette sempre di avere una convergenza lineare, con un fattore di contrazione che cambia al variare della dimensione campionaria locale n. Nel Teorema 1.2.3 infatti, si vede che, scegliendo $\alpha \asymp \rho\eta$, si hanno sia garanzie di convergenza degli Algoritmi 1.3 e 1.4 nei casi più problematici in cui le funzioni obiettivo locali non sono abbastanza simili, sia un tasso di convergenza veloce, dello stesso ordine di quello degli Algoritmi 1.1 e 1.2, nelle situazioni in cui n è elevato (si veda il Teorema 1.2.4). Sotto questo aspetto quindi, gli algoritmi della metodologia CEASE risolvono perfettamente le problematiche principali dell'approccio CSL.

Inoltre si sottolinea che, considerando che l'errore statistico $\|\hat{\theta} - \theta^*\|_2$ è dell'ordine di $O_p(\sqrt{d/N})$ sotto condizioni deboli, per raggiungere l'efficienza statistica tramite gli Algoritmi 1.3 e 1.4 sono necessarie $O(\frac{\log\|\theta^{(0)}-\hat{\theta}\|_2+\log(N/d)}{\log(1/\eta)})$ iterazioni.

Analisi statistica con n elevato

Ora si presentano dei risultati analoghi relativi all'errore di ottimizzazione per gli Algoritmi 1.1 e 1.2 in un contesto in cui la numerosità campionaria locale n è sufficientemente elevata (si veda Fan et al., 2021, Teorema 4.2).

Teorema 1.2.4. Si considerino valide le Assunzioni 1.2.4 e 1.2.5, e si supponga che con alta probabilità $\theta^{(0)} \in B(\hat{\theta}, R)$ per qualche $R > \|\hat{\theta} - \theta^*\|_2$. Per gli Algoritmi 1.1 e 1.2 si ha che

$$\|\theta^{(t)} - \hat{\theta}\|_2 = O_p(\eta^{t/2}\|\theta^{(0)} - \hat{\theta}\|_2), \quad \forall t \geq 0,$$

dove $\eta = \kappa^2 p(logN)/n$. In aggiunta, si ritenga valida anche l'Assunzione 1.2.6. Esiste allora una certa costante C tale che per l'Algoritmo 1.4 si ha

$$\|\theta^{(t)} - \hat{\theta}\|_2 = O_p(\eta^{t-t_0}\|\theta^{(t_0)} - \hat{\theta}\|_2), \quad \forall t \geq t_0,$$

dove $t_0 = \lceil \frac{2\log(CMR/\rho)}{\log(1/\eta}} \rceil$.

Il Teorema 1.2.4 mostra che, quando n è sufficientemente grande, gli Algoritmi 1.1 e 1.2 hanno caratteristiche simili a quelle degli Algoritmi 1.3 e 1.4:

- convergenza più veloce con n elevato;
- l'operazione di media degli stimatori locali permette di velocizzare significativamente la convergenza se la funzione di rischio $\mathcal{L}^*$ è abbastanza liscia;
- deboli restrizioni nell'inizializzazione.

Tuttavia, tale teorema non fornisce alcuna garanzia di convergenza in contesti generali in cui la dimensione campionaria locale non è sufficiente, ed è quindi del tutto in linea con i problemi riscontrati nel Paragrafo 1.1.4.

1.2.4 Indicazioni pratiche

In questo breve paragrafo si forniscono delle indicazioni generali sull'implementazione pratica dell'algoritmo CEASE (si veda Fan et al., 2021, Paragrafo 4.3).

In merito all'inizializzazione, viene consigliato di scegliere un valore che dipenda dal numero di osservazioni n contenute localmente. Se ad esempio n non è sufficientemente elevato in relazione al numero di variabili esplicative d, un'inizializzazione pari a zero potrebbe essere più robusta. Nel caso in cui n sia moderato o grande, lo stimatore dato dalla media degli stimatori locali porta a una convergenza estremamente veloce.

Relativamente alla scelta di α invece, secondo il Teorema 1.2.3, è sufficiente fissarlo ad un valore dell'ordine di $\rho\kappa^2 d\log(N)/n$. Nella pratica, impostare α ad essere un piccolo multiplo di d/n sembra essere una scelta adeguata. Successivamente infatti, nell'implementazione, si considererà $\alpha = 0.15d/n$, valore considerato nello studio di simulazione svolto in Fan et al. (2021, Paragrafo 5.1).

1.2.5 Intervalli di confidenza

Come sottolineato in Fan et al. (2021), la metodologia CEASE finora è stata valutata a livello di stima puntuale. Uno degli interessi di questo lavoro di tesi sarà infatti quello di proporre per questo approccio un'opportuna definizione degli standard error degli stimatori, da poter usare successivamente per costruire intervalli di confidenza o svolgere test d'ipotesi.

Essendo il metodo CEASE una generalizzazione del *framework* CSL, in cui viene aggiunto un termine di prossimità nelle funzioni obiettivo, risulta adeguato procedere in modo analogo a quanto visto nel Paragrafo 1.1.3.

Se si effettuano un numero sufficiente t di iterazioni degli Algoritmi 1.3 e 1.4 e $\alpha \asymp \rho\eta$, grazie al Teorema 1.2.3 si ha la convergenza alla stima globale $\hat{\theta}$. Risulta quindi lecito ipotizzare che tale stimatore $\theta^{(t)}$ abbia la stessa distribuzione asintotica dello stimatore $\hat{\theta}$, cioè

$$\sqrt{N}(\theta^{(t)} - \theta^*) \to \mathcal{N}_d(0, \Sigma) \quad \text{in distribuzione con } N \to \infty.$$

È dunque d'interesse definire delle stime della matrice di varianza e covarianza asintotica Σ dello stimatore $\theta^{(t)}$. Nell'aggiornamento (1.16) da eseguire in ogni iterazione del metodo CEASE, è possibile notare che in questo caso il surrogato da minimizzare è $\tilde{\mathcal{L}}^t(\theta) + (\alpha/2)\|\theta - \theta^{(t)}\|_2^2$ con il termine aggiuntivo di penalità derivante dall'idea dell'algoritmo *proximal point*. Tenendo conto di questa aggiunta gli stimatori proposti nel Paragrafo 1.1.3 vengono modificati come mostrato nel seguito:

- stimatore *plug-in* locale,

$$\tilde{\Sigma} = (\nabla^2\tilde{\mathcal{L}}^t(\theta^{(t)}) + \alpha\mathbf{I})^{-1}\left(\frac{1}{n}\sum_{i=1}^{n}\nabla\tilde{\mathcal{L}}^t(\theta^{(t)}; z_{i1})\nabla\tilde{\mathcal{L}}^t(\theta^{(t)}; z_{i1})^T\right)(\nabla^2\tilde{\mathcal{L}}^t(\theta^{(t)}) + \alpha\mathbf{I})^{-1};$$

- stimatore *plug-in* globale,

$$\tilde{\Sigma}' = (\nabla^2\tilde{\mathcal{L}}^t(\theta^{(t)}) + \alpha\mathbf{I})^{-1}\left(\frac{n}{k}\sum_{j=1}^{k}\nabla\mathcal{L}_j(\theta^{(t)})\nabla\mathcal{L}_j(\theta^{(t)})^T\right)(\nabla^2\tilde{\mathcal{L}}^t(\theta^{(t)}) + \alpha\mathbf{I})^{-1}.$$

Si ricorda che nel *framework* CSL, quando $\mathcal{L}(\theta)$ è l'opposto della funzione di log-verosimiglianza, si può utilizzare $\nabla^2\tilde{\mathcal{L}}(\tilde{\theta})^{-1}$ come stimatore *plug-in* di Σ. Per il metodo CEASE si propone di modificarlo in

$$\tilde{\Sigma}'' = \left(\nabla^2\tilde{\mathcal{L}}^t(\theta^{(t)}) + \alpha\mathbf{I}\right)^{-1}. \tag{1.24}$$

La definizione degli intervalli di confidenza è in linea con quella vista in (1.12), in cui andrà utilizzata una delle stime di Σ appena specificate.

1.2.6 CEASE nella regressione logistica

In questo paragrafo si vuole specificare l'algoritmo CEASE nel caso della regressione logistica per dati binari (si veda Fan et al., 2021, Appendice C) e la relativa specificazione degli intervalli di confidenza. Tali considerazioni varranno anche per il metodo CSL, in quanto si ricorda che esso non è altro che un caso particolare dell'algoritmo CEASE in cui $\alpha = 0$.

Come prima cosa si illustrerà come eseguire la minimizzazione esatta e quella approssimata della funzione surrogata $\tilde{\mathcal{L}}_j^t(\theta)$ ottenuta nella macchina $\mathcal{M}_j$ all'iterazione t. In particolare per l'ottimizzazione numerica si considererà il metodo *Newton-Raphson*, il quale in questo caso corrisponde al metodo *Fisher scoring*, in quanto nella regressione logistica l'informazione osservata corrisponde all'informazione attesa. Verrà preso come riferimento l'Algoritmo 1.4 dato che gli altri algoritmi rientrano in quest'ultimo come specificazioni particolari.

Dato $z_1^N = \{(x_{ij}, y_{ij}) : i = 1, \dots, n,\ j = 1, \dots, k\}$, si ricorda che la funzione obiettivo nella macchina $\mathcal{M}_j$ è

$$\mathcal{L}_j(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\big\{y_{ij}(x_{ij}^T\theta) - \log\left[1 + \exp\left(x_{ij}^T\theta\right)\right]\big\}.$$

Per ogni iterazione dell'Algoritmo 1.4 abbiamo che lo stimatore locale è

$$\begin{aligned}\theta_j^{(t+1)} &= \arg\min_{\theta\in\Theta}\big\{\mathcal{L}_j(\theta) - \langle\nabla\mathcal{L}_j(\theta^{(t)}) - \nabla\mathcal{L}_{\mathcal{N}}(\theta^{(t)}), \theta\rangle + (\alpha/2)\|\theta - \theta^{(t)}\|_2^2\big\} \\ &= \arg\min_{\theta\in\Theta}\Big\{\frac{1}{n}\sum_{i=1}^{n}\log\left[1 + \exp\left(x_{ij}^T\theta\right)\right] - \langle\hat{w}_j + \eta_j^t + \alpha\theta^{(t)}, \theta\rangle + (\alpha/2)\|\theta\|_2^2\Big\},\end{aligned}$$

dove $\hat{w}_j = \frac{1}{k}\sum_{i=1}^{n} y_{ij}x_{ij}$ e $\eta_j^t = \nabla\mathcal{L}_j(\theta^{(t)}) - \nabla\mathcal{L}_{\mathcal{N}}(\theta^{(t)})$. Si assume di inizializzare il metodo di ottimizzazione con $\theta_{j,(0)}^{(t+1)} = \theta^{(t)}$. Un'iterazione dell'algoritmo *Newton-Raphson* applicato al problema di minimizzazione appena riportato è

$$\theta_{j,(r+1)}^{(t+1)} = \theta_{j,(r)}^{(t+1)} - \Big(\frac{1}{n}\mathbf{X}_j^T\mathbf{W}_{j,(r)}^{(t+1)}\mathbf{X}_j + \alpha\mathbf{I}\Big)^{-1}\Big[\frac{1}{n}\mathbf{X}_j^T(\mathbf{p}_{j,(r)} - \mathbf{y}_j) - \eta_j^t + \alpha(\theta_{j,(r)}^{(t+1)} - \theta^{(t)})\Big], \quad (1.25)$$

in cui l'indice r identifica il numero dell'iterazione, $\mathbf{X}_j \in \mathbb{R}^{n\times p}$ è la matrice di disegno con i dati della macchina $\mathcal{M}_j$, $\mathbf{y}_j \in \mathbb{R}^n$ denota il vettore $\{y_{ij}\}_{i=1}^n$, $\mathbf{p}_{j,(r)} = \{\pi_{ij}(\theta_{j,(r)}^{(t+1)})\}_{i=1}^n \in \mathbb{R}^n$, e $\mathbf{W}_{j,(r)}^{(t+1)}$ è la matrice diagonale con elementi $\{\pi_{ij}(\theta_{j,(r)}^{(t+1)})(1 - \pi_{ij}(\theta_{j,(r)}^{(t+1)}))\}_{i=1}^n$. Se

tale aggiornamento viene effettuato fino a convergenza allora si sta ottimizzando in modo esatto, se invece viene effettuata una sola iterazione, si ottiene l'aggiornamento tramite approssimazione quadratica descritto nei paragrafi precedenti. Dopo di ciò è sufficiente effettuare la media degli stimatori locali ottenuti nelle singole macchine ed ottenere $\theta^{(t+1)}$. Come accennato in precedenza, per l'Algoritmo 1.2 è sufficiente impostare $\alpha = 0$, mentre per gli Algoritmi 1.3 e 1.1 è necessario omettere l'operazione di media rispettivamente negli Algoritmi 1.4 e 1.2.

Per quanto riguarda la definizione degli intervalli di confidenza, ci si focalizza sugli Algoritmi 1.3 e 1.1, dato che viene definito un unico surrogato utilizzando una macchina predefinita, da noi considerata la prima. La stima della matrice di varianza e covarianza asintotica Σ, riportata nella formula (1.24), valutata nella stima restituita dall'ultima iterazione $\theta^{(T)}$, diventa

$$\hat{\Sigma}'' = \Big(\frac{1}{n}\mathbf{X}_1^T\mathbf{W}_1^{(T)}\mathbf{X}_1 + \alpha\mathbf{I}\Big)^{-1}, \tag{1.26}$$

dove $\mathbf{W}_1^{(T)}$ è la matrice diagonale con elementi $\{\pi_{i1}(\theta^{(T)})(1-\pi_{i1}(\theta^{(T)}))\}_{i=1}^n$. Per la definizione degli intervalli di confidenza è necessario procedere come nella formula (1.12) in cui è sufficiente sostituire la stima di Σ appena formulata.

1.2.7 CEASE con correzione di Firth

Vista l'utilità della correzione di Firth nel garantire stime finite e trovandosi in un contesto in cui i dati sono distribuiti e la numerosità campionaria per macchina potrebbe non essere molto elevata, ci si è interessati a inserirla all'interno del metodo CEASE, cercando di capire se potesse apportare alcune migliorie a quest'ultimo. Ci si aspetta che tale modifica non aiuti a ridurre necessariamente l'errore di ottimizzazione $\|\theta^{(t)}-\hat{\theta}\|_2$ ma che piuttosto contribuisca alla diminuzione dell'errore statistico $\|\theta^{(t)}-\theta^*\|_2$. Infatti, la successione di stime $\{\theta^{(t)}\}_{t=0}^T$ potrebbe convergere in qualche modo non più alla stima di massima verosimiglianza globale $\hat{\theta}$ ma bensì alla stima globale ottenuta con correzione di Firth $\hat{\theta}^F$, la quale si sa avere una minore distorsione della precedente.

Un primo approccio naturale per riuscire in tale intento sarebbe quello di definire una nuova funzione obiettivo globale aggiungendo a quella basata sulla verosimiglianza, denotata con $\mathcal{L}_N(\theta)$, la correzione di Firth, come mostrato di seguito

$$\mathcal{L}_N^F(\theta) = \mathcal{L}_N(\theta) - \frac{1}{2N}\log|i_N(\theta)|,$$

dove $i_N(\theta)$ è l'informazione attesa valutata usando tutti i dati. A questo punto, per ottenere la formulazione delle funzioni obiettivo locali, sarebbe necessario suddividere

$\mathcal{L}_N^F(\theta)$ nei contributi additivi apportati dai vari sottogruppi di dati memorizzati nelle macchine. Purtroppo però, ciò non è possibile per il termine additivo dato dalla correzione di Firth, infatti

$$\log|i_N(\theta)| = \log\left|\sum_{j=1}^{k} i_j(\theta)\right| \neq \sum_{j=1}^{k}\log|i_j(\theta)|.$$

Quindi, non potendo procedere sotto questa prospettiva, l'idea su cui ci si è basati è stata analoga a quella utilizzata per definire la prima modifica della metodologia CSL nel Paragrafo 1.1.4, in cui le penalità sono state definite localmente. Nel dettaglio, si è considerata tale correzione solo nella specificazione dei gradienti delle funzioni obiettivo locali, ricavando $\nabla\mathcal{L}_j^F(\theta)$ come visto nella formula (1.13) per ogni macchina, e inserendo la loro media al posto di $\nabla\mathcal{L}_N(\theta)$. Dopo di che queste nuove formulazioni dei gradienti locali e di quello globale sono state utilizzate all'interno dell'algoritmo CEASE ogni qualvolta venissero richieste.

Così facendo, nella regressione logistica, l'r-esima iterazione dell'algoritmo *Newton-Raphson* (1.25) si modifica in

$$\theta_{j,(r+1)}^{(t+1)} = \theta_{j,(r)}^{(t+1)} - \left(\frac{1}{n}\mathbf{X}_j^T\mathbf{W}_{j,(r)}^{(t+1)}\mathbf{X}_j + \alpha\mathbf{I}\right)^{-1}\left[\frac{1}{n}\mathbf{X}_j^T(\mathbf{p}_{j,(r)} - \mathbf{y}_j) + \mathbf{X}_j^T\mathbf{W}\xi_{j,(r)} - \eta_j^t + \alpha(\theta_{j,(r)}^{(t+1)} - \theta^{(t)})\right],$$

dove $\mathbf{W}\xi_{j,(r)}$ è un vettore n-dimensionale con i-esimo elemento $h_{ij}(\pi_{ij}(\theta_{j,(r)}^{(t+1)} - \frac{1}{2})$ e h_{ij} è l'i-esimo elemento diagonale della matrice $\mathbf{H}_j$, definita come nella formula (10) ma utilizzando le matrici $\mathbf{X}_j$ e $\mathbf{W}_j$.

Relativamente agli intervalli di confidenza invece, non considerando la correzione di Firth all'interno dell'hessiano, la formulazione dello stimatore di Σ rimane invariata e quindi uguale a quella riportata in (1.26).

Nella Capitolo 3 tale modifica verrà confrontata con gli altri metodi in esame ma non la si valuterà con $\alpha = 0$ in quanto, già in precedenza, si è potuto riscontrare che conduce l'algoritmo verso performance peggiori.

1.3 Approccio bayesiano

In Jordan et al. (2019, Paragrafo 3.3) viene presentato l'approccio all'inferenza distribuita bayesiana, per modelli parametrici regolari, all'interno del *framework* CSL. Si

consideri la generica formulazione di una distribuzione a posteriori globale

$$p(\theta|Z_1^N) = D \exp\left\{-\sum_{i=1}^{n}\sum_{j=1}^{k}\mathcal{L}(\theta; z_{ij})\right\}p(\theta),$$

dove D è la costante di normalizzazione e $p(\theta)$ è una distribuzione a priori definita nello spazio parametrico Θ. In questo paragrafo si assumerà sempre che la funzione di perdita $\mathcal{L}$ sia l'opposto della funzione di log-verosimiglianza. La soluzione proposta consiste nella definizione di un surrogato della posteriori

$$\tilde{p}_N(\theta) \propto \exp\big(-N\tilde{\mathcal{L}}(\theta)\big)p(\theta), \tag{1.27}$$

in cui il log-verosimiglianza globale viene sostituita dal suo surrogato $\tilde{\mathcal{L}}(\theta)$. Tale surrogato della posteriori può quindi essere utilizzato per l'implementazione di algoritmi MCMC (*Markov Chain Monte Carlo*) con cui ottenere valori simulati dalla posteriori da utilizzare per condurre l'inferenza statistica. Affinché il surrogato $\tilde{p}_N(\cdot)$ sia abbastanza simile alla distribuzione a posteriori globale $p(\cdot|Z_1^N)$, è necessario che il primo venga costruito a partire da una stima sufficientemente vicina al vero valore θ^*. Nella pratica quindi si potrebbe utilizzare lo stimatore $\theta^{(T)}$ derivante dall'algoritmo ILEA (si veda l'Algoritmo 1.1) con un numero adeguato T di iterazioni.

L'approccio appena descritto riguarda la versione asimmetrica della metodologia CSL, in quanto la funzione $\tilde{\mathcal{L}}$, utilizzata per la costruzione del surrogato della posteriori, viene definita prendendo come riferimento una singola macchina. Non è ancora stata formulata in letteratura un'alternativa simmetrica per la risoluzione dell'inferenza bayesiana. Rispetto ad altri approcci distribuiti in ambito bayesiano, quello preso in analisi è efficiente dal punto di vista del costo di comunicazione, infatti consiste nell'esecuzione di un'unica catena di Markov in una macchina locale, non richiedendo il trasferimento di più campioni dalla posteriori ricavati da diverse macchine locali. Si noti che l'algoritmo MCMC eseguito localmente per generare dal surrogato della posteriori, utilizza n osservazioni anziché N per la valutazione del surrogato $\tilde{\mathcal{L}}$, comportando la riduzione della complessità computazionale di un fattore pari a k. Per questo motivo la metodologia proposta può risultare utile anche in un contesto bayesiano non distribuito, aiutando a ridurre l'onere computazionale e garantendo risultati molto simili a quelli che si possono ottenere utilizzando tutti i dati.

Per quanto riguarda la metodologia CEASE, in Fan et al. (2021) non è stata trattata una sua formulazione per l'inferenza bayesiana, ma essendo un'estensione del metodo CSL, si proporrà una naturale modifica dell'approccio definito per quest'ultimo. Si

ricorda che il metodo CEASE ha come obiettivo quello di risolvere le problematiche di convergenza, riscontrabili con n ridotto rispetto a d, dell'algoritmo iterativo ILEA. Esso quindi si concentra nella definizione di stime puntuali ma sarebbe sicuramente d'interesse ampliarlo a diversi aspetti dell'inferenza statistica, ottenendo quindi una vera e propria estensione del *framework* inferenziale CSL, che ne permetta di risolvere le problematiche derivanti da una numerosità campionaria locale insufficiente.

L'estensione dell'inferenza bayesiana che si propone per l'algoritmo CEASE, consiste nel aggiungere la penalizzazione quadratica, derivante dall'algoritmo *proximal-point*, al surrogato della log-verosimiglianza $\tilde{\mathcal{L}}$ nella formula (1.27), ottenendo il seguente nuovo surrogato della posteriori

$$\tilde{p}_N^{\text{CEASE}}(\theta) \propto \exp\left(-N(\tilde{\mathcal{L}}(\theta) + (\alpha/2)\|\theta - \tilde{\theta}\|_2^2)\right)p(\theta), \tag{1.28}$$

dove $\tilde{\theta}$ è la stima utilizzata per la definizione di $\tilde{\mathcal{L}}$. Chiaramente le osservazioni fatte in precedenza valgono anche in questo caso. Considerazioni aggiuntive riguardano la scelta del parametro α, la quale non necessariamente coincide con la scelta da effettuare per garantire le proprietà di convergenza dell'algoritmo. In questa situazione infatti l'obiettivo è differente ed è quello di aggiungere una penalità alle funzioni obiettivo locali in modo tale che non solo si assomiglino tra di loro ma siano il più simile possibile alla funzione obiettivo globale. A tal proposito sarà dunque opportuno approfondire, come interesse futuro, la scelta di questo parametro.

Capitolo 2

Sottocampionamento ottimale

Nel capitolo corrente si andranno a considerare delle tecniche basate sul sottocampionamento tramite pesi ottimali opportunamente definiti. In particolare, si analizzerà il metodo di sottocampionamento ottimale di Poisson nel contesto delle stime di quasi-verosimiglianza proposto in Yu et al. (2020). Tale approccio non è necessariamente pensato per architetture distribuite ma si può applicare in una qualsiasi situazione in cui si ha a che fare con grandi moli di dati da analizzare. Proprio per questo motivo, nella quasi completa totalità della trattazione che segue, non si ricorrerà alla notazione introdotta all'inizio del Capitolo 1 per il *framework* distribuito.

In ogni caso la metodologia di sottocampionamento presa in esame si presta bene quando i dati sono suddivisi in sottogruppi contenuti in diverse macchine e si analizzerà anche una proposta di sottocampionamento distribuito. Infatti il sottocampionamento di Poisson (Sarndal et al., 1992), a differenza del sottocampionamento con reinserimento con probabilità non uguali, non richiede di accedere contemporaneamente alle probabilità di estrazione di tutti i dati ma necessità unicamente di averle a disposizione una alla volta. Quindi questo permette di utilizzare meno memoria e di avere una maggiore efficienza computazionale, soprattutto se permette di evitare di trasmettere informazioni tra macchine.

Per non limitarsi al modello di regressione logistica, tale approccio si basa sugli stimatori di quasi-verosimiglianza i quali richiedendo unicamente assunzioni sui momenti della variabile risposta e permettono quindi di supportare più modelli statistici.

Si introduce ora il modello e le notazioni ad esso associate, le quali verranno utilizzate per tutto il capitolo. Sia $\{(x_i, y_i)\}_{i=1}^N$ una sequenza di realizzazioni di variabili casuali indipendenti, dove, come visto in precedenza, $x_i \in \mathbb{R}^d$ è il vettore delle covariate e y_i è la variabile risposta. Si consideri Y_i la variabile casuale da cui è stato generato y_i, e si

ipotizza che

$$\mathbb{E}(Y_i|x_i) = \psi(x_i^T\theta), \quad i = 1, \ldots, N,$$

dove $\theta \in \mathbb{R}^d$ è sempre un vettore di parametri ignoti e $\psi(\cdot)$ è una funzione continua e differenziabile due volte tale che $\dot{\psi}(t) = d\psi(t)/dt > 0$ per ogni t. Lo stimatore di quasi-verosimiglianza $\hat{\theta}_{QLE}$ è la soluzione della seguente equazione di stima:

$$Q(\theta) = \sum_{i=1}^{N} \{y_i - \psi(x_i^T\theta)\} x_i = 0. \tag{2.1}$$

Avendo quindi introdotto il modello di riferimento, la trattazione si andrà principalmente a sviluppare presentando in modo generico il sottocampionamento di Poisson, con due alternative di scelta ottimale delle probabilità di estrazione, per poi analizzare un'implementazione pratica di queste ultime.

Per maggiori dettagli dei metodi proposti e delle relative analisi che verranno presentate in questo capitolo, si consiglia di consultare Yu et al. (2020) e Wang et al. (2018).

2.1 Algoritmo Generale di Sottocampionamento di Poisson

In primo luogo è opportuno analizzare in generale il metodo di sottocampionamento e le sue proprietà asintotiche. Sia p_i la probabilità di estrarre l'osservazione i-esima per $i = 1, \ldots, N$, e sia S un insieme delle osservazioni campionate con le corrispondenti probabilità di estrazione. Un algoritmo generale di sottocampionamento di Poisson è presentato nell'Algoritmo 2.1.

Come accennato in precedenza, un vantaggio evidente del sottocampionamento di Poisson è che la decisione di includere o meno ogni osservazione (x_i, y_i) viene fatta basandosi solamente su p_i. Scansionando tutti i dati per effettuare la selezione,le probabilità p_i possono essere usate una alla volta o a blocchi comportando che per tale procedura non ci siano vincoli di memoria quando si ha a che fare con tanti dati.

La dimensione del sottocampione selezionato tramite l'Algoritmo 2.1, denotata con r^*, è casuale e tale che $\mathbb{E}(r^*) = \sum_{i=1}^{N} p_i$. In seguito si indicherà con $r = \sum_{i=1}^{N} p_i$ la dimensione attesa del sottocampione ed inoltre si assumerà che $r < N$, che è una condizione naturale nel contesto dei *big data*.

