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

Scienze Statistiche

Tesi di Laurea

Dati funzionali di traffico telefonico: un approccio bayesiano non parametrico

Relatore Prof. Bruno Scarpa
Dipartimento di Scienze Statistiche

Laureando Tommaso Rigon
Matricola 1066585

Anno Accademico 2014/2015

Indice

Introduzione

In questa tesi, così come in ogni analisi statistica, le prime mosse vengono prese a partire dai dati. Si tratta di informazioni provenienti da un'azienda di telecomunicazioni ed in particolare le attività mensili (numero di telefonate, numero di *SMS*), compiute dai vari singoli utenti. L'azienda è interessata a comprendere il comportamento dei suoi clienti e possibilmente anticipare le loro decisioni, onde evitare che passino alla concorrenza o smettano di utilizzare i servizi offerti. Nel primo Capitolo vengono presentati i dati grezzi, poi ripuliti e organizzati. Il problema aziendale viene quindi formalizzato matematicamente e varie tecniche classificazione ben note in letteratura sono state applicate: i modelli lineari generalizzati, modelli *MARS, gradient boosting* (Hastie et al., 2001).

Per poter procedere con l'implementazione e la discussione di modelli più sofisticati, nel secondo Capitolo si è resa necessaria una breve parentesi teorica volta ad introdurre alcuni concetti chiave. Vengono presentati da un punto di vista strettamente probabilistico due processi stocastici di fondamentale importanza per gli scopi di questa tesi: il processo gaussiano (Rasmussen & Williams, 2006) ed il processo di Dirichlet (Hjort et al., 2010).

Nel terzo capitolo, la struttura dei dati stessi viene trattata per quello che è: una collezione di osservazioni di tipo funzionale (Ramsay & Silverman, 2005). Essi vengono modellati esplicitamente come tali, evidenziando, da un punto di vista frequentista, alcune caratteristiche "latenti". In particolare, è stata condotta un'analisi di *clustering funzionale* che ha permesso di identificare le forme funzionali di alcuni gruppi.

Nel quarto capitolo, la struttura funzionale viene ripresa, ma i dati vengono modellati con un approccio di tipo bayesiano non parametrico, inducendo una particolare forma di *clustering funzionale*. Viene introdotto il processo di Dirichlet funzionale e descritta nel dettaglio una procedura Markov Chain Monte Carlo (MCMC), utilizzata per la simulazione dalla distribuzione a posteriori. Essa è stata implementata tramite una combinazione del software R e del più efficiente linguaggio C++. Questa procedura viene poi applicata ai dati in questione e i risultati sono stati analizzati nel dettaglio.

Nel quinto Capitolo il problema di classificazione discusso nel primo Capitolo è affrontato nuovamente, facendo leva, questa volta, sui risultati derivanti dal *clustering funzionale* del quarto Capitolo. Algoritmi di tipo frequentista vengono accostati a modelli bayesiani con effetti casuali. Nel secondo caso, recenti sviluppi (Polson et al., 2013), hanno suggerito

una procedura MCMC adeguata, sfruttando il concetto di *data-augmentation*.

Tutte le metodologie utilizzate sono state messe a confronto, sia da un punto di vista interpretativo che in termini di capacità previsive. Come spesso accade, non esiste un reale "vincitore", ma pregi e difetti di ciascun metodo sono messi in evidenza.

Capitolo 1

Dati di telefonia mobile

1.1 Motivazioni

Una compagnia telefonica è interessata a comprendere il comportamento dei suoi clienti per poter attuare successivamente azioni di marketing opportune. Ciascun cliente è identificato da una *scheda SIM*, che è contrassegnata da un codice identificativo. Si supponga che esse siano di tipo prepagato, ovvero il cliente paga in anticipo il servizio. Fintanto che la scheda risulta attiva, il cliente accumula traffico telefonico, pagando un corrispettivo all'azienda. Per motivi differenti, il cliente può decidere di disattivare la propria scheda per passare alla concorrenza. Questo fenomeno è noto come *churn*.

L'azienda quindi si propone un duplice obiettivo: identificare i clienti a maggior rischio di *churn* ed in seguito effettuare delle azioni di marketing mirate, per contenere il loro abbandono. Il primo problema può essere affrontato utilizzando un approccio statistico, in cui la propensione al *churn* per ciascun cliente è identificata da una probabilità:

$$\pi_i = \mathbb{P}\left(\text{L'}i\text{-esimo cliente disattiva la sim card}\right), \quad i = 1, \ldots, n.$$

dove n indica il numero complessivo di clienti presenti nel database aziendale. Supponiamo, per il momento, che ciascun cliente possieda un'unica scheda sim.

Il secondo problema, ovvero rendere un cliente a rischio di abbandono più fedele alla compagnia, è generalmente di competenza di altre figure professionali, anche se una corretta interpretazione dei risultati da parte dello statistico può fornire suggerimenti. Per un'esposizione più dettagliata circa l'uso della statistica per l'identificazione del *churn*, si può fare riferimento a Berry & Linoff (2004, Capitolo 4).

Per ciascun cliente sono disponibili numerose informazioni, che vengono immagazzinate massivamente nei database aziendali. Esse sono sia di tipo demografico (età, sesso, residenza, etc.), sia legate al comportamento d'uso del servizio (numero di telefonate, numero di sms, piano tariffario, etc.). Tramite un approccio statistico, è possibile sostituire

l'ignota propensione al *churn* π_i con una sua stima $\hat{\pi}_i$, che viene ottenuta come funzione delle informazioni addizionali relative al cliente:

$$\hat{\pi}_i = \hat{f}(\text{informazioni per l'}i\text{-esimo cliente}), \quad i = 1, \ldots, n.$$

Il processo di stima può avvenire attraverso differenti vie: ipotizzando un modello probabilistico ben preciso oppure seguendo un approccio più algoritmico (Breiman, 2001). Alcuni modelli hanno notevoli vantaggi interpretativi, mentre altri tentano principalmente di ottenere previsioni di qualità migliore. Talvolta, esiste un *trade-off* tra capacità predittiva e interpretabilità del modello. In questi casi, c'è una certa discrezionalità nella scelta del modello migliore, poiché la limpidezza dei risultati è valutata soggettivamente dall'analista. Non di rado però accade che modelli "semplici" godano di ottime capacità previsive, permettendo all'analista di contribuire sia nell'identificazione del *churn*, sia nella comprensione delle sue cause.

1.2 Presentazione del dataset

Il *dataset* che verrà preso come esempio è stato analizzato precedentemente da Canale (2012). I dati grezzi sono di due tipologie diverse: statici e dinamici. Le informazioni dinamiche sono fotografie di diversi istanti temporali, mentre quelle statiche non mutano nel tempo. Per 215324 sim card sono disponibili varie informazioni statiche, riportate nella Tabella 1.1.

Il traffico telefonico è invece registrato a cadenza mensile, a partire da Novembre 2004 fino ad Aprile 2006. Ciascuna osservazione è pertanto un evento dinamico, identificato congiuntamente dalla data, dal codice identificativo della sim card e dalla tipologia di evento. In Tabella 1.2 è presente una descrizione dettagliata.

Anzitutto sono state condotte alcune banali operazioni preliminari qui non discusse nel dettaglio, ad esempio la corretta codifica delle date. Ciascun *dataset* è stato presentato separatamente per maggior chiarezza espositiva, ma per effettuare l'analisi è stato necessario accorpare i dati in un unico *dataset*.

E' necessario definire una variabile dicotomica, chiamata `Churn`, che indichi se il cliente è da considerarsi attivo oppure se ha abbandonato la compagnia telefonica. E' importante notare che la data di disattivazione della *SIM card* non coincide con la data in cui il cliente ha preso la decisione di abbandonare la compagnia, che è invece l'oggetto di interesse. Poiché non sono note le effettive intenzioni del cliente, si tenterà di desumerle dai dati stessi. Si definisca una *SIM card* come *silente* se non si registrano telefonate in uscita in una determinata finestra di tempo. Se nell'ultimo periodo osservato una scheda risulta *silente*, allora essa sarà definita come in stato di *churn* a partire dal primo mese di inattività. Si sta infatti assumendo che la data di inizio del periodo di inattività coincida con il momento in cui il cliente ha preso la decisione di abbandonare la compagnia.

Tabella 1.1: Struttura dataset con informazioni statiche. Ciascuna sim card rappresenta un'osservazione, per un totale di 215324. Le informazioni riguardano *SIM card* attivate tra Marzo 2013 e Dicembre 2013.

Variabile	**Descrizione**	**Note**
ID	Codice interno che identifica univocamente la sim card.	
Stato	Indica se la sim card è attiva, disattiva o sospesa, a Maggio 2006	
Causale	In caso di avvenuta disattivazione, specifica il motivo per quale essa è avvenuta.	
Attivazione	Data di attivazione della sim card.	
Piano tariffario	Variabile qualitativa che specifica il piano tariffario attivo.	
Ultima	Data dell'ultima chiamata effettuata dal cliente al momento della rilevazione.	
Sesso	Sesso del cliente.	Presenza di dati mancanti.
Nascita	Data di nascita del cliente.	Presenza di dati mancanti. Informazioni a volte inaffidabili.
Provincia	Provincia di residenza del cliente.	Presenza di dati mancanti.

Tabella 1.2: Struttura dataset con informazioni dinamiche. Ciascuna riga è un evento, identificato dalla terna *ID, Tipo, Traffico*. Le informazioni riguardano il traffico tra Novembre 2004 ed Aprile 2006

Variabile	**Descrizione**	**Note**
ID	Codice interno che identifica univocamente la sim card.	
Tipo	Tipologia di evento: SMS, MMS, chiamate verso fisso, chiamate verso differenti operatori	
Traffico	Mese di riferimento per ciascun evento.	
Eventi	Numero di volte che si è verificato l'evento.	
Minuti	Variabile quantitativa che misura la durata complessiva delle telefonate, espressa in minuti.	Vale 0 in caso di SMS o MMS

Le operazioni effettuate sono pertanto riassunte:

1. Le diverse tipologie di traffico sono state aggregate in due grandi categorie: il traffico proveniente dalle telefonate in uscita e quello proveniente dai servizi di messaggistica in uscita. Per ciascuna macro categoria sono state ottenute la somma degli eventi e la somma delle durate delle telefonate.

2. Unione dei due dataset tramite un'operazione *JOIN*, utilizzando la variabile *ID* come identificativo unico.

3. Creazione della variabile qualitativa `Churn`, che indica la disattivazione della *SIM card*. Per ottenere la variabile `Churn` sono state considerate le schede *silenti* nei mesi di Febbraio, Marzo ed Aprile 2006, ovvero quelle che non presentavano telefonate in uscita in nessuno dei tre mesi.

4. Le *SIM card* che risultavano in stato di *churn* in un periodo precedente a Febbraio 2006 sono state eliminate dal dataset. In Tabella 1.3 è spiegato, tramite un esempio, come sono state trattate queste variabili.

Tabella 1.3: Esempio delle operazioni preliminari. I valori della tabella rappresentano il numero di telefonate avvenuto nel mese corrispettivo per differenti *SIM card*.

Operazione effettuata	*Gennaio 2006*	*Febbraio 2006*	*Marzo 2006*	*Aprile 2006*
Churn: No	5	5	5	5
Churn: No	0	0	0	5
Churn: Si	5	0	0	0
Eliminazione dal dataset	0	0	0	0

5. Eliminazione delle variabili non utilizzate per l'analisi. Eliminazione dei dati di traffico relativi al periodo Febbraio, Marzo ed Aprile 2006. Il motivo è semplice: volendo prevedere un evento avvenuto all'inizio di Febbraio 2006, non è lecito sfruttare informazioni avvenute solo in seguito.

6. Riorganizzazione del dataset. Ciascuna riga rappresenta nuovamente una singola *SIM card*. Le informazioni relative al traffico mensile sono riorganizzate per colonna invece che per riga: ciascuna mensilità diviene una variabile.

Queste operazioni sono state svolte tramite il software R Core Team (2015), ed in particolare grazie al recente ed efficiente pacchetto R `dplyr` di Wickham & Francois (2015).

1.3 Analisi descrittive

Il *dataset* ottenuto nel paragrafo precedente è pertanto il punto di partenza di tutte le analisi successive. Esso ha ora una struttura semplice, riportata in Tabella 1.4. Per il momento, si è deciso di focalizzare l'attenzione sul numero di telefonate mensili, escludendo altre possibili variabili. Il numero di osservazioni, a seguito di tutte le modifiche, è pari a $n = 26.010$. La grande perdita di informazione avviene principalmente nell'unificazione dei due *dataset*. Infatti i dati relativi al traffico telefonico (Tabella 1.2), contengono informazioni di sole 34.101 *SIM card* distinte.

Tabella 1.4: Sono mostrate alcune osservazioni del dataset oggetto di analisi. Dove non diversamente specificato, ciascun numero rappresenta il numero di telefonate in uscita nel mese corrispettivo.

	ID	*churn*	*Novembre 2004*	*Dicembre 2004*	*Gennaio 2005*	$\cdots$	*Gennaio 2006*
1	78506	No	5	18	8	$\cdots$	17
2	85377	No	6	31	7	$\cdots$	12
3	85831	Si	29	2	0	$\cdots$	7
5	100174	No	8	21	2	$\cdots$	4
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
26010	5101053	No	0	0	0	$\cdots$	0

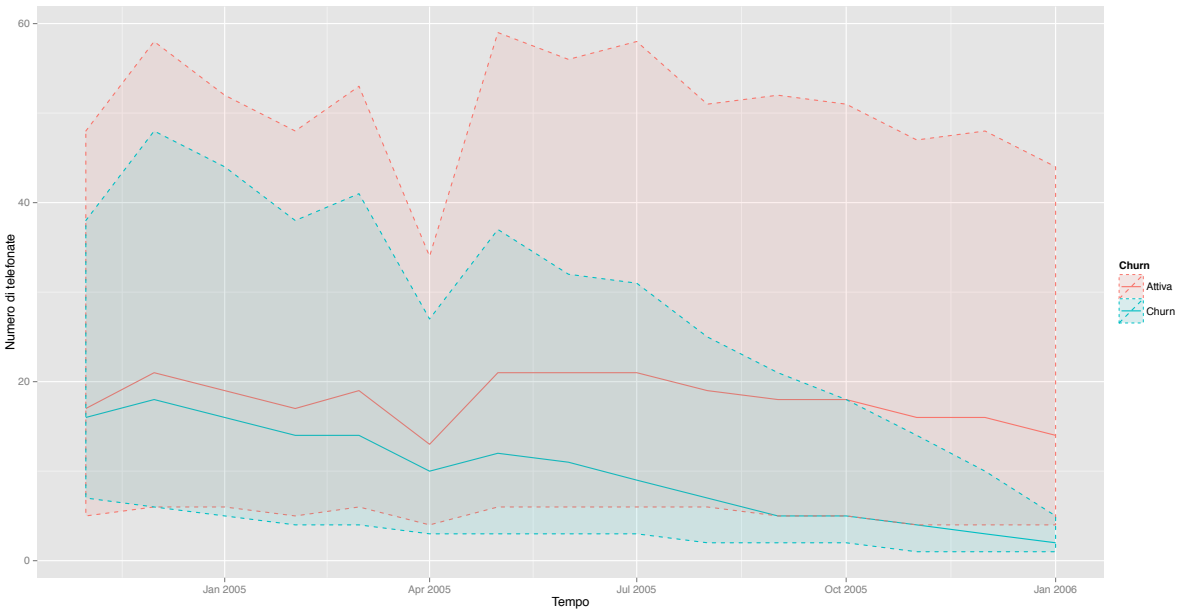
A ciascuna *SIM card* è associata una breve serie storica a cadenza mensile. In Figura 1.1(a) sono rappresentati i trend mediani rispetto a tutte le possibili *SIM card*. La distinzione tra *SIM* inattive e *SIM* ancora attive è piuttosto evidente già da una prima ispezione grafica. I clienti che tendono ad abbandonare la compagnia telefonica sono principalmente coloro che meno utilizzano il servizio, fenomeno accentuato in prossimità del periodo di disattivazione. Entrambi i *trend* manifestano un forte calo nel mese di Aprile 2005: ciò è presumibilmente dovuto ad un errore tecnico presente nei dati originari.

In Figura 1.1(b) viene presentato un grafico analogo in forma di *boxplot*, per evidenziare la forte asimmetria che caratterizza le distribuzioni marginali, causata da alcuni conteggi estremamente elevati.

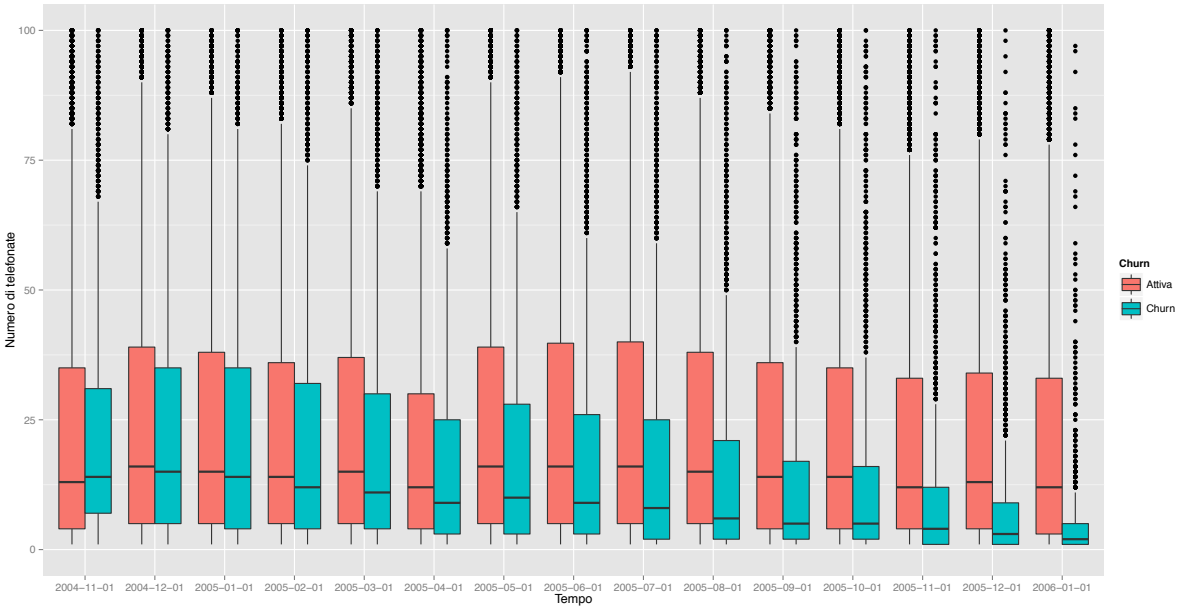
1.4 Previsione del *churn*

1.4.1 Operazioni preliminari

Si è intenzionati a prevedere se una *SIM card* risulterà inattiva o meno. Questo problema è detto di *classificazione* (Azzalini & Scarpa, 2011, Capitolo 5). La disattivazione



(a) Le linee continue rappresentano i trend mediani rispettivamente in presenza ed in assenza di *churn*. Le aree colorate, delimitate dalle linee tratteggiate, rappresentano il primo ed il terzo quartile.



(b) I boxplot delle *SIM* disattive (verde) sono messi a confronto con quelli delle *SIM* attive (blu), mese per mese. Il grafico è limitato all'intervallo $(0, 100)$ per facilitarne la lettura. I "pallini" rappresentano valori anomali di ciascuna distribuzione.

Figura 1.1: Analisi descrittive

rappresenta la realizzazione di una variabile casuale Y_i che assume valore 1 (*SIM* inattiva), oppure 0 (*SIM* attiva). Sia X una matrice di dati contenente il numero di telefonate mensili per ciascuna sim card. Il generico elemento x_i è esprimibile dal vettore

$$x_i = (x_{i1}, \ldots, x_{ip})^T.$$

Il modello associato è nella forma

$$\pi_i = \mathbb{P}(Y_i = 1) = f(x_i), \quad i = 1, \ldots, n, \tag{1.1}$$

in cui $f(\cdot) : \mathbb{R}^p \to (0, 1)$, essendo π_i una probabilità. Esistono, in realtà, modelli per la classificazione in cui $f(\cdot)$ ha ad esempio codominio reale ma che non verranno trattati. Supposto di disporre di una stima $\hat{\pi}_i$, ciascuna osservazione verrà identificata come inattiva se $\hat{\pi}_i > c$ e attiva altrimenti, dove c è una costante fissata.

Per poter successivamente valutare la bontà del modello senza distorsioni, il dataset è stato suddiviso in due parti dette *insieme di stima* e *insieme di verifica*, selezionate casualmente, nella proporzione del 75% e 25%, rispettivamente. Ciascun modello verrà stimato utilizzando la prima partizione mentre verrà valutato utilizzando la seconda. Questa operazione è necessaria per evitare il fenomeno chiamato *sovradattamento*. La motivazione di questa operazione è fornita ad esempio in Hastie et al. (2001, Capitolo 7) e ne verrà qui data solo un'intuitiva spiegazione. Se si suppone di utilizzare l'errore quadratico medio (*MSE, Mean Squared Error*), come misura complessiva di bontà della stima per $f(\cdot)$, si ha che

$$\begin{aligned} \text{MSE} &= \mathbb{E}\left[\left(f(x) - \hat{f}(x)\right)^2\right] \\ &= \mathbb{E}\left[\left(f(x) - \hat{f}(x)\right)\right]^2 + \text{Var}\left(\hat{f}(x)\right) = \text{Distorsione}^2 + \text{Varianza}. \end{aligned}$$

Auspicabilmente, l'indice *MSE* è il più piccolo possibile. La scomposizione precedente suggerisce l'esistenza di un compromesso tra distorsione e varianza: non è infatti possibile minimizzare entrambi contemporaneamente. Non potendo calcolare l'indice *MSE* analiticamente, dato che esso dipende da quantità ignote, questo viene stimato a sua volta. La stima $\hat{MSE}$ basata sull'insieme di stima generalmente sottovaluta la reale variabilità di $\hat{f}(\cdot)$. Pertanto, tale indicatore, o altri equivalenti, vengono ottenuti grazie all'insieme di verifica.

Sono stati quindi stimati tre differenti modelli di crescente ordine di complessità. In tutti e tre i casi, la dipendenza temporale tra le osservazioni è stata trascurata. Le variabili esplicative infatti contengono informazioni di tipo longitudinale che possono essere modellate includendo tra le covariate dei semplici tassi di crescita. In questa fase si è preferito una più semplice analisi mentre nei Capitoli successivi tali informazioni verranno opportunamente trattate.

1.4.2 Regressione logistica

I modelli lineari generalizzati, ed in particolare il modello di regressione logistica, sono stati ampiamente discussi in letteratura. Si rimanda a McCullagh & Nelder (1989) oppure Azzalini (2008, Capitolo 6), per ogni dettaglio. Viene specificato un predittore lineare tale che

$$\text{logit}(\pi_i) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}.$$

Il vettore di parametri $\beta = (\beta_0, \ldots, \beta_p)$ è stimato tramite il criterio di massima verosimiglianza (*MLE, Maximum Likelihood Estimate*), sfruttando l'efficiente algoritmo IWLS (*Iterated Weighted Least Squares*). Il vettore delle stime $\hat{\beta}$ è pertanto pari a

$$\hat{\beta} = \arg\sup_{\beta} l(\beta; y), \tag{1.2}$$

dove $l(\beta; y)$ è la funzione di logverosimiglianza per il modello di regressione logistica. Il modello "ottimale" è stato ottenuto tramite una procedura *stepwise*, utilizzando il criterio di informazione AIC (Akaike, 1973), in cui le variabili esplicative vengono aggiunte ed eliminate sequenzialmente.

Il metodo della massima verosimiglianza non è l'unica procedura inferenziale esistente: è possibile stimare lo stesso modello con metodi bayesiani, i quali sono descritti ad esempio in Carlin & Louis (2008). Si definisca la cosiddetta *distribuzione a posteriori* come

$$p(\beta \mid y) \propto L(y \mid \beta) p(\beta), \tag{1.3}$$

in cui $L(y \mid \beta)$ è la funzione di verosimigliaza e $p(\beta)$ la *distribuzione a priori*. Si noti come nella transizione concettuale tra approccio frequentista e bayesiano la notazione per le quantità di verosimiglianza sia mutata. Il modello è inteso ora *condizionatamente* ai parametri, così come evidenziato dalla notazione. La distribuzione a posteriori richiede il calcolo della cosiddetta *costante di normalizzazione*

$$\int L(y \mid \beta) p(\beta) d\beta. \tag{1.4}$$

Poichè essa è, in questo caso, sostanzialmente impossibile da ottenere anche numericamente, si seguirà l'approccio proposto da Polson et al. (2013) per questa classe di modelli, che verrà descritto nel dettaglio nel Capitolo 5. Esso consente di simulare abbastanza efficientemente valori dalla distribuzione a posteriori $p(\beta \mid y)$, da cui si ottiene un'approssimazione per la cosiddetta media a posteriori

$$\beta_B = \mathbb{E}\left[p(\beta \mid y)\right], \tag{1.5}$$

spesso utilizzata come stima puntuale bayesiana. Sono stati selezionati degli iperparametri che hanno indotto una distribuzione a priori estremamente vaga e quindi poco informativa. Il modello bayesiano stimato è lo stesso ottenuto tramite la procedura frequentista

stepwise, per consentire un confronto. Il risultato di entrambe le stime è riportato in Tabella 1.5: non sorprendentemente i due metodi sono estremamente simili, tanto da essere indistinguibili se si considera un'approssimazione alla terza cifra decimale.