Per procedere nella presentazione dei risultati asintotici sono necessarie le seguenti assunzioni, spiegate solo intuitivamente di seguito (per maggiori dettagli si veda Yu

Algorithm 2.1: Algoritmo generale di sottocampionamento di Poisson

Input: numerosità campionaria r

Inizializzazione: $S = \varnothing$ e definizione di $\mathbf{p} = \{p_i\}_{i=1}^N$;

for $i = 1, \ldots, N$ **do**

  Generare una variabile bernoulliana $\delta_i \sim \text{Bernoulli}(p_i)$;

  if $\delta_i = 1$ **then**

    Aggiornare $S = S \cup \{(x_i, y_i, p_i)\}$;

Stima: risoluzione della seguente equazione di stima pesata per ottenere $\tilde{\theta}$ basato sul sottocampione S,

$$Q^*(\theta) = \sum_S \frac{1}{p_i}\{y_i - \psi(x_i^T\theta)\}x_i = 0; \tag{2.2}$$

Output: $\tilde{\theta}$

et al., 2020, Assunzioni 1-5):

1. un'assunzione sullo spazio parametrico Θ e sul vero valore del parametro θ^* che permette di garantire la consistenza dello stimatore;

2. alcune assunzioni sui momenti, comunemente soddisfatte da molti esempi di modelli lineari generalizzati;

3. un'assunzione che riguarda la convessità della funzione di quasi-verosimiglianza e che quindi assicura che lo stimatore associato sia unico;

4. alcune restrizioni sulla regolarità delle funzioni $\psi(\cdot)$ e $\dot{\psi}(\cdot)$;

5. un'ultima assunzione che pone dei limiti ai pesi che si utilizzano nell'equazione di stima (2.19).

Si definisca $\Sigma_\psi(\theta) = N^{-1}\sum_{i=1}^N \dot{\psi}(x_i^T\theta)x_i x_i^T$ e si riutilizzi la notazione introdotta nel capitolo precedente $Z_1^N = \{(x_i, y_i)\}$, in cui in questo caso non si considerà più la suddivisione dei dati in più macchine.

Il seguente teorema presenta la consistenza dello stimatore ottenuto dall'Algoritmo 2.1 rispetto allo stimatore $\hat{\theta}_{QLE}$ ottenuto su tutti i dati (si veda Yu et al., 2020, Teorema 1).

Teorema 2.1.1. Se le assunzioni riportate in precedenza sono valide, allora con $N \to \infty$ e $r \to \infty$, $\tilde{\theta}$ è consistente in probabilità rispetto a $\hat{\theta}_{QLE}$, condizionatamente a Z_1^N. Inoltre, il tasso di convergenza è $r^{-1/2}$. Cioè, con una probabilità prossima ad uno, per ogni $\epsilon > 0$, esistono dei valori finiti Δ_ϵ e r_ϵ tali che

$$P(\|\tilde{\theta} - \hat{\theta}_{QLE}\| \geq r^{-1/2}\Delta_\epsilon | Z_1^N) < \epsilon, \quad \forall r > r_\epsilon. \tag{2.3}$$

Il risultato sulla consistenza mostra che l'errore di approssimazione può essere reso il più piccolo possibile utilizzando una dimensione del sottocampione r abbastanza grande, dato che l'errore di approssimazione è dell'ordine di $O_{p|Z_1^N}(r^{-1/2})$, notazione che indica che per definire l'ordine asintotico in probabilità ci si sta condizionando alla totalità dei dati a disposizione.

Il risultato riguardante la normalità asintotica è invece presentato nel teorema sottostante (si veda Yu et al., 2020, Teorema 2).

Teorema 2.1.2. Se le assunzioni specificate in precedenza sono valide, allora con $N \to \infty$ e $r \to \infty$, condizionatamente a Z_1^N, si ha che

$$\mathbf{V}^{-1/2}(\tilde{\theta} - \hat{\theta}_{QLE}) \xrightarrow{d} N_d(0, \mathbf{I}), \tag{2.4}$$

dove

$$\mathbf{V} = \Sigma_\psi(\hat{\theta}_{QLE})^{-1}\mathbf{V}_c\Sigma_\psi(\hat{\theta}_{QLE})^{-1} \tag{2.5}$$

e

$$\mathbf{V}_c = \frac{1}{N^2}\sum_{i=1}^{N}\frac{\{y_i - \psi(x_i^T\hat{\theta}_{QLE})\}^2 x_i x_i^T}{p_i} - \frac{1}{N^2}\sum_{i=1}^{N}\{y_i - \psi(x_i^T\hat{\theta}_{QLE})\}^2 x_i x_i^T. \tag{2.6}$$

Quando $r/N \to 0$, il secondo termine a destra dell'uguale nella formula (2.6) può essere ignorato. In Yu et al. (2020) si osserva che, in questo caso, il risultato ottenuto nel Teorema 2.1.2 è lo stesso di quello ottenuto per il campionamento con reinserimento nella regressione logistica (si veda Wang et al., 2018, Teorema 2). In ogni caso, quando $r/N \to c \in (0, 1]$, il sottocampionamento di Poisson porta ad avere una varianza più piccola per lo stimatore.

Nei Teoremi 2.1.1 e 2.1.2 i risultati distributivi che vengono riportati valgono condizionatamente ai dati osservati. L'inferenza condizionata in statistica è abbastanza comune e il metodo più popolare in questo ambito è il bootstrap (Efron, 1979). Si noti che il bootstrap non parametrico non è altro che il campionamento con reinserimento con pesi uniformi, in cui la dimensione del sottocampione corrisponde alla dimensione campionaria totale. I risultati asintotici in Wang et al. (2018) per il campionamento con reinserimento nella regressione logistica si riconducono a quelli del bootstrap impostando probabilità uniformi e $r = N$. In ogni caso, il boostrap e i metodi di sottocampionamento hanno degli obiettivi differenti: il primo si focalizza nell'approssimare distribuzioni complicate e viene usato quando non ci sono soluzioni esplicite; il secondo ha come motivazione principale il raggiungimento di un costo computazionale sostenibile e viene usato anche quando ci sono soluzioni in forma chiusa.

2.2 Sottocampionamento di Poisson ottimale

Per implementare l'Algoritmo 2.1, bisogna specificare delle probabilità di selezione $\mathbf{p} = \{p_i\}_{i=1}^N$ per tutti i dati. Una scelta semplice è quella uniforme $\mathbf{p}^{\text{UNIF}} = \{p_i = r/N\}_{i=1}^N$. Tuttavia, utilizzando delle probabilità uniformi non si ottiene un algoritmo ottimale e una scelta non uniforme di tali probabilità potrebbe portare a performance migliori. Yu et al. (2020) si sono quindi interessati di definire delle procedure di sottocampionamento più efficienti scegliendo delle probabilità p_i non uniformi che permettano di *minimizzare* la matrice di varianza e covarianza asintotica $\mathbf{V}$ in (2.21). Però, essendo $\mathbf{V}$ una matrice, risulta necessario specificare che cosa si intende per *minimizzare*. Nel seguito saranno descritti due criteri di minimizzazione e le due corrispondenti definizioni delle probabilità ottimali.

2.2.1 Minimo MSE asintotico di $\tilde{\theta}$

Un primo criterio di minimizzazione suggerito da Yu et al. (2020) si basa sull'idea dell'*A-ottimalità* nell'ambito del disegno ottimale degli esperimenti, la quale induce un ordinamento tra le matrici di varianza e covarianza utilizzando la traccia di una matrice (Kiefer, 1959). Perciò l'intento è quello di minimizzare la traccia $tr(\mathbf{V})$, operazione che è equivalente a minimizzare l'errore quadratico medio (MSE) asintotico di $\tilde{\theta}$ nell'approssimare $\hat{\theta}_{QLE}$. Nel dettaglio, se identifichiamo con $\mathbf{u}$ una variabile casuale con distribuzione $N_d(0, \mathbf{V})$, si ha

$$\tilde{\theta} - \hat{\theta}_{QLE} | Z_1^N \dot{\sim} \mathbf{u},$$

dove $\dot{\sim}$ indica che asintoticamente i due termini hanno la stessa distribuzione, e l'MSE asintotico di $\tilde{\theta}$ corrisponde a $\mathbb{E}(\|\mathbf{u}\|^2 | Z_1^N)$.

Il seguente teorema fornisce una formulazione delle probabilità di estrazione p_i basandosi proprio sul suddetto criterio e indicheremo tale approccio di sottocampionamento con la sigla mMSE (si veda Yu et al., 2020, Teorema 3).

Teorema 2.2.1. Per facilità di presentazione, si definiscono

$$\hbar_i^{\text{mMSE}} = |y_i - \psi(x_i^T \hat{\theta})| \|\Sigma_\psi(\hat{\theta}_{QLE})^{-1} x_i\|, \quad i = 1, \ldots, N, \tag{2.7}$$

e si denoti con $\hbar_{(1)}^{\text{mMSE}} \leq \hbar_{(2)}^{\text{mMSE}} \leq \cdots \leq \hbar_{(N)}^{\text{mMSE}}$ la statistica ordinata di $\{\hbar_i^{\text{mMSE}}\}_{i=1}^N$. Per convenienza si considera $\hbar_{(N+1)}^{\text{mMSE}} = +\infty$ e si assume che $\hbar_{(N-r)}^{\text{mMSE}} > 0$. L'MSE asintotico di $\tilde{\theta}$, $tr(\mathbf{V})$, raggiunge il suo minimo se le probabilità p_i nell'Algoritmo 2.1 sono scelte

essere

$$p_i^{\text{mMSE}} = r\frac{\hbar_i^{\text{mMSE}} \wedge M}{\sum_{j=1}^{N}(\hbar_j^{\text{mMSE}} \wedge M)}, \tag{2.8}$$

dove $a \wedge b = \min(a, b)$,

$$M = \frac{1}{r-K}\sum_{i=1}^{N-K}\hbar_{(i)}^{\text{mMSE}}, \tag{2.9}$$

e

$$K = \min\left\{ s \;\middle|\; 0 \le s \le r,\ (r-s)\hbar_{(N-s)}^{\text{mMSE}} < \sum_{i=1}^{N-s}\hbar_{(i)}^{\text{mMSE}} \right\}. \tag{2.10}$$

Si osservi che nella formula (2.8), se $r\hbar_{(N)}^{\text{mMSE}}/\left(\sum_{j=1}^{N}\hbar_j^{\text{mMSE}}\right) < 1$, allora $\hbar_{(N)}^{\text{mMSE}} < M = r^{-1}\sum_{j=1}^{N}\hbar_j^{\text{mMSE}}$ e le probabilità di sottocampionamento ottimali si riducono a $p_i^{\text{mMSE}} = r\hbar_i^{\text{mMSE}}/\left(\sum_{j=1}^{N}\hbar_j^{\text{mMSE}}\right)$. In questo caso, tutte le probabilità p_i^{mMSE} sono minori di uno e quindi l'estrazione di ogni osservazione è casuale. Se invece $r\hbar_i^{\text{mMSE}}/\left(\sum_{j=1}^{N}\hbar_j^{\text{mMSE}}\right) \ge 1$ per alcuni i, di conseguenza alcuni p_i^{mMSE} saranno posti uguali a uno. In questa situazione, K corrisponde al numero di p_i^{mMSE} che sono uguali a uno e M è la soglia che soddisfa

$$\max_{i=1,\dots,N}\frac{r(\hbar_i^{\text{mMSE}} \wedge M)}{\sum_{j=1}^{N}(\hbar_j^{\text{mMSE}} \wedge M)} = 1.$$

Come si può osservare dalle quantità definite in (2.7), le probabilità ottimali di sottocampionamento $\mathbf{p}^{\text{mMSE}} = \{p_i^{\text{mMSE}}\}_{i=1}^{N}$ dipendono direttamente dai dati sia tramite le covariate che attraverso la risposta. Per quanto riguarda le covariate, i termini $\|\Sigma_\psi(\hat{\theta}_{QLE})^{-1}x_i\|$ descrivono la struttura informativa di quest'ultime e sono simili a dei punteggi leva. La variabile risposta invece, influisce sulle probabilità ottimali di sottocampionamento tramite $|y_i - \psi(x_i^T\hat{\theta}_{QLE})|$, il che intuitivamente permette allo stimatore basato sul sottocampione di essere più robusto.

Nel caso della regressione logistica per dati binari, si ha che $\psi(x_i^T\theta) = \pi_i(\theta)$ e, per come è definito l'approccio mMSE di sottocampionamento, si avrà che osservazioni con $\pi_i(\hat{\theta}_{QLE})$ piccolo e $y_i = 1$ e osservazioni con $\pi_i(\hat{\theta}_{QLE})$ grande e $y_i = 0$ verranno selezionate più probabilmente. Dunque, viene data una preferenza maggiore alle osservazioni che sono più propense ad essere classificate erroneamente.

Come si è già trattato nella parte introduttiva di questo lavoro di tesi, l'esistenza delle stime nella regressione logistica non è sempre garantita. Infatti si ottengono delle stime infinite quando si riscontrano situazioni di perfetta separazione o quasi-perfetta separazione nei dati. Tali situazioni problematiche, equivalentemente a quanto detto nel capitolo introduttivo, si riscontrano quando

$$\pi_i(\theta) \le 0.5\ \forall i \in \{i : y_i = 0\}, \quad \pi_i(\theta) \ge 0.5\ \forall i \in \{i : y_i = 1\}. \tag{2.11}$$

A tal proposito, il sottocampionamento ottimale si sforza di aumentare la sovrapposizione tra questi due insiemi di osservazioni, definiti dalle modalità della risposta, portandoli nella direzione di $\pi_i(\hat{\theta}_{QLE})$. Così facendo, viene diminuita la probabilità che le stime basate su un sottocampione non esistano finite.

2.2.2 Minimo MSE asintotico di $\Sigma_\psi(\hat{\theta}_{QLE})\tilde{\theta}$

Il sottocampionamento ottimale mMSE richiede il calcolo di $\|\Sigma_\psi(\hat{\theta}_{QLE})^{-1}x_i\|$ per $i = 1, \ldots, N$, il che necessita di un tempo di esecuzione dell'ordine di $O(Nd^2)$. In questo paragrafo verrà descritto un criterio di ottimalità modificato rispetto a quello visto in precedenza, che permetterà di ridurre il tempo di calcolo delle probabilità di selezione ottimali.

In Wang et al. (2018), il nuovo criterio ottimale viene formulato tramite una nuova definizione di ordinamento parziale di matrici definite positive. Per due matrici definite positive $\mathbf{A}_1$ e $\mathbf{A}_2$, $\mathbf{A}_1 \geq \mathbf{A}_2$ se e solo se $\mathbf{A}_1 - \mathbf{A}_2$ è una matrice semidefinita positiva. Si noti che la matrice di varianza e covarianza asintotica $\mathbf{V}$ in (2.21) dipende da $\mathbf{p}$ solo attraverso $\mathbf{V}_c$. Dati due vettori di probabilità $\mathbf{p}^{(1)}$ e $\mathbf{p}^{(2)}$, $\mathbf{V}(\mathbf{p}^{(1)}) \leq \mathbf{V}(\mathbf{p}^{(2)})$ se e solo se $\mathbf{V}_c(\mathbf{p}^{(1)}) \leq \mathbf{V}_c(\mathbf{p}^{(2)})$. Questa osservazione ha permesso di cambiare prospettiva e di focalizzarsi su $\mathbf{V}_c$ per la definizione del nuovo criterio. Nello specifico, anziché minimizzare $tr(\mathbf{V})$ come proposto nel Paragrafo 2.2.1, si minimizza $tr(\mathbf{V}_c)$. In questo modo si sta minimizzando l'MSE asintotico di $\Sigma_\psi(\hat{\theta}_{QLE})\tilde{\theta}$ nell'approssimare $\Sigma_\psi(\hat{\theta}_{QLE})\hat{\theta}_{QLE}$. La procedura che ne risulta essenzialmente corrisponde con il criterio di ottimalità lineare (L-ottimalità) per il disegno ottimale degli esperimenti (si veda Atkinson et al., 2007, Capitolo 10), il quale mira a migliorare la qualità dello stimatore per alcune sue combinazioni lineari di coefficienti ignoti.

Il seguente teorema specifica le probabilità ottimali che si ottengono seguendo questo nuovo criterio di ottimalità, il quale verrà riferito nel seguito con la sigla mVc (si veda Yu et al., 2020, Teorema 4).

Teorema 2.2.2. Siano

$$\hbar_i^{\mathrm{mVc}} = |y_i - \psi(x_i^T\hat{\theta})|\|x_i\|, \quad i = 1, \ldots, N, \tag{2.12}$$

e si denoti con $\hbar_{(1)}^{\mathrm{mVc}} \leq \hbar_{(2)}^{\mathrm{mVc}} \leq \cdots \leq \hbar_{(N)}^{\mathrm{mVc}}$ la statistica ordinata di $\{\hbar_i^{\mathrm{mVc}}\}_{i=1}^N$. Per convenienza si considera $\hbar_{(N+1)}^{\mathrm{mVc}} = +\infty$ e si assume che $\hbar_{(N-r)}^{\mathrm{mVc}} > 0$. La traccia di $\mathbf{V}_c$ definita in (2.6) raggiunge il suo minimo se le probabilità p_i nell'Algoritmo 2.1 sono

scelte essere

$$p_i^{\mathrm{mVc}} = r\frac{\hbar_i^{\mathrm{mVc}} \wedge M}{\sum_{j=1}^{N}(\hbar_j^{\mathrm{mVc}} \wedge M)}, \tag{2.13}$$

dove

$$M = \frac{1}{r-K}\sum_{i=1}^{N-K} \hbar_{(i)}^{\mathrm{mVc}}, \tag{2.14}$$

e

$$K = \min\left\{ s \;\middle|\; 0 \le s \le r,\ (r-s)\hbar_{(N-s)}^{\mathrm{mVc}} < \sum_{i=1}^{N-s} \hbar_{(i)}^{\mathrm{mVc}} \right\}. \tag{2.15}$$

I risultati strutturali per $\mathbf{p}^{\mathrm{mVc}} = \{p_i^{\mathrm{mVc}}\}_{i=1}^{N}$ e $\mathbf{p}^{\mathrm{mMSE}}$ sono simili. Ciò che cambia tra i due metodi è l'effetto delle covariate: $\mathbf{p}^{\mathrm{mVc}}$ utilizza $\|x_i\|$ anziché $\|\Sigma_\psi(\hat{\theta}_{QLE})^{-1}x_i\|$. L'obiettivo primario di questa modifica risulta quindi evidentemente soddisfatto: per il calcolo di $\|x_i\|$ per $i, \dots, N$ viene richiesto un tempo dell'ordine di $O(Nd)$ mentre, come si è visto in precedenza, tale ordine è di $O(Nd^2)$ per la metodologia mMSE.

2.2.3 Implementazione

Per facilità di presentazione, nel seguito si utilizzerà la notazione unificata p_i^{os} per indicare le probabilità ottimali p_i^{mMSE} e p_i^{mVc} ottenute nei Teoremi 2.2.1 e 2.2.2 rispettivamente. Si precisa quindi che,

$$p_i^{\mathrm{os}} = r\frac{\hbar_i^{\mathrm{os}} \wedge M}{\sum_{j=1}^{N}(\hbar_j^{\mathrm{os}} \wedge M)} = r\frac{\hbar_i^{\mathrm{os}} \wedge M}{N\Psi}, \quad i = 1, \dots, N, \tag{2.16}$$

dove $M = (r-K)^{-1}\sum_{i=1}^{N-K} \hbar_{(i)}^{\mathrm{os}}$, $\Psi = N^{-1}\sum_{j=1}^{N}(\hbar_j^{\mathrm{os}} \wedge M)$, e $\hbar_i^{\mathrm{os}}$ corrisponde o a $\hbar_i^{\mathrm{mMSE}}$ o a $\hbar_i^{\mathrm{mVc}}$. Osservando le formule (2.7) e (2.12) ci si può accorgere che le probabilità ottimali definite dipendono da $\hat{\theta}_{QLE}$, ossia dallo stimatore che si ottiene utilizzando tutti i dati, quindi non si possono calcolare esattamente. Per questo motivo Yu et al. (2020) suggeriscono, per un'implementazione pratica, di sostituire l'ignoto stimatore $\hat{\theta}_{QLE}$ con uno stimatore pilota, detto $\tilde{\theta}_0$, il quale può essere ottenuto effettuando ad esempio un sottocampionamento con probabilità uniformi. Inoltre, per sfruttare il vantaggio del campionamento di Poisson e determinare le probabilità di inclusione di ogni osservazione separatamente, viene consigliato di utilizzare il campione pilota per approssimare M e Ψ.

D'ora in poi la numerosità del campione pilota verrà specificata tramite r_0. Sono stati proposti diversi modi per approssimare M:

- un primo metodo denotato con $M = E$, in cui M viene calcolato in linea con le formule riportate nei Teoremi 2.2.1 e 2.2.2 ad eccezione del fatto che $\hat{\theta}_{QLE}$ è sostituito con $\tilde{\theta}_0$;
- un secondo metodo indicato con $M = Q$, in cui M è approssimato utilizzando il quantile di ordine $(1-r/(2N))$ delle quantità $\{\hbar_i^{\text{mMSE}}\}_{i=1}^{r_0}$ o di $\{\hbar_i^{\text{mVc}}\}_{i=1}^{r_0}$, calcolate tramite il campione pilota;
- un'ultimo criterio il quale consiste nel porre $M = \infty$.

Nei contesti in cui il sottocampionamento risulta utile, è molto comune che $r \ll N$ e che quindi il numero di casi in cui $\hbar_i^{\text{os}} > M$ sia piccolo. Per questo motivo l'ultima proposta, in cui $M = \infty$, di norma non influenza significativamente le probabilità ottimali di sottocampionamento. Infatti, le simulazioni effettuate in Yu et al. (2020, Paragrafo 5.1) mostrano che la scelta $M = \infty$ non riduce l'efficienza delle stime finché il rapporto r/N è piccolo.

Per quanto riguarda Ψ, indicando con $\tilde{S}_{r_0}$ l'insieme corrispondente al sottocampione pilota, una sua approssimazione è data da

$$\hat{\Psi} = \frac{1}{|\tilde{S}_{r_0}|} \sum_{\tilde{S}_{r_0}} |y_i - \psi(x_i^T \tilde{\theta}_0)| h(x_i), \tag{2.17}$$

dove $|\tilde{S}_{r_0}|$ indica la cardinalità dell'insieme $\tilde{S}_{r_0}$, e $h(x) = \|x\|$ per mVc o $h(x) = \|\Sigma_\psi(\tilde{\theta}_0)^{-1}x\|$ per mMSE con $\Sigma_\psi(\tilde{\theta}_0)$ calcolato come $\Sigma_\psi(\tilde{\theta}_0) = |\tilde{S}_{r_0}|^{-1} \sum_{\tilde{S}_{r_0}} \dot{\psi}(x_i^T\tilde{\theta}_0) x_i x_i^T$.

Si indichino allora con $\tilde{p}_i^{\text{os}}$ le probabilità ottimali di sottocampionamento approssimate con le quantità $\hat{\theta}_{QLE}$, M e Ψ nella formula (2.16) sostituite con $\tilde{\theta}_0$, $M = \infty$ e $\hat{\Psi}$. Lo stimatore ottenuto con l'equazione di stima (2.19) e utilizzando $\tilde{p}_i^{\text{os}}$ potrebbe essere sensibile alle osservazioni con $y_i - \dot{\psi}(x_i^T\tilde{\theta}_0) \approx 0$ nel caso in cui quest'ultime venissero incluse all'interno del sottocampione. A tal riguardo, per rendere lo stimatore più stabile e robusto, Yu et al. (2020) suggeriscono di avvalersi dell'idea del sottocampionamento basato sullo *shrinkage* proposto da Ma et al. (2015). Nel dettaglio, le probabilità vengono modificate nel modo seguente

$$\tilde{p}_i^{\text{sos}} = (1-\varrho)\frac{r|y_i - \psi(x_i^T\tilde{\theta}_0)|h(x_i)}{N\hat{\Psi}} + \varrho r N^{-1}, \quad i = 1, \ldots, N, \tag{2.18}$$

dove $\varrho \in (0, 1)$.

Si osservi che, essendo $\tilde{\theta}_0$ e $\hat{\Psi}$ calcolati tramite il campione pilota, le probabilità $\tilde{p}_i^{\text{sos}}$ dipendono solo dall'osservazione i-esima (x_i, y_i). Ciò comporta che ogni probabilità può essere calcolata separatamente e che quindi non c'è la necessità di avere tutti i

dati contemporaneamente in memoria, aspetto particolarmente benefico in termini di utilizzo della memoria. Tale vantaggio è apprezzabile anche in un contesto distribuito in cui, dopo aver estratto il campione pilota ed aver calcolato e trasmesso le quantità d'interesse, la definizione delle probabilità e la successiva estrazione delle osservazioni può avvenire separatamente in ogni singola macchina, senza alcun bisogno di comunicare ulteriori informazioni tra di esse.

Nella formula (2.18), le probabilità definite sono una combinazione convessa di $\tilde{\mathbf{p}}^{\text{os}} = \{\tilde{p}_i^{\text{os}}\}_{i=1}^N$ e le probabilità di selezione uniformi e una scelta appropriata di ϱ può permettere di sfruttare i benefici di entrambe nel modo più efficiente possibile. Negli studi di simulazione condotti in Yu et al. (2020, Paragrafo 5.1) si è potuto riscontrare empiricamente che un valore $\varrho \approx 0.2$ permette di ottenere delle performance migliori a livello di MSE. Per questo motivo, quando si utilizzerà questa metodologia nello studio di simulazione svolto nel Capitolo 3, si considererà $\varrho = 0.2$.

Dato che Ψ e M vengono approssimate, alcune probabilità $\tilde{p}_i^{\text{sos}}$ potrebbero essere più grandi di uno. Per questo motivo, al posto di tali probabilità, si utilizzerà $\tilde{p}_i^{\text{sos}} \wedge 1$.

La procedura pratica che ne deriva dal calcolo e dall'utilizzo delle quantità approssimate definite in questo paragrafo è presentata nell'Algoritmo 2.2, anche definito algoritmo *two-step*.

Rispetto a quanto detto in precedenza, nell'Algoritmo 2.2 viene specificata un'operazione di unione tra il campione pilota e quello ottenuto tramite sottocampionamento ottimale, con lo scopo di poter usufruire del contenuto informativo di entrambi. Nell'unione le probabilità devono essere riscalate affinché i due campioni abbiano un peso opportuno all'interno dell'equazione di stima (2.19).

Le quantità arbitrarie definite nell'Algoritmo 2.2 sono le due numerosità campionarie r_0 e r. A tal proposito potrebbe essere d'interesse capire se vi è un criterio di scelta ottimale nel definirle e, in merito a questo aspetto, Wang et al. (2018, Paragrafo 5.1), tramite uno studio di simulazione, sono riusciti ad evidenziare empiricamente che valori del rapporto $r_0/(r_0+r)$ vicino a 0.2 sono un'ottima scelta per ottenere un algoritmo *two-step* efficiente. Sebbene questo risultato non sia sistematico, lo si terrà in considerazione come linea guida nell'implementazione di tale algoritmo.