Tabella 1.5: Modello di regressione logistica: stima frequentiste e bayesiane. Solamente l'errore standard frequentista, con relativo p-value, è riportato in tabella, per semplicità di lettura. La log-verosimiglianza e l'indice AIC si riferiscono al modello frequentista.

	$\hat{\beta}_B$	$\hat{\beta}$ *MLE*	Err. std. (*MLE*)	Valore z	$\Pr(>\|z\|)$
(Intercept)	-0.8180	-0.817	0.0268	-30.502	$< 10^{-16}$
Novembre 2004	0.007	0.007	0.0009	7.979	1.48×10^{-15}
Dicembre 2004	0.001	0.001	0.0010	1.559	0.11893
Gennaio 2005	0.002	0.001	0.0009	1.842	0.06546
Luglio 2005	-0.002	-0.002	0.0007	-3.150	0.00163
Dicembre 2005	-0.011	-0.011	0.0021	-5.178	2.25
Gennaio 2006	-0.139	-0.139	0.0054	-25.671	$< 10^{-16}$
log-verosimiglianza:	-7221.456	(g.d.l. =7)		AIC:	14456.91

I coefficienti ottenuti tramite la procedura frequentista non necessariamente risultano significativamente diversi da 0, poiché nessun test statistico è stato condotto in fase di selezione del modello. Per quel che riguarda la procedura bayesiana, invece, si riporta in Figura 1.2 la distribuzione a posteriori per la probabilità di *churn* $\pi_i \mid y$ per un cliente estratto casualmente dal *dataset*. L'approccio bayesiano consente di comunicare graficamente quanto sia affidabile la stima, stante la corretta specificazione del modello: ciò può risultare più intuitivo rispetto all'interpretazione di un intervallo di confidenza frequentista.

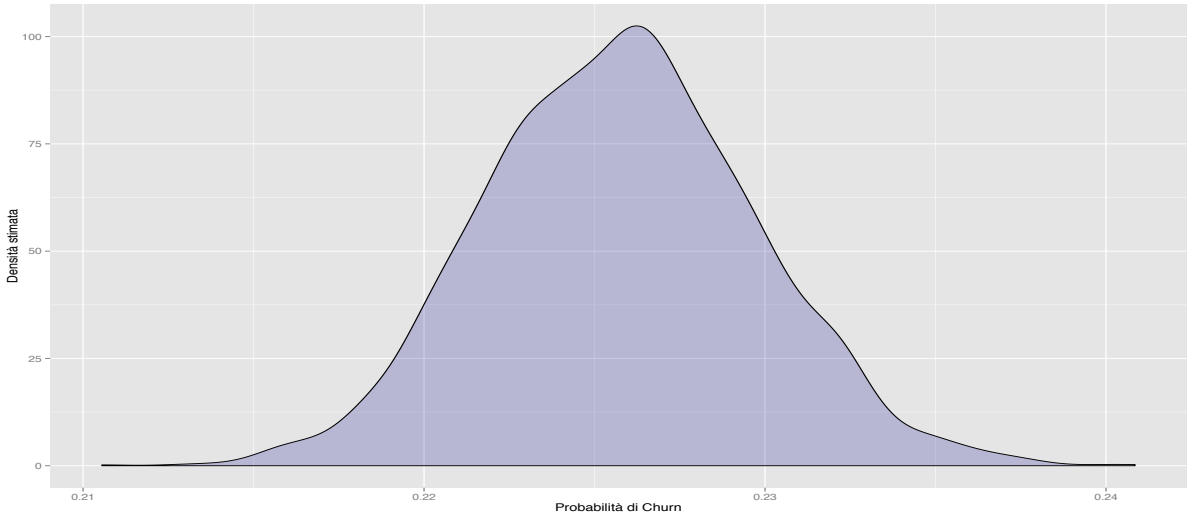
1.4.3 MARS (*Multivariate Adaptive Regression Splines*)

I modelli MARS vennero inizialmente introdotti da Friedman (1991), ma sono descritti in forma più discorsiva ad esempio in Hastie et al. (2001, Capitolo 9), a cui si rimanda per approfondimenti. Rispetto ai modelli lineari generalizzati, essi sono più flessibili e consentono una modellazione automatica di effetti di interazione tra covariate. Nel caso di problemi di classificazione, con funzione legame logistica si ha che

$$\text{logit}(\pi_i) = \beta_0 + \sum_{m=1}^{M} \beta_m \phi_m(x_i), \tag{1.6}$$

in cui le funzione di base $\phi_m(\cdot)$ sono particolari splines, le quali verranno descritte più nel dettaglio nel Capitolo 3. Essi prevedono una procedura di stima sequenziale che consiste in tre fasi:

Figura 1.2: Distribuzione a posteriori della probabilità di *churn* per cliente estratto casualmente dal dataset.



1. **Fase di crescita**. Si comincia con il modello costituito dalla sola intercetta. Ad ogni ciclo, si procede come in una procedura *stepwise* in cui, tuttavia, sono aggiunte al predittore lineare coppie di funzioni di base del tipo $\phi_m(\cdot)$. Si procede fino a saturazione o fino ad una soglia prefissata.

2. **Fase di potatura**. Tramite il metodo della convalida incrociata generalizzata (Hastie et al., 2001, Capitolo 7), o metodi equivalenti, alcune basi vengono eliminate sequenzialmente, non necessariamente a coppie.

3. **Fase di stima**. Utilizzando le basi selezionate alla fine della fase di potatura, viene stimato un modello di regressione logistica utilizzando gli usuali algoritmi di stima.

Nel caso specifico, le funzioni di base sono splines della forma

$$(x_j - \xi)_+ \text{ oppure } (\xi - x_j)_+, \tag{1.7}$$

dove ξ è a sua volta un parametro ignoto, chiamato nodo, che assume valori solo nello spazio dei valori osservati. La determinazione dei nodi è quindi parte integrante del processo di stima. La notazione $(x)_+$ indica la parte positiva di x.

Sebbene i MARS ammettano la formulazione di basi di funzioni più complesse rispetto alla (1.7), in questo caso l'attenzione è stata ristretta a termini privi di interazione, per evitare eccessiva instabilità nella fase di stima. I risultati sono riportati in Tabella 1.6, ottenuti tramite il pacchetto R `earth` di Milborrow (2015). I gradi di libertà indicati sono un indicatore distorto della complessità del modello, poiché è stata trascurata la variabilità relativa ai nodi stimati $\hat{\xi}$.

Tabella 1.6: Risultati del modello MARS

	Coefficiente	Errore standard	z value	$\Pr(>\|z\|)$
(Intercept)	-4.994	0.136	-36.621	$< 10^{-16}$
$(\textit{Gennaio 2006} - 13)_+$	-0.022	0.004	-6.178	6.51×10^{-10}
$(15 - \textit{Novembre 2004})_+$	-0.114	0.004	-28.946	$< 10^{-16}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$(19 - \textit{Agosto 2005})_+$	0.019	0.004	4.761	1.92×10^{-6}
log-verosimiglianza:	-6374.094	(g.d.l. =10)	AIC:	12768.19

1.4.4 Gradient Boosting

Il *gradient boosting* è un metodo piuttosto generale che può essere utilizzato sia per problemi di classificazione che di stima. Inizialmente, Freund & Schapire (1997) proposero l'algoritmo *Adaboost.M1* il quale venne in seguito collocato in un contesto statistico da Hastie et al. (2000). Ne verrà data qui una breve rassegna mentre per una presentazione semplice ma esaustiva, si veda Hastie et al. (2001, Capitolo 10). Si supponga di voler approssimare una generica funzione come sommatoria di funzioni di base

$$f(x_i) = \sum_{m=1}^{M} \beta_m \phi(x_i; \gamma_m), \tag{1.8}$$

in cui $\phi(x; \gamma_m) : \mathbb{R}^p \to \mathbb{R}$, sono funzioni con argomento multivariato che dipendono da un ignoto insieme di parametri γ. I valori $m = 1, \ldots, M$ sono indici che identificano la m-esima funzione di base. I modelli MARS sono pertanto un caso particolare della (1.8), in cui la funzione $\phi(x; \gamma)$ ha una forma particolare ed i parametri ignoti sono i nodi ξ. La stima degli ignoti parametri (β, γ) avviene tramite la minimizzazione di una particolare funzione di perdita L. Idealmente, vorremmo ottenere i valori che rendono minima la quantità

$$\arg\min_{\beta,\gamma} \sum_{i=1}^{n} L\left(y_i, \sum_{m=1}^{M} \beta_m \phi(x_i; \gamma_m)\right). \tag{1.9}$$

Tuttavia, ciò è computazionalmente troppo oneroso nella quasi totalità dei casi pratici. La minimizzazione di un singolo elemento è invece più trattabile

$$\arg\min_{\beta_j,\gamma_j} \sum_{i=1}^{n} L\left(y_i, \beta_j \phi(x_i; \gamma_j)\right). \tag{1.10}$$

L'idea sottostante alla procedura chiamata *Forward Stagewise Additive Modeling* è quindi di approssimare l'equazione (1.8) tramite minimizzazioni successive, così come descritto nell'Algoritmo 1.

Algorithm 1 Forward Stagewise Additive Modeling

1. Inizializzare $f_0(x) = 0$
2. Per $m = 1$ a M
   (a) Si ottenga
   $$(\hat{\beta}_m, \hat{\gamma}_m) = \arg\min_{\beta_m, \gamma_m} \sum_{i=1}^{n} L\left(y_i, f_{m-1}(x) + \beta_m \phi(x_i; \gamma_m)\right)$$
   (b) Si imposti
   $$f_m(x) = f(x)_{m-1} + \hat{\beta}_m \phi(x; \hat{\gamma}_m)$$

A ciascun ciclo dell'Algoritmo 1, la minimizzazione parziale migliora l'approssimazione. Si può dimostrare che se la funzione di perdita è esponenziale, cioè se $L(y_i, f(x_i)) = \exp\{y_i f(x_i)\}$, allora questa procedura coincide con l'algoritmo *AdaBoost.M1*. Al posto della funzione $\phi(x; \gamma)$ vengono spesso utilizzate particolari basi, ovvero gli alberi di regressione. Formalmente infatti un albero può essere scritto come

$$A(x; \Theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j), \tag{1.11}$$

dove l'insieme dei parametri è dato da $\Theta = \{R_j, \gamma_j\}_1^J$.

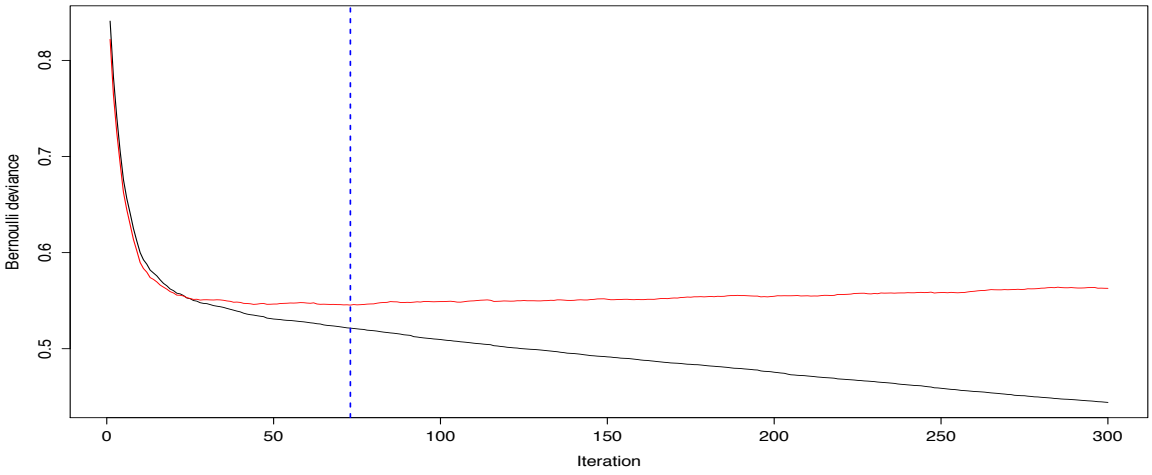
Il gradient stochastic boosting è un'ulteriore modificazione di quanto esposto finora proposta in Friedman (2002). Tralasciando ulteriori formalismi, esso prevede:

1. Una correzione nella stima dell'm-esimo albero basata sul gradiente della funzione di perdita.
2. Un aggiornamento debole, ponderato per un parametro di apprendimento, chiamato *tasso di apprendimento*. Valori piccoli del tasso di apprendimento migliorano la stima, ma rallentano la convergenza.
3. A ciascuna interazione, non vengono utilizzate le osservazioni originali per la stima, ma un sotto campione estratto casualmente, senza reinserimento. Friedman (2002) ha mostrato che questo ha il duplice effetto di aumentare la capacità previsiva pur diminuendo il costo computazionale.

Il pacchetto R `gbm` di Ridgeway et al. (2015) conduce abbastanza efficientemente l'intera procedura. Essa si basa su di un'ulteriore ripartizione dell'originario insieme di stima, per un totale di tre suddivisioni. La prima parte viene utilizzata per l'effettiva stima, la seconda per selezionare il numero ottimale di iterazioni, la terza per il confronto finale.

Per il caso specifico, il *tasso di apprendimento* è stato fissato pari a 0.15, selezionato dopo alcuni tentativi volti a verificare l'effettivo assestamento del processo. Si è posto inoltre $M = 300$ come numero massimo di iterazioni. Trattandosi di un problema di classificazione, la funzione di perdita è la devianza di una distribuzione binomiale. In Figura 1.3 sono stati calcolati i valori della funzione di perdita rispetto al primo insieme (stima) e alla seconda porzione (verifica). La terza porzione è invece riservata per la verifica conclusiva. Si noti come nell'insieme di verifica il miglioramento sia evidente nelle prime interazioni ma tenda ad assestarsi rapidamente. Il modello selezionato è quindi quello che, alla m-esima iterazione dell'Algoritmo 1, minimizza la funzione di perdita nell'insieme di verifica.

Figura 1.3: Assestamento del gradient boosting. Nell'asse delle ascisse si trova il valore m di interazioni. Sull'asse delle ordinate è rappresentata la devianza binomiale valutata nell'insieme di stima (linea nera) e nell'insieme di validazione (linea rossa). La linea tratteggiata verticale invece evidenza il numero di interazioni ottimale.



Generalmente il *gradient boosting* è piuttosto accurato in termini previsivi, ma la semplicità interpretativa dei modelli lineari viene quasi completamente persa.

1.5 Confronto complessivo

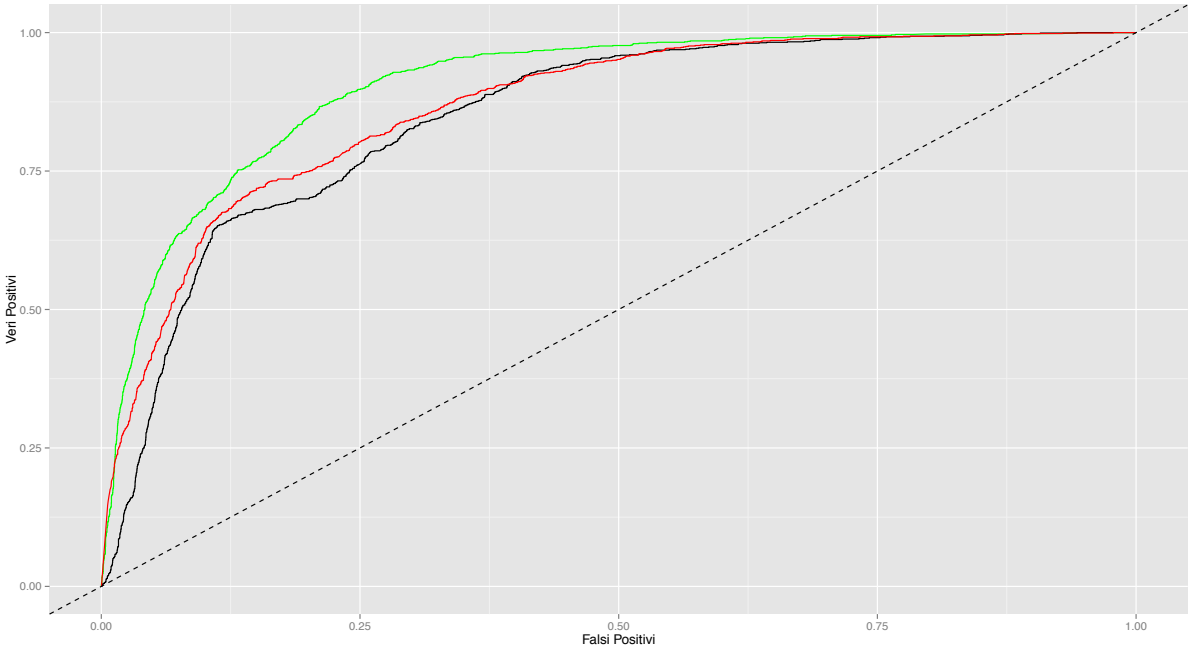
Il tasso di errata classificazione viene spesso usato come indicatore di bontà di previsione. Esso tuttavia dipende fortemente dalla soglia c utilizzata. Per appurare quale sia il modello migliore in termini di previsione, verrà quindi utilizzato l'indice AUC (*Area under the ROC curve*), valutato nell'insieme di verifica. Esso infatti non dipende dalla soglia c poichè, in un certo senso, tiene conto di tutte le possibili scelte. Quando l'indice AUC

è pari a 1, il modello è sostanzialmente deterministico e classifica senza errore ciascuna osservazione. Quando invece esso è pari a 0.5, allora il modello ha la stessa capacità previsiva delle proporzioni iniziali. La curva ROC è descritta brevemente in Azzalini & Scarpa (2011, Capitolo 5). Un secondo indice spesso utilizzato è il tasso di errata classificazione, definito come

$$\varepsilon(c) = \frac{\#\text{ di classificazioni errate}}{\#\text{ di osservazioni nell'insieme di verifica}} \tag{1.12}$$

Altri indicatori utili sono il tasso dei falsi positivi e il tasso dei falsi negativi. La soglia è stata fissata ponendo $c = 0.5$. In Tabella 1.7 sono riportati gli indici per i modelli precedentemente descritti mentre in Figura 1.4 le curve ROC sono state rappresentate.

Figura 1.4: Confronto tra curve ROC. Linea nera: modello di regressione logistica; linea rossa: modello MARS; linea verde: gradient boosting.



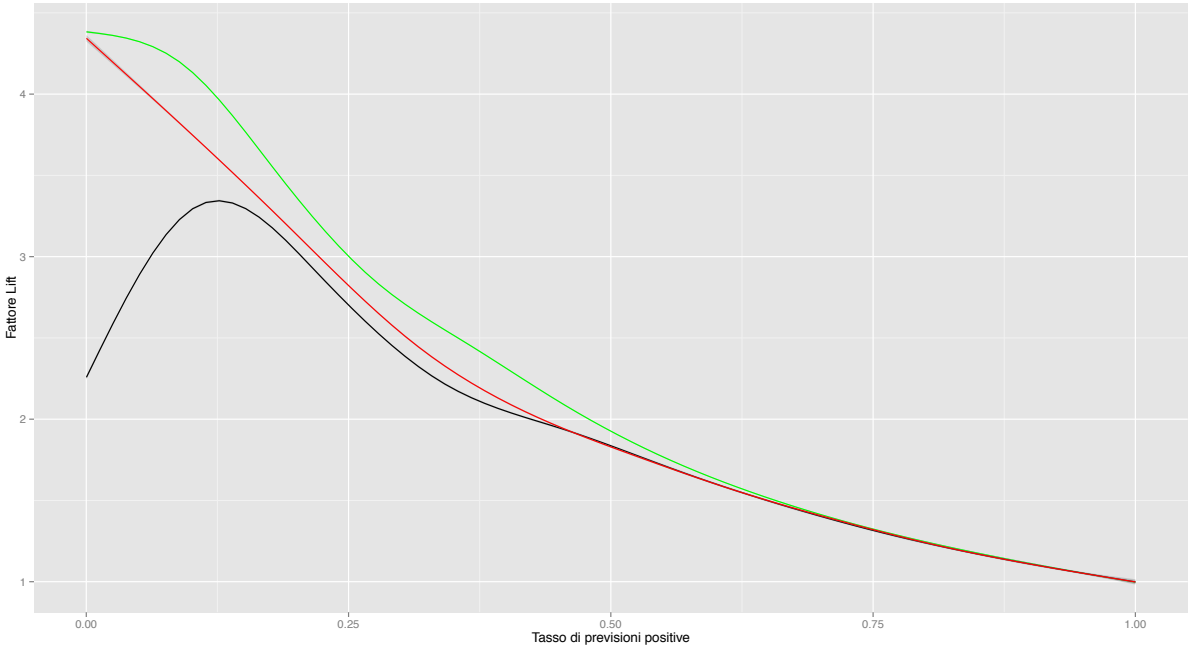
Come spesso accade, la tecnica del gradient boosting fornisce una previsione leggermente migliore rispetto agli altri modelli. In questo caso l'interpretabilità risulta difficile anche per un modello di regressione logistica e pertanto un modello più complesso che tuttavia primeggia in termini previsivi è indubbiamente preferibile.

Un grafico diverso, ma altrettanto importante per il caso specifico è mostrato in Figura 1.5. Esso rappresenta il confronto tra curve Lift per ciascun modello. Il fattore "Lift" è il rapporto tra la previsione fornita dal modello e quella che si otterrebbe utilizzando le proporzioni iniziali. Pertanto un fattore Lift pari a 5 indica che il modello prevede 5 volte meglio rispetto al classificatore banale. Nell'asse delle ascisse, invece, è rappresentata la quota di osservazioni previste.

Tabella 1.7: Confronto degli indici AUC, tasso di errata classificazione, falsi positivi, falsi negativi

	AUC	Errore complessivo	Falsi Positivi	Falsi Negativi
Binomiale	0.849	0.184	0.500	0.176
Binomiale bayesiano	0.849	0.185	0.503	0.176
MARS	0.867	0.147	0.339	0.121
Gradient boosting	0.904	0.126	0.290	0.099

Figura 1.5: Confronto tra curve Lift. Linea nera: modello di regressione logistica; linea rossa: modello MARS; linea verde: gradient boosting. Tutte e tre le curve sono state preventivamente "lisciate", per migliorarne la leggibilità.



1.6 Metodi bayesiani, metodi frequentisti

In questo Capitolo, così come in quelli successivi, procedure inferenziali di tipo bayesiano si affiancano a quelle di carattere frequentista. Esse differiscono notevolmente da un punto di vista interpretativo: alcuni principi, ad esempio quello del *campionamento ripetuto*, sono accettati in un paradigma ma non nell'altro. La probabilità stessa è intesa come *frequenza limite*, in un caso, e basata sul concetto soggettivo di scommessa nell'altro. Entrambi gli approcci hanno solide motivazioni teoriche che solo in parte si sono conciliate nell'arco degli anni. Vi sono stati, ad ogni modo, punti di contatto. Si citano solo alcuni esempi rilevanti: la consistenza frequentista di stime bayesiane è trattata in Diaconis & Freedman (1986). Tecniche come il *bootstrap* sono risultate legate alla statistica bayesiana, dando luogo al cosiddetto *bayesian bootstrap*: si veda ad esempio Newton & Raftery (1994), oppure Efron (2012). Inoltre, esiste una branca della statistica bayesiana, detta bayesiana oggettiva, che si occupa della specificazione di distribuzioni a priori che contengano meno informazione possibile, nel tentativo di riottenere l'oggettività tipicamente frequentista in un contesto bayesiano. Si veda Berger (2006) in proposito.

Da un punto di vista puramente applicato, si potrebbe ignorare ogni implicazione di tipo "filosofico" fintanto che i differenti approcci risolvono il problema in oggetto. Sebbene questa sia una pratica diffusa, si è deciso di quantomeno menzionare il problema, consapevoli della sua complessità. In Rao (2007) il ruolo della statistica viene descritto nella sua evoluzione durante gli anni, fino ai giorni nostri, in cui discipline come il *data mining* ed il *machine learning* (ancor più di recente è in voga il termine *data science*), sono diventate sempre più rilevanti. Il prof. Rao, nelle conclusioni, suggerisce che la convalida incrociata o la semplice ripartizione in insieme di stima ed insieme di verifica, permettono di evitare eventuali considerazioni di tipo stocastico.