Ci si interessa ora ad analizzare brevemente il costo di comunicazione dell'Algoritmo 2.2 in un contesto distribuito con k macchine. Se si ipotizza di avere una numerosità campionaria locale n sufficientemente grande, il campione pilota può essere estratto unicamente da una singola macchina, in modo da evitare la comunicazione tra le macchine delle osservazioni estratte. Le quantità stimate $\tilde{\theta}_0$ e $\hat{\Psi}$ vanno dunque trasmesse comportando un costo di $(k-1)(d+1)$. A questo punto ogni macchina può, indipendentemente

Algorithm 2.2: Algoritmo pratico *two-step*

Input: numerosità campionarie r_0 e r

Sottocampionamento pilota: eseguire l'Algoritmo 2.1 con dimensione media del sottocampione r_0 e $\mathbf{p}^{\text{UNIF}} = \{p_i = r_0/N\}_{i=1}^N$ per selezionare un sottocampione corrispondente all'insieme $\tilde{S}_{r_0}$, e utilizzarlo per ottenere una stima $\tilde{\theta}_0$ e $\hat{\Psi}$ come nella formula (2.17);

Inizializzazione: $S_0 = \varnothing$;

for $i = 1, \dots, N$ **do**

  Generare una variabile bernoulliana $\delta_i \sim \text{Bernoulli}(p_i)$ con $p_i = \tilde{p}_i^{\text{sos}} \wedge 1$, dove $\tilde{p}_i^{\text{sos}}$ è definito in (2.18);

  if $\delta_i = 1$ **then**

    Aggiornare $S_i = S_{i-1} \cup \{(x_i, y_i, p_i)\}$;

  else

    Impostare $S_i = S_{i-1}$;

Unione: il campione pilota e il campione ottimale vengono uniti in $S_{\text{tot}} = S_{r_0} \cup S_N$, modificando le probabilità p_i del campione pilota moltiplicandole per $(r_0 + r)/r_0$ mentre quelle del campione ottimale per $(r_0 + r)/r$, in modo tale che nell'equazione di stima (2.19) i pesi $1/p_i$ facciano influire i due campioni proporzionalmente al loro numero di osservazioni;

Stima: risoluzione della seguente equazione di stima pesata per ottenere $\breve{\theta}$ basato sul sottocampione S_{tot},

$$Q^*(\theta) = \sum_{S_{\text{tot}}} \frac{1}{p_i}\{y_i - \psi(x_i^T\theta)\}x_i = 0; \tag{2.19}$$

Output: $\breve{\theta}$

dalle altre, procedere nella selezione delle osservazioni che andranno a comporre il campione ottimale. Finita questa fase, le osservazioni campionate dovranno essere trasferite alla macchina centrale comportando un'ultimo *step* di comunicazione con costo pari all'incirca a rd. Complessivamente quindi, il costo di comunicazione è dell'ordine di $O(d(k + r))$, di netto superiore a quello dei metodi *gradient-enhanced* approfonditi nel Capitolo 1. È opportuno però sottolineare che l'utilizzo dei metodi di sottocampionamento in architetture distribuite non è possibile quando ci sono limitazioni dovute alla privacy, dato che nell'implementazione si rende necessario il trasferimento di intere osservazioni tra le macchine.

2.2.4 Standard error e intervalli di confidenza

Si andranno ora a illustrare le proprietà asintotiche derivate in Yu et al. (2020, Teoremi 5-6) per l'Algoritmo 2.2, utilizzate successivamente per definire degli opportuni intervalli di confidenza.

Teorema 2.2.3. Si ritengano valide le assunzioni elencate nel Paragrafo 2.1, se $r_0 r^{-1/2} \to 0$, allora per lo stimatore $\breve{\theta}$ ottenuto dall'Algoritmo 2.2, se $r \to \infty$ e $N \to \infty$, con probabilità prossima a uno, per ogni $\epsilon > 0$ esistono finiti Δ_ϵ e r_ϵ tali che

$$P(\|\breve{\theta} - \hat{\theta}_{QLE}\| \geq r^{-1/2}\Delta_\epsilon | Z_1^N) < \epsilon, \quad \forall r > r_\epsilon.$$

Teorema 2.2.4. Si ritengano valide le assunzioni elencate nel Paragrafo 2.1 e $r_0 r^{-1/2} \to 0$, allora con $N \to \infty$, $r \to \infty$ e $r_0 \to \infty$, condizionatamente a Z_1^N, si ha che

$$\mathbf{V}^{-1/2}(\breve{\theta} - \hat{\theta}_{QLE}) \xrightarrow{d} N_d(0, \mathbf{I}), \tag{2.20}$$

dove

$$\mathbf{V} = \Sigma_\psi(\hat{\theta}_{QLE})^{-1}\mathbf{V}_c\Sigma_\psi(\hat{\theta}_{QLE})^{-1} \tag{2.21}$$

e

$$\mathbf{V}_c = \frac{1}{N^2}\sum_{i=1}^{N} \frac{\{1 - (p_i^{\text{sos}} \wedge 1)\}\{y_i - \psi(x_i^T\hat{\theta}_{QLE})\}^2 x_i x_i^T}{p_i^{\text{sos}} \wedge 1}, \tag{2.22}$$

con p_i^{sos} definiti come nelle formula (2.18) sostituendo $\tilde{\theta}_0$ con $\hat{\theta}_{QLE}$ e $\hat{\Psi}$ con il suo valore effettivo Ψ.

La normalità asintotica nel Teorema 2.2.4 può essere utilizzata per la formulazione degli standard error. Basandosi sulla formulazione proposta da Wang et al. (2018, Paragrafo 4.2), si forniscono degli stimatori per la matrice di varianza e covarianza calcolati includendo solo i dati del sottocampione selezionato. Tale stima della matrice di varianza e covarianza di $\breve{\theta}$ è la seguente

$$\breve{\mathbf{V}} = \breve{\Sigma}_\psi(\breve{\theta})^{-1}\breve{\mathbf{V}}_c\breve{\Sigma}_\psi(\breve{\theta})^{-1}, \tag{2.23}$$

dove

$$\breve{\Sigma}_\psi(\breve{\theta}) = \frac{1}{N}\sum_{S_{\text{tot}}} \frac{\dot{\psi}(x_i^T\breve{\theta})x_i x_i^T}{p_i},$$

e

$$\breve{\mathbf{V}}_c = \frac{1}{N^2}\sum_{S_{\text{tot}}} \frac{(1-p_i)\{y_i - \psi(x_i^T\breve{\theta})\}^2 x_i x_i^T}{p_i^2}$$

Le probabilità $\{p_i\}_{i=1}^{r+r_0}$ riportate nelle formule sopra, corrispondono alle probabilità riscalate presenti nell'insieme complessivo S_{tot} introdotto nell'Algoritmo 2.2. Le stime $\breve{\Sigma}_\psi(\breve{\theta})$ e $\breve{\mathbf{V}}_c$ sono motivate dal metodo dei momenti. Gli standard error delle componenti di $\breve{\theta}$ possono essere ottenuti tramite radice quadrata degli elementi sulla diagonale della matrice $\breve{\mathbf{V}}$. La definizione di un intervallo di confidenza di livello $1-\alpha$ per l'i-esima

componente di θ risulta dunque immediata

$$\left(\breve{\theta}_i - Z_{1-\alpha/2}\sqrt{\breve{\sigma}^2_{i,i}},\ \tilde{\theta}_i + Z_{1-\alpha/2}\sqrt{\breve{\sigma}^2_{i,i}}\right),$$

dove $\breve{\sigma}^2_{i,i}$ è l'i-esimo elemento sulla diagonale della matrice $\breve{\mathbf{V}}$.

2.3 Sottocampionamento di Poisson distribuito

Come già accennato in precedenza, Yu et al. (2020, Paragrafo 4) formulano anche una proposta distribuita del sottocampionamento di Poisson, che permette di evitare di comunicare i sottocampioni estratti tra le diverse macchine, evitando problematiche dovute alla privacy e diminuendo il costo di comunicazione. L'idea è quella di trasmettere gli stimatori ottenuti localmente alla macchina principale, la quale dovrà aggregarli in modo tale da approssimare lo stimatore di quasi-verosimiglianza ottenibile con tutti i dati. Ritrovandosi nuovamente in una situazione di riferimento con architetture distribuite, si ricorrerà alla notazione introdotta all'inizio del Capitolo 1. Nell'Algoritmo 2.3 viene illustrata la procedura d'interesse.

In tale algoritmo, r_0 rimane il numero di osservazioni da estrarre per comporre il campione pilota mentre r corrisponde alla dimensione campionaria di ognuno dei sottogruppi da estrarre in ogni macchina. Quindi nel complesso si utilizza l'informazione di $r_0 + kr$ osservazioni. Siccome r_0 solitamente non è elevato, il campione pilota può venire estratto sottocampionando in ogni macchina e successivamente mettendo insieme le osservazioni, ritenendo trascurabile il costo di comunicazione. Nelle fasi seguenti invece, in termini di trasmissione di informazione, viene richiesta la comunicazione ad una macchina centrale delle stime locali $\tilde{\theta}_j$ e delle matrici $\dot{Q}^*_j(\tilde{\theta}_j)$, procedura sicuramente meno onerosa del raggruppare sottogruppi di osservazioni in un unico campione. Nonostande ciò, la trasmissione delle matrici $\dot{Q}^*_j(\tilde{\theta}_j)$ di dimensione $d \times d$ comporta un costo di comunicazione maggiore rispetto ai metodi *gradient-enhanced*, i quali richiedono solo la trasmissione dei gradienti, ossia di vettori d-dimensionali.

Risultati analoghi a quelli visti in precedenza per la proposta non distribuita sulla consistenza e la normalità asintotica sono presentati in Yu et al. (2020, Teoremi 7-8), a cui si rimanda anche per la formulazione della stima della matrice di varianza e covarianza asintotica.

Algorithm 2.3: Algoritmo di sottocampionamento ottimale di Poisson distribuito

Input: numerosità campionarie r_0 e r

Step 1: Ottenimento dello stimatore pilota

for $i = 1, \ldots, N$ **do**

Generare una variabile bernoulliana $\delta_i \sim \text{Bernoulli}(p_i)$ con $p_i = r_0/N$;

if $\delta_i = 1$ **then**

Aggiungere (x_i, y_i, p_i) all'insieme S_{r_0};

Con il sottocampione ottenuto S_{r_0}, calcolare lo stimatore pilota $\tilde{\theta}_0$,$\hat{\Psi}$ e $\dot{Q}_0$ secondo le formule (2.19), (2.17) e (2.25);

Step 2: Sottocampionamento e compressione

foreach $Z_j, j = 1, \ldots, k$ **do**

Inizializzazione: $S_{0j} = \varnothing$;

for $i = 1, \ldots, n$ **do**

Calcolare la corrispondente probabilità di sottocampionamento $\tilde{p}_{ij}^{\text{sos}}$ in linea con (2.18);

Generare una variabile bernoulliana $\delta_{ij} \sim \text{Bernoulli}(p_{ij})$ con $p_{ij} = \tilde{p}_{ij}^{\text{sos}} \wedge 1$;

if $\delta_{ij} = 1$ **then**

Aggiornare $S_{ij} = S_{i-1j} \cup \{(x_{ji}, y_{ji}, p_{ji})\}$;

else

Impostare $S_{ij} = S_{i-1j}$;

Ottenere $\tilde{\theta}_j$ risolvendo

$$Q_j^*(\theta) = \frac{1}{n} \sum_{S_{nj}} \frac{1}{p_{ij}} \{y_{ij} - \psi(x_{ij}^T \theta)\} x_{ij} = 0, \tag{2.24}$$

e calcolare

$$\dot{Q}_j^*(\tilde{\theta}_j) = -\sum_{S_{nj}} \frac{1}{p_{ij}} \dot{\psi}(x_{ij}^T \tilde{\theta}_j) x_{ij} x_{ij}^T. \tag{2.25}$$

Step 3: Combinazione

Combinare i k stimatori e lo stimatore pilota calcolando

$$\tilde{\theta}_{kr} = \left\{ \sum_{j=0}^{k} \dot{Q}_j^*(\tilde{\theta}_j) \right\}^{-1} \sum_{j=0}^{k} \dot{Q}_j^*(\tilde{\theta}_j) \tilde{\theta}_j. \tag{2.26}$$

Output: $\tilde{\theta}_{kr}$

2.4 Sottocampionamento nella regressione logistica

Essendo il modello di regressione logistica di riferimento in questo lavoro di tesi, si andranno ora a specificare per tale modello le quantità generali introdotte in questo capitolo sui metodi di sottocampionamento ottimale. In particolare si ha che

$$\psi(x_i^T\theta) = \pi_i(\theta) = \frac{\exp(x_i^T\theta)}{1+\exp(x_i^T\theta)},$$

$$\dot{\psi}(x_i^T\theta) = \pi_i(\theta)(1-\pi_i(\theta)),$$

e quindi

$$\Sigma_\psi(\theta) = N^{-1}\sum_{i=1}^{N}\pi_i(\theta)(1-\pi_i(\theta))x_i x_i^T = N^{-1}\mathbf{X}^T\mathbf{W}\mathbf{X}.$$

Dall'ultima formulazione si può notare che $\Sigma_\psi(\theta)$ corrisponde all'informazione osservata e attesa, divisa per il numero di osservazioni.

Inoltre si osservi che l'equazione di stima pesata (2.19) in questo caso non è altro che l'equazione di stima ottenuta uguagliando a zero la funzione *score* (3) valutata con le osservazioni estratte presenti in S e introducendo gli adeguati pesi $1/p_i$, ossia

$$Q^*(\theta) = \sum_S \frac{1}{p_i}\{y_i - \pi_i(\theta)\}x_i = 0; \tag{2.27}$$

2.4.1 Sottocampionamento con correzione di Firth

Anche per le metodologie di sottocampionamento ci si è interessati a valutare l'utilizzo dell'inserimento della correzione di Firth. Si proporrà di impiegare tale penalità nell'Algoritmo 2.2 in entrambe le fasi e cioè sia nell'ottenimento dello stimatore pilota, sia nel calcolo dello stimatore finale.

Nello studio di simulazione che si presenterà nel Capitolo 3, le dimensioni r_0 e r del sottocampione valutate non sempre saranno sufficientemente elevate rispetto al numero di variabili d e sarà quindi d'interesse analizzare l'utilità della correzione di Firth nell'ottenere performance migliori ed eventualmente sotto quali condizioni.

La modifica che ne deriva riguarda solamente la specifica dell'equazione di stima pesata (2.19) che in questo caso diventa

$$Q^{*F}(\theta) = \sum_S \frac{1}{p_i}\{(y_i + h_i/2) - (1+h_i)\pi_i(\theta)\}x_i,$$

con S generico sottoinsieme estratto e h_i i-esimo elemento sulla diagonale della matrice $\mathbf{H}$ definita in (10) utilizzando solo le osservazioni contenute in S.

2.5 Approccio bayesiano

I metodi di sottocampionamento presi in considerazione in questo capitolo e introdotti da Yu et al. (2020) sono stati pensati per l'inferenza frequentista. In questo breve paragrafo, si faranno delle considerazioni su una loro possibile applicazione in un contesto bayesiano.

Innanzitutto è da notare che per il metodo distribuito descritto nel Paragrafo 2.3 non risulta immediata una possibile formulazione bayesiana, perché con le quantità trasmesse alla macchina centrale non vi è un loro naturale utilizzo per la definizione della distribuzione a posteriori. Se invece si considera la versione non distribuita, si potrebbe definire, con le unità sottocampionate tramite l'Algoritmo 2.2 e contenute nell'insieme S_{tot}, la seguente pseudo-distribuzione a posteriori:

$$\breve{p}(\theta|S_{\text{tot}}) \propto \exp\left\{\frac{1}{N}\sum_{S_{\text{tot}}}\frac{1}{p_i}\ell(\theta;y_i)\right\}p(\theta).$$

In tale distribuzione vengono utilizzate le probabilità ottimali p_i per pesare i contributi delle singole osservazioni estratte nella definizione della log-verosimiglianza, analogamente a quanto fatto nell'equazione di stima (2.19). Anche in questo caso, equivalentemente a quanto visto per i metodi *gradient-enhanced*, la definizione della distribuzione a posteriori si basa su $r + r_0$ osservazioni anziché N e quindi questo porta a una netta diminuzione del costo computazionale degli algoritmi MCMC.

Si ricorda che anche in questo approccio all'inferenza, i metodi di sottocampionamento mirano ad avvicinarsi il più possibile ai risultati ottenibili con tutti i dati utilizzando un minor numero di osservazioni, mentre i metodi descritti nel Capitolo 1 si interessano di replicare nel modo più fedele possibile l'inferenza globale, avendo i dati divisi in sottogruppi.

Capitolo 3

Confronto metodologie

3.1 Confronti iniziali

In questo lavoro di tesi ci si è interessati a valutare le performance delle metodologie presentate nei capitoli precedenti, in un modello di regressione logistica per dati binari, con particolare attenzione alle modifiche proposte tramite l'inserimento della correzione di Firth. In particolare le metodologie analizzate sono state:

- CSL (Paragrafo 1.1);
- CEASE con parametro α fissato (Paragrafo 1.2);
- CEASE con correzione di Firth, introdotto nel Paragrafo 1.2.7 e con parametro α fissato (spesso indicato nelle analisi con CEASE$_F$);
- sottocampionamento ottimale non distribuito (metodi mMSE e mVc, introdotti nel Paragrafo 2.2);
- sottocampionamento ottimale non distribuito con utilizzo della correzione di Firth, come descritto nel Paragrafo 2.4.1 (metodi indicati con mMSE$_F$ e mVc$_F$).

La valutazione di ognuno di questi metodi è avvenuta sia a livello di stima puntuale che a livello di intervalli di confidenza.

All'interno delle due macroclassi di approcci gli interessi delle analisi sono stati molteplici. Per quanto riguarda le metodologie *gradient-enhanced*, gli obiettivi dello studio sono stati:

- avere una valutazione empirica più generale delle problematiche di convergenza dell'approccio CSL e della loro risoluzione da parte dell'algoritmo CEASE;

- confronto tra minimizzazione esatta e approssimata della funzione surrogata ad ogni iterazione degli algoritmi;
- analisi della copertura degli intervalli di confidenza proposti per la metodologia CEASE nel Paragrafo 1.2.5;
- studio dell'inserimento della correzione di Firth nell'algoritmo CEASE.

Il secondo e il terzo obiettivo appena elencati, sono stati infatti proposti come possibili spunti di analisi future per l'algoritmo CEASE in Fan et al. (2021).

In merito alle metodologie di sottocampionamento invece, i maggiori interessi sono stati:

- valutazione di questi metodi in un contesto a dimensionalità non irrilevante e cioè con un rapporto d/r tale da poter rendere instabili i processi di stima (dove si ricorda che r non è altro che la dimensione del sottocampione estratto);
- analisi dell'introduzione della correzione di Firth nella procedura di stima.

A proposito del primo aspetto d'interesse, sia in Yu et al. (2020) che in Wang et al. (2018), queste metodologie di sottocampionamento sono state analizzate in contesti in cui la numerosità campionaria totale era estremamente elevata e anche la dimensione del sottocampione era nettamente superiore al numero dei parametri di regressione considerati. Per questo motivo ci si è occupati di studiare le sue performance in situazioni a più alta dimensionalità in cui anche lo stimatore pilota potrebbe risultare poco efficiente ed affidabile. Si sono considerate diverse specificazioni della quantità M per vedere se, empiricamente, in questo contesto le conclusioni sono le stesse di quelle ottenute in Yu et al. (2020).

Come si è già potuto in parte evincere dalla trattazione svolta finora, e riprendendo quanto è stato appena sottolineato per i metodi di sottocampionamento, il contesto d'interesse è quello distribuito in cui il numero d di parametri da stimare non è sufficientemente elevato rispetto alla numerosità globale N da rappresentare una problematica nell'inferenza su tutti i dati, ma ha un valore abbastanza grande da complicare il processo inferenziale a livello locale quando i dati vengono suddivisi in sottoinsiemi, nel caso dei metodi *gradient-enhanced*, o quando si va a considerare un sottocampione, nel caso dei metodi di sottocampionamento ottimale. A tal proposito, tutti i confronti che verranno effettuati sono stati svolti tenendo in considerazione la suddivisione dei dati in sottogruppi di numerosità diversa. In questo modo si sono quindi potute analizzare le diverse metodologie sotto scenari con diversi gradi di problematicità.

Tutti i confronti svolti in questo capitolo si basano su uno studio di simulazione, la cui impostazione, che ora si andrà a presentare, è stata definita in modo analogo a quanto fatto in Fan et al. (2021, Paragrafo 5.1) per gli esperimenti numerici. Innanzitutto si sono fissate la dimensione campionaria totale $N = 10000$ e il numero di parametri di regressione $d = 101$ e si è proceduto a determinare la matrice del disegno $\mathbf{X}$ con i-esima riga $x_i = (1, u_i^T)^T$, dove u_i è un vettore di osservazioni di dimensione $d-1$ generato da una variabile $N_{d-1}(0_{d-1}, \Sigma)$ e $\Sigma = \text{diag}(10, 5, 2, 1, 1, \ldots, 1) \in \mathbb{R}^{(d-1)(d-1)}$. Per quanto riguarda il veri valori del parametro, ossia θ^*, sono stati generati uniformemente da una sfera con raggio di norma 3 (si veda la Tabella 3.1 per i valori effettivi ottenuti). Infine sono stati simulati 10^3 vettori risposta $y = (y_1, \ldots, y_N)$ da una variabile casuale bernoulliana con probabilità di successo $P(Y_i = 1|x_i) = \pi_i(\theta^*) = \frac{\exp(x_i^T\theta^*)}{1+\exp(x_i^T\theta^*)}$, tenendo fissa la matrice di disegno $\mathbf{X}$ in ogni simulazione.

TABELLA 3.1: Veri valori del parametro θ nello studio di simulazione per il modello di regressione logistica.

0.222	0.251	0.489	0.189	-0.208	0.202	-0.330	0.359	0.136	0.330
0.386	0.472	-0.302	0.513	-0.332	-0.574	-0.054	-0.488	0.245	-0.095
0.183	-0.256	-0.386	0.217	-0.054	-0.157	0.023	0.365	0.216	-0.123
-0.325	-0.242	-0.193	0.417	-0.218	0.212	0.367	-0.238	0.248	-0.015
-0.497	-0.138	-0.123	0.135	0.018	-0.085	-0.182	-0.053	-0.121	0.116
-0.311	0.136	-0.030	0.432	0.097	0.104	-0.146	0.608	0.227	0.144
-0.259	0.646	0.200	-0.070	0.352	0.077	-0.196	0.177	-0.049	-0.142
0.385	-0.120	-0.289	-0.225	0.275	0.035	0.069	-0.458	-0.157	-0.066
-0.488	-0.593	0.003	0.587	-0.028	-0.142	0.636	-0.005	-0.001	-0.155
-0.438	0.007	-0.279	0.326	-0.157	0.456	0.705	-0.055	-0.595	-0.081
-0.240									

Sono state considerate tre situazioni differenti di suddivisione dei dati, denominate "n grande", "n moderato" e "n piccolo" con numerosità campionaria locale e numero di sottogruppi pari rispettivamente a $(n, k) = (2000, 5)$, $(1000, 20)$ e $(250, 40)$. Risulta chiaro come questa suddivisione vada ad influire in pratica all'interno delle metodologie *gradient-enhanced*, mentre non lo è altrettanto se si considerano le metodologie di sottocampionamento, dato che si stanno analizzando le loro formulazioni non distribuite. Infatti è opportuno specificare che, affinché il confronto tra le due classi di approcci fosse adeguato, per ognuna delle tre situazioni definite, si sono applicati i metodi di sottocampionamento in modo tale che il campione finale estratto avesse una dimensione pari a $r_0 + r = n$.

Focalizzandosi sui metodi *gradient-enhanced*, il valore del parametro α negli algoritmi CEASE e CEASE$_F$ è stato considerato pari a $0.15d/n$, valore utilizzato anche in Fan

et al. (2021) negli studi di simulazione effettuati. Inoltre, per studiare le loro performance con diversi tipi di inizializzazione, si sono considerati due criteri per definire il valore iniziale del parametro θ:

- $\theta^{(0)} = 0$, detta *inizializzazione zero*;
- $\theta^{(0)} = \bar{\theta}$, detta *inizializzazione ottimale*, in cui $\bar{\theta}$ non è altro che la media degli stimatori locali ottenuti nelle singole macchine o tramite massima verosimiglianza o, qualora quest'ultima restituisca stime infinite, tramite l'aggiunta della correzione di Firth.

Gli algoritmi CSL e CEASE sono stati utilizzati nella loro versione asimmetrica, dato che per queste nel Capitolo 1 si sono presentate delle possibili definizioni degli intervalli di confidenza.Come detto in precedenza, la matrice di disegno $\mathbf{X}$ è stata considerata fissata in tutte le 10^3 simulazioni effettuate, in cui pure i sottogruppi sono stati definiti sempre uguali e come riferimento è sempre stato preso il primo di essi. Di conseguenza, all'interno di ognuna delle tre casistiche di suddivisione dei dati, si è sempre andati a considerare lo stesso sottogruppo di riferimento e quindi sempre la stessa sottomatrice di disegno. Nonostante ciò si pensa che questa impostazione non abbia intaccato la generalità dei risultati ottenuti dallo studio e se ne è avuto conferma dal secondo studio di simulazione effettuato nel Paragrafo 3.2. Per questo motivo si ritiene che le valutazioni complessive siano valide e che ci permettano di avere un'idea delle modifiche proposte nelle metodologie.

A proposito invece dei metodi di sottocampionamento, si è ritenuto opportuno prendere in considerazione anche il campionamento uniforme (indicato con UNIF) in modo tale da avere un riferimento che permettesse di evidenziare l'utilità delle probabilità ottimali. Come visto nel Paragrafo 2.2.3, nella formula (2.18) si è utilizzato $\varrho = 0.2$, in linea con quanto è stato suggerito dallo studio di simulazione svolto in Yu et al. (2020, Paragrafo 5.1). Un'ulteriore scelta è stata quella di definire un rapporto $r_0/(r_0 + r)$ pari a 0.2. Così facendo però, nel caso in cui $n = 250$, r_0 sarebbe stato minore di d. Dunque, solo per questa casistica si è deciso di impostare $r_0 = d + 50$, permettendo di non ottenere uno stimatore pilota eccessivamente instabile e facendo in modo che il sottocampione ottimale influisse abbastanza nella stima finale. L'elevata dimensionalità d del problema rispetto alla dimensione del sottocampione pilota r_0, ha comportato una definizione delle probabilità di estrazione p_i tali da produrre, nel secondo step dell'Algoritmo 2.2, sottocampioni di numerosità campionaria eccessiva rispetto a r, ossia quella dichiarata. Per questo motivo si è ritenuto necessario riscalare le probabilità ottimali definite in (2.18) in modo tale che sommassero a r. Pertanto si è riusciti ad ottenere dei

sottocampioni con numerosità media pari a quella desiderata e rendere così possibile il confronto con le metodologie *gradient-enhanced*.

Si ricorda inoltre che i metodi mMSE_F e mVc_F fanno utilizzo della correzione di Firth sia nell'ottenere lo stimatore pilota, sia nella definizione dello stimatore finale. Nell'implementazione pratica in R si è fatto utilizzo della libreria `brglm2` (Kosmidis, 2020) per il calcolo delle stime di Firth.

3.1.1 Confronti stime puntuali

Nel paragrafo corrente si andranno ad analizzare i risultati dello studio di simulazione svolto a livello di stime puntuali. Ci si è interessati a controllare quanto gli stimatori ottenuti si allontanassero dal vero valore del parametro, ossia θ^*. In particolare, per un generico stimatore $\hat{\theta}_{\text{gen}}$, si è considerata la distanza euclidea $\|\hat{\theta}_{\text{gen}} - \theta^*\|_2$ per ognuna delle 10^3 simulazioni effettuate, di cui si è calcolata la media.

Partendo dai metodi *gradient-enhanced*, se ne è studiata la convergenza iterazione dopo iterazione. Nella Figura 3.1 sono riportati tali confronti al variare della tipologia di inizializzazione e della numerosità campionaria locale n. In questo caso, per ottenere una rappresentazione grafica più chiara, nelle ordinate è stata riportata la media dei $\log \|\theta^{(t)} - \theta^*\|_2$ ottenuti. È da sottolineare che nell'ottenere le medie non si sono considerati eventuali stime mancanti dovute all'impossibilità di invertire l'hessiano della funzione surrogata. Tale problematica è stata riscontrata esclusivamente con il metodo CSL, quando le stime divergevano eccessivamente. Si vedrà nella Tabella 3.3 con che frequenza e in quali situazioni si è presentato questo problema. Come prima cosa si può notare come, in qualsiasi scenario considerato, il metodo CEASE_F, sia con minimizzazione esatta che approssimata, non permette di ottenere delle performance migliori rispetto alla sua versione senza correzione di Firth. In particolare, sembra che converga verso un valore del parametro diverso da quello dello stimatore globale $\hat{\theta}$ e più distante di quest'ultimo dal vero valore θ^*. Il metodo CSL invece, come ci si aspettava, ha delle performance che peggiorano visibilmente al diminuire di n. Se con $n = 2000$ arriva velocemente a convergenza, sia con la versione esatta che con quella approssimata, ciò non accade con gli altri valori di n considerati dove l'algoritmo fallisce in modo evidente. Nel caso in cui si utilizza l'inizializzazione $\theta^{(0)} = 0$ e $n = 1000$, la versione esatta presenta un picco, questo perché diverse stime hanno riportato valori estremamente grandi, portando a non ottenere dei risultati nell'iterazione successiva, motivo per cui si osserva tale scostamento. Sempre in questo scenario CSL^{ex} e CSL^{appr} sembrano avere dei comportamenti differenti. Gli algoritmi $\text{CEASE}^{\text{appr}}$ e CEASE^{ex} sono gli unici a convergere

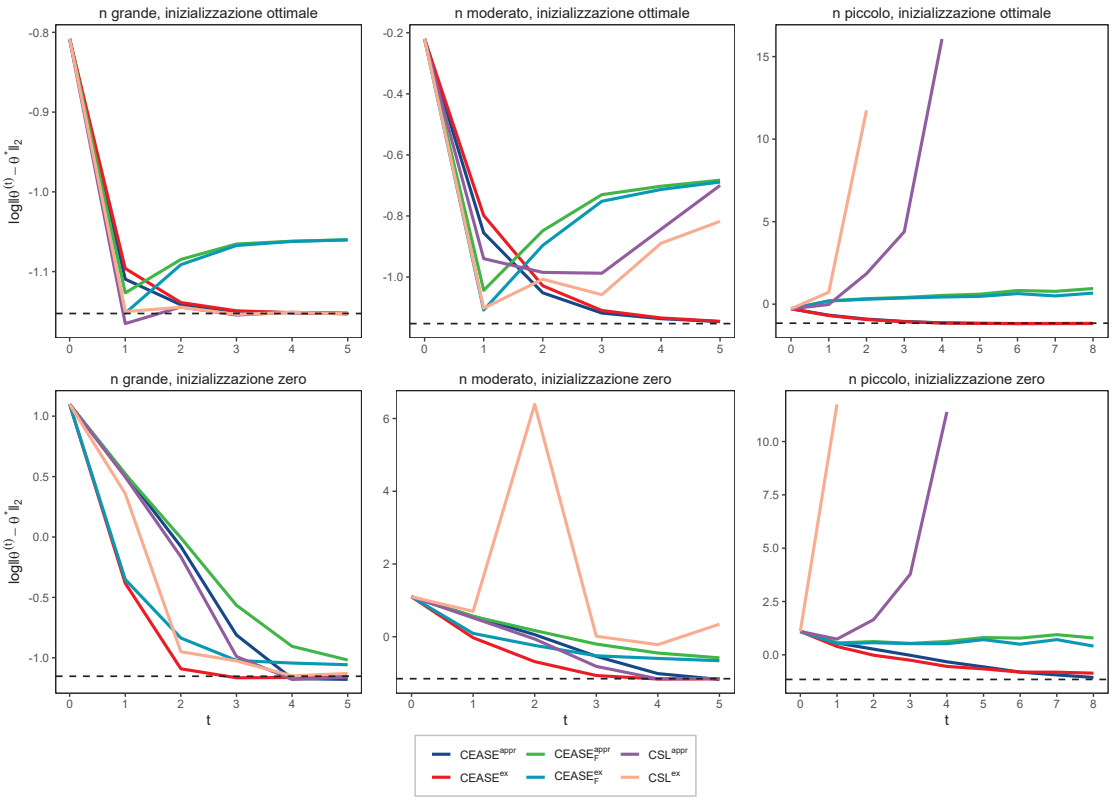


FIGURA 3.1: Convergenza degli algoritmi *gradient-enhanced* al variare della numerosità campionaria locale e del tipo di inizializzazione. Nell'asse delle ascisse e in quello delle ordinate sono riportati rispettivamente il numero di iterazioni effettuate e il valore medio di $\log \|\theta^{(t)} - \theta^*\|_2$ ottenuto nelle 10^3 simulazioni. La linea orizzontale tratteggiata corrisponde allo stimatore globale $\hat{\theta}$ ottenuto utilizzando tutti i dati.

rapidamente in ogni scenario, anche con valore iniziale nullo, a conferma che la convergenza non dipende dalla bontà dell'inizializzazione. In particolare si osserva come il numero di iterazioni considerate siano sempre sufficienti per garantire che le stime arrivino ad ottenere dei valori che presentano una distanza dal vero valore θ^* simile a quella dello stimatore globale $\hat{\theta}$. Il tipo di ottimizzazione utilizzata non sembra influire sulle performance dell'algoritmo se si considera l'inizializzazione ottimale mentre con l'altra tipologia di inizializzazione la versione CEASE$^{\text{ex}}$ ha una velocità di convergenza maggiore. In ogni caso i risultati ottenuti ci evidenziano che ricorrerendo all'ottimizzazione approssimata vengono mantenute le garanzie di convergenza, permettendo allo stesso tempo di ridurre l'onere computazionale. Inoltre, si noti che, in qualsiasi scenario, nell'ultima iterazione la versione approssimata presenta in media le stesse performance ottenute con ottimizzazione esatta, motivandone ulteriormente il suo utilizzo.

Per avere un'idea del numero di iterazioni necessarie in media per l'ottimizzazione esatta del surrogato della verosimiglianza, si veda la Tabella A.1.

Per i metodi di sottocampionamento invece, nella Tabella 3.2 si sono riportate le

TABELLA 3.2: Media delle distanze $\|\hat{\theta}_{\text{gen}} - \theta^*\|_2$ ottenute nelle 10^3 simulazioni per le diverse metodologie al variare della numerosità campionaria locale. Si ricorda che per i metodi di sottocampionamento $n = r + r_0$. Per i metodi *gradient-enhanced* i valori riportati si riferiscono a quelli ottenuti nell'ultima iterazione e con inizializzazione ottimale.

	n = 2000	n = 1000	n = 250
$\hat{\theta}$ globale	0.32	0.32	0.32
CSL^{ex}	0.32	0.45	∞
CSL^{appr}	0.32	0.51	∞
CEASE^{ex}	0.32	0.32	0.31
$\text{CEASE}^{\text{appr}}$	0.32	0.32	0.31
$\text{CEASE}_F^{\text{ex}}$	0.35	0.50	1.96
$\text{CEASE}_F^{\text{appr}}$	0.35	0.51	2.58
UNIF	0.83	1.49	2.72
mMSE con $M = \infty$	0.77	1.27	33.65
mMSE con $M = Q$	0.72	1.39	∞
mMSE con $M = E$	0.76	1.29	29.97
mVc con $M = \infty$	0.82	1.31	32.71
mVc con $M = Q$	0.77	1.27	32.39
mVc con $M = E$	0.82	1.32	30.26
mMSE_F con $M = \infty$	0.71	0.99	2.15
mMSE_F con $M = Q$	0.65	1.02	2.63
mMSE_F con $M = E$	0.71	1.00	2.16
mVc_F con $M = \infty$	0.75	1.01	2.15
mVc_F con $M = Q$	0.70	0.97	2.24
mVc_F con $M = E$	0.75	1.01	2.14

medie delle distanze euclidee $\|\hat{\theta}_{\text{gen}} - \theta^*\|_2$ al variare della dimensione $n = r + r_0$ del campione estratto. Il primo aspetto da notare riguarda le performance nettamente migliori che si sono ottenute con la correzione di Firth con qualsiasi valore di n considerato. Nel dettaglio, al diminuire di n il miglioramento risulta più evidente. Con $n = 250$, come si vedrà meglio successivamente, la correzione di Firth evita di ottenere delle stime infinite, motivo per il quale, nelle metodologie che non ne fanno utilizzo, i valori delle distanze medie che si osservano sono molto elevati. Le restanti osservazioni che possono essere fatte, sono in linea con quanto viene descritto in Yu et al. (2020), ossia:

- il metodo mMSE di definizione delle probabilità risulta essere migliore rispetto al metodo mVc, sebbene si ricordi che quest'ultimo comporta un minore tempo di calcolo;
- le performance del sottocampionamento non sembrano essere troppo sensibili rispetto alla scelta di M e per questo motivo si può semplicemente utilizzare $M =$

∞.

Inoltre, come ci si poteva aspettare, l'utilizzo di $M = \infty$ risulta essere migliore con valori di n più bassi. Come ultima osservazione si consideri il campionamento uniforme: tutti i metodi basati sulle probabilità ottimali sono migliori ad eccezione delle versioni senza correzione di Firth quando $n = 250$. Questo probabilmente accade perchè lo stimatore pilota che si ottiene è pessimo e questo comporta un'errata definizione delle probabilità ottimali di estrazione.

Sempre nella Tabella 3.2 sono riportate le prestazioni delle ultime iterazioni degli algoritmi *gradient-enhanced* e dello stimatore globale $\hat{\theta}$. In questo modo si ha la conferma, a livello numerico, che sia CEASE$^{\text{ex}}$ che CEASE$^{\text{appr}}$ convergono allo stimatore globale e che, rispetto ai metodi di sottocampionamento, permettono di ottenere dei risultati nettamente migliori. Si noti infine che i valori riportati pari a ∞ corrispondono all'ottenimento di sole stime infinite per i metodi di sottocampionamento e di sole stime divergenti per i metodi *gradient-enhanced*.

La Tabella 3.3 aiuta a capire come e in quali situazioni il metodo CSL diverge al punto tale da non riuscire a fornire una stima. In accordo con quanto visto nella Figura 3.1, tale problematica si riscontra in modo evidente, dopo poche iterazioni, con una numerosità campionaria locale piccola. Analogamente ci si è interessati anche a capire

Tabella 3.3: Percentuale di stime mancanti nelle iterazioni per l'approccio CSL, al variare del numero di sottogruppi k e del tipo di inizializzazione.

		Iterazioni							
		1	2	3	4	5	6	7	8
k = 5	CSL$^{\text{ex}}$ con $\theta^{(0)} = \bar{\theta}$	0	0	0	0	0			
	CSL$^{\text{appr}}$ con $\theta^{(0)} = \bar{\theta}$	0	0	0	0	0			
	CSL$^{\text{ex}}$ con $\theta^{(0)} = 0$	0	0	0	0	0			
	CSL$^{\text{appr}}$ con $\theta^{(0)} = 0$	0	0	0	0	0			
k = 10	CSL$^{\text{ex}}$ con $\theta^{(0)} = \bar{\theta}$	0	0	0	0	0			
	CSL$^{\text{appr}}$ con $\theta^{(0)} = \bar{\theta}$	0	0	0	0	0			
	CSL$^{\text{ex}}$ con $\theta^{(0)} = 0$	0	0	47	47	47			
	CSL$^{\text{appr}}$ con $\theta^{(0)} = 0$	0	0	0	0	0			
k = 40	CSL$^{\text{ex}}$ con $\theta^{(0)} = \bar{\theta}$	0	0	100	100	100	100	100	100
	CSL$^{\text{appr}}$ con $\theta^{(0)} = \bar{\theta}$	0	0	0	6	100	100	100	100
	CSL$^{\text{ex}}$ con $\theta^{(0)} = 0$	0	100	100	100	100	100	100	100
	CSL$^{\text{appr}}$ con $\theta^{(0)} = 0$	0	0	0	0	100	100	100	100

la frequenza delle stime infinite ottenute con i metodi di sottocampionamento. Come già

si era potuto intuire in precedenza, nella Tabella 3.4 si nota che con $n = 250$ quasi tutte le simulazioni delle metodologie senza correzione di Firth incorrono in questo problema. Con un'analisi più attenta si è potuto capire che ciò è dovuto alla stima pilota della fase iniziale dell'Algoritmo 2.2. Quindi questo non fa altro che avvalorare l'utilità della correzione di Firth.

Tabella 3.4: Percentuale di stime infinite per i metodi di sottocampionamento.

	n=2000	n=1000	n=250
UNIF	0	0	0
mMSE con $M = \infty$	0	0	98.9
mMSE con $M = Q$	0	0	100
mMSE con $M = E$	0	0	99.1
mVc con $M = \infty$	0	0	99.1
mVc con $M = Q$	0	0	99.7
mVc con $M = E$	0	0	99
mMSE_F con $M = \infty$	0	0	0
mMSE_F con $M = Q$	0	0	0
mMSE_F con $M = E$	0	0	0
mVc_F con $M = \infty$	0	0	0
mVc_F con $M = Q$	0	0	0
mVc_F con $M = E$	0	0	0

3.1.2 Confronti intervalli di confidenza

In questo paragrafo si riportano i confronti, a livello di intervalli di confidenza, ottenuti tramite lo studio di simulazione descritto in precedenza. Dato l'elevato numero di covariate a disposizione, non vengono riportati i risultati per tutti i parametri. In alcuni controlli preliminari si è riscontrato che il segno del parametro non influiva sui risultati di copertura, mentre il valore assoluto del parametro sì. Si è quindi deciso di focalizzarsi solo sui parametri positivi. In particolare, si sono considerati tre valori (uno prossimo a 0, uno intermedio e il massimo osservato), riportati di seguito:

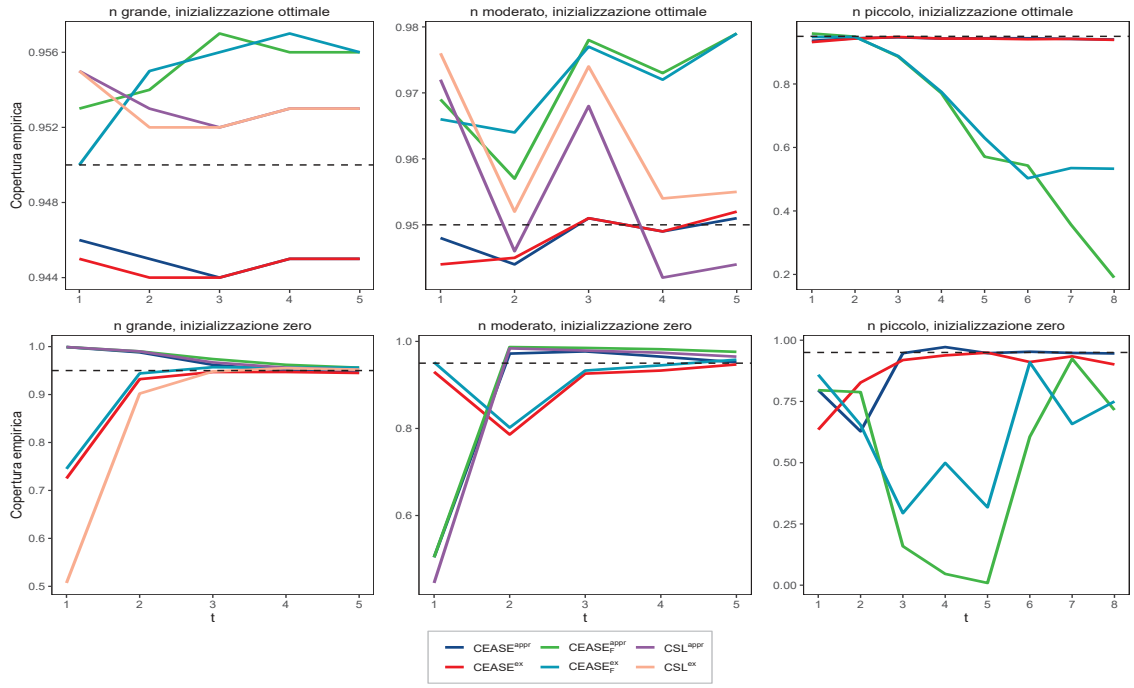
θ^*_{84}	θ^*_{24}	θ^*_{97}
0.0028	0.2171	0.7055

In questo modo si è ritenuto di poter svolgere un'analisi sufficientemente esaustiva.

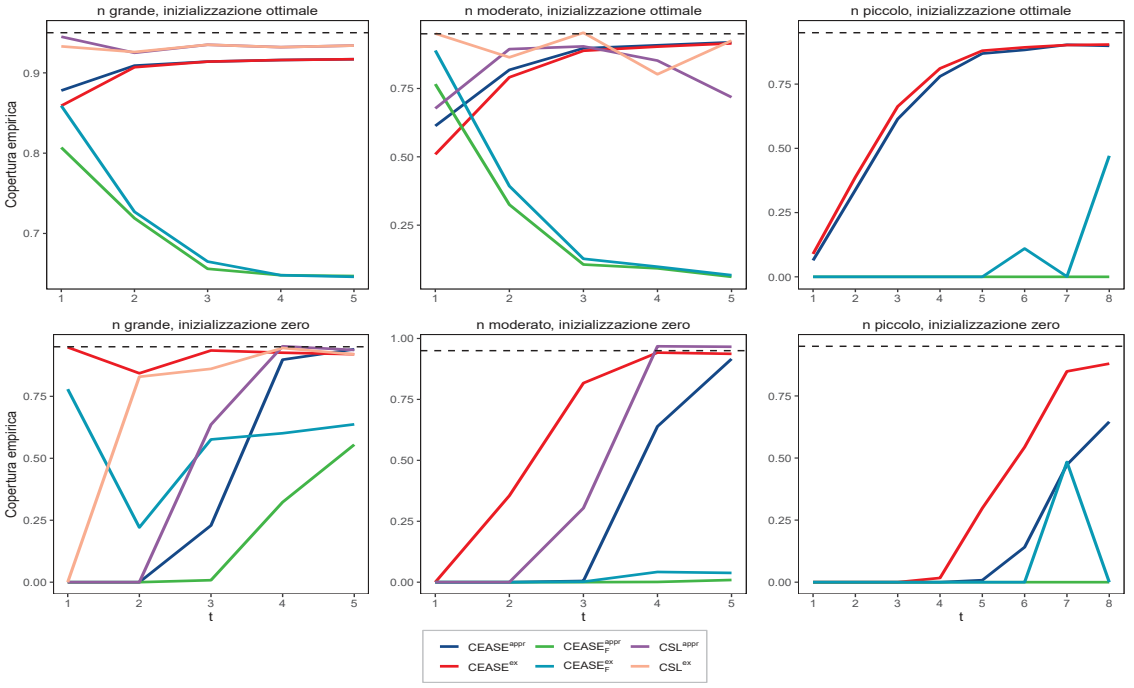
Prima di procedere con la presentazione dei risultati è opportuno sottolineare che le conclusioni che si possono trarre con le coperture empiriche ottenute devono tenere

conto che lo studio di simulazione è stato svolto con 1000 campioni e quindi la variabilità Monte Carlo non è poi così trascurabile.

Anche in questo caso per i metodi *gradient-enhanced* lo studio è stato effettuato al variare della tipologia di inizializzazione e della numerosità campionaria locale n. Per completezza, come modalità di ottimizzazione si sono confrontate sia quella esatta che quella approssimata, anche se in precedenza si è notato come questa scelta non porti a grandi cambiamenti, se non a una diversa velocità di convergenza nelle prime iterazioni quando si utilizza l'inizializzazione nulla. Lo studio della copertura empirica per i parametri θ^*_{84} e θ^*_{97} è riportato nella Figura 3.2. Le stesse analisi per il parametro θ^*_{24} sono presentate in appendice nella Figura A.1 e non sono state incluse con quelle dagli altri due parametri per motivi di chiarezza, dato che presentavano dei risultati intermedi. Sia per θ^*_{84} che per θ^*_{97} le conclusioni che si possono trarre a livello di confronti delle metodologie sono le stesse; l'unica differenza riguarda una maggiore difficoltà a raggiungere il livello di copertura nominale per il parametro θ^*_{97}, in qualsiasi scenario considerato. In particolare, per quest'ultimo, appaiono più problematici i risultati ottenuti con $n = 250$ e inizializzazione nulla. Soffermandosi invece sugli aspetti in comune tra le coperture empiriche per i due parametri, si può notare nuovamente come i metodi CEASE con la correzione di Firth abbiano le performance peggiori e la loro copertura sia completamente sbagliata in quasi tutti gli scenari. Questo non è sorprendente dato che probabilmente è dovuto alla distorsione osservata precedentemente nella Figura 3.1. I metodi CSL invece, presentano delle prestazioni migliori rispetto agli algoritmi CEASE quando la numerosità n è molto elevata e cioè quando anche per i primi viene garantita la convergenza. Questo si pensa sia dovuto alla diversa specificazione degli standard error per le due metodologie. Proprio quest'ultimo aspetto si ritiene sia cruciale per ottenere delle coperture degli intervalli di confidenza vicine a quella nominale nel momento in cui viene garantita la convergenza. Infatti, per i metodi CEASE non corretti tramite Firth, dato che convergono sempre, è possibile avere una valutazione più completa degli standard error proposti, i quali non sempre sembrano portare a risultati vicini a quelli dichiarati, soprattutto con un valore di n piccolo. Un dettaglio generale da osservare riguarda l'utilizzo dell'inizializzazione nulla con $n = 250$, la quale rende più problematico l'ottenimento di intervalli di confidenza affidabili. Ulteriore aspetto da evidenziare è che, analogamente a quanto visto nei confronti delle stime puntuali, l'utilizzo dell'ottimizzazione esatta e di quella approssimata tendono a portare agli stessi risultati nell'ultima iterazione. Ciò è sicuramente un punto a favore delle versioni approssimate.



(a) $\theta^*_{83} = 0.0028$



(b) $\theta^*_{97} = 0.7055$

FIGURA 3.2: Copertura empirica degli algoritmi *gradient-enhanced* al variare della numerosità campionaria locale e del tipo di inizializzazione, per i parametri θ^*_{83} e θ^*_{97}. Nell'asse delle ascisse è riportato il numero di iterazioni effettuate. La linea orizzontale tratteggiata corrisponde al livello di copertura nominale 0.95.

Nella Tabella 3.5 vengono prese in analisi anche le metodologie di sottocampionamento oltre che i metodi *gradient-enhanced*. Per questi ultimi vengono prese in considerazione le ultime iterazioni degli algoritmi CSL$^{\text{ex}}$ e CEASE$^{\text{ex}}$, dato che la tipologia di ottimizzazione non si è dimostrata essere determinante nel differenziare i risultati. Per gli approcci di sottocampionamento, per motivi di chiarezza rappresentativa, si sono riportati i metodi mMSE$_F$ e mVc$_F$ con $M = \infty$, essendo quelli con le migliori performance in generale. Analogamente a prima e per gli stessi motivi, vengono esclusi i risultati relativi al parametro θ^*_{24}, i quali vengono esposti in appendice nella Tabella A.2. Innanzitutto appare evidente che l'utilizzo del sottocampionamento con probabilità ottimali non permette di ottenere una copertura empirica soddisfacente soprattutto con campioni di dimensione $n = r + r_0$ ridotta. Questo è sicuramente dovuto al fatto che i risultati distributivi, riportati nel Paragrafo 2.2.4, su cui ci si è basati per la definizione degli intervalli di confidenza, sono definiti condizionatamente all'insieme dei dati complessivo Z_1^N e quindi considerando fissato il valore della stima globale, in quel paragrafo indicato con $\hat{\theta}_{QLE}$. Ciò non sarebbe un problema se la variabilità dello stimatore globale fosse trascurabile ma, nello studio di simulazione in esame, questo non accade, dato che con $d = 101$ la numerosità campionaria totale N dovrebbe essere maggiore di quella considerata. Si noti inoltre che l'ampiezza media degli intervalli per questi metodi sia nettamente superiore a quella degli approcci *gradient-enhanced*, dato che per i primi il contenuto informativo considerato per definire la variabilità è inferiore rispetto agli ultimi, dove invece ci si propone di avvicinarsi il più possibile ai risultati inferenziali ottenibili avendo a disposizione tutti i dati. Per quanto riguarda le metodologie CSL e CEASE le conclusioni che si possono trarre sono analoghe a quelle precedenti. In particolare, avendo a disposizione come riferimento nella Tabella 3.5 la copertura empirica ottenuta con l'inferenza a livello globale, è possibile capire quanto tali approcci si avvicinino a quest'ultima. Il metodo CSL con $n = 2000$ ($k = 5$) permette di ottenere gli stessi risultati ottenuti globalmente, mentre il metodo CEASE produce una copertura empirica che al diminuire di n si allontana sempre di più da quella globale. Inoltre, tale differenza tra copertura ottenuta con l'algoritmo CEASE e quella con tutti i dati a disposizione risulta più marcata per il parametro θ^*_{97}, dove già si era potuta riscontrare un'inferenza più problematica. Dunque all'aumentare del valore del parametro di regressione, la copertura empirica associata ottenuta con gli standard error specificati per il metodo CEASE si allontana maggiormente da quella ottenibile globalmente. Ulteriore aspetto da sottolineare riguarda l'ampiezza media degli intervalli: in quasi tutti gli scenari considerati la metodologia CEASE produce degli intervalli di ampiezza minore rispetto a quelli ottenuti con tutti i dati. Questo probabilmente succede a causa del

TABELLA 3.5: Copertura empirica percentuale e ampiezza media (riportata tra parentesi) degli intervalli di confidenza al 95% per i parametri $\theta^*_{83} = 0.0028$ e $\theta^*_{97} = 0.7055$. Viene fatto variare il numero di sottogruppi k e di conseguenza la numerosità campionaria locale n. Per i metodi *gradient-enhanced* si considera l'ultima iterazione ottenuta con inizializzazione ottimale. I metodi di sottocampionamento riportati invece sono stati implementati con $M = \infty$.

		$\hat{\theta}$ globale	CSL^{ex}	CEASE^{ex}	mMSE_F	mVc_F
θ^*_{83}	k=5	94.6 (0.121)	95.3 (0.124)	94.5 (0.119)	87.4 (0.215)	86.5 (0.225)
	k=10	94.6 (0.121)	95.5 (0.135)	95.2 (0.123)	91.7 (0.331)	91.8 (0.332)
	k=40	94.6 (0.121)		93.9 (0.115)	86.7 (0.572)	86.5 (0.566)
θ^*_{97}	k=5	93.1 (0.131)	93.4 (0.133)	91.7 (0.126)	83.6 (0.228)	83.0 (0.236)
	k=10	93.1 (0.131)	92.5 (0.145)	91.5 (0.128)	90.1 (0.348)	87.8 (0.347)
	k=40	93.1 (0.131)		90.4 (0.110)	62.1 (0.580)	64.1 (0.575)

termine aggiuntivo $\alpha\mathbf{I}$ nella definizione della stima proposta della matrice di varianza e covarianza asintotica nella formula (1.24), il quale tende a restringere gli standard error. Quest'ultimo aspetto evidenzia la necessità di approfondire ulteriormente la definizione degli standard error per la metodologia CEASE, proponendo delle alternative da poter confrontare. Nel paragrafo seguente infatti ci si porrà proprio questo obiettivo, analizzando i risultati ottenuti da un ulteriore studio di simulazione.

3.2 Confronti ulteriori

Visti i precedenti risultati ottenuti sulla copertura empirica, si è approfondita ulteriormente la definizione di opportuni standard error per la metodologia CEASE. Come già osservato, quest'ultima è l'unica ad arrivare a convergenza in qualsiasi scenario proposto e quindi una stima adeguata della varianza degli stimatori, che si avvicini a quella globale, potrebbe permetterci di ricavare le stesse coperture che si otterrebbero avendo tutti i dati a disposizione su un'unica macchina.

L'impostazione dello studio di simulazione adottato è rimasta uguale a quella adottata in precedenza, con l'unica differenza che, in questo caso, i sottogruppi sono stati definiti in modo casuale in ognuna delle 10^3 simulazioni, e quindi non tenuti fissi. In questo modo si pensa di aver assicurato una maggiore generalità dei risultati ottenuti, soprattutto nello scenario in cui $n = 250$ ed è più probabile che gli hessiani delle funzioni obiettivo locali siano diversi tra di loro. L'utilizzo di questa modifica ci ha permesso di avvalorare i risultati ottenuti precedentemente, confermando che l'aver utilizzato sempre lo stesso gruppo non ha intaccato le conclusioni e i risultati dello studio.