Un po' semplicisticamente, si potrebbe quindi concludere che la statistica abbia come unico scopo la generazione di "algoritmi" che funzionino efficacemente in termini previsivi, evitando di porre troppa enfasi sul processo inferenziale. Non si vuole sostenere ciò in questa tesi: nonostante le difficoltà che ne conseguono, la specificazione di un modello probabilistico sottostante alla procedura di stima permette una maggiore interpretabilità e può facilitare i processi decisionali. La previsione, inoltre, non è l'unica applicazione della statistica, la quale può essere usata per verificare e quantificare teorie note in letteratura e non solo per definire nuove relazioni.

Un approccio algoritmico è pertanto parte integrante del bagaglio culturale di uno statistico ma non è autosufficiente. Tuttavia, esso giustifica la commistione di paradigmi inferenziali differenti, trattando ciascuna tecnica, anche quelle bayesiane, come generiche "funzioni dei dati". Nel corso della tesi questa visione viene abbracciata, giustificata dal contesto applicato, ponendo quindi maggiore enfasi sul risultato. La ripartizione dei dati in insieme di stima e di verifica non può prescindere dal *principio del campionamento*

ripetuto. Questa tesi pertanto rimane di stampo frequentista, nonostante il largo utilizzo di tecniche bayesiane. Un approccio puramente bayesiano richiederebbe il condizionamento rispetto al campione osservato e questa ipotesi non verrà rispettata.

Capitolo 2

Strumenti probabilistici

2.1 Processo gaussiano (GP)

In questo Capitolo verranno presentati alcuni strumenti probabilistici ben noti in letteratura, ma ritenuti non elementari. Di rado vengono menzionati in un corso di Laurea Magistrale. Inoltre, questo consente di definire una notazione opportuna, che verrà utilizzata anche nei Capitoli successivi.

In accordo con Rasmussen & Williams (2006), si definisce un processo gaussiano (*Gaussian Process, GP*)

Definizione 1 *Un processo gaussiano è una raccolta di variabili casuali e ciascun sottonsieme di essa ha distribuzione gaussiana.*

Un GP è pertanto un processo stocastico, completamente identificato dalla sua funzione media e funzione covarianza. Si può quindi scrivere:

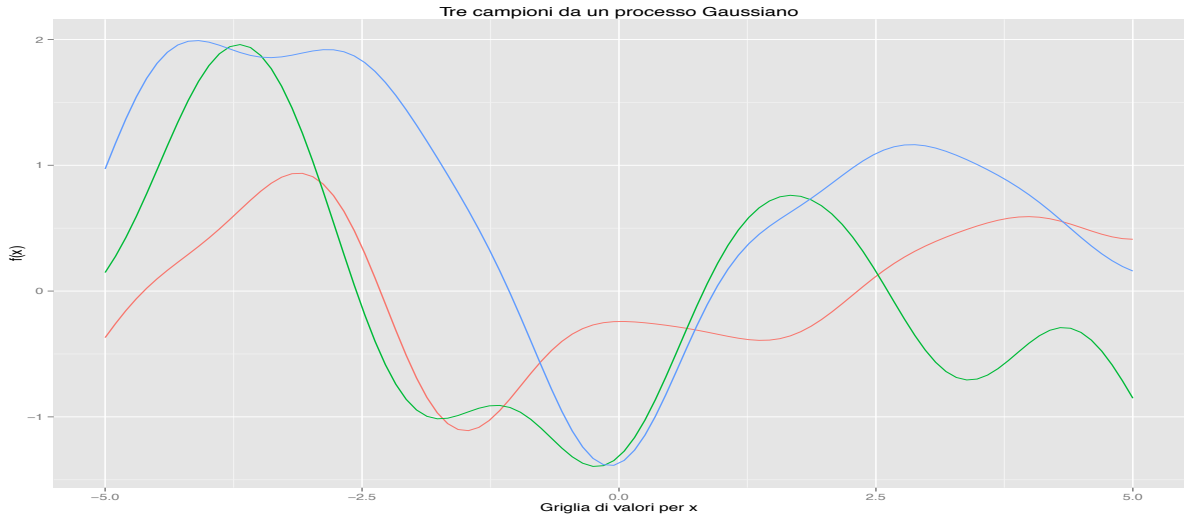
$$f \sim GP\left(m(x), C(x, x')\right), \tag{2.1}$$

in cui $m(x) : \mathbb{R} \to \mathbb{R}$ è la funzione media e $C(x, x') : \mathbb{R}^2 \to \mathbb{R}$ è la funzione covarianza. Ciascuna estrazione da un GP è una funzione. Nei casi pratici esso viene generalmente valutato in una griglia finita di punti x_i per $i = 1, \ldots, n$, ed in tal caso esso ha distribuzione normale multivariata di media μ_x e matrice di covarianza C_{xx}. Nei Capitoli successivi, per semplificare la notazione e qualora ciò non comportasse ambiguità, questa matrice verrà denotata semplicemente con C oppure con $C(\kappa)$, per sottolineare la dipendenza da eventuali parametri aggiuntivi. Entrambe le quantità vengono ottenute a partire dalla funzione media e funzione covarianza, calcolate nella griglia x. Si scriverà

$$f(x) \sim N_n\left(\mu_x, C_{xx}\right). \tag{2.2}$$

La forma della funzione estratta dipende fortemente dalla funzione di covarianza, che è chiamata *funzione nucleo*.

Figura 2.1: Esempio di estrazione pseudocasuale da un GP a media nulla e funzione covarianza definita dall'equazione (2.3).



Generazione di valori pseudocasuali

Sia per fini inferenziali che illustrativi è utile saper generare valori pseudocasuali da una determinata distribuzione. Trattandosi di un processo stocastico continuo, esso si intende valutato in un insieme finito di punti. Per poter generare un campione da un GP, per semplicità a media nulla, è sufficiente specificare una *funzione nucleo*. Ad esempio, si supponga che valga

$$C(x_i, x_j) = \exp\left\{-\frac{1}{2}(x_i - x_j)^2\right\}, \qquad \forall i, j, \tag{2.3}$$

che è nota come covarianza esponenziale. Si anticipa che essa può esser generalizzata come segue (Rasmussen & Williams, 2006, Pag. 20):

$$C(x_i, x_j; \kappa) = \kappa_2 \exp\left\{-\kappa_1(x_i - x_j)^2\right\} + \kappa_3 \delta_{ij}, \qquad \forall i, j, \tag{2.4}$$

dove δ_{ij} è in genere la funzione indicatrice $I(x_i = x_j)$. Per un rapido calcolo della matrice di covarianza è stato usato il pacchetto R `fields` di Nychka et al. (2015). Il GP può essere estratto da una distribuzione gaussiana multivariata tramite, ad esempio, il pacchetto R `mvtnorm` di Genz et al. (2015). Per questioni di stabilità numerica, è stata utilizzata la scomposizione in valori singolari all'interno dell'algoritmo di generazione dei valori pseudocasuali. In Figura 2.1 sono stati rappresentate tre estrazioni provenienti da un processo gaussiano. Come si può notare le tre estrazioni sono molto "liscie" e ciò è indotto dalla funzione nucleo dell'equazione (2.3), che è differenziabile infinite volte.

2.1.1 Processi gaussiani condizionati

Si supponga di osservare una realizzazione f di un GP. Si è interessati a caratterizzare la distribuzione del processo f^* che abbia la medesima distribuzione, condizionatamente a f. Supponendo che f abbia media nulla, si ha che

$$f \sim GP(0, C(x, x')), \qquad f^* \sim GP\left(0, C(x, x')\right). \tag{2.5}$$

Siano $f^*(x^*)$ e $f(x)$ le variabili normali multivariate relative alle griglie di punti x e x^*, rispettivamente di n e n^* elementi. Sia $f^*(x^*) \mid f(x)$ la distribuzione condizionata, mentre sia $(f(x), f^*(x^*))$ la distribuzione congiunta. Si ottiene che

$$(f(x), f^*(x^*)) \sim N_{n+n^*}\left(0, \Sigma\right), \qquad \Sigma = \begin{pmatrix} C_{xx} & C_{xx^*} \\ C_{x^*x} & C_{x^*x^*} \end{pmatrix}, \tag{2.6}$$

in cui si ha che $C_{x^*x} = C_{xx^*}^T$. Grazie alle proprietà della distribuzione normale, la distribuzione condizionata è pari a

$$f^*(x^*) \mid f(x) \sim N_{n^*}\left(\tilde{\mu}, \tilde{\Sigma}\right), \tag{2.7}$$

dove

$$\tilde{\mu} = C_{x^*x} C_{xx}^{-1} f, \qquad \tilde{\Sigma} = C_{x^*x^*} - C_{x^*x} C_{xx}^{-1} C_{xx^*}. \tag{2.8}$$

Quando $x = x^*$ la distribuzione condizionata è una variabile casuale degenere valutata nei valori osservati f. Infatti, non sono presenti altre fonti di variabilità. Questo comporta una perfetta interpolazione delle osservazioni: la distribuzione diventa degenere nei punti osservati f.

2.2 Processo di Dirichlet (DP)

Un differente processo stocastico, il quale verrà utilizzato in seguito, è noto come processo di Dirichlet (*Dirichlet Process, DP*). Esso è un processo stocastico utilizzato prevalentemente in ambito bayesiano non parametrico. A livello intuitivo, esso è una distribuzione casuale: ciascuna estrazione da un DP è a sua volta una distribuzione. Inoltre, ciascuna estrazione da un DP è discreta, ma non può essere descritta da un numero finito di parametri, da cui la dicitura di modello non parametrico. Deve il suo nome al fatto che ogni distribuzione marginale con numero di componenti finiti è distribuita come una Dirichlet. Un riferimento per questo processo e le sue applicazioni nel campo bayesiano non parametrico è dato da Hjort et al. (2010). Il processo di Dirichlet venne introdotto in Ferguson (1973) e Ferguson (1974) nel modo seguente

Definizione 2 (Processo di Dirichlet) *Sia G_0 una distribuzione (detta di base) nello spazio Ω e α un valore reale positivo (detto parametro di concentrazione). Si dirà che $G \sim DP(\alpha, G_0)$ se per ogni partizione finita e misurabile $\mathcal{A}_1, \dots, \mathcal{A}_r$ di Ω si ha che*

$$(G(\mathcal{A}_1), \dots, G(\mathcal{A}_r)) \sim Dir(\alpha G_0(\mathcal{A}_1), \dots, \alpha G_0(\mathcal{A}_r)). \tag{2.9}$$

Si noti che la misura indotta da G è a sua volta casuale e questo giustifica l'interpretazione in termini di distribuzione casuale. Inoltre, sia $\mathcal{A}$ una partizione misurabile di Ω, allora

$$\mathbb{E}[G(\mathcal{A})] = G_0(\mathcal{A}), \quad \text{Var}(G(\mathcal{A})) = \frac{G_0(\mathcal{A})(1 - G_0(\mathcal{A}))}{1 + \alpha}. \tag{2.10}$$

La distribuzione di base G_0 e il parametro α sono quindi interpretabili: la prima rappresenta la media del DP, mentre α regola la concentrazione del processo intorno alla media stessa. Vale inoltre che $G(\mathcal{A}) \to G_0(\mathcal{A})$ per $\alpha \to \infty$. Si noti, tuttavia, che questo non implica che $G \to G_0$. Infatti, il DP è discreto anche nel caso in cui la distribuzione G_0 non lo sia.

Il processo di Dirichlet gode di una ulteriore proprietà. Siano $\theta_i \overset{i.i.d.}{\sim} G, i = 1, \dots, n$ estrazioni da un DP, allora si può dimostrare che

$$G \mid \theta_1, \dots, \theta_n \sim DP\left(\alpha + n, \frac{\alpha}{\alpha + n} G_0 + \frac{n}{\alpha + n} \frac{1}{n} \sum_{i=1}^{n} \delta_{\theta_i}\right), \tag{2.11}$$

dove δ_{θ_i} è un punto massa concentrato in θ_i. Il DP condizionato ai dati è pertanto nuovamente un DP con differenti parametri. Esso è maggiormente concentrato nella nuova distribuzione di base $\tilde{G}_0$, la quale è una mistura tra G_0 e i valori osservati. All'aumentare di n, il processo $G \mid \theta_1, \dots, \theta_n$ tende ad assumere la forma della distribuzione empirica.