Si presentano ora le stime della matrice di varianza e covarianza asintotica Σ dello stimatore $\tilde{\theta}$ ottenuto dall'algoritmo CEASE, considerate nelle analisi svolte in questo paragrafo. La prima proposta è

$$\hat{\Sigma}_1 = \nabla^2 \tilde{\mathcal{L}}(\tilde{\theta})^{-1}, \tag{3.1}$$

la quale corrisponde alla stima utilizzata precedentemente per l'algoritmo CSL e che non si è potuto valutare adeguatamente dato che quest'ultimo, in molte delle situazioni considerate, divergeva. Si noti che questa scelta corrisponde all'inversa dell'informazione attesa della macchina di riferimento (in questo caso la prima) divisa per la numerosità campionaria locale, ossia

$$\hat{\Sigma}_1 = \nabla^2 \mathcal{L}_1(\tilde{\theta})^{-1} = \big(i_1(\tilde{\theta})/n)\big)^{-1}.$$

La seconda stima considerata non è altro che lo stimatore *plug-in* globale definito in (1.11)

$$\hat{\Sigma}_2 = \nabla^2 \tilde{\mathcal{L}}(\tilde{\theta})^{-1} \left(\frac{n}{k} \sum_{j=1}^{k} \nabla \mathcal{L}_j(\tilde{\theta}) \nabla \mathcal{L}_j(\tilde{\theta})^T \right) \nabla^2 \tilde{\mathcal{L}}(\tilde{\theta})^{-1}. \tag{3.2}$$

Anche se in questo caso, essendo la funzione obiettivo l'opposto della funzione di log-verosimiglianza, sarebbe sufficiente utilizzare lo stimatore (3.1), si è voluto vedere se l'influenza del termine centrale in (3.2) permettesse di migliorare la stima qualora l'informazione nel sottogruppo di riferimento non fosse adeguata e quindi al diminuire di n. I successivi due stimatori vengono definiti dall'estensione dei due precedenti nel momento in cui si sta valutando come funzione obiettivo quella corrispondente all'algoritmo CEASE. Tali stimatori sono

$$\hat{\Sigma}_3 = (\nabla^2 \tilde{\mathcal{L}}(\tilde{\theta}) + \alpha \mathbf{I})^{-1}, \tag{3.3}$$

e la modifica dello stimatore *plug-in* globale (3.2)

$$\hat{\Sigma}_4 = (\nabla^2 \tilde{\mathcal{L}}(\tilde{\theta}) + \alpha \mathbf{I})^{-1} \left(\frac{n}{k} \sum_{j=1}^{k} \nabla \mathcal{L}_j(\tilde{\theta}) \nabla \mathcal{L}_j(\tilde{\theta})^T \right) (\nabla^2 \tilde{\mathcal{L}}(\tilde{\theta}) + \alpha \mathbf{I})^{-1}. \tag{3.4}$$

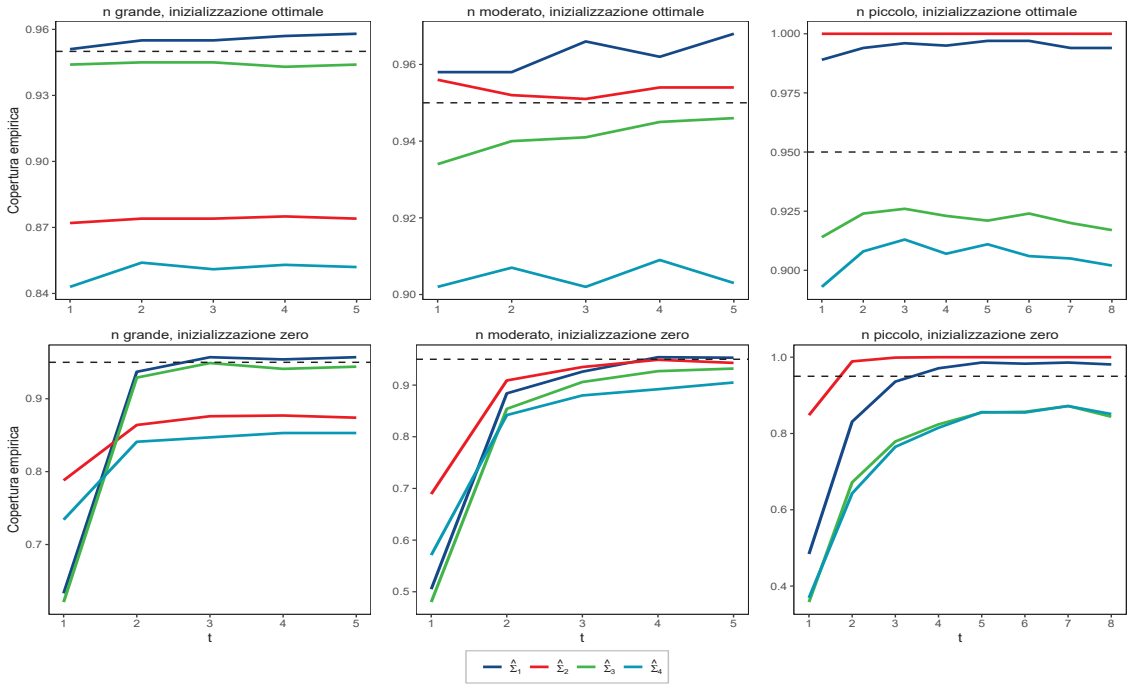
Lo stimatore in (3.3) è lo stesso che è stato proposto e utilizzato per il metodo CEASE nello studio precedente. Se il parametro α viene posto uguale a zero, allora gli ultimi due stimatori corrispondono esattamente ai primi due. Si osservi che ognuna delle alternative in esame ha come comune idea di base quella di ridurre al minimo i costi di comunicazione tra le macchine, dato che si ricorda che il contesto distribuito è quello

di riferimento. Le loro formulazioni infatti o non richiedono il trasferimento di alcuna quantità come accade in (3.1) e (3.3) oppure richiedono il trasferimento dei k gradienti d-dimensionali, come in (3.2) e (3.4), il che non comporta un costo eccessivo. Nel caso in cui si ricorra ai metodi *gradient-enhanced* in situazioni di analisi di *big data* contenuti in una singola macchina, potrebbe essere fattibile il calcolo delle informazioni attese $i_j(\tilde{\theta})$, per $j = 1, \ldots, k$, e la loro successiva somma per ottenere l'informazione attesa globale $i(\tilde{\theta})$ e dunque ottenere degli intervalli di confidenza equivalenti a quelli definiti con tutti i dati. Chiaramente nel caso di architetture distribuite, questa procedura potrebbe comportare un eccessivo costo di comunicazione e per questo motivo non viene presa in esame.

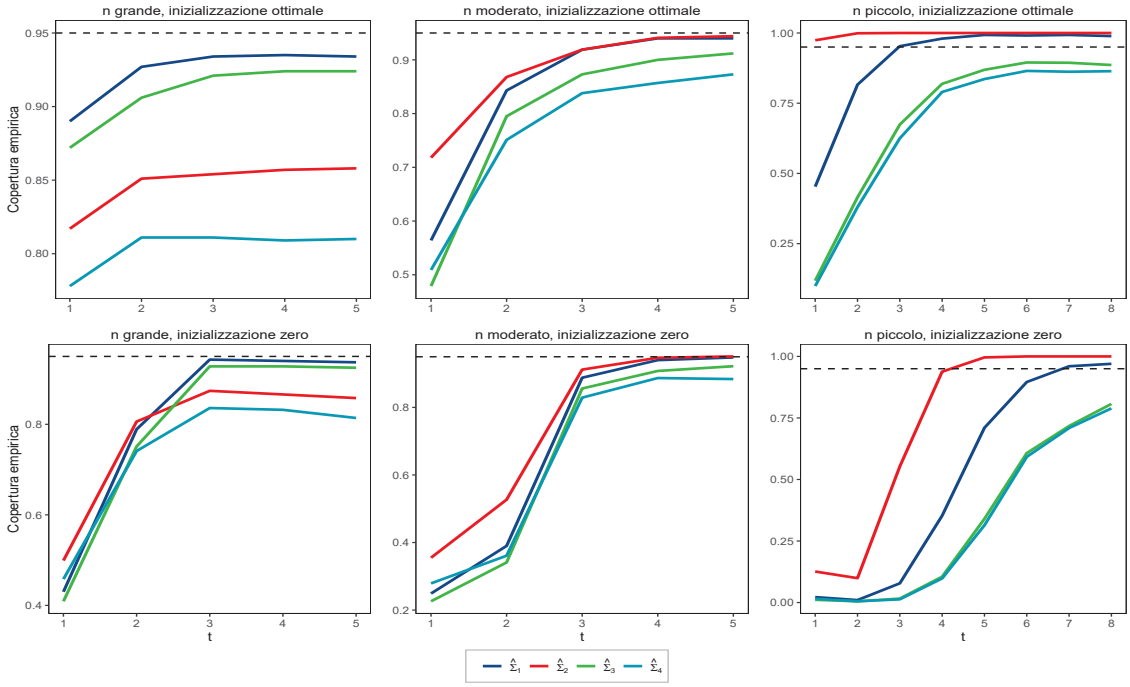
Si procede ora con la presentazione dei risultati ottenuti. I parametri di regressione considerati sono sempre gli stessi e siccome l'utilizzo dell'ottimizzazione approssimata si è rivelata equivalente a quella esatta, si riporteranno solo le analisi inerenti a quest'ultima. Si ricorda che anche in questa simulazione si è fissato $\alpha = 0.15d/n$.

Nella Figura 3.3 viene esaminata la copertura empirica per i parametri θ^*_{84} e θ^*_{97} perché, anche in questo caso, per il parametro θ^*_{24} si sono riscontrati dei risultati intermedi (si veda la Figura A.2). Un primo aspetto da notare riguarda la tipologia di inizializzazione, la quale non sembra condurre a livelli di copertura empirica differenti nell'ultima iterazione. Tale equivalenza viene un po' meno nel momento in cui n è piccolo, caso in cui l'inizializzazione nulla porta a livelli di copertura empirica inferiori. Si ipotizza che questo accada perchè partendo da $\theta^{(0)} = 0$ l'algoritmo converge verso la stima globale da una direzione in cui solitamente l'hessiano locale tende ad essere maggiore, portando a degli intervalli di confidenza con un'ampiezza più piccola del previsto. Lo stimatore $\hat{\Sigma}_4$ è quello che presenta le performance peggiori in ognuno degli scenari considerati. Per quanto riguarda lo stimatore $\hat{\Sigma}_2$, è l'alternativa che presenta la migliore copertura empirica con numerosità campionaria locale moderata, mentre nelle altre due situazioni non risulta avere un comportamento soddisfacente. La bassa copertura con n grande e quindi con poche macchine a disposizione, è conforme a quanto teorizzato in Jordan et al. (2019), mentre l'eccessiva copertura con n piccolo potrebbe essere dovuta all'insufficente numero di osservazioni locali rispetto alla dimensionalità d del problema inferenziale. Tale stimatore dunque potrebbe essere utile nelle situazioni in cui si hanno a disposizione molte macchine con una numerosità campionaria sufficientemente elevata. Spostando invece l'attenzione sullo stimatore $\hat{\Sigma}_1$, si può vedere che presenta dei livelli di copertura empirica soddisfacenti con n elevato e moderato, mentre con un gran numero di sottoinsiemi la sua copertura risulta essere quasi pari ad 1. Anche lo stimatore $\hat{\Sigma}_3$ riporta delle coperture empiriche soddisfacenti in generale, sempre più basse rispetto a

quelle ottenute con $\hat{\Sigma}_1$.



(a) $\theta^*_{83} = 0.0028$



(b) $\theta^*_{97} = 0.7055$

FIGURA 3.3: Copertura empirica dell'algoritmo CEASE^{ex}, con diverse specificazioni degli standard error, al variare della numerosità campionaria locale e del tipo di inizializzazione, per i parametri θ^*_{83} e θ^*_{97}. Nell'asse delle ascisse è riportato il numero di iterazioni effettuate. La linea orizzontale tratteggiata corrisponde al livello di copertura nominale 0.95.

TABELLA 3.6: Copertura empirica percentuale e ampiezza media (riportata tra parentesi) degli intervalli di confidenza al 95% per i parametri $\theta^*_{83} = 0.0028$ e $\theta^*_{97} = 0.7055$, ottenuti utilizzando diverse specificazioni degli standard error. I risultati sono relativi all'algoritmo CEASE e in particolare all'ultima iterazione ottenuta con inizializzazione ottimale. Viene fatto variare il numero di sottogruppi k e di conseguenza la numerosità campionaria locale n.

		$\hat{\theta}$ globale	$\hat{\Sigma}_1$	$\hat{\Sigma}_2$	$\hat{\Sigma}_3$	$\hat{\Sigma}_4$
θ^*_{83}	k=5	94.6 (0.121)	95.8 (0.125)	87.4 (0.114)	94.4 (0.120)	85.2 (0.105)
	k=10	94.6 (0.121)	96.8 (0.131)	95.4 (0.142)	94.6 (0.120)	90.3 (0.116)
	k=40	94.6 (0.121)	99.4 (0.187)	100 (0.431)	91.7 (0.108)	90.2 (0.106)
θ^*_{97}	k=5	93.1 (0.131)	93.4 (0.135)	85.8 (0.122)	92.4 (0.127)	81.0 (0.109)
	k=10	93.1 (0.131)	94.0 (0.141)	94.4 (0.153)	91.2 (0.125)	87.3 (0.120)
	k=40	93.1 (0.131)	98.9 (0.195)	100 (0.438)	88.6 (0.110)	86.4 (0.105)

Nella Tabella 3.6 sono riportate le coperture empiriche per i parametri θ^*_{84} e θ^*_{97} relative alle ultime iterazioni dell'algoritmo CEASE^{ex} con inizializzazione ottimale (si veda la Tabella A.3 per il parametro θ^*_{24}). Tali risultati ci permettono di confermare quanto detto finora e mettono in evidenza che, come si poteva notare nella Tabella 3.5, i livelli di copertura ottenuti diminuiscono all'aumentare del vero valore del parametro. I parametri con valori più estremi si riconfermano essere quelli più problematici con cui ottenere una copertura vicini a quella nominale. I due stimatori migliori si dimostrano essere nuovamente $\hat{\Sigma}_1$ e $\hat{\Sigma}_3$ i quali, all'aumentare della numerosità campionaria locale n, presentano due comportamenti opposti: con $\hat{\Sigma}_1$ la copertura aumenta mentre con $\hat{\Sigma}_3$ diminuisce rispetto a quella ottenuta globalmente. I livelli di copertura prodotti con la prima proposta infatti risultano sempre essere maggiori rispetto a quelli ottenuti con la terza e lo stesso vale per l'ampiezza media degli intervalli. Appare quindi evidente come un andamento intermedio tra quello ottenuto con questi due stimatori sarebbe auspicabile e permetterebbe di ottenere delle prestazioni vicine a quelle globali. Si riuscirebbe a perseguire tale proposito scegliendo in modo accurato il valore di α da utilizzare nella formula (3.3). Negli studi svolti infatti, nella formulazione della stima $\hat{\Sigma}_3$ si è utilizzato lo stesso valore di α utilizzato anche per l'implementazione dell'algoritmo CEASE, ossia $\alpha = 0.15d/n$. Tale valore però è sicuramente adatto per assicurare la convergenza dell'algoritmo e per garantirne una velocità di convergenza adeguata, ma non è detto che sia la scelta ottimale per far sì che gli hessiani delle funzioni obiettivo locali si assomiglino tra di loro e in particolare siano simili a quello globale. Ad esempio, in questo caso il valore scelto sembra essere eccessivo, data l'ampiezza media ridotta degli intervalli di confidenza. Sarebbe quindi opportuno approfondire ulteriormente tale

scelta del parametro. A tal proposito, usando come spunto di partenza il risultato in (1.23), il quale permette di quantificare l'ordine della differenza massima degli hessiani locali da quello globale, si è proposto un valore di α che permettesse di uniformarli. In particolare, in tale relazione il termine $\|\Sigma\|_2$ è stato considerato pari al suo limite inferiore ipotizzato nel Paragrafo 1.2.3 e quindi costante. In questo modo il nuovo valore di α è stato definito considerando il contributo di un singolo coefficiente dell'hessiano nel risultato (1.23), ossia

$$\alpha_2 = \sqrt{\frac{(\log d + \log N)}{dn}}. \tag{3.5}$$

D'ora in poi si considererà $\alpha_1 = 0.15d/n$. Siccome il risultato in (3.5) è stato formulato tramite un ordine di grandezza, si è deciso di analizzare anche le prestazioni ottenibili con un sottomultiplo di α_2, cioè

$$\alpha_3 = 0.5\sqrt{\frac{(\log d + \log N)}{dn}}. \tag{3.6}$$

I risultati ottenuti per i tre parametri in analisi sono stati riportati nella Tabella 3.7. Si noti che le scelte di α specificate sono state utilizzate all'interno dello stimatore $\hat{\Sigma}_3$. Come si è potuto osservare in precedenza, le coperture ottenute globalmente sono intermedie rispetto a quelle relative agli stimatori $\hat{\Sigma}_1$ e $\hat{\Sigma}_3(\alpha_1)$. L'aspetto di principale

TABELLA 3.7: Copertura empirica percentuale e ampiezza media (riportata tra parentesi) degli intervalli di confidenza al 95% per i parametri $\theta^*_{83} = 0.0028$, $\theta^*_{24} = 0.217$ e $\theta^*_{97} = 0.7055$, ottenuti utilizzando diverse specificazioni degli standard error. I risultati sono relativi all'algoritmo CEASE e in particolare all'ultima iterazione ottenuta con inizializzazione ottimale. Viene fatto variare il numero di sottogruppi k e di conseguenza la numerosità campionaria locale n.

		$\hat{\theta}$ globale	$\hat{\Sigma}_1$	$\hat{\Sigma}_3(\alpha_1)$	$\hat{\Sigma}_3(\alpha_2)$	$\hat{\Sigma}_3(\alpha_3)$
θ^*_{83}	k=5	94.6 (0.121)	95.6 (0.125)	94.7 (0.120)	94.7 (0.120)	95.1 (0.122)
	k=10	94.6 (0.121)	96.5 (0.131)	94.7 (0.120)	94.8 (0.122)	95.6 (0.126)
	k=40	94.6 (0.121)	99.7 (0.187)	91.7 (0.108)	96.7 (0.136)	98.1 (0.153)
θ^*_{24}	k=5	94.5 (0.122)	95.3 (0.126)	94.3 (0.121)	94.1 (0.121)	94.9 (0.123)
	k=10	94.5 (0.122)	96.0 (0.132)	94.0 (0.121)	94.5 (0.123)	95.3 (0.127)
	k=40	94.5 (0.122)	99.4 (0.189)	90.3 (0.108)	96.5 (0.136)	98.0 (0.154)
θ^*_{97}	k=5	93.1 (0.131)	93.5 (0.135)	92.0 (0.127)	91.9 (0.127)	93.1 (0.131)
	k=10	93.1 (0.131)	94.2 (0.141)	91.3 (0.125)	92.0 (0.128)	93.2 (0.134)
	k=40	93.1 (0.131)	98.9 (0.196)	88.6 (0.110)	94.6 (0.139)	97.3 (0.158)

interesse è che la scelta di α_2 in generale permette di migliorare le coperture ottenute con α_1 in qualsiasi scenario considerato. Tale specifica risulta dunque appropriata per

conseguire degli standard error più simili a quelli globali. Anche il valore α_3 sembra portare a degli ottimi risultati, soprattuto con coefficienti di regressione con valori alti. Rispetto ad α_2 però produce delle coperture superiori ed eccessive soprattutto con $k = 40$.

Complessivamente le proposte formulate per il parametro α sono soddisfacenti, in particolare α_2, ma si potrebbero ottenere delle ulteriori migliorie modificando queste ultime in modo tale da avere dei valori minori con n elevato e dei valori più grandi con n piccolo. A questo riguardo potrebbe essere utile definire un risultato analogo a quello in (1.23) ma relativo all'ordine di grandezza della differenza media tra gli hessiani locali e quello globale.

Conclusioni

Nella stesura di questo lavoro di tesi l'idea iniziale era di confrontare il metodo CSL proposto in Jordan et al. (2019) con le metodologie di sottocampionamento, in particolare quelle presentate nel lavoro di Yu et al. (2020), nell'ottica dell'inferenza distribuita per un modello di regressione logistica per dati binari. In un'implementazione preliminare dell'algoritmo ILEA del *framework* CSL, si sono riscontrate alcune problematiche di convergenza al diminuire della numerosità campionaria locale n. Ci si è quindi interessati ad analizzare e risolvere tali problemi, tentando di regolarizzare le funzioni obiettivo locali in modo tale che assomigliassero di più a quella globale. Ognuna delle proposte considerate non ha permesso di garantire la convergenza dell'algoritmo, probabilmente perchè le tipologie di regolarizzazione utilizzate contraevano le funzioni obiettivo verso lo zero e non erano adatte a perseguire tale scopo. Dopo di che si è considerata l'estensione proposta da Fan et al. (2021), denominata CEASE (*Communication-Efficient Accurate Statistical Estimation*), in cui vengono definiti in modo più dettagliato i problemi di convergenza del metodo CSL e se ne propone una risoluzione pratica tramite l'inserimento di una penalità quadratica adattiva all'interno di ogni iterazione dell'algoritmo ILEA.

In ognuna delle metodologie considerate si è proposto un possibile utilizzo della correzione di Firth (1993), ritenendo che quest'ultima potesse svolgere un ruolo chiave nell'ottenimento di prestazioni migliori, soprattutto nei casi in cui la dimensione dei sottogruppi considerati non fosse abbastanza elevata.

Si è svolto uno studio di simulazione per confrontare i metodi *gradient-enhanced* con i metodi di sottocampionamento ottimale, in cui si sono analizzate le suddivisioni dei dati in 5, 10 e 40 sottogruppi. Oltre ai confronti tra le due macroclassi di metodologie, si sono potute fare delle considerazioni interne ad ognuna di esse.

Per gli algoritmi CEASE si è avuto nuovamente conferma del fatto che la loro convergenza è sempre garantita, con qualsiasi suddivisione ed inizializzazione, e che la velocità di tale convergenza è simile a quella dei metodi CSL. Sempre per tale metodologia si

è riscontrato che, all'interno di ogni iterazione dell'algoritmo di stima, l'utilizzo dell'ottimizzazione approssimata anziché di quella esatta non determina sostanziali differenze nella velocità di convergenza. Dopo un numero sufficiente di iterazioni, infatti, le prestazioni ottenute con minimizzazione esatta e approssimata si sono rivelate pressochè uguali. Perciò la versione approssimata è risultata essere una valida modifica per poter diminuire il costo computazionale dell'algoritmo. Ulteriore aspetto che è stato evidenziato riguarda l'inadeguatezza della versione dell'approccio CEASE con l'inserimento della correzione di Firth. Tale modifica è stata considerata nella speranza di poter sfruttare i benefici della correzione di Firth nell'inferenza distribuita e di ottenere uno stimatore finale che fosse vicino a quello globale di Firth, il quale si sa essere migliore, in termini di distorsione, dello stimatore di massima verosimiglianza. Purtroppo però l'utilizzo che se ne è proposto ha peggiorato le prestazioni dell'algoritmo facendolo convergere ad una stima decisamente più lontana dal vero valore del parametro rispetto alla stima globale. La penalità introdotta tramite Firth infatti non può essere suddivisa in una somma dei contribuiti relativi ai dati contenuti nelle singole macchine e quindi la proposta di inserimento presa in analisi non è equivalente all'utilizzo di tale correzione globalmente. Potrebbe essere d'interesse indagare ulteriori possibilità di inserimento che permettano di avvicinarsi il più possibile allo stimatore di Firth globale.

Per quanto riguarda la costruzione di intervalli di confidenza, si sono valutate diverse definizioni della matrice di varianza e covarianza asintotica dello stimatore risultate tramite metodologia CEASE. Si è visto che sarebbe desiderabile specificare degli standard error intermedi rispetto a quelli definiti per il metodo CSL in Jordan et al. (2019) e quelli proposti in questo lavoro in cui si tiene conto della penalità quadratica introdotta nel metodo CEASE. Tale compromesso può essere conseguito con un'adeguata scelta del parametro α, il quale non deve necessariamente avere lo stesso valore consigliato per ottenere delle ottime proprietà di convergenza dell'algoritmo ma dovrebbe essere tale da poter uniformare l'hessiano locale a quello globale. In questo lavoro di tesi vengono dati alcuni spunti ma sarà sicuramente d'interesse in futuro valutare in modo approfondito la scelta di α.

Per i metodi di sottocampionamento si è invece riscontrato che le loro performance sono sempre peggiori rispetto alla controparte, sia in termini di stime puntuali che in termini di intervalli di confidenza. Dopotutto, l'informazione utilizzata è quella di un sottocampione e non quella di tutti i dati, seppur divisi in gruppi, come nei metodi *gradient-enhanced*. Si ricorda infatti che questo approccio è pensato non per replicare esattamente l'inferenza che si ottiene globalmente ma piuttosto per avvicinarsi a quest'ultima riducendo drasticamente i tempi di calcolo. In particolare le proprietà

distributive utilizzate per la definizione degli intervalli di confidenza sono utili nel momento in cui si può considerare che lo stimatore globale coincida quasi esattamente con il vero valore del parametro, fatto che nel contesto di riferimento non accade necessariamente. In questa classe di metodi si è potuto inoltre osservare l'influenza positiva della correzione di Firth, la quale permette sia di ottenere delle stime migliori che delle stime finite quando la numerosità del sottocampione estratto non è sufficientemente elevata rispetto al numero d di parametri da stimare.

Rimangono infine alcuni aspetti che potrebbero essere approfonditi in futuro. Innanzitutto, si potrebbe cercare di combinare le due classi di metodi considerati. In particolare una possibilità potrebbe essere quella di sottocampionare in ogni macchina le osservazioni da utilizzare per la costruzione delle funzioni obiettivo locali, con l'utilizzo delle probabilità di estrazione come pesi, da utilizzare all'interno degli algoritmi CSL e CEASE. In secondo luogo, potrebbe essere d'interesse analizzare anche la versione simmetrica degli algoritmi *gradient-enhanced* all'interno dello studio di simulazione. Ulteriore aspetto che potrebbe valere la pena prendere in considerazione è la valutazione delle prestazioni delle metodologie analizzate tenendo fissa la numerosità locale n e aumentando il numero di sottoinsiemi, oltre a delle modifiche sull'impostazione dello studio, come ad esempio una diversa generazione delle variabili esplicative e/o dei valori dei parametri di regressione. Queste eventuali modifiche dello studio di simulazione potrebbero portare a evidenziare le problematiche dell'algoritmo CSL anche con numerosità locali superiori. L'ultimo punto di interesse futuro riguarda la valutazione delle prestazioni di tali metodi in un contesto distribuito di stima bayesiana. In questo lavoro si sono accennati dei possibili approcci bayesiani per ognuno dei metodi e si potrebbero quindi valutare le loro performance a livello di distribuzione a posteriori ottenibili tramite algoritmi MCMC. In particolare, per la metodologia CEASE e, più in generale, per i metodi *gradient-enhanced* si ritiene che anche in questo caso sia essenziale uno studio approfondito della scelta del parametro α, scelta che si pensa sia connessa a quella per l'ottenimento di intervalli di confidenza che abbiano una buona copertura empirica in ogni possibile scenario.