2.2.1 Urne di Polya e proprietà di *clustering*

Il DP si presta a numerose interpretazioni, una delle quali è definita in Blackwell & MacQueen (1973). Sia θ_{n+1} la $n+1$-esima estrazione da $G \sim DP(\alpha, G_0)$. La distribuzione condizionata rispetto a tutti gli altri valori osservati è pari a

$$\theta_{n+1} \mid \theta_1, \dots, \theta_n \sim \frac{\alpha}{\alpha + n} G_0 + \frac{n}{\alpha + n} \sum_{i=1}^{n} \delta_{\theta_i}/n. \tag{2.12}$$

La distribuzione "previsiva" per θ_{n+1} pertanto coincide con la distribuzione di base $\tilde{G}_0$ del DP condizionato. La sequenza dei valori condizionati $\theta_1, \theta_2, \dots$ segue lo schema delle *urne di Polya*. L'interpretazione è la seguente: inizialmente, viene presa una biglia trasparente la quale viene dipinta di un colore ed inserita in un urna. Ai passi successivi, ad esempio l'$n+1$-esimo, si prende una nuova biglia trasparente. Con probabilità $\frac{\alpha}{\alpha+n}$, essa viene dipinta con un nuovo colore. Se ciò non accade, viene estratta una biglia precedentemente

inserita nell'urna e quella trasparente viene dipinta dello stesso colore di quella estratta; entrambe le biglie vengono infine inserite nell'urna.

Questo schema, nonostante non costituisca in sé una formale dimostrazione, suggerisce che il DP è intrinsecamente discreto, data la presenza di valori ripetuti. Siano $\theta_1^*, \dots, \theta_H^*$ i valori unici contenuti nell'insieme $\theta_1, \dots, \theta_n$ e n_h il numero di ripetizioni di θ_h^*, allora l'equazione (2.12) può essere riscritta come

$$\theta_{n+1} \mid \theta_1, \dots, \theta_n \sim \frac{\alpha}{\alpha + n} G_0 + \frac{n}{\alpha + n} \sum_{i=1}^{H} n_h \delta_{\theta_i^*} / n. \tag{2.13}$$

La presenza di valori ripetuti implica sostanzialmente una suddivisione dei valori θ in $k \leq n$ gruppi, i quali vengono determinati casualmente. Questo processo viene generalmente chiamato del "ristorante cinese" (*Chinese restaurant process, CRP*), in seguito alla metafora utilizzata da Pitman (1996). Nel CRP si suppone l'esistenza di un locale con un numero infinito di tavoli. Il primo cliente della giornata si siede nel primo tavolo libero. Il secondo, si siede in compagnia del primo oppure sceglie di sedersi in un tavolo diverso, con probabilità definita dalla (2.13). Tutti i clienti successivi possono scegliere, tramite un esperimento casuale a sua volta definito dalla (2.13), di sedersi in un tavolo già occupato oppure occuparne uno nuovo. Inoltre, quante più persone sono sedute al tavolo maggiore sarà la probabilità che il nuovo cliente vi prenda posto. Di conseguenza, quanto più un tavolo sarà affollato, tanto più tenderà a crescere. In questa metafora, ciascun tavolo rappresenta un *cluster* mentre ciascun cliente rappresenta un'osservazione.

Condizionatamente al numero di osservazioni n, il numero di *cluster* è una variabile casuale ben definita. Sia k la variabile casuale che indica il numero di cluster, allora direttamente dall'equazione (2.13) si ricava che

$$\mathbb{E}[k \mid n] = \sum_{i=1}^{n} \frac{\alpha}{\alpha + i - 1} = \alpha(\psi(\alpha + n) - \psi(\alpha)) = O\left(\alpha \log n\right), \tag{2.14}$$

in cui $\psi(\cdot)$ è la funzione digamma (Abramowitz & Stegun, 1972). Appare quindi chiaro che esiste una stretta relazione tra il numero medio di *cluster* ed il parametro di concentrazione α. Inoltre, il numero di *cluster* dipende dalla numerosità campionaria: maggiore differenziazione tra le osservazioni è possibile con l'aumentare dell'informazione.

2.2.2 Rappresentazione *stick-breaking* e modelli di mistura

Il processo di Dirichlet ha una rappresentazione alternativa, dovuta a Sethuraman (1994). Essa è decisamente più intuitiva rispetto a quella utilizzata finora, ma venne storicamente introdotta più tardi.

Definizione 3 (Processo di Dirichlet, stick-breaking) *Si definiscano le seguenti quantità:*

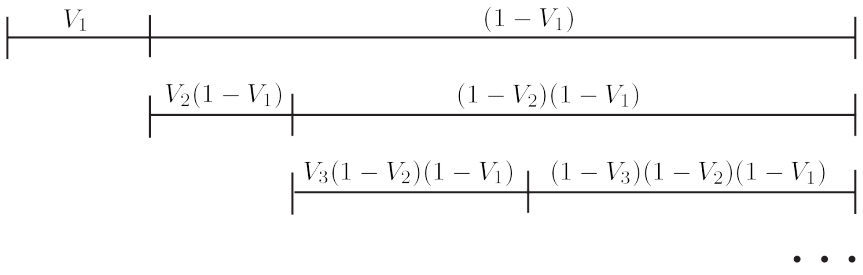
$$V_h \overset{i.i.d.}{\sim} Beta(1, \alpha), \qquad \theta_h \overset{i.i.d.}{\sim} G_0, \tag{2.15}$$

$$\pi_h = V_h \prod_{i=1}^{h-1}(1 - V_i), \qquad G = \sum_{h=1}^{\infty} \pi_h \delta_{\theta_h}, \tag{2.16}$$

allora si ha che $G \sim DP(\alpha, G_0)$.

Si faccia attenzione al leggero cambio di notazione rispetto alla (2.11). In questo caso, infatti, la generica variabile casuale θ_h indica un'estrazione proveniente dalla distribuzione di base G_0 mentre prima indicava un'estrazione da G. Il nome *stick-breaking* deriva dal fatto che la misura di probabilità casuale viene costruita iterativamente "spezzando casualmente" un ipotetico segmento unitario come riportato in Figura 2.2. La misura casuale $\pi = (\pi_1, \pi_2, \dots)$ è spesso indicata con la dicitura $\pi \sim \text{GEM}(\alpha)$, da Griffiths, Engen e McCloskey.

Figura 2.2: Rappresentazione grafica dei primi tre *step* del processo *stick-breaking*. Le lettere indicate rappresentano le probabilità corrispondenti, in cui $V_1, V_2, V_3 \sim \text{Beta}(1, \alpha)$.



La rappresentazione (2.15)-(2.16) permette quindi di interpretare il DP come una mistura infinita di variabili casuali provenienti da G_0. Questa scrittura ha senso se, come accade, i pesi della mistura convergono rapidamente a 0. In particolare, vale che

$$\sum_{h=1}^{\infty} \pi_h = 1.$$

Come notarono Muliere & Tardella (1998), è possibile quindi approssimare il DP utilizzando solamente i primi H pesi della mistura, imponendo $V_H = 1$. Si avrà quindi che

$$G = \sum_{h=1}^{H} \pi_h \delta_{\theta_h}, \qquad \sum_{h=1}^{H} \pi_h = 1. \tag{2.17}$$

Questa rappresentazione è, tuttavia, estremamente più trattabile, poiché coinvolge un numero finito di termini. Allo stesso tempo, il troncamento ad H termini non riconduce il processo ad una mistura tra H distribuzioni: esso rappresenta piuttosto il limite superiore di componenti utilizzate.

2.3 Applicazioni del DP alla statistica bayesiana

Il DP è stato ampiamente usato in campo bayesiano: esso viene infatti scelto come distribuzione a priori in un'impostazione non parametrica, ad esempio nella modellazione di dati gerarchici (Gelman & Hill, 2006), per la stima di funzioni densità o per dati funzionali. I modelli parametrici possono infatti soffrire di sovradattamento o sottodattamento a causa della discrepanza che c'è tra modello ipotizzato e quello "vero". Di conseguenza, l'identificazione della corretta specificazione richiede una certa attenzione, come fatto ad esempio nel Capitolo 1. Inoltre, la scelta di una specifica forma funzionale per la distribuzione a priori è spesso legata alla semplicità computazionale piuttosto che a reali convinzioni a priori. L'approccio bayesiano non parametrico indotto dal DP porta con sé tre grandi vantaggi:

1. Supporre che una determinata quantità si distribuisca a priori come un DP significa ipotizzare a priori una ignota distribuzione, all'interno di un ampia classe. Qualora non siano disponibili ulteriori informazioni a priori, ciò risulta sicuramente un approccio più coerente rispetto all'utilizzo di una forma funzionale ben precisa.

2. La complessità del modello è parte integrante del processo inferenziale. Il numero di *cluster* infatti regola la flessibilità del modello stesso che, a sua volta, è controllata dal parametro α.

3. Il *clustering* di cui al punto precedente è di per sé un effetto collaterale, ma spesso risulta essere il vero oggetto di interesse.

Un esempio di applicazione è illustrato in Dunson (2009), in cui il DP è utilizzato come distribuzione a priori degli effetti casuali in presenza di dati gerarchici. Nella formulazione più semplice esso prevede che

$$y_{ij} = z_{ij}^T \beta_i + \varepsilon_{ij}, \qquad \varepsilon_{ij} \sim N\left(0, \sigma^2\right), \tag{2.18}$$

$$\beta_i \sim DP(\alpha, G_0), \qquad G_0 \sim N\left(\tilde{\mu}, \tilde{\Sigma}\right), \tag{2.19}$$

in cui y_{ij} è la j-esima osservazione dell'i-esimo soggetto, z_{ij} è un vettore di p costanti note e $\beta_i = (\beta_{i1}, \ldots, \beta_{ip})$. Si tratta pertanto di un modello di regressione gaussiano con effetti casuali. La distribuzione di base G_0 ha una sua interpretazione: rappresenta il migliore tentativo di specificazione per la distribuzione dell'effetto casuale. Infatti per $\alpha \to \infty$ gli effetti casuali convergono puntualmente alla distribuzione gaussiana. Una specificazione parametrica prevederebbe ad esempio $\beta_i \sim N(\tilde{\mu}, \tilde{\Sigma})$ che risulta essere, all'atto pratico, un caso particolare della specificazione (2.19). Le distribuzioni a priori per i restanti parametri non sono qui di interesse e quindi non discusse.

Capitolo 3

Analisi funzionale

3.1 Dati funzionali

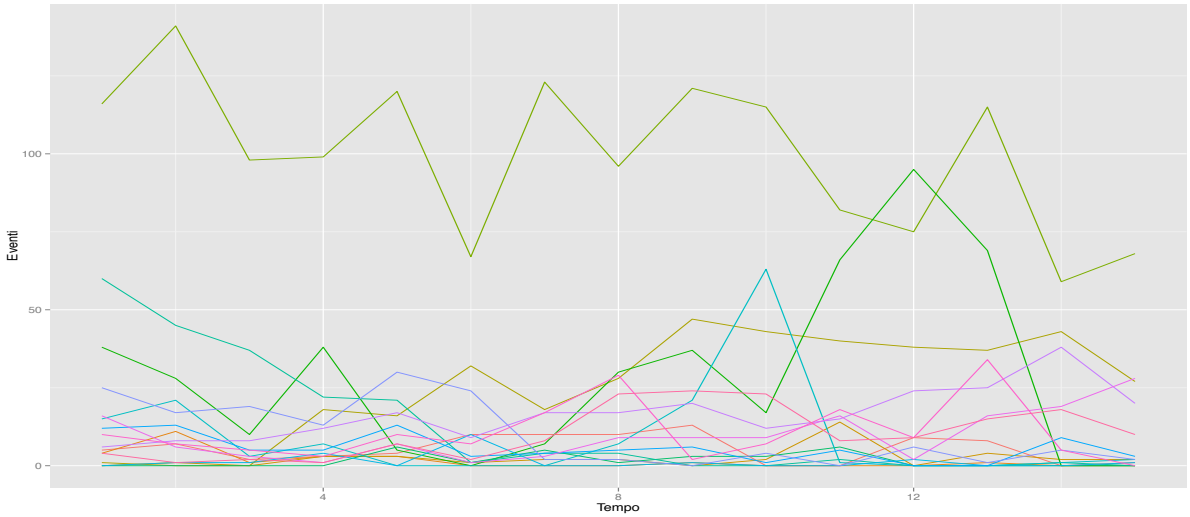
In alcune circostanza, come nel caso dei dati di telefonia mobile, sono disponibili misurazioni ripetute dello stesso processo aleatorio. La variabile casuale oggetto di analisi giace in uno spazio funzionale, di cui si osserva una porzione finita, definita dagli istanti di tempo $t_1, \dots, t_T$. Supponendo che gli intervalli siano equispaziati, ciascuna estrazione proviene quindi da un processo del tipo

$$x_i(t) = f_i(t) + \varepsilon_i, \qquad t = 1, \dots, T, \quad i = 1, \dots, n. \tag{3.1}$$

in cui ε_i è un errore casuale. In Figura 3.1 sono riportate 10 realizzazioni, estratte dal dataset di telefonia mobile analizzato finora. Si tenga presente, comunque, che ciascuna osservazione è costituita da una serie storica mensile: non è una valutazione istantanea di un processo continuo, come nell'equazione (3.1), ma un valore aggregato rispetto all'intero mese. Tuttavia, a livello operativo, ciascuna osservazione è trattata *come se* provenisse effettivamente da un processo del tipo (3.1). Come si può notare, il comportamento appare piuttosto irregolare e difficilmente interpretabile. Se fosse stato rappresentato l'intero dataset, ovvero circa 25.000 osservazioni, esso sarebbe stato sostanzialmente illeggibile. Per questo motivo sono necessarie analisi più sofisticate, in grado di gestire e riassumere la grande mole di dati. Come nei problemi di regressione presentati nel capitolo precedente, è di interesse ottenere una stima sufficientemente accurata per ciascuna funzione $f_i(t)$. Nel campo dell'analisi funzionale, tuttavia, ricostruire la relazione tra diversi processi è altrettanto importante. Tre grandi classi di problemi possono essere posti

1. **Clustering funzionale**. Lo scopo è raggruppare in maniera quanto più omogenea possibile ciascuna funzione. Ciò può avvenire seguendo differenti metodologie e nell'arco del Capitolo ne verranno presentate almeno un paio.

Figura 3.1: Dati di telefonia mobile. Numero di telefonata mensili per 10 clienti estratti casualmente. In questo caso $n = 10, T = 15$.



2. **Regressione e classificazione con predittore funzionale**. Lo scopo è ottenere una previsione per una variabile esplicativa scalare y_i, utilizzando come predittore un'intera funzione.

3. **Regressione con variabile risposta funzionale**. Lo scopo è prevedere l'andamento di una intera funzione, utilizzando variabili esplicative scalari.

Riferimenti recenti per analisi funzionale dal punto di vista frequentista sono Ramsay & Silverman (2005) per una trattazione teorica, mentre in Ramsay et al. (2009) maggior enfasi è data all'implementazione software.

3.2 Approccio classico

3.2.1 Dai dati grezzi a funzioni "lisciate"

Gestire direttamente i dati grezzi è di scarso interesse, poiché essi sono "sporcati" da un errore casuale. Un approccio comune consiste nel considerare, invece, una trasformazione più regolare, ottenuta ad esempio con i metodi non parametrici presentati nel Capitolo precedente. Anche grazie alla loro efficienza computazionale, i metodi basati sulla *basis expansions* sono molto popolari. Seguendo la definizione data in Ramsay & Silverman (2005), si ha che

Definizione 4 (*Basis expansion*) *Un sistema di funzioni di base è un insieme di funzioni note $\phi(t)$ linearmente indipendenti, le quali, tramite combinazione lineare composta*

da un numero di termini M sufficientemente elevato, possono approssimare arbitrariamente bene qualsiasi funzione.

Pertanto è possibile approssimare una generica funzione tramite la seguente combinazione lineare:

$$f(t) = \sum_{m=1}^{M} \beta_m \phi_m(t), \tag{3.2}$$

dove $\phi_m(t)$ rappresenta l'm-esimo elemento della base di funzioni, analogamente a quanto illustrato nell'equazione (1.8). In questo caso, però, la base di funzioni non dipende da un ulteriore insieme di parametri. La base polinomiale è probabilmente la più nota, i cui elementi sono

$$\left\{1, t, t^2, t^3, \dots\right\}. \tag{3.3}$$

Essa sicuramente rappresenta un buon punto di partenza, ma spesso risulta poco efficiente in termini inferenziali rispetto ad altre base di funzioni. Altre basi di funzioni note in letteratura sono, ad esempio, le serie di Fourier, le Wavelets, o le basi indotte dalle splines. Una splines di ordine R con J nodi è esprimibile come

$$f(t) = \sum_{i=0}^{R-1} \theta_i t^i + \sum_{j=1}^{J} \delta_j \left(t - \xi_j\right)_+^{R-1}, \tag{3.4}$$

in cui $\xi = (\xi_1, \dots, \xi_J)$ è un insieme di nodi. A seconda dei contesti, questi noti vengono trattati come costanti note o parametri da stimare. Ad esempio, nei MARS presentati nel Capitolo 1 essi sono stati stimati. Questa struttura implica che:

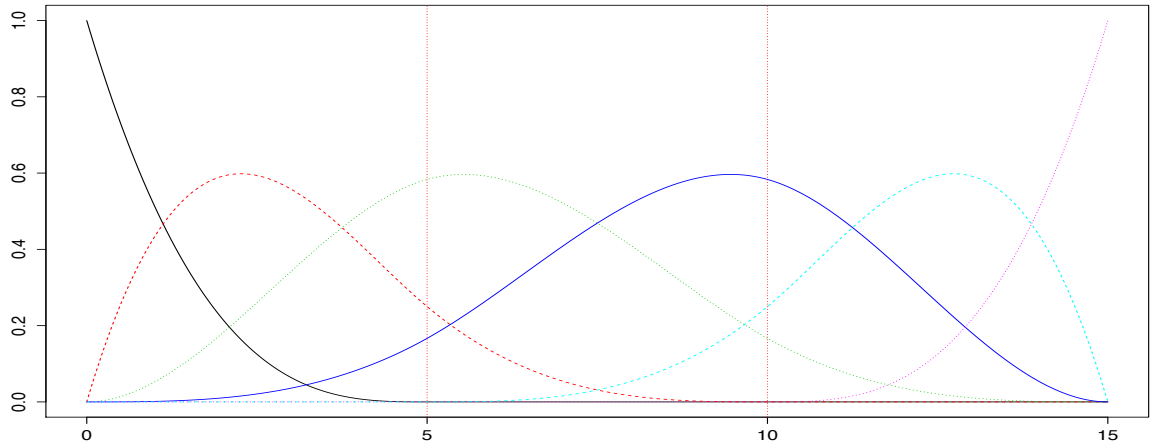
1. $f(t)$ è polinomiale a tratti di ordine $R-1$ in ciascun intervallo $[\xi_i, \xi_{i+1})$.
2. $f(t)$ possiede $R-2$ derivate continue.

ed induce una base di funzioni chiamata *truncated power basis*. I nodi vengono generalmente scelti equispaziati rispetto agli estremi dell'insieme finito che si sta valutando. Tipicamente viene scelto $R = 4$ che genera le cosiddette splines cubiche. Il numero di nodi J invece va scelto con una certa attenzione, al fine di evitare sovradattamento. Una formulazione completamente equivalente, ovvero una trasformazione lineare della base appena descritta è costituita dalla cosiddetta base di funzioni *B-splines*. I vettori che la costituiscono sono meno correlati della *truncated power basis* e pertanto risultano numericamente più stabili in contesti di regressione. In Figura 3.2 viene rappresentata una base di funzioni *B-splines* con $R = 4$ e $J = 2$.

3.2.2 Stima ai minimi quadrati

Si richiami ora l'obiettivo iniziale, ovvero ricostruire n funzioni lisciate, eliminando il rumore. Ciascuna realizzazione del processo viene approssimata con una base di funzioni

Figura 3.2: Base di funzioni *B-splines* con con $R = 4$ e $J = 2$, nell'intervallo $[0, 15]$



prestabilita, ad esempio una base di funzioni *B-splines*. Supposto che i nodi siano trattati come costanti note, l'insieme di coefficienti $\beta_i, i = 1, \ldots, n$, è ignoto e può essere stimato utilizzando il criterio dei minimi quadrati. Ciascun insieme di coefficienti è quindi stimato separatamente per ciascuna realizzazione del processo stocastico. Sia $\Phi = (\phi_1, \ldots, \phi_M)$ la matrice contenente le funzioni di base, allora la stima ai minimi quadrati per β_i è pari a

$$\hat{\beta}_i = \left(\Phi^T \Phi\right)^{-1} \Phi^T x_i, \tag{3.5}$$

in cui x_i è il vettore delle osservazioni per la i-esima realizzazione del processo. Allora, l'i-esimo vettore delle osservazioni lisciate è pari a

$$\hat{x}_i = \Phi \hat{\beta}_i = \Phi \left(\Phi^T \Phi\right)^{-1} \Phi^T x_i. \tag{3.6}$$

Un metodo di stima leggermente differente che verrà successivamente utilizzato, prevede invece che la stima per i coefficienti β_i sia pari a

$$\hat{\beta}_i = \left(\Phi^T \Phi + \lambda R_\phi\right)^{-1} \Phi^T x_i, \tag{3.7}$$

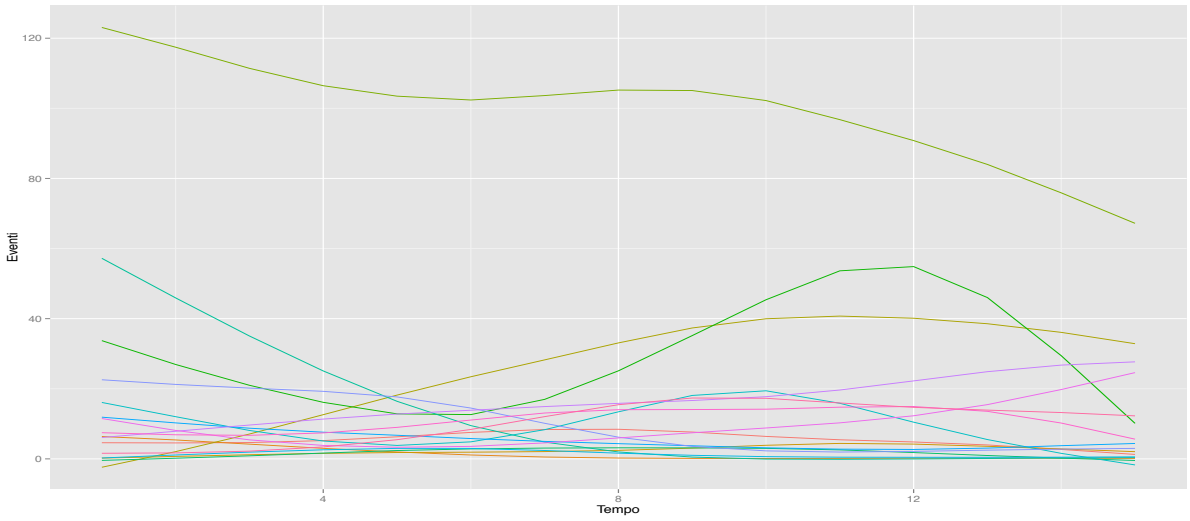
in cui λ è un parametro di lisciamento mentre R_ϕ è una matrice che generalmente coinvolge la derivata seconda della base di funzioni. Questo approccio pertanto è simile alle *splines di lisciamento* (Azzalini & Scarpa, 2011), in cui però il numero di nodi è inferiore al numero di osservazioni. Il numero M di funzioni di base coinvolte congiuntamente al parametro di lisciamento λ viene infine selezionato tramite convalida incrociata generalizzata. Si noti, comunque, che il parametro di lisciamento λ è unico e non varia da un processo all'altro.

3.3 *Clustering funzionale*: i dati di telefonia mobile

Viene qui ripreso il *dataset* di telefonia mobile. Sono state prese in considerazione solamente $n = 2000$ osservazioni, estratte casualmente. Ciò è stato necessario per evitare un carico computazionale eccessivo: la perdita di informazione è quindi evidente ma purtroppo inevitabile. Estendere l'analisi all'intero *dataset* non comporta nessun sforzo intellettuale aggiuntivo, ma richiede tempo e risorse informatiche maggiori.

In primo luogo, è stata identificata una base di funzioni di tipo *B-splines*, considerata "ottimale" secondo il criterio della convalida incrociata generalizzata. Le funzioni sono state stimate seguendo l'equazione (3.7). I valori ottimali per M, il numero di elementi della base di funzioni, e per λ, il parametro di lisciamento, sono stati ottenuti tramite il comando `min.basis` del pacchetto R `fda.usc` di Febrero-Bande & Oviedo de la Fuente (2012). Gli stessi dati della Figura 3.1 sono quindi riproposti in forma lisciata in Figura 3.3.

Figura 3.3: Dati di telefonia mobile. Numero di telefonata mensili per 10 clienti estratti casualmente. Le osservazioni sono state lisciate con tramite una base di funzioni *B-splines*, in cui $M = 14$ e $\lambda = 5.86$.