Appendice A

Materiale supplementare

Tabella A.1: Numero di iterazioni medie effettuate in ogni passo di ottimizzazione esatta di ognuno degli algoritmi *gradient-enhanced*, al variare della numerosità campionaria locale e della tipologia di inizializzazione

		Iterazioni							
		1	2	3	4	5	6	7	8
k = 5	CSL^{ex} con $\theta^{(0)} = \bar{\theta}$	3.00	2.42	2.01	2.00	2.00			
	CEASE^{ex} con $\theta^{(0)} = \bar{\theta}$	3.00	2.14	2.00	2.00	2.00			
	CEASE$^{ex}_F$ con $\theta^{(0)} = \bar{\theta}$	3.99	3.00	2.00	2.00	2.00			
	CSL^{ex} con $\theta^{(0)} = 0$	4.00	3.01	3.16	3.39	3.76			
	CEASE^{ex} con $\theta^{(0)} = 0$	3.00	3.00	2.99	2.10	2.02			
	CEASE$^{ex}_F$ con $\theta^{(0)} = 0$	4.00	3.00	3.00	3.00	2.99			
k = 10	CSL^{ex} con $\theta^{(0)} = \bar{\theta}$	4.00	3.01	3.16	3.39	3.76			
	CEASE^{ex} con $\theta^{(0)} = \bar{\theta}$	3.00	3.00	2.99	2.10	2.02			
	CEASE$^{ex}_F$ con $\theta^{(0)} = \bar{\theta}$	4.00	3.00	3.00	3.00	2.99			
	CSL^{ex} con $\theta^{(0)} = 0$	7.00	5.67	3.90	3.00	3.00			
	CEASE^{ex} con $\theta^{(0)} = 0$	6.00	4.00	3.00	2.95	2.00			
	CEASE$^{ex}_F$ con $\theta^{(0)} = 0$	6.00	4.00	3.00	3.00	2.00			
k = 40	CSL^{ex} con $\theta^{(0)} = \bar{\theta}$	5.57	3.35						
	CEASE^{ex} con $\theta^{(0)} = \bar{\theta}$	3.00	3.00	3.00	2.89	2.64	2.46	2.50	2.44
	CEASE$^{ex}_F$ con $\theta^{(0)} = \bar{\theta}$	6.12	5.99	5.85	6.27	7.01	8.09	9.04	9.10
	CSL^{ex} con $\theta^{(0)} = 0$	5.00							
	CEASE^{ex} con $\theta^{(0)} = 0$	5.00	4.00	4.00	3.00	3.06	3.00	3.04	3.02
	CEASE$^{ex}_F$ con $\theta^{(0)} = 0$	7.00	7.00	6.00	7.02	8.04	9.83	9.16	10.02

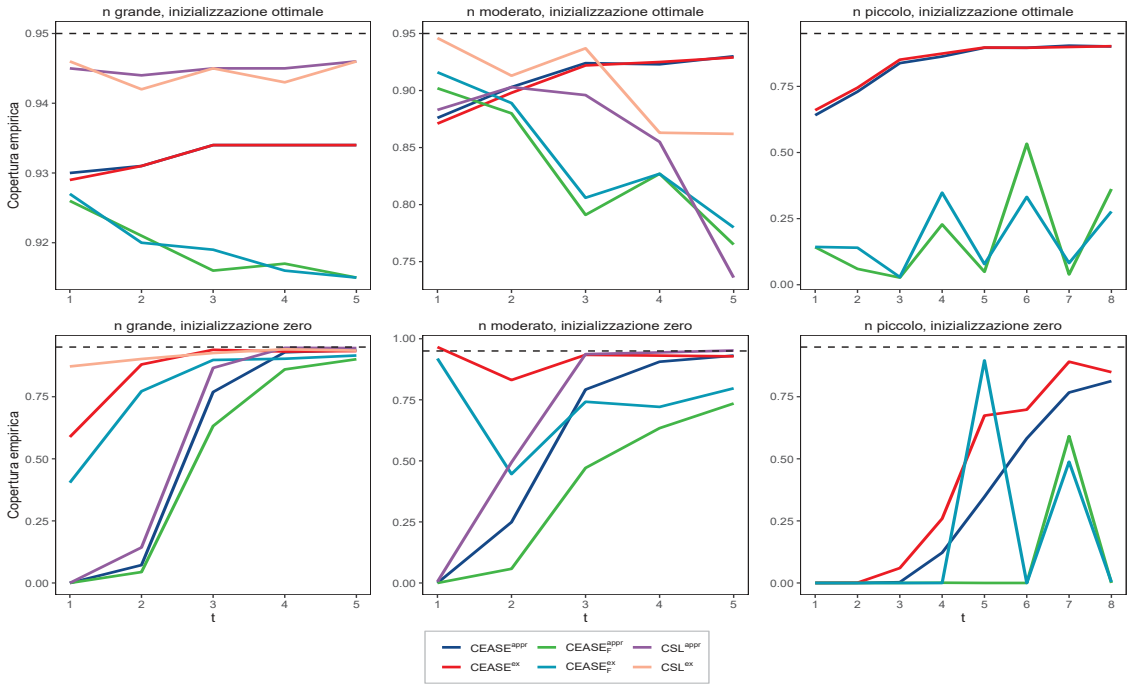


FIGURA A.1: Copertura empirica degli algoritmi *gradient-enhanced* al variare della numerosità campionaria locale e del tipo di inizializzazione, per il parametro $\theta^*_{24} = 0.217$. Nell'asse delle ascisse è riportato il numero di iterazioni effettuate. La linea orizzontale tratteggiata corrisponde al livello di copertura nominale 0.95.

TABELLA A.2: Copertura empirica percentuale e ampiezza media (riportata tra parentesi) degli intervalli di confidenza al 95% per il parametro $\theta^*_{24} = 0.217$. Viene fatto variare il numero di sottogruppi k e di conseguenza la numerosità campionaria locale n. Per i metodi *gradient-enhanced* si considera l'ultima iterazione ottenuta con inizializzazione ottimale. I metodi di sottocampionamento riportati invece sono stati implementati con $M = \infty$.

		$\hat{\theta}$ globale	CSL^{ex}	CEASE^{ex}	mMSE_F	mVc_F
θ^*_{24}	k=5	94.5 (0.122)	94.6 (0.122)	93.4 (0.117)	89.9 (0.217)	86.3 (0.227)
	k=10	94.5 (0.122)	86.2 (0.125)	92.9 (0.115)	91.6 (0.335)	90.7 (0.335)
	k=40	94.5 (0.122)		90.2 (0.104)	83.7 (0.575)	82.5 (0.572)

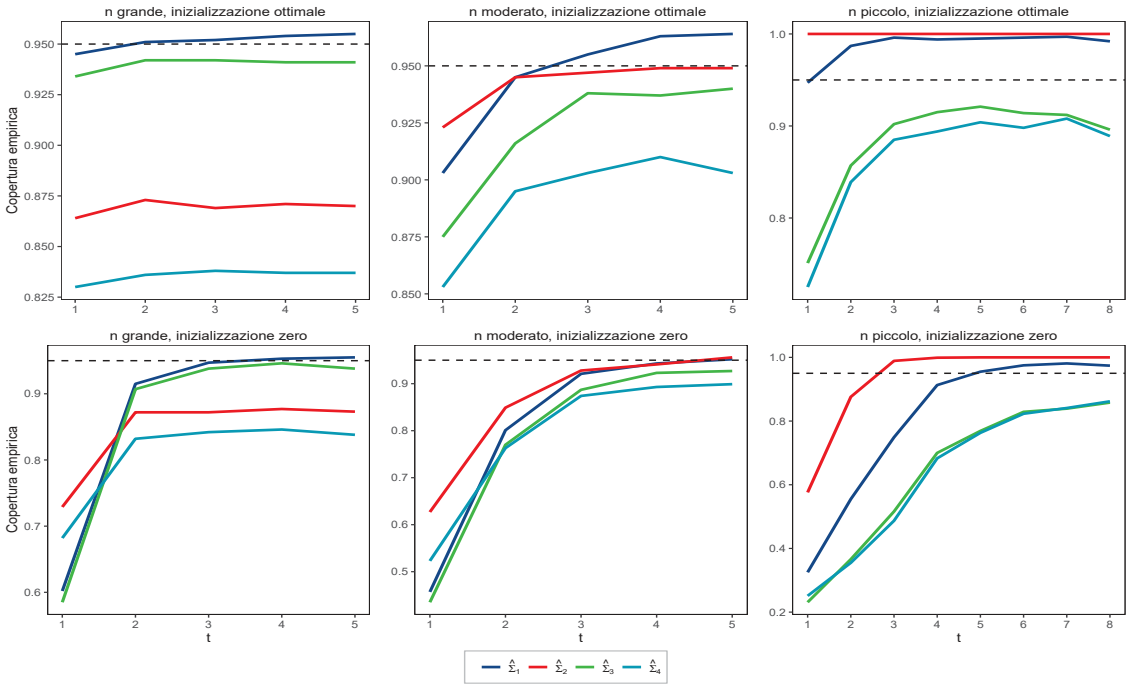


FIGURA A.2: Copertura empirica dell'algoritmo CEASE, con le diverse specificazioni degli standard error, al variare della numerosità campionaria locale e del tipo di inizializzazione, per il parametro $\theta^*_{24} = 0.217$. Nell'asse delle ascisse è riportato il numero di iterazioni effettuate. La linea orizzontale tratteggiata corrisponde al livello di copertura nominale 0.95.

TABELLA A.3: Copertura empirica percentuale e ampiezza media (riportata tra parentesi) degli intervalli di confidenza al 95% per il parametro $\theta^*_{24} = 0.217$, ottenuti utilizzando diverse specificazioni degli standard error. I risultati sono relativi all'algoritmo CEASE e in particolare all'ultima iterazione ottenuta con inizializzazione ottimale. Viene fatto variare il numero di sottogruppi k e di conseguenza la numerosità campionaria locale n.

		$\hat{\theta}$ globale	$\hat{\Sigma}_1$	$\hat{\Sigma}_2$	$\hat{\Sigma}_3$	$\hat{\Sigma}_4$
θ^*_{24}	k=5	94.5 (0.122)	95.5 (0.126)	87.0 (0.115)	94.1 (0.121)	83.7 (0.105)
	k=10	94.5 (0.122)	96.4 (0.132)	94.9 (0.143)	94.0 (0.121)	90.3 (0.116)
	k=40	94.5 (0.122)	99.2 (0.188)	100 (0.429)	89.6 (0.108)	88.9 (0.106)

Appendice B

Codice R per simulazioni

B.1: Funzione per simulare parametri e osservazioni

```
# p è il numero di variabili compresa l'intercetta,
# n è il numero di unità totali e
# R il numero di campioni della risposta simulati.
sim_data <- function(p, n, R, seed = 23)
{
  set.seed(seed)
  # Matrice del disegno
   X <- matrix(rnorm(n*(p - 1),
                     sd = sqrt(c(10, 5, 2, rep(1, (p - 4))))),
                           nrow = n, ncol = (p - 1), byrow = T)
  X <- cbind(rep(1, n), X)
  # Coefficienti di regressione generati uniformemente in
  # una sfera di raggio 3
  u <- rnorm(p)
  norm <- sqrt(sum(u^2))
  r <- runif(1)^(1/p)
  betas0 <- 3 * (r * u/norm)

  pi0 <- plogis(X %*% betas0)
  # Risposta
  all_y <- list()
  for(i in 1:R)
  {
    y <- rbinom(n, size = 1, prob = pi0)
    all_y[[i]] <- y
  }
  list(all_y = all_y, X = X, betas0 = betas0, seed = seed)
}
```

B.1 Simulazioni metodi *gradient-enhanced*

B.2: Metodo Newton-Raphson per la regressione logistica

```
# Funzione per il calcolo della stima di massima verosimiglianza
# nella regressione logistica.
getMLE <- function(x, y) {
  beta <- rep(0, ncol(x))
  loop <- 1
  Loop <- 100
  msg <- "NA"
  while (loop <= Loop) {
    pr <- c(plogis(x %*% beta))
    H <- t(x) %*% (x * pr * (1 - pr))
    S <- colSums((y - pr) * x)
    tryCatch(
      {shs <- NA
      shs <- solve(H, S) },
      error = function(e){
        cat("\n ERROR :", loop, conditionMessage(e), "\n")})
    if (is.na(shs[1])) {
      msg <- "Not converge"
      beta <- loop <- NA
      break
    }
    beta.new <- beta + shs
    tlr  <- sum((beta.new - beta)^2)
    beta  <- beta.new
    if(tlr < 0.000001) {
      msg <- "Successful convergence"
      break
    }
    if (loop == Loop)
      warning("Maximum iteration reached")
    loop  <- loop + 1
  }
  list(par = beta, message = msg)
}
```

B.3: Funzione per calcolare le quantità d'interesse localmente

```
# Viene calcolato il gradiente ed eventualmente l'hessiano per un
# dato sottoinsieme di dati e una data stima dei coefficienti.
quantities.i <- function(coef, data.i, hessian = FALSE, Firth = FALSE)
{
```

```
  n <- length(data.i$y)
  eta <- as.vector(data.i$X%*%coef)
  mu <- plogis(eta)
  if (Firth){
    W <- diag(mu * (1 - mu))
    tryCatch(expr = {
      grad <- rep(NA, ncol(data.i$X))
      H <- (W^(1 / 2)) %*% data.i$X %*% solve(t(data.i$X) %*%
                                    W %*% data.i$X) %*%
                                        t(data.i$X) %*% (W^(1 / 2))
      grad <- - colSums(((data.i$y + diag(H)/2) - (1 + diag(H)) *
                           mu) * data.i$X) / n
    }, error = function(e){
      cat("\n ERROR :", conditionMessage(e), "\n")
    })
  } else{
    grad <- - colSums((data.i$y - mu) * data.i$X) / n
  }
  hess <- NULL
  if (hessian)
    hess <- t(data.i$X) %*% diag(mu * (1 - mu)) %*% data.i$X / n
  list(grad = grad, hess = hess)
}
```

B.4: Funzione per implementare un'iterazione dell'algoritmo CSL o CEASE

```
# Se alpha = 0 si ottiene un'iterazione dall'algoritmo CEASE,
    altrimenti
# con alpha >= 0 corrisponde al metodo CEASE.
# Porre maxiter = 1 equivale a svolgere un'ottimizzazione non esatta
    della
# funzione surrogata.
newestimate <- function(start, datak, alpha = 0, maxiter = 100,
                        Firth = FALSE)
{
  loop <- 1
  Loop <- maxiter
  msg <- "NA"
  theta.t <- start
  require(plyr)
  quant_all <- llply(datak, function(x) quantities.i(start,
                                    data.i = x, Firth = Firth))
  k <- length(quant_all)
  p <- length(start)
```

```
grad_tot <- rep(0, p)
for (i in 1:k){
  if (any(is.na(quant_all[[i]]$grad))){
    theta.t <- loop <- NA
    msg <- "Firth undefined global gradient"
    return(list(par = theta.t, message = msg, iter = loop))
  }
  grad_tot <- grad_tot + quant_all[[i]]$grad
}
grad_tot <- grad_tot / k
# Correzione aggiunta al gradiente della prima macchina
corr_grad <- quant_all[[1]]$grad - grad_tot

while(loop <= Loop){
  quant1.t <- quantities.i(theta.t, datak[[1]], hessian = T,
                           Firth = Firth)
  if (any(is.na(quant1.t$grad))){
    msg <- "Firth undefined local gradient"
    return(list(par = theta.t, message = msg, iter = loop))
  }
  H.t <- quant1.t$hess + alpha * diag(p)
  S.t <- quant1.t$grad - corr_grad + alpha * (theta.t - start)
  tryCatch(
    { shs <- NA
    shs <- drop(solve(H.t, S.t)) },
    error = function(e){
      cat("\n ERROR :", loop, conditionMessage(e), "\n")})
  if (is.na(shs[1]) & (maxiter == 1)){
    msg <- "Not converge"
    theta.t <- loop <- NA
    break
  }
  if (is.na(shs[1])){
    msg <- "Not converge"
    break
  }
  theta.new <- drop(theta.t - shs)
  tlr  <- sum((theta.new - theta.t)^2)
  theta.t  <- theta.new
  if ((tlr < 0.000001) | (maxiter == 1)){
    msg <- "Successful convergence"
    break
  }
  if ((loop == Loop) & (maxiter > 1)){
```

```
      warning("Maximum iteration reached")
      msg <- "Maximum iteration reached"
    }
    loop  <- loop + 1
  }
  list(par = theta.t, message = msg, iter = loop)
}
```

B.5: Funzione per implementare i metodi *gradient-enhanced* per un valore di k

```
get_est_enhanced <- function(X, Y, k, alpha = NA, theta0 = NULL){
  n <- nrow(X); p <- ncol(X)
  alg_iter <- ifelse(k == 40, yes = 8, no = 5)
  # Valore di alpha
  if (is.na(alpha))
    alpha <- 0.15 * p/(n/k)
  # Suddivisione dei dati in k macchine
  list_data <- list()
  for (j in 1:k)
  {
    list_data[[j]] <- list()
    list_data[[j]]$X <- X[((j-1)*(n/k)+1):(j*(n/k)),]
    list_data[[j]]$y <- Y[((j-1)*(n/k)+1):(j*(n/k))]
  }

  # Possibili inizializzazioni
  start.zero <- rep(0, p)

  require(plyr)
  require(brglm2)
  startk <- llply(list_data, function(x) getMLE(x$X, x$y)$par)
  # Controllo eventuali stime infinte con relativa sostituzione
  # con stime di Firth (se non converge nemmeno Firth non si
  # considera tale macchina nella media)
  err_k <- which(unlist(llply(startk, function(x) any(is.na(x)))))
  for (i in err_k){
    tryCatch(
      { startk[[i]] <- NA
      startk[[i]] <- unname(coef(glm(list_data[[i]]$y ~
                                     -1 + list_data[[i]]$X,
                                      family = binomial(link = "logit"),
                                       method = "brglmFit",
                                        maxit = 300)))},
      error = function(e){
```

```
      cat("\n Firth ERROR :", conditionMessage(e), "\n")})
}
startk <- Filter(function(x) !any(is.na(x)), startk)
start.good <- drop(Reduce("+", startk)/length(startk))


## CSL - exact - zero initialization
estCSL_ex_zero <- matrix(NA, nrow = p, ncol = alg_iter + 1)
sigCSL_ex_zero <- matrix(NA, nrow = p, ncol = alg_iter)
# Numero di iterazioni dell'algoritmo Newton-Raphson
iterCSL_ex_zero <- rep(NA, alg_iter)
msgCSL_ex_zero <- rep("NA", alg_iter)
estCSL_ex_zero[, 1] <- start.zero
for (i in 1:alg_iter){
  temp <- newestimate(estCSL_ex_zero[, i],
                      list_data, alpha = 0, maxiter = 100,
                       Firth = F)
  iterCSL_ex_zero[i] <- temp$iter
  msgCSL_ex_zero[i] <- temp$message
  if (any(is.na(temp$par))){
    msgCSL_ex_zero[(i + 1):alg_iter] <- msgCSL_ex_zero[i]
    break
  } else{
    estCSL_ex_zero[, i + 1] <- temp$par
    tryCatch(expr = {
    sigCSL_ex_zero[, i] <- diag(solve(quantities.i(temp$par,
                                        data.i = list_data[[1]],
                                         hessian = T,
                                          Firth = F)$hess))/n
    }, error = function(e){cat("\n Error sigCSL_ex_zero \n")})
    if (any(is.na(sigCSL_ex_zero[, i]))){
      msgCSL_ex_zero[(i + 1):alg_iter] <- "Not converge"
      break
    }
  }
}


## CSL - approx - zero initialization
estCSL_appr_zero <- matrix(NA, nrow = p, ncol = alg_iter + 1)
sigCSL_appr_zero <- matrix(NA, nrow = p, ncol = alg_iter)
msgCSL_appr_zero <- rep("NA", alg_iter)
estCSL_appr_zero[, 1] <- start.zero
for (i in 1:alg_iter){
```

```
  temp <- newestimate(estCSL_appr_zero[, i], list_data,
                      alpha = 0, maxiter = 1, Firth = F)
  msgCSL_appr_zero[i] <- temp$message
  if (any(is.na(temp$par))){
    msgCSL_appr_zero[(i + 1):alg_iter] <- msgCSL_appr_zero[i]
    break
  } else{
    estCSL_appr_zero[, i + 1] <- temp$par
    tryCatch(expr = {
      sigCSL_appr_zero[, i] <- diag(solve(quantities.i(temp$par,
                                            data.i = list_data[[1]],
                                              hessian = T,
                                               Firth = F)$hess))/n
    }, error = function(e){ cat("\n Error sigCSL_appr_zero \n") })
    if (any(is.na(sigCSL_appr_zero[, i]))){
      msgCSL_appr_zero[(i + 1):alg_iter] <- "Not converge"
      break
    }
  }
}

## CSL - exact - good initialization
estCSL_ex_good <- matrix(NA, nrow = p, ncol = alg_iter + 1)
sigCSL_ex_good <- matrix(NA, nrow = p, ncol = alg_iter)
iterCSL_ex_good <- rep(NA, alg_iter)
msgCSL_ex_good <- rep("NA", alg_iter)
estCSL_ex_good[, 1] <- start.good
for (i in 1:alg_iter){
  temp <- newestimate(estCSL_ex_good[, i], list_data,
                      alpha = 0, maxiter = 100, Firth = F)
  iterCSL_ex_good[i] <- temp$iter
  msgCSL_ex_good[i] <- temp$message
  if (any(is.na(temp$par))){
    msgCSL_ex_good[(i + 1):alg_iter] <- msgCSL_ex_good[i]
    break
  } else{
    estCSL_ex_good[, i + 1] <- temp$par
    tryCatch(expr = {
      sigCSL_ex_good[, i] <- diag(solve(quantities.i(temp$par,
                                          data.i = list_data[[1]],
                                            hessian = T,
                                             Firth = F)$hess))/n
    }, error = function(e){ cat("\n Error sigCSL_ex_good \n") })
    if (any(is.na(sigCSL_ex_good[, i]))){
```

```
      msgCSL_ex_good[(i + 1):alg_iter] <- "Not converge"
      break
    }
  }
}

## CSL - approx - good initialization
estCSL_appr_good <- matrix(NA, nrow = p, ncol = alg_iter + 1)
sigCSL_appr_good <- matrix(NA, nrow = p, ncol = alg_iter)
msgCSL_appr_good <- rep("NA", alg_iter)
estCSL_appr_good[, 1] <- start.good
for (i in 1:alg_iter){
  temp <- newestimate(estCSL_appr_good[, i], list_data,
                      alpha = 0, maxiter = 1, Firth = F)
  msgCSL_appr_good[i] <- temp$message
  if (any(is.na(temp$par))){
    msgCSL_appr_good[(i + 1):alg_iter] <- msgCSL_appr_good[i]
    break
  } else{
    estCSL_appr_good[, i + 1] <- temp$par
    tryCatch(expr = {
      sigCSL_appr_good[, i] <- diag(solve(quantities.i(temp$par,
                                            data.i = list_data[[1]],
                                              hessian = T,
                                               Firth = F)$hess))/n
    }, error = function(e){ cat("\n Error sigCSL_appr_good \n") })
    if (any(is.na(sigCSL_appr_good[, i]))){
      msgCSL_appr_good[(i + 1):alg_iter] <- "Not converge"
      break
    }
  }
}

## CEASE - exact - zero initialization
estCEASE_ex_zero <- matrix(NA, nrow = p, ncol = alg_iter + 1)
sigCEASE_ex_zero <- matrix(NA, nrow = p, ncol = alg_iter)
iterCEASE_ex_zero <- rep(NA, alg_iter)
msgCEASE_ex_zero <- rep("NA", alg_iter)
estCEASE_ex_zero[, 1] <- start.zero
for (i in 1:alg_iter){
  temp <- newestimate(estCEASE_ex_zero[, i], list_data,
                      alpha = alpha, maxiter = 100, Firth = F)
  iterCEASE_ex_zero[i] <- temp$iter
  msgCEASE_ex_zero[i] <- temp$message
```

```
  if (any(is.na(temp$par))){
    msgCEASE_ex_zero[(i + 1):alg_iter] <- msgCEASE_ex_zero[i]
    break
  } else{
    estCEASE_ex_zero[, i + 1] <- temp$par
    tryCatch(expr = {
      sigCEASE_ex_zero[, i] <- diag(solve(quantities.i(temp$par,
                                            data.i = list_data[[1]],
                                              hessian = T,
                                               Firth = F)$hess +
                                                alpha * diag(p)))/n
    }, error = function(e){ cat("\n Error sigCEASE_ex_zero \n") })
    if (any(is.na(sigCEASE_ex_zero[, i]))){
      msgCEASE_ex_zero[(i + 1):alg_iter] <- "Not converge"
      break
    }
  }
}

## CEASE - approx - zero initialization
estCEASE_appr_zero <- matrix(NA, nrow = p, ncol = alg_iter + 1)
sigCEASE_appr_zero <- matrix(NA, nrow = p, ncol = alg_iter)
msgCEASE_appr_zero <- rep("NA", alg_iter)
estCEASE_appr_zero[, 1] <- start.zero
for (i in 1:alg_iter){
  temp <- newestimate(estCEASE_appr_zero[, i], list_data,
                      alpha = alpha, maxiter = 1, Firth = F)
  msgCEASE_appr_zero[i] <- temp$message
  if (any(is.na(temp$par))){
    msgCEASE_appr_zero[(i + 1):alg_iter] <- msgCEASE_appr_zero[i]
    break
  } else{
    estCEASE_appr_zero[, i + 1] <- temp$par
    tryCatch(expr = {
      sigCEASE_appr_zero[, i] <- diag(solve(quantities.i(temp$par,
                                        data.i = list_data[[1]],
                                          hessian = T,
                                           Firth = F)$hess +
                                            alpha * diag(p)))/n
    }, error = function(e){ cat("\n Error sigCEASE_appr_zero \n") })
    if (any(is.na(sigCEASE_appr_zero[, i]))){
      msgCEASE_appr_zero[(i + 1):alg_iter] <- "Not converge"
      break
    }
```

```
  }
}

## CEASE - exact - good initialization
estCEASE_ex_good <- matrix(NA, nrow = p, ncol = alg_iter + 1)
sigCEASE_ex_good <- matrix(NA, nrow = p, ncol = alg_iter)
iterCEASE_ex_good <- rep(NA, alg_iter)
msgCEASE_ex_good <- rep("NA", alg_iter)
estCEASE_ex_good[, 1] <- start.good
for (i in 1:alg_iter){
  temp <- newestimate(estCEASE_ex_good[, i], list_data,
                      alpha = alpha, maxiter = 100, Firth = F)
  iterCEASE_ex_good[i] <- temp$iter
  msgCEASE_ex_good[i] <- temp$message
  if (any(is.na(temp$par))){
    msgCEASE_ex_good[(i + 1):alg_iter] <- msgCEASE_ex_good[i]
    break
  } else{
    estCEASE_ex_good[, i + 1] <- temp$par
    tryCatch(expr = {
      sigCEASE_ex_good[, i] <- diag(solve(quantities.i(temp$par,
                                                       data.i = list_data[[1]],
                                                         hessian = T,
                                                          Firth = F)$hess +
                                                           alpha * diag(p)))/n
    }, error = function(e){ cat("\n Error sigCEASE_ex_good \n") })
    if (any(is.na(sigCEASE_ex_good[, i]))){
      msgCEASE_ex_good[(i + 1):alg_iter] <- "Not converge"
      break
    }
  }
}

## CEASE - approx - good initialization
estCEASE_appr_good <- matrix(NA, nrow = p, ncol = alg_iter + 1)
sigCEASE_appr_good <- matrix(NA, nrow = p, ncol = alg_iter)
msgCEASE_appr_good <- rep("NA", alg_iter)
estCEASE_appr_good[, 1] <- start.good
for (i in 1:alg_iter){
  temp <- newestimate(estCEASE_appr_good[, i], list_data,
                      alpha = alpha, maxiter = 1, Firth = F)
  msgCEASE_appr_good[i] <- temp$message
  if (any(is.na(temp$par))){
    msgCEASE_appr_good[(i + 1):alg_iter] <- msgCEASE_appr_good[i]
```

```
    break
  } else{
    estCEASE_appr_good[, i + 1] <- temp$par
    tryCatch(expr = {
      sigCEASE_appr_good[, i] <- diag(solve(quantities.i(temp$par,
                                        data.i = list_data[[1]],
                                            hessian = T,
                                             Firth = F)$hess +
                                              alpha * diag(p)))/n
    }, error = function(e){ cat("\n Error sigCEASE_appr_good \n") })
    if (any(is.na(sigCEASE_appr_good[, i]))){
      msgCEASE_appr_good[(i + 1):alg_iter] <- "Not converge"
      break
    }
  }
}

## CEASE + Firth - exact - zero initialization
estCEASE_F_ex_zero <- matrix(NA, nrow = p, ncol = alg_iter + 1)
sigCEASE_F_ex_zero <- matrix(NA, nrow = p, ncol = alg_iter)
iterCEASE_F_ex_zero <- rep(NA, alg_iter)
msgCEASE_F_ex_zero <- rep("NA", alg_iter)
estCEASE_F_ex_zero[, 1] <- start.zero
for (i in 1:alg_iter){
  temp <- newestimate(estCEASE_F_ex_zero[, i], list_data,
                      alpha = alpha, maxiter = 100, Firth = T)
  iterCEASE_F_ex_zero[i] <- temp$iter
  msgCEASE_F_ex_zero[i] <- temp$message
  if (any(is.na(temp$par))){
    msgCEASE_F_ex_zero[(i + 1):alg_iter] <- msgCEASE_F_ex_zero[i]
    break
  } else{
    estCEASE_F_ex_zero[, i + 1] <- temp$par
    tryCatch(expr = {
      # Firth = F perché l'hessiano non risente della
      # correzione di Firth
      sigCEASE_F_ex_zero[, i] <- diag(solve(quantities.i(temp$par,
                                        data.i = list_data[[1]],
                                            hessian = T,
                                             Firth = F)$hess +
                                              alpha * diag(p)))/n
    }, error = function(e){ cat("\n Error sigCEASE_F_ex_zero \n") })
    if (any(is.na(sigCEASE_F_ex_zero[, i]))){
      msgCEASE_F_ex_zero[(i + 1):alg_iter] <- "Not converge"
```

```
      break
    }
  }
}

## CEASE + Firth - approx - zero initialization
estCEASE_F_appr_zero <- matrix(NA, nrow = p, ncol = alg_iter + 1)
sigCEASE_F_appr_zero <- matrix(NA, nrow = p, ncol = alg_iter)
msgCEASE_F_appr_zero <- rep("NA", alg_iter)
estCEASE_F_appr_zero[, 1] <- start.zero
for (i in 1:alg_iter){
  temp <- newestimate(estCEASE_F_appr_zero[, i], list_data,
                      alpha = alpha, maxiter = 1, Firth = T)
  msgCEASE_F_appr_zero[i] <- temp$message
  if (any(is.na(temp$par))){
    msgCEASE_F_appr_zero[(i + 1):alg_iter] <-
                                      msgCEASE_F_appr_zero[i]
    break
  } else{
    estCEASE_F_appr_zero[, i + 1] <- temp$par
    tryCatch(expr = {
      sigCEASE_F_appr_zero[, i] <- diag(solve(quantities.i(temp$par,
                                      data.i = list_data[[1]],
                                        hessian = T,
                                          Firth = F)$hess +
                                           alpha * diag(p)))/n
    }, error = function(e){ cat("\n Error sigCEASE_F_appr_zero \n")
 })
    if (any(is.na(sigCEASE_F_appr_zero[, i]))){
      msgCEASE_F_appr_zero[(i + 1):alg_iter] <- "Not converge"
      break
    }
  }
}

## CEASE + Firth - exact - good initialization
estCEASE_F_ex_good <- matrix(NA, nrow = p, ncol = alg_iter + 1)
sigCEASE_F_ex_good <- matrix(NA, nrow = p, ncol = alg_iter)
iterCEASE_F_ex_good <- rep(NA, alg_iter)
msgCEASE_F_ex_good <- rep("NA", alg_iter)
estCEASE_F_ex_good[, 1] <- start.good
for (i in 1:alg_iter){
  temp <- newestimate(estCEASE_F_ex_good[, i], list_data,
                      alpha = alpha, maxiter = 100, Firth = T)
```

```
  iterCEASE_F_ex_good[i] <- temp$iter
  msgCEASE_F_ex_good[i] <- temp$message
  if (any(is.na(temp$par))){
    msgCEASE_F_ex_good[(i + 1):alg_iter] <- msgCEASE_F_ex_good[i]
    break
  } else{
    estCEASE_F_ex_good[, i + 1] <- temp$par
    tryCatch(expr = {
      sigCEASE_F_ex_good[, i] <- diag(solve(quantities.i(temp$par,
                                      data.i = list_data[[1]],
                                           hessian = T,
                                            Firth = F)$hess +
                                             alpha * diag(p)))/n
    }, error = function(e){ cat("\n Error sigCEASE_F_ex_good \n") })
    if (any(is.na(sigCEASE_F_ex_good[, i]))){
      msgCEASE_F_ex_good[(i + 1):alg_iter] <- "Not converge"
      break
    }
  }
}

## CEASE + Firth - approx - good initialization
estCEASE_F_appr_good <- matrix(NA, nrow = p, ncol = alg_iter + 1)
sigCEASE_F_appr_good <- matrix(NA, nrow = p, ncol = alg_iter)
msgCEASE_F_appr_good <- rep("NA", alg_iter)
estCEASE_F_appr_good[, 1] <- start.good
for (i in 1:alg_iter){
  temp <- newestimate(estCEASE_F_appr_good[, i], list_data,
                      alpha = alpha, maxiter = 1, Firth = T)
  msgCEASE_F_appr_good[i] <- temp$message
  if (any(is.na(temp$par))){
    msgCEASE_F_appr_good[(i + 1):alg_iter] <-
                                           msgCEASE_F_appr_good[i]
    break
  } else{
    estCEASE_F_appr_good[, i + 1] <- temp$par
    tryCatch(expr = {
      sigCEASE_F_appr_good[, i] <- diag(solve(quantities.i(temp$par,
                                       data.i = list_data[[1]],
                                           hessian = T,
                                            Firth = F)$hess +
                                             alpha * diag(p)))/n
    }, error = function(e){ cat("\n Error sigCEASE_F_appr_good \n")
 })
```

```
    if (any(is.na(sigCEASE_F_appr_good[, i]))){
      msgCEASE_F_appr_good[(i + 1):alg_iter] <- "Not converge"
      break
    }
  }
}


for (el in ls())
{
  if(!(el %in% c("estCSL_ex_zero", "estCSL_appr_zero",
                 "estCSL_ex_good", "estCSL_appr_good",
                 "estCEASE_ex_zero", "estCEASE_appr_zero",
                 "estCEASE_ex_good", "estCEASE_appr_good",
                 "estCEASE_F_ex_zero", "estCEASE_F_appr_zero",
                 "estCEASE_F_ex_good", "estCEASE_F_appr_good",
                 "sigCSL_ex_zero", "sigCSL_appr_zero",
                 "sigCSL_ex_good", sigCSL_appr_good",
                 "sigCEASE_ex_zero", "sigCEASE_appr_zero",
                 "sigCEASE_ex_good", "sigCEASE_appr_good",
                 "sigCEASE_F_ex_zero", "sigCEASE_F_appr_zero",
                 "sigCEASE_F_ex_good", "sigCEASE_F_appr_good",
                 "iterCSL_ex_zero", "iterCSL_ex_good",
                 "iterCEASE_ex_zero", "iterCEASE_ex_good"
                 "iterCEASE_F_ex_zero", "iterCEASE_F_ex_good",
                 "msgCSL_ex_zero", "msgCSL_appr_zero",
                 "msgCSL_ex_good", "msgCSL_appr_good",
                 "msgCEASE_ex_zero", "msgCEASE_appr_zero",
                 "msgCEASE_ex_good", "msgCEASE_appr_good",
                 "msgCEASE_F_ex_zero", "msgCEASE_F_appr_zero",
                 "msgCEASE_F_ex_good", "msgCEASE_F_appr_good",
                 "n", "k", "theta0")))
  {
    rm(list = el)
  }
}
for (j in 1:5){gc()}

return(list(estCSL_ex_zero = estCSL_ex_zero,
            estCSL_appr_zero = estCSL_appr_zero,
            estCSL_ex_good = estCSL_ex_good,
            estCSL_appr_good = estCSL_appr_good,
            estCEASE_ex_zero = estCEASE_ex_zero,
            estCEASE_appr_zero = estCEASE_appr_zero,
```

```
                estCEASE_ex_good = estCEASE_ex_good,
                estCEASE_appr_good = estCEASE_appr_good,
                estCEASE_F_ex_zero = estCEASE_F_ex_zero,
                estCEASE_F_appr_zero = estCEASE_F_appr_zero,
                estCEASE_F_ex_good = estCEASE_F_ex_good,
                estCEASE_F_appr_good = estCEASE_F_appr_good,
                sigCSL_ex_zero = sigCSL_ex_zero,
                sigCSL_appr_zero = sigCSL_appr_zero,
                sigCSL_ex_good = sigCSL_ex_good,
                sigCSL_appr_good = sigCSL_appr_good,
                sigCEASE_ex_zero = sigCEASE_ex_zero,
                sigCEASE_appr_zero = sigCEASE_appr_zero,
                sigCEASE_ex_good = sigCEASE_ex_good,
                sigCEASE_appr_good = sigCEASE_appr_good,
                sigCEASE_F_ex_zero = sigCEASE_F_ex_zero,
                sigCEASE_F_appr_zero = sigCEASE_F_appr_zero,
                sigCEASE_F_ex_good = sigCEASE_F_ex_good,
                sigCEASE_F_appr_good = sigCEASE_F_appr_good,
                iterCSL_ex_zero = iterCSL_ex_zero,
                iterCSL_ex_good = iterCSL_ex_good,
                iterCEASE_ex_zero = iterCEASE_ex_zero,
                iterCEASE_ex_good = iterCEASE_ex_good,
                iterCEASE_F_ex_zero = iterCEASE_F_ex_zero,
                iterCEASE_F_ex_good = iterCEASE_F_ex_good,
                msgCSL_ex_zero = msgCSL_ex_zero,
                msgCSL_appr_zero = msgCSL_appr_zero,
                msgCSL_ex_good = msgCSL_ex_good,
                msgCSL_appr_good = msgCSL_appr_good,
                msgCEASE_ex_zero = msgCEASE_ex_zero,
                msgCEASE_appr_zero = msgCEASE_appr_zero,
                msgCEASE_ex_good = msgCEASE_ex_good,
                msgCEASE_appr_good = msgCEASE_appr_good,
                msgCEASE_F_ex_zero = msgCEASE_F_ex_zero,
                msgCEASE_F_appr_zero = msgCEASE_F_appr_zero,
                msgCEASE_F_ex_good = msgCEASE_F_ex_good,
                msgCEASE_F_appr_good = msgCEASE_F_appr_good,
                theta0 = theta0, n_mch = n/k, k = k))

}
```

B.6: Funzione per implementare i metodi *gradient-enhanced* per più valori di k

```
get_est_enhanced_multi <- function(X, Y, list_k, alpha = NA, theta0){
  lapply(list_k, function(k) get_est_enhanced(X, Y, k, alpha, theta0))
```

```
}
```

B.7: Esecuzione in parallelo su 10^3 campioni

```
R <- 10^3 # Numero di campioni
p <- 101 # Numero di variabili
n <- 10^4 # Numero di osservazioni
data_R <- sim_data(p = p, n = n, R = R, seed = 23)
require(snow); require(foreach); require(doSNOW);
ncores <- parallel::detectCores()
cl <- makeSOCKcluster(ncores - 1, outfile = "")
registerDoSNOW(cl)
progress <- function(r) cat(sprintf("task %d is complete\n", r))
progress <- function(nfin, tag) {
  cat(sprintf('tasks completed: %d; tag: %d\n', nfin, tag))
}
opts <- list(progress = progress)

out <- foreach(y_input = data_R$all_y, .options.snow = opts,
                    .verbose =  T) %dopar%
  {
    get_est_enhanced_multi(X = data_R$X, Y = y_input,
                                    list_k = c(5, 10, 40),
                                      theta0 = data_R$betas0)
  }

stopCluster(cl)
```

B.2 Simulazioni metodi di sottocampionamento

B.8: Metodo Newton-Raphson per la regressione logistica

```
# Funzione per il calcolo della stima di massima verosimiglianza
# nella regressione logistica con pesi w per le osservazioni.
getMLE <- function(x, y, w) {
  beta <- rep(0, ncol(x))
  loop <- 1
  Loop <- 100
  msg <- "NA"
  while (loop <= Loop) {
    pr <- c(plogis(x %*% beta))
    H <- t(x) %*% (x * pr * (1 - pr) * w)
    S <- colSums((y - pr) * w * x)
```

```
    tryCatch(
      {shs <- NA
      shs <- solve(H, S) },
      error = function(e){
        cat("\n ERROR :", loop, conditionMessage(e), "\n")})
    if (is.na(shs[1])) {
      msg <- "Not converge"
      beta <- loop <- NA
      break
    }
    beta.new <- beta + shs
    tlr <- sum((beta.new - beta)^2)
    beta <- beta.new
    if(tlr < 0.000001) {
      msg <- "Successful convergence"
      break
    }
    if (loop == Loop)
      warning("Maximum iteration reached")
    loop <- loop + 1
  }
  list(par = beta, message = msg, iter = loop, Infor = H)
}
```

B.9: Sottocampionamento con probabilità uniformi

```
AlgTwoStpsimp <- function(X, Y, r0, rho = 0.2, beta = 0, Psi = 0){
  n <- nrow(X)
  idx <- 1:n
  pi <- rep((r0/n), n)
  decision <- rbinom(n, rep(1, n), prob = pi)
  idx.simp <- idx[decision == 1]
  X.simp <- X[idx.simp, ]
  Y.simp <- Y[idx.simp]
  pinv.simp <- n / r0
  fit.simp <- getMLE(x = X.simp, y = Y.simp, w = pinv.simp)
  msg <- fit.simp$message
  if (msg != "Successful convergence") {
    warning(paste("not converge", msg))
    beta.simp <- rep(NA, ncol(X))
    return(list(simp= beta.simp, msg = msg, infor = NA, V_c = NA))
  }
  beta.simp <- fit.simp$par
  hatpsidot <- plogis(X.simp %*% beta.simp)
```

```
  V_c <- t(X.simp) %*% diag(c((Y.simp - hatpsidot)^2 *
                           (pinv.simp^2))) %*% X.simp - t(X.simp) %*%
                             diag(c((Y.simp - hatpsidot)^2 *
                              (pinv.simp))) %*% X.simp
  return(list(simp = beta.simp, msg = msg,
              infor = fit.simp$Infor, V_c = V_c))
}
```

B.10: Sottocampionamento mMSE

```
AlgTwoStpmse <- function(X, Y, r1, rho = 0.2, beta, W.prop,
                         Psi, M = Inf){
  n <- nrow(X)
  psi.dot  <-  (plogis(X %*% beta))
  idx <- 1:n
  PI.mMSE <- pmin(abs(Y - psi.dot) * rowSums((X %*% W.prop)^2), M)
  PI.mMSE <- PI.mMSE / (n * Psi)
  PI.mMSE1 <- abs((1 - rho) * PI.mMSE + rho/n + 1/(r1 + 1e-6))/2 -
    abs((1 - rho) * PI.mMSE + rho/n - 1/(r1 + 1e-6))/2

  # Facciamo in modo che sommino a 1 se la loro somma è
  # maggiore di 1
  if (sum(PI.mMSE1) > 1)
    PI.mMSE1 <- PI.mMSE1 / sum(PI.mMSE1)

  decision <- rbinom(n, rep(1, n), prob = r1 * PI.mMSE1)
  idx.mMSE <- idx[decision == 1]
  X.mMSE <- X[idx.mMSE,]
  Y.mMSE <- Y[idx.mMSE]
  pinv.mMSE <- 1 / (r1 * PI.mMSE1[idx.mMSE])
  return(list(pinv.mMSE = pinv.mMSE, idx.mMSE = idx.mMSE))
}
```

B.11: Sottocampionamento mVc

```
AlgTwoStpmvc <- function(X, Y, r1, rho = 0.2, beta, Psi, M = Inf){
  n <- nrow(X)
  psi.dot  <-  (plogis(X %*% beta))
  idx <- 1:n
  PI.mVc <- pmin(abs(Y - psi.dot) * rowSums(X^2), M)
  PI.mVc <- PI.mVc / (n * Psi)
  PI.mVc1 <- abs((1 - rho) * PI.mVc + rho/n + 1/(r1 + 1e-6))/2 -
    abs((1 - rho) * PI.mVc + rho/n - 1/(r1 + 1e-6))/2
```

```
  # Facciamo in modo che sommino a 1 se la loro somma è
  # maggiore di 1
  if (sum(PI.mVc1) > 1)
    PI.mVc1 <- PI.mVc1 / sum(PI.mVc1)

  decision <- rbinom(n, rep(1, n), prob = r1 * PI.mVc1)
  idx.mVc <- idx[decision == 1]
  X.mVc <- X[idx.mVc,]
  Y.mVc <- Y[idx.mVc]
  pinv.mVc <- 1 / (r1 * PI.mVc1[idx.mVc])
  return(list(pinv.mVc = pinv.mVc, idx.mVc = idx.mVc))
}
```

B.12: Implementazione dei metodi di sottocampionamento per un valore di k

```
# Funzione che per un dato valore di k, ossia per una fissata
# dimensione del sottocampione da estrarre, implementa i
# metodi mMSE e mVc con diverse scelte del parametro M e con
# eventuale inserimento della correzione di Firth. La funzione
# viene applicata ad un fissato insieme di dati (X e Y).
AlgTwoStp <- function(X, Y, k, rho = 0.2, theta0 = NULL){
  # Definizione di r1 e r2, ossia rispettivamente la numerosità
  # del campione pilota e del campione ottimale
  n <- nrow(X)
  p <- ncol(X)
  r1 <- floor(0.2 * n/k)
  r2 <- (n/k) - r1
  # Controllo dimensione campione pilota rispetto al numero
  # di variabili
  if (r1 <= p){
    r1 <- p + 50
    r2 <- (n/k) - r1
  }
  cat("r1 =", r1, ",r2 =", r2 )

  ## Sottocampionamento con pesi uniformi
  pilot <- AlgTwoStpsimp(X, Y, (r1 + r2))
  # Controllo convergenza ed eventuale stima di Firth
  if (pilot$msg == "Not converge"){
    require(brglm2)
    idx.unif <- sample(1:n, r1 + r2, T)
    tryCatch(
      { beta.simp <- rep(NA,ncol(X))
      beta.simp <- c(unname(coef(glm(Y[idx.unif] ~
```

```
                                  -1 + X[idx.unif, ],
                                   family = binomial(link = "logit"),
                                     method = "brglmFit",
                                      maxit = 200))))},
    error = function(e){
      cat("\n Firth ERROR :", conditionMessage(e), "\n")}
  )
  if (any(is.na(beta.simp))){
    msg.simp <- "Not converge"
    sigma.simp <- rep(NA,ncol(X))
  } else{
    msg.simp <- "Successful convergence"
    pr <-  c(plogis(X[idx.unif, ] %*% beta.simp))
    pinv.unif <- rep(n/(r1 + r2), r1 + r2)
    Infor <- t(X[idx.unif, ]) %*% (X[idx.unif, ] * pr * (1 - pr)
                                    * pinv.unif)
    V_c <- t(X[idx.unif, ]) %*% diag(c((Y[idx.unif] - pr)^2 *
                  (pinv.unif^2))) %*% X[idx.unif, ] -
                    t(X[idx.unif, ]) %*% diag(c((Y[idx.unif] -
                                                   pr)^2 *
                      (pinv.unif))) %*% X[idx.unif, ]
    sigma.simp <- diag(solve(Infor) %*% V_c %*% solve(Infor))
  }
} else{
  beta.simp <- pilot$simp
  msg.simp <- pilot$msg
  sigma.simp <- diag(solve(pilot$infor) %*% pilot$V_c %*%
                       solve(pilot$infor))
}


## Sottocampionamento con pesi ottimali: campione pilota
idx <- 1:n
idx.prop <- sample(1:n, r1, T)
x.prop <- X[idx.prop,]
y.prop <- Y[idx.prop]
pinv.prop <- rep(n/r1, r1)
fit.prop <- getMLE(x = x.prop, y = y.prop, w = pinv.prop)
# Controllo convergenza ed eventuale stima di Firth
if (fit.prop$message == "Not converge"){
  require(brglm2)
  tryCatch(
    { beta.prop <- rep(NA,ncol(X))
    beta.prop <- unname(coef(glm(y.prop ~
```

```
                                    -1 + x.prop,
                                     family = binomial(link = "logit"),
                                       method = "brglmFit",
                                        maxit = 200)))},
    error = function(e){
      cat("\n Firth ERROR :", conditionMessage(e), "\n")}
  )
  if (any(is.na(beta.prop))){
    for (el in ls())
    {
      if (!(el %in% c("beta.simp", "beta.mMSE_Inf",
                      "beta.mMSE_Inf_F", "beta.mMSE_Q",
                      "beta.mMSE_Q_F", "beta.mMSE_E",
                      "beta.mMSE_E_F", "beta.mVc_Inf",
                      "beta.mVc_Inf_F", "beta.mVc_Q",
                      "beta.mVc_Q_F", "beta.mVc_E", "beta.mVc_E_F",
                      "sigma.simp", "sigma.mMSE_Inf",
                      "sigma.mMSE_Inf_F", "sigma.mMSE_Q",
                      "sigma.mMSE_Q_F", "sigma.mMSE_E",
                      "sigma.mMSE_E_F", "sigma.mVc_Inf",
                      "sigma.mVc_Inf_F", "sigma.mVc_Q",
                      "sigma.mVc_Q_F", "sigma.mVc_E",
                      "sigma.mVc_E_F",
                      "r.mMSE_Inf", "r.mMSE_Inf_F", "r.mMSE_Q",
                      "r.mMSE_Q_F", "r.mMSE_E", "r.mMSE_E_F",
                      "r.mVc_Inf", "r.mVc_Inf_F", "r.mVc_Q",
                      "r.mVc_Q_F", "r.mVc_E", "r.mVc_E_F",
                      "msg.simp", "msg.mMSE_Inf", "msg.mMSE_Inf_F",
                      "msg.mMSE_Q", "msg.mMSE_Q_F", "msg.mMSE_E",
                      "msg.mMSE_E_F", "msg.mVc_Inf", "msg.mVc_Inf_F",
                      "msg.mVc_Q", "msg.mVc_Q_F", "msg.mVc_E",
                      "msg.mVc_E_F", "n", "k", "theta0")))
      {
        rm(list = el)
      }
    }
    for (j in 1:5){gc()}

    return(list(beta.simp = beta.simp, beta.mMSE_Inf = rep(NA, p),
                beta.mMSE_Q = rep(NA, p), beta.mMSE_E = rep(NA, p),
                beta.mVc_Inf = rep(NA, p), beta.mVc_Q = rep(NA, p),
                beta.mVc_E = rep(NA, p),
                beta.mMSE_Inf_F = rep(NA, p),
                beta.mMSE_Q_F = rep(NA, p),
```

```
                beta.mMSE_E_F = rep(NA, p),
                beta.mVc_Inf_F = rep(NA, p),
                beta.mVc_Q_F = rep(NA, p),
                beta.mVc_E_F = rep(NA, p),
                sigma.simp = rep(NA, p),
                sigma.mMSE_Inf = rep(NA, p),
                sigma.mMSE_Q = rep(NA, p),
                sigma.mMSE_E = rep(NA, p),
                sigma.mVc_Inf = rep(NA, p),
                sigma.mVc_Q = rep(NA, p),
                sigma.mVc_E = rep(NA, p),
                sigma.mMSE_Inf_F = rep(NA, p),
                sigma.mMSE_Q_F = rep(NA, p),
                sigma.mMSE_E_F = rep(NA, p),
                sigma.mVc_Inf_F = rep(NA, p),
                sigma.mVc_Q_F = rep(NA, p),
                sigma.mVc_E_F = rep(NA, p),
                r.mMSE_Inf = NA, r.mMSE_Q = NA, r.mMSE_E = NA,
                r.mVc_Inf = NA, r.mVc_Q = NA, r.mVc_E = NA,
                r.mMSE_Inf_F = NA, r.mMSE_Q_F = NA, r.mMSE_E_F = NA,
                r.mVc_Inf_F = NA, r.mVc_Q_F = NA, r.mVc_E_F = NA,
                msg.simp = "First stage not converge",
                msg.mMSE_Inf = "First stage not converge",
                msg.mMSE_Q = "First stage not converge",
                msg.mMSE_E = "First stage not converge",
                msg.mVc_Inf = "First stage not converge",
                msg.mVc_Q = "First stage not converge",
                msg.mVc_E = "First stage not converge",
                msg.mMSE_Inf_F = "First stage not converge",
                msg.mMSE_Q_F = "First stage not converge",
                msg.mMSE_E_F = "First stage not converge",
                msg.mVc_Inf_F = "First stage not converge",
                msg.mVc_Q_F = "First stage not converge",
                msg.mVc_E_F = "First stage not converge",
                theta0 = theta0, n_mch = n/k, k = k))
  }
} else{
  beta.prop <- fit.prop$par
}

psi.dot <- (plogis(X %*% beta.prop))
pinv.prop <- pinv.prop * r1/(r1 + r2)
```

```
## mMSE - M = Inf
M <- Inf
psi.ddot <- psi.dot[idx.prop]
W.prop <- solve(t(x.prop) %*% (x.prop * psi.ddot *
                                  (1 - psi.ddot) * pinv.prop))
Psi <- sum(pmin(abs(Y[idx.prop] - psi.ddot) *
                  rowSums((X[idx.prop,] %*% W.prop)^2), M))/r1
# Otteniamo gli indici delle unità selezionate e l'inverso
# delle loro probabilità di estrazione
result <- AlgTwoStpmse(X, Y, r2, rho, beta.prop, W.prop, Psi, M = M)
# Uniamo le unità e l'inverso delle probabilità con quelle
# ottenute nel campione pilota
X.mMSE <- X[c(idx.prop, result$idx.mMSE),]
Y.mMSE <- Y[c(idx.prop, result$idx.mMSE)]

result$pinv.mMSE <- result$pinv.mMSE * n/sum(result$pinv.mMSE)
result$pinv.mMSE <- result$pinv.mMSE * r2/(r1 + r2)
pinv.mMSE <- c(pinv.prop, result$pinv.mMSE)
r.mMSE_Inf <- length(pinv.mMSE)

fit.mMSE <- getMLE(x = X.mMSE, y = Y.mMSE, w = pinv.mMSE)
msg.mMSE_Inf <- fit.mMSE$message
if (msg.mMSE_Inf != "Successful convergence") {
  beta.mMSE_Inf <- rep(NA,ncol(X))
  sigma.mMSE_Inf <- rep(NA,ncol(X))
} else{
  beta.mMSE_Inf <- fit.mMSE$par
  hatpsidot.mMSE <- c(plogis(X.mMSE %*% beta.mMSE_Inf))
  V_c.mMSE <- t(X.mMSE) %*% diag(c((Y.mMSE - hatpsidot.mMSE)^2 *
                       (pinv.mMSE^2))) %*% X.mMSE - t(X.mMSE) %*%
                          diag(c((Y.mMSE - hatpsidot.mMSE)^2 *
                                   (pinv.mMSE))) %*% X.mMSE
  sigma.mMSE_Inf <- diag(solve(fit.mMSE$Infor) %*% V_c.mMSE %*%
                             solve(fit.mMSE$Infor))
}


## mMSE - M = Inf , Firth
beta.mMSE_Inf_F <- rep(NA,ncol(X))
sigma.mMSE_Inf_F <- rep(NA,ncol(X))
r.mMSE_Inf_F <- length(pinv.mMSE)
require(brglm2)
tryCatch(
  {
```

```
    beta.mMSE_Inf_F <- unname(coef(glm(Y.mMSE ~
                              - 1 + X.mMSE,
                                family = binomial(link = "logit"),
                                  method = "brglmFit",
                                    maxit = 200,
                             weights = pinv.mMSE*(nrow(X.mMSE)/n))))
    msg.mMSE_Inf_F <- "Successful convergence"},
  error = function(e){
    cat("\n Firth ERROR :", conditionMessage(e), "\n")}
)
if (any(is.na(beta.mMSE_Inf_F))){
  beta.mMSE_Inf_F <- rep(NA, ncol(X))
  msg.mMSE_Inf_F <- "Not converge"
} else{
  hatpsidot.mMSE <- c(plogis(X.mMSE %*% beta.mMSE_Inf_F))
  V_c.mMSE <- t(X.mMSE) %*% diag(c((Y.mMSE - hatpsidot.mMSE)^2 *
                (pinv.mMSE^2))) %*% X.mMSE - t(X.mMSE) %*%
                   diag(c((Y.mMSE - hatpsidot.mMSE)^2 *
                             (pinv.mMSE))) %*% X.mMSE
  Infor <- t(X.mMSE) %*% (X.mMSE * hatpsidot.mMSE *
                             (1 - hatpsidot.mMSE) * pinv.mMSE)
  sigma.mMSE_Inf_F <- diag(solve(Infor) %*% V_c.mMSE %*%
                              solve(Infor))
}


## mMSE - M = Q
M <- quantile(abs(Y[idx.prop] - psi.ddot) *
                rowSums((X[idx.prop,] %*% W.prop)^2),
              (1 - r2/(2 * n)))

Psi <- sum(pmin(abs(Y[idx.prop] - psi.ddot) *
                  rowSums((X[idx.prop,] %*% W.prop)^2), M))/r1
# Otteniamo gli indici delle unità selezionate e l'inverso
# delle loro probabilità di estrazione
result <- AlgTwoStpmse(X, Y, r2, rho, beta.prop,
                       W.prop, Psi, M = M)
# Uniamo le unità e l'inverso delle probabilità con quelle
# ottenute nel campione pilota
X.mMSE <- X[c(idx.prop, result$idx.mMSE),]
Y.mMSE <- Y[c(idx.prop, result$idx.mMSE)]

result$pinv.mMSE <- result$pinv.mMSE * n/sum(result$pinv.mMSE)
result$pinv.mMSE <- result$pinv.mMSE * r2/(r1 + r2)
```

```
pinv.mMSE <- c(pinv.prop, result$pinv.mMSE)
r.mMSE_Q <- length(pinv.mMSE)

fit.mMSE <- getMLE(x = X.mMSE, y = Y.mMSE, w = pinv.mMSE)
msg.mMSE_Q <- fit.mMSE$message
if (msg.mMSE_Q != "Successful convergence") {
  warning(paste("mMSE not converge", msg.mMSE_Q))
  beta.mMSE_Q <- rep(NA,ncol(X))
  sigma.mMSE_Q <- rep(NA,ncol(X))
} else{
  beta.mMSE_Q <- fit.mMSE$par
  hatpsidot.mMSE <- plogis(X.mMSE%*%beta.mMSE_Q)
  V_c.mMSE <- t(X.mMSE) %*% diag(c((Y.mMSE - hatpsidot.mMSE)^2 *
                (pinv.mMSE^2))) %*% X.mMSE - t(X.mMSE) %*%
    diag(c((Y.mMSE - hatpsidot.mMSE)^2 * (pinv.mMSE))) %*% X.mMSE
  sigma.mMSE_Q <- diag(solve(fit.mMSE$Infor) %*% V_c.mMSE %*%
                         solve(fit.mMSE$Infor))
}


## mMSE - M = Q, Firth
beta.mMSE_Q_F <- rep(NA, ncol(X))
sigma.mMSE_Q_F <- rep(NA, ncol(X))
r.mMSE_Q_F <- length(pinv.mMSE)
require(brglm2)
tryCatch(
  {
    beta.mMSE_Q_F <- unname(coef(glm(Y.mMSE ~
                            - 1 + X.mMSE,
                             family = binomial(link = "logit"),
                              method = "brglmFit", maxit = 200,
                        weights = pinv.mMSE*(nrow(X.mMSE)/n))))
    msg.mMSE_Q_F <- "Successful convergence"},
  error = function(e){
    cat("\n Firth ERROR :", conditionMessage(e), "\n")}
)
if (any(is.na(beta.mMSE_Q_F))){
  beta.mMSE_Q_F <- rep(NA, ncol(X))
  msg.mMSE_Q_F <- "Not converge"
} else{
  hatpsidot.mMSE <- c(plogis(X.mMSE%*%beta.mMSE_Q_F))
  V_c.mMSE <- t(X.mMSE) %*% diag(c((Y.mMSE - hatpsidot.mMSE)^2 *
                (pinv.mMSE^2))) %*% X.mMSE - t(X.mMSE) %*%
    diag(c((Y.mMSE - hatpsidot.mMSE)^2 * (pinv.mMSE))) %*% X.mMSE
```

```
  Infor <- t(X.mMSE) %*% (X.mMSE * hatpsidot.mMSE *
                            (1 - hatpsidot.mMSE) * pinv.mMSE)
  sigma.mMSE_Q_F <- diag(solve(Infor) %*% V_c.mMSE %*%
                           solve(Infor))
}


## mMSE - M = E
h_mMSE <- drop(sort(abs(Y - psi.dot) *
                      rowSums((X %*% W.prop)^2)))
k_mMSE <- min(which(sapply(0:r2, function(s){ ((r2 - s) *
                      h_mMSE[n-s]) < sum(h_mMSE[1:(n-s)])})) - 1)
M <- sum(h_mMSE[1:(n-k_mMSE)])/(r2 - k_mMSE)

Psi <- sum(pmin(abs(Y[idx.prop] - psi.ddot) *
                  rowSums((X[idx.prop,] %*% W.prop)^2), M))/r1
result <- AlgTwoStpmse(X, Y, r2, rho, beta.prop,
                       W.prop, Psi, M = M)

X.mMSE <- X[c(idx.prop, result$idx.mMSE),]
Y.mMSE <- Y[c(idx.prop, result$idx.mMSE)]

result$pinv.mMSE <- result$pinv.mMSE * n/sum(result$pinv.mMSE)
result$pinv.mMSE <- result$pinv.mMSE * r2/(r1 + r2)
pinv.mMSE <- c(pinv.prop, result$pinv.mMSE)
r.mMSE_E <- length(pinv.mMSE)

fit.mMSE <- getMLE(x = X.mMSE, y = Y.mMSE, w = pinv.mMSE)
msg.mMSE_E <- fit.mMSE$message
if (msg.mMSE_E != "Successful convergence") {
  warning(paste("mMSE not converge", msg.mMSE_E))
  beta.mMSE_E <- rep(NA,ncol(X))
  sigma.mMSE_E <- rep(NA,ncol(X))
} else{
  beta.mMSE_E <- fit.mMSE$par
  hatpsidot.mMSE <- plogis(X.mMSE%*%beta.mMSE_E)
  V_c.mMSE <- t(X.mMSE) %*% diag(c((Y.mMSE - hatpsidot.mMSE)^2 *
              (pinv.mMSE^2))) %*% X.mMSE - t(X.mMSE) %*%
    diag(c((Y.mMSE - hatpsidot.mMSE)^2 * (pinv.mMSE))) %*% X.mMSE
  sigma.mMSE_E <- diag(solve(fit.mMSE$Infor) %*% V_c.mMSE %*%
                         solve(fit.mMSE$Infor))
}
```

```
## mMSE - M = E, Firth
beta.mMSE_E_F <- rep(NA, ncol(X))
sigma.mMSE_E_F <- rep(NA, ncol(X))
r.mMSE_E_F <- length(pinv.mMSE)
require(brglm2)
tryCatch(
  {
    beta.mMSE_E_F <- unname(coef(glm(Y.mMSE ~
                          - 1 + X.mMSE,
                            family = binomial(link = "logit"),
                              method = "brglmFit", maxit = 200,
                          weights = pinv.mMSE*(nrow(X.mMSE)/n))))
    msg.mMSE_E_F <- "Successful convergence"},
  error = function(e){
    cat("\n Firth ERROR :", conditionMessage(e), "\n")}
)
if (any(is.na(beta.mMSE_E_F))){
  beta.mMSE_E_F <- rep(NA, ncol(X))
  msg.mMSE_E_F <- "Not converge"
} else{
  hatpsidot.mMSE <- c(plogis(X.mMSE %*% beta.mMSE_E_F))
  V_c.mMSE <- t(X.mMSE) %*% diag(c((Y.mMSE - hatpsidot.mMSE)^2 *
                (pinv.mMSE^2))) %*% X.mMSE - t(X.mMSE) %*%
    diag(c((Y.mMSE - hatpsidot.mMSE)^2 * (pinv.mMSE))) %*% X.mMSE
  Infor <- t(X.mMSE) %*% (X.mMSE * hatpsidot.mMSE *
                            (1 - hatpsidot.mMSE) * pinv.mMSE)
  sigma.mMSE_E_F <- diag(solve(Infor) %*% V_c.mMSE %*%
                           solve(Infor))
}


## mVc - M = Inf
M <- Inf
# PI.mVc <- abs(Y - psi.dot) * rowSums(X^2) # quantita'mai
 utilizzata
Psi <- sum(pmin(abs(Y[idx.prop] - psi.ddot) *
                  rowSums((X[idx.prop,])^2), M))/r1

result <- AlgTwoStpmvc(X, Y, r2, rho, beta.prop, Psi, M = M)
X.mVc <- X[c(idx.prop, result$idx.mVc),]
Y.mVc <- Y[c(idx.prop, result$idx.mVc)]

result$pinv.mVc <- result$pinv.mVc * n/sum(result$pinv.mVc)
result$pinv.mVc <- result$pinv.mVc * r2/(r1 + r2)
```

```
pinv.mVc <- c(pinv.prop, result$pinv.mVc)
r.mVc_Inf <- length(pinv.mVc)

fit.mVc <- getMLE(x = X.mVc, y = Y.mVc, w = pinv.mVc)
msg.mVc_Inf <- fit.mVc$message
if (msg.mVc_Inf != "Successful convergence") {
  warning(paste("mVc not converge:", msg.mVc_Inf))
  beta.mVc_Inf <- rep(NA,ncol(X))
  sigma.mVc_Inf <- rep(NA,ncol(X))
} else{
  beta.mVc_Inf <- fit.mVc$par
  hatpsidot.mVc <- plogis(X.mVc %*% beta.mVc_Inf)
  V_c.mVc <- t(X.mVc) %*% diag(c((Y.mVc - hatpsidot.mVc)^2 *
              (pinv.mVc^2))) %*% X.mVc - t(X.mVc) %*%
    diag(c((Y.mVc - hatpsidot.mVc)^2 * (pinv.mVc))) %*% X.mVc
  sigma.mVc_Inf <- diag(solve(fit.mVc$Infor) %*% V_c.mVc
                        %*% solve(fit.mVc$Infor))
}


## mVc - M = Inf , Firth
beta.mVc_Inf_F <- rep(NA, ncol(X))
sigma.mVc_Inf_F <- rep(NA, ncol(X))
r.mVc_Inf_F <- length(pinv.mVc)
require(brglm2)
tryCatch(
  {
    beta.mVc_Inf_F <- unname(coef(glm(Y.mVc ~
                             - 1 + X.mVc,
                               family = binomial(link = "logit"),
                                 method = "brglmFit", maxit = 200,
                               weights = pinv.mVc*(nrow(X.mVc)/n))))
    msg.mVc_Inf_F <- "Successful convergence"},
  error = function(e){
    cat("\n Firth ERROR :", conditionMessage(e), "\n")}
)
if (any(is.na(beta.mVc_Inf_F))){
  beta.mVc_Inf_F <- rep(NA, ncol(X))
  msg.mVc_Inf_F <- "Not converge"
} else{
  hatpsidot.mVc <- c(plogis(X.mVc%*%beta.mVc_Inf_F))
  V_c.mVc <- t(X.mVc) %*% diag(c((Y.mVc - hatpsidot.mVc)^2 *
              (pinv.mVc^2))) %*% X.mVc - t(X.mVc) %*%
    diag(c((Y.mVc - hatpsidot.mVc)^2 * (pinv.mVc))) %*% X.mVc
```

```
  Infor <- t(X.mVc) %*% (X.mVc * hatpsidot.mVc *
                           (1 - hatpsidot.mVc) * pinv.mVc)
  sigma.mVc_Inf_F <- diag(solve(Infor) %*%
                           V_c.mVc %*% solve(Infor))
}


## mVc - M = Q
PI.mVc <- abs(Y - psi.dot) * rowSums(X^2) # quantita'mai utilizzata

M <- quantile(abs(Y[idx.prop] - psi.ddot) *
                rowSums((X[idx.prop,])^2), (1 - r2/(2*n)))

Psi <- sum(pmin(abs(Y[idx.prop] - psi.ddot) *
                  rowSums((X[idx.prop,])^2), M))/r1

result <- AlgTwoStpmvc(X, Y, r2, rho, beta.prop, Psi, M = M)
X.mVc <- X[c(idx.prop, result$idx.mVc),]
Y.mVc <- Y[c(idx.prop, result$idx.mVc)]

result$pinv.mVc <- result$pinv.mVc * n/sum(result$pinv.mVc)
result$pinv.mVc <- result$pinv.mVc * r2/(r1 + r2)
pinv.mVc <- c(pinv.prop, result$pinv.mVc)
r.mVc_Q <- length(pinv.mVc)

fit.mVc <- getMLE(x = X.mVc, y = Y.mVc, w = pinv.mVc)
msg.mVc_Q <- fit.mVc$message
if (msg.mVc_Q != "Successful convergence") {
  warning(paste("mVc not converge:", msg.mVc_Q))
  beta.mVc_Q <- rep(NA,ncol(X))
  sigma.mVc_Q <- rep(NA,ncol(X))
} else{
  beta.mVc_Q <- fit.mVc$par
  hatpsidot.mVc <- plogis(X.mVc %*% beta.mVc_Q)
  V_c.mVc <- t(X.mVc) %*% diag(c((Y.mVc - hatpsidot.mVc)^2 *
              (pinv.mVc^2))) %*% X.mVc - t(X.mVc) %*%
    diag(c((Y.mVc - hatpsidot.mVc)^2 * (pinv.mVc))) %*% X.mVc
  sigma.mVc_Q <- diag(solve(fit.mVc$Infor) %*% V_c.mVc %*%
                        solve(fit.mVc$Infor))
}


## mVc - M = Q , Firth
beta.mVc_Q_F <- rep(NA, ncol(X))
```

```
sigma.mVc_Q_F <- rep(NA, ncol(X))
r.mVc_Q_F <- length(pinv.mVc)
require(brglm2)
tryCatch(
  {
    beta.mVc_Q_F <- unname(coef(glm(Y.mVc ~
                              - 1 + X.mVc,
                                family = binomial(link = "logit"),
                                 method = "brglmFit", maxit = 200,
                            weights = pinv.mVc * (nrow(X.mVc)/n))))
    msg.mVc_Q_F <- "Successful convergence"},
  error = function(e){
    cat("\n Firth ERROR :", conditionMessage(e), "\n")}
)
if (any(is.na(beta.mVc_Q_F))){
  beta.mVc_Q_F <- rep(NA, ncol(X))
  msg.mVc_Q_F <- "Not converge"
} else{
  hatpsidot.mVc <- c(plogis(X.mVc %*% beta.mVc_Q_F))
  V_c.mVc <- t(X.mVc) %*% diag(c((Y.mVc - hatpsidot.mVc)^2 *
              (pinv.mVc^2))) %*% X.mVc - t(X.mVc) %*%
    diag(c((Y.mVc - hatpsidot.mVc)^2 * (pinv.mVc))) %*% X.mVc
  Infor <- t(X.mVc) %*% (X.mVc * hatpsidot.mVc *
                           (1 - hatpsidot.mVc) * pinv.mVc)
  sigma.mVc_Q_F <- diag(solve(Infor) %*% V_c.mVc %*%
                          solve(Infor))
}


## mVc - M = E
PI.mVc <- abs(Y - psi.dot) * rowSums(X^2) # quantita'mai utilizzata

h_mVc <- drop(sort(abs(Y - psi.dot) * rowSums((X)^2)))
k_mVc <- min(which(sapply(0:r2, function(s){ ((r2 - s) *
                    h_mVc[n-s]) < sum(h_mVc[1:(n-s)])})) - 1)
M <- sum(h_mVc[1:(n-k_mVc)])/(r2 - k_mVc)

Psi <- sum(pmin(abs(Y[idx.prop] - psi.ddot) *
                  rowSums((X[idx.prop,])^2), M))/r1

result <- AlgTwoStpmvc(X, Y, r2, rho, beta.prop, Psi, M = M)
X.mVc <- X[c(idx.prop, result$idx.mVc),]
Y.mVc <- Y[c(idx.prop, result$idx.mVc)]
```

```
result$pinv.mVc <- result$pinv.mVc * n/sum(result$pinv.mVc)
result$pinv.mVc <- result$pinv.mVc * r2/(r1 + r2)
pinv.mVc <- c(pinv.prop, result$pinv.mVc)
r.mVc_E <- length(pinv.mVc)

fit.mVc <- getMLE(x = X.mVc, y = Y.mVc, w = pinv.mVc)
msg.mVc_E <- fit.mVc$message
if (msg.mVc_E != "Successful convergence") {
  warning(paste("mVc not converge:", msg.mVc_E))
  beta.mVc_E <- rep(NA,ncol(X))
  sigma.mVc_E <- rep(NA,ncol(X))
} else{
  beta.mVc_E <- fit.mVc$par
  hatpsidot.mVc <- plogis(X.mVc %*% beta.mVc_E)
  V_c.mVc <- t(X.mVc) %*% diag(c((Y.mVc - hatpsidot.mVc)^2 *
              (pinv.mVc^2))) %*% X.mVc - t(X.mVc) %*%
    diag(c((Y.mVc - hatpsidot.mVc)^2 * (pinv.mVc))) %*% X.mVc
  sigma.mVc_E <- diag(solve(fit.mVc$Infor) %*% V_c.mVc %*%
                        solve(fit.mVc$Infor))
}


## mVc - M = E , Firth
beta.mVc_E_F <- rep(NA, ncol(X))
sigma.mVc_E_F <- rep(NA, ncol(X))
r.mVc_E_F <- length(pinv.mVc)
require(brglm2)
tryCatch(
  {
    beta.mVc_E_F <- unname(coef(glm(Y.mVc ~
                                - 1 + X.mVc,
                                family = binomial(link = "logit"),
                                 method = "brglmFit", maxit = 200,
                          weights = pinv.mVc * (nrow(X.mVc)/n))))
    msg.mVc_E_F <- "Successful convergence"},
  error = function(e){
    cat("\n Firth ERROR :", conditionMessage(e), "\n")}
)
if (any(is.na(beta.mVc_E_F))){
  beta.mVc_E_F <- rep(NA, ncol(X))
  msg.mVc_E_F <- "Not converge"
} else{
  hatpsidot.mVc <- c(plogis(X.mVc %*% beta.mVc_E_F))
  V_c.mVc <- t(X.mVc) %*% diag(c((Y.mVc - hatpsidot.mVc)^2 *
```

```
              (pinv.mVc^2))) %*% X.mVc - t(X.mVc) %*%
    diag(c((Y.mVc - hatpsidot.mVc)^2 * (pinv.mVc))) %*% X.mVc
  Infor <- t(X.mVc) %*% (X.mVc * hatpsidot.mVc *
                          (1 - hatpsidot.mVc) * pinv.mVc)
  sigma.mVc_E_F <- diag(solve(Infor) %*% V_c.mVc %*%
                        solve(Infor))
}


for(el in ls())
{
  if(!(el %in% c("beta.simp", "beta.mMSE_Inf",
                 "beta.mMSE_Inf_F", "beta.mMSE_Q",
                "beta.mMSE_Q_F", "beta.mMSE_E",
                "beta.mMSE_E_F", "beta.mVc_Inf",
                "beta.mVc_Inf_F", "beta.mVc_Q",
                "beta.mVc_Q_F", "beta.mVc_E",
                "beta.mVc_E_F",
                "sigma.simp", "sigma.mMSE_Inf",
                "sigma.mMSE_Inf_F", "sigma.mMSE_Q",
                "sigma.mMSE_Q_F", "sigma.mMSE_E",
                "sigma.mMSE_E_F",
                "sigma.mVc_Inf", "sigma.mVc_Inf_F",
                "sigma.mVc_Q",  "sigma.mVc_Q_F",
                "sigma.mVc_E", "sigma.mVc_E_F",
                "r.mMSE_Inf", "r.mMSE_Inf_F", "r.mMSE_Q",
                "r.mMSE_Q_F", "r.mMSE_E", "r.mMSE_E_F",
                "r.mVc_Inf", "r.mVc_Inf_F", "r.mVc_Q",
                "r.mVc_Q_F", "r.mVc_E", "r.mVc_E_F",
                "msg.simp", "msg.mMSE_Inf", "msg.mMSE_Inf_F",
                "msg.mMSE_Q", "msg.mMSE_Q_F", "msg.mMSE_E",
                "msg.mMSE_E_F", "msg.mVc_Inf", "msg.mVc_Inf_F",
                "msg.mVc_Q", "msg.mVc_Q_F", "msg.mVc_E",
                "msg.mVc_E_F", "n", "k", "theta0")))
  {
    rm(list = el)
  }
}
for(j in 1:5){gc()}

return(list(beta.simp = beta.simp,
            beta.mMSE_Inf = beta.mMSE_Inf,
            beta.mMSE_Q = beta.mMSE_Q,
            beta.mMSE_E = beta.mMSE_E,
```

```
                beta.mVc_Inf = beta.mVc_Inf,
                beta.mVc_Q = beta.mVc_Q,
                beta.mVc_E = beta.mVc_E,
                beta.mMSE_Inf_F = beta.mMSE_Inf_F,
                beta.mMSE_Q_F = beta.mMSE_Q_F,
                beta.mMSE_E_F = beta.mMSE_E_F,
                beta.mVc_Inf_F = beta.mVc_Inf_F,
                beta.mVc_Q_F = beta.mVc_Q_F,
                beta.mVc_E_F = beta.mVc_E_F,
                sigma.simp = sigma.simp,
                sigma.mMSE_Inf = sigma.mMSE_Inf,
                sigma.mMSE_Q = sigma.mMSE_Q,
                sigma.mMSE_E = sigma.mMSE_E,
                sigma.mVc_Inf = sigma.mVc_Inf,
                sigma.mVc_Q = sigma.mVc_Q,
                sigma.mVc_E = sigma.mVc_E,
                sigma.mMSE_Inf_F = sigma.mMSE_Inf_F,
                sigma.mMSE_Q_F = sigma.mMSE_Q_F,
                sigma.mMSE_E_F = sigma.mMSE_E_F,
                sigma.mVc_Inf_F = sigma.mVc_Inf_F,
                sigma.mVc_Q_F = sigma.mVc_Q_F,
                sigma.mVc_E_F = sigma.mVc_E_F,
                r.mMSE_Inf = r.mMSE_Inf, r.mMSE_Q = r.mMSE_Q,
                r.mMSE_E = r.mMSE_E, r.mVc_Inf = r.mVc_Inf,
                r.mVc_Q = r.mVc_Q, r.mVc_E = r.mVc_E,
                r.mMSE_Inf_F = r.mMSE_Inf_F, r.mMSE_Q_F = r.mMSE_Q_F,
                r.mMSE_E_F = r.mMSE_E_F, r.mVc_Inf_F = r.mVc_Inf_F,
                r.mVc_Q_F = r.mVc_Q_F,
                r.mVc_E_F = r.mVc_E_F,
                msg.simp = msg.simp, msg.mMSE_Inf = msg.mMSE_Inf,
                msg.mMSE_Q = msg.mMSE_Q, msg.mMSE_E = msg.mMSE_E,
                msg.mVc_Inf = msg.mVc_Inf, msg.mVc_Q = msg.mVc_Q,
                msg.mVc_E = msg.mVc_E,
                msg.mMSE_Inf_F = msg.mMSE_Inf_F,
                msg.mMSE_Q_F = msg.mMSE_Q_F,
                msg.mMSE_E_F = msg.mMSE_E_F,
                msg.mVc_Inf_F = msg.mVc_Inf_F,
                msg.mVc_Q_F = msg.mVc_Q_F, msg.mVc_E_F = msg.mVc_E_F,
                theta0 = theta0, n_mch = n/k, k = k))
}
```

B.13: Implementazione dei metodi di sottocampionamento per più valori di k

```
get_AlgTwoStp_multi <- function(X, Y, list_k, rho = 0.2, theta0){
```

```
  lapply(list_k, function(k) AlgTwoStp(X, Y, k, rho, theta0))
}
```

B.14: Esecuzione in parallelo su 10^3 campioni

```
R <- 10^3 # Numero di campioni
p <- 101 # Numero di variabili
n <- 10^4 # Numero di osservazioni
data_R <- sim_data(p = p, n = n, R = R, seed = 23)
rho <- 0.2
require(snow); require(foreach); require(doSNOW);
ncores <- parallel::detectCores()
cl <- makeSOCKcluster(ncores - 1, outfile="")
registerDoSNOW(cl)
progress <- function(r) cat(sprintf("task %d is complete\n", r))
opts <- list(progress = progress)
out <- foreach(y_input = data_R$all_y, .options.snow = opts,
               .verbose = T) %dopar%
  {
    get_AlgTwoStp_multi(X = data_R$X, Y = y_input,
                        list_k = c(5, 10, 40), rho = rho,
                        theta0 = data_R$betas0)
  }

stopCluster(cl)
```

Bibliografia

Arjevani, Y. & Shamir, O. (2015). Communication complexity of distributed convex learning and optimization. *Advances in Neural Information Processing Systems* , 1756–1764.

Atkinson, A., Donev, A. & Tobias, R. (2007). *Optimum Experimental Designs, With SAS.* Oxford University Press.

Candès, E. J. & Sur, P. (2020). The phase transition for the existence of the maximum likelihood estimate in high-dimensional logistic regression. *The Annals of Statistics* **48**, 27–42.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics* **7**, 1–26.

Fan, J., Guo, Y. & Wang, K. (2021). Communication-efficient accurate statistical estimation. *Journal of the American Statistical Association* , `https://doi.org/10.1080/01621459.2021.1969238`.

Firth, D. (1993). Bias reduction of maximum likelihood estimates. *Biometrika* **80**, 27–38.

Gao, Y., Liu, W., Wang, H., Wang, X., Yan, Y. & Zhang, R. (2022). A review of distributed statistical inference. *Statistical Theory and Related Fields* **6**, 89–99.

Jordan, M. I., Lee, J. D. & Yang, Y. (2019). Communication-efficient distributed statistical inference. *Journal of the American Statistical Association* **114**, 668–681.

Kiefer, J. (1959). Optimum experimental designs. *Journal of the Royal Statistical Society: Series B* **21**, 272–319.

Kosmidis, I. (2007). *Bias Reduction in Exponential Family Nonlinear Models.* Tesi di Dottorato, Department of Statistics, University of Warwick.

Kosmidis, I. (2020). *brglm2: Bias Reduction in Generalized Linear Models*. R package version 0.6.2.

Kosmidis, I. & Firth, D. (2021). Jeffreys-prior penalty, finiteness and shrinkage in binomial-response generalized linear models. *Biometrika* **108**, 71–82.

Ma, P., Mahoney, M. W. & Yu, B. (2015). A statistical perspective on algorithmic leveraging. *Journal of Machine Learning Research* **16**, 861–919.

Nocedal, J. & Wright, S. J. (2006). *Numerical optimization (Second Edition).* Springer.

Pace, L. & Salvan, A. (1997). *Principles of Statistical Inference from a Neo-fisherian Perspective.* World Scientific Publishing Company.

Rigon, D. & Aliverti, A. (2022). Conjugate priors and bias reduction for logistic regression models. , `https://arxiv.org/abs/2202.08734`.

Rockafellar, R. T. (1976). Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization* **14**, 877–898.

Sarndal, C. E., Swensson, B. & Wretman, J. (1992). *Model Assisted Survey Sampling.* Springer.

Shamir, O., Srebro, N. & Zhang, T. (2014). Communication-efficient distributed optimization using an approximate newton-type method. In *Proceedings of the 31st International Conference on Machine Learning*, E. P. Xing & T. Jebara, eds., vol. 32 of *Proceedings of Machine Learning Research*. Bejing, China: PMLR.

Wang, F., Kolar, M., Srebro, N. & Zhang, T. (2017). Efficient distributed learning with sparsity. In *Proceedings of the 34th International Conference on Machine Learning*, vol. 70 of *Proceedings of Machine Learning Research*.

Wang, H. Y., Zhu, R. & Ma, P. (2018). Optimal subsampling for large sample logistic regression. *Journal of the American Statistical Association* **113**, 829–844.

Yu, J., Wang, H. Y., Ai, M. & Zhang, H. (2020). Optimal distributed subsampling for maximum quasi-likelihood estimators with massive data. *Journal of the American Statistical Association* **117**, 265–276.