A fini esplorativi, è utile tentare di formare dei gruppi (*cluster*), di tipo funzionale, i quali siano quanto più omogenei al loro interno e quanto più eterogenei tra loro. Questo infatti permette di interpretare meglio i dati, individuando differenti tipologie di comportamento.

Per una recente rassegna dei metodi di *clustering funzionale* si può fare riferimento a Jacques & Preda (2014). Gli autori identificano almeno tre categorie di possibili approcci, il primo dei quali consiste nell'applicare le usuali tecniche di *clustering* ai dati grezzi,

ovvero quelli rappresentati in Figura 3.1. In questo caso si può fare nuovamente riferimento ad Azzalini & Scarpa (2011). Un secondo approccio consiste nello specificare un modello probabilistico tale per cui una forma di raggruppamento sia intrinseca nel modello stesso: un esempio verrà presentato nel Capitolo 4.

In alternativa, il processo di *clustering* può avvenire in due stadi: inizialmente le osservazioni vengono lisciate tramite, ad esempio, una base di funzioni e solo successivamente raggruppate utilizzando le tradizionali tecniche. Per fare ciò, è necessario quantificare la diversità tra due differenti curve. Una scelta frequente è definire la *dissimilarità* tra due funzioni f ed f_* pari a

$$d_l(f, f_*) = \left(\int_0^T \left(f^{(l)}(t) - f_*^{(l)}(t) \right)^2 dt \right)^{1/2}, \tag{3.8}$$

dove $f^{(l)}(t) = \frac{\partial^l}{\partial t^l} f(t)$ indica la derivata l-esima, mentre l è un parametro tipicamente posto pari a 0. In questo caso specifico, si è scelto, anche per motivi computazionali, di approssimare il precedente integrale tramite la seguente quantità

$$d_l(f, f_*) = \left(\sum_{t=1}^{T} \left(f^{(l)}(t) - f_*^{(l)}(t) \right)^2 \right)^{1/2}, \tag{3.9}$$

che pertanto coincide con la distanza euclidea tra i valori lisciati dei punti osservati $t = 1, \ldots, T$. Altre misure di *dissimilarità* possono essere definite a partire dai coefficienti stimati tramite la (3.7), qualora il lisciamento avvenga tramite base di funzioni.

Algoritmo *PAM (Partitioning Around Medoids)*

E' stato utilizzato il pacchetto R `cluster` di Maechler et al. (2015) ed in particolare la funzione `pam`, che implementa un algoritmo di clustering denominato, appunto, *PAM* (Kaufman & Rousseeuw, 1987). Si tratta di un algoritmo di tipo non gerarchico, simile al noto algoritmo *kmeans*, che suddivide i dati in K gruppi. Il numero di gruppi deve essere specificato a priori. Esso si basa sulla ricerca di $(m_1, \ldots, m_K)$ osservazioni rappresentative, chiamate medoidi (*medoids*). Tutte le altre osservazioni vengono quindi assegnate al medoide più vicino, sulla base di una generica matrice di *dissimilarità*. L'algoritmo *kmeans* è simile: le osservazioni sono infatti assegnate al *centroide* più vicino, che è un punto nello spazio e non necessariamente corrispondente ad un'osservazione. Si vuole quindi minimizzare la seguente quantità

$$\sum_{i=1}^{n} \min_{k=1,\ldots,K} d(f_i, m_k). \tag{3.10}$$

Quindi, la generica funzione f_i è assegnata al medoide m_k se

$$d(f_i, m_k) \leq d(f_i, m_{k^*}), \quad \forall k \neq k^*. \tag{3.11}$$

L'identificazione dei medoidi è ottenuta ricorsivamente e la procedura è descritta dettagliatamente nell'Algoritmo 2. Rispetto all'algoritmo *kmeans* i vantaggi sono i seguenti:

Algorithm 2 *Partitioning Around Medoids*

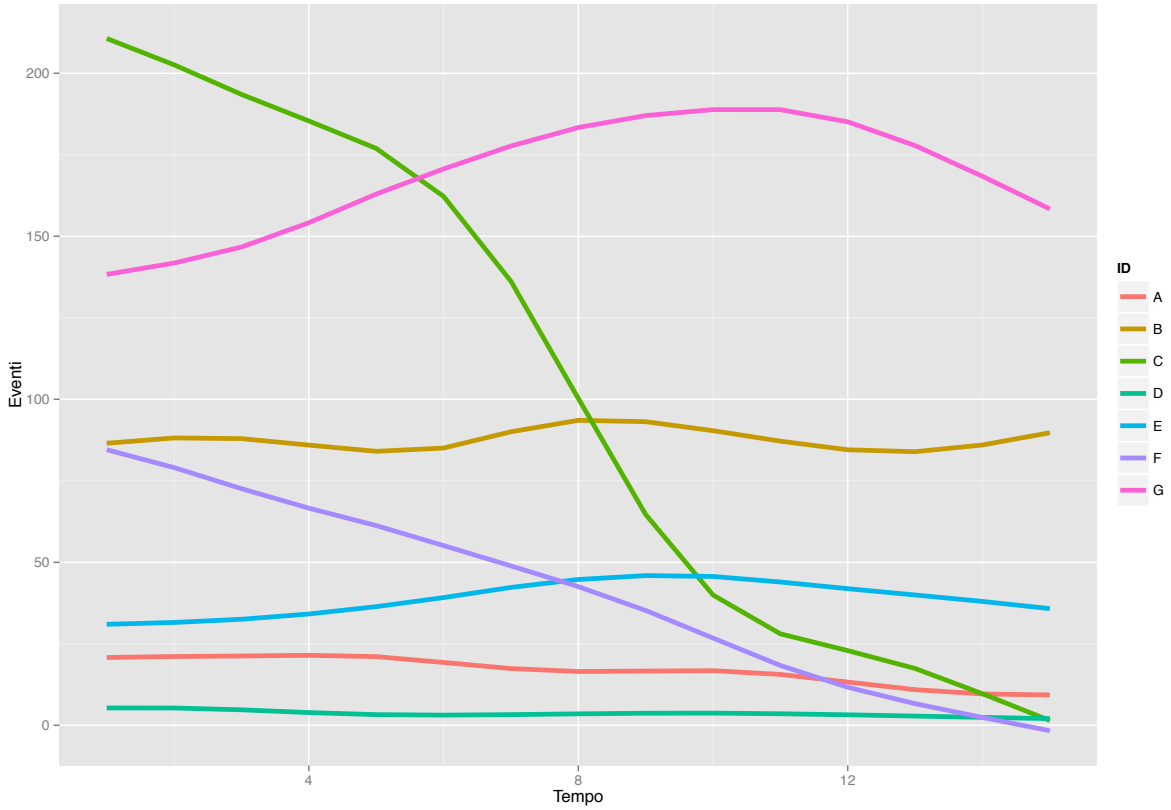
1. **Costruzione dei medoidi iniziali**
   (a) m_1 è l'osservazione in cui $\sum_{i=1}^{n} d(f_i, m_1)$ è più piccola possibile.
   (b) m_2 è l'osservazione che diminuisce la funzione obiettivo (3.10) il più possibile, stante m_1.
   (c) ...
   (d) m_K è l'osservazione che diminuisce la funzione obiettivo (3.10) il più possibile, stanti $m_1, \ldots, m_{K-1}$.
2. **Allineamento dei medoidi**. Si ripeta fino a convergenza.
   (a) Si considerino tutte le possibili coppie di elementi (i, j), tali che
   $$i \in \{m_1, \ldots, m_K\}, \quad j \notin \{m_1, \ldots, m_K\}$$
   (b) Si scambino gli indici i, j delle due osservazioni che diminuiscono maggiormente la (3.10), se queste esistono.

1. L'algoritmo *PAM* non necessita di un punto iniziale e pertanto è garantita l'unicità del risultato.
2. L'algoritmo *PAM* è più robusto rispetto ad osservazioni anomale in quanto minimizza la somma delle distanze e non la somma delle distanze elevata al quadrato.
3. I *medoidi* ottenuti sono più facilmente interpretabili nel caso di osservazioni funzionali: i *centroidi* che si otterrebbero con l'algoritmo *kmeans* perdono la struttura lisciata delle osservazioni originarie.

Per i dati in questione, sono stati ottenuti $K = 7$ *cluster* e i corrispondenti *medoidi*, i quali sono stati rappresentati in Figura 3.4. Il numero K di *cluster* è stato scelto sia sulla base di informazioni soggettive che tramite alcune operazioni preliminari basate su algoritmi di clustering di tipo gerarchico. Il numero di *cluster* è stato determinato in maniera tale che le funzioni fossero sufficientemente eterogenee tra loro ma contemporaneamente in numero ridotto, così da facilitarne l'interpretazione. La suddivisione agisce in primo luogo sui livelli medi, identificando i clienti con diverse quote di traffico mensile. Tuttavia

i medoidi C ed F, identificano due gruppi caratterizzati da un *trend* decrescente, di diversa pendenza. Questa caratteristica non sarebbe stata individuata se non si fosse tenuto conto della longitudinalità dei dati.

Figura 3.4: *Medoidi* ottenuti tramite l'algoritmo *PAM*. In legenda è riportato il codice identificativo della *SIM card* di ciascun *medoide*.



Capitolo 4

Modello funzionale bayesiano non parametrico

4.1 Regressione tramite processi gaussiani

4.1.1 Inferenza bayesiana con parametri di disturbo noti

In questo paragrafo viene discusso il problema di regressione tramite GP da un punto di vista bayesiano. Sebbene ciò possa sembrare un'eccessiva complicazione, questo modello è una sorta di "mattoncino" del più complesso approccio bayesiano non parametrico sviluppato in seguito. Si supponga di osservare un campione y composto da n osservazioni tale che

$$y = f(x) + \varepsilon, \qquad \varepsilon \sim N(0, \sigma^2 I_n). \tag{4.1}$$

Si supponga, per semplicità, che il parametro σ^2 sia noto. Il modello, e la sua verosimiglianza, viene quindi espresso condizionatamente alla funzione f. Si ha che

$$y \mid f(x) \sim N\left(f(x), \sigma^2 I_n\right), \qquad L(y \mid f(x)) \propto \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - f(x_i))^2\right\}. \tag{4.2}$$

Utilizzando un approccio bayesiano, si utilizza una *distribuzione a priori funzionale* per f, ovvero si suppone $f \sim GP\left(0, C(x, x')\right)$. La distribuzione a posteriori è quindi proporzionale a

$$p(f^* \mid y) \propto L(y \mid f)p(f), \tag{4.3}$$

ma può essere ottenuta con calcoli analoghi a quelli fatti nel Capitolo 2. Restringendosi ad una griglia di punti x^* si ha quindi che

$$f^*(x^*) \mid y \sim N_{n^*}\left(\tilde{\mu}, \tilde{\Sigma}\right),$$

in cui

$$\tilde{\mu} = C_{x^*x}(C_{xx} + \sigma^2 I_n)^{-1} y, \qquad \tilde{\Sigma} = C_{x^*x^*} - C_{x^*,x}(C_{xx} + \sigma^2 I_n)^{-1} C_{xx^*}. \tag{4.4}$$

Si tratta pertanto dell'equazione (2.8) leggermente modificata, ma con una sostanziale differenza. Nel primo caso, si tratta di una interpolazione dei valori osservati, mentre nel secondo caso viene condotta una effettiva regressione. In Figura 4.1(b) viene evidenziata questa differenza utilizzando lo stesso insieme di dati.

Aspetti computazionali: scomposizione di Cholesky

Al fine di ottenere, ad esempio, la media a posteriori dell'equazione (4.4), è necessario invertire una matrice di dimensione $n \times n$, operazione che può risultare non banale quando il numero di osservazioni n è elevato. Seguendo le indicazioni di Rasmussen & Williams (2006, Pag. 19), tutte le quantità di interesse possono essere ottenute efficientemente seguendo l'Algoritmo 3. Ciò è possibile poiché la matrice di covarianza è definita positiva.

Algorithm 3 Scomposizione di Cholesky per la stima di un GP

1. Si assegni $L = \text{Cholesky}(C_{xx} + \sigma^2 I_n))$
2. Si risolva il sistema $Lz = y$ e si salvi il vettore z, sfruttando la triangolarità di L.
3. Si risolva il sistema $L^T \alpha = z$ e si salvi α, sfruttando la triangolarità di L^T.
4. Si ottenga $\tilde{\mu} = C_{x^*x}\alpha$.
5. Si risolva il sistema $Lv = C_{x^*x}$ e si salvi v.
6. Si ottenga $\tilde{\Sigma} = C_{x^*x^*} - v^T v$.

Differenti *funzioni nucleo*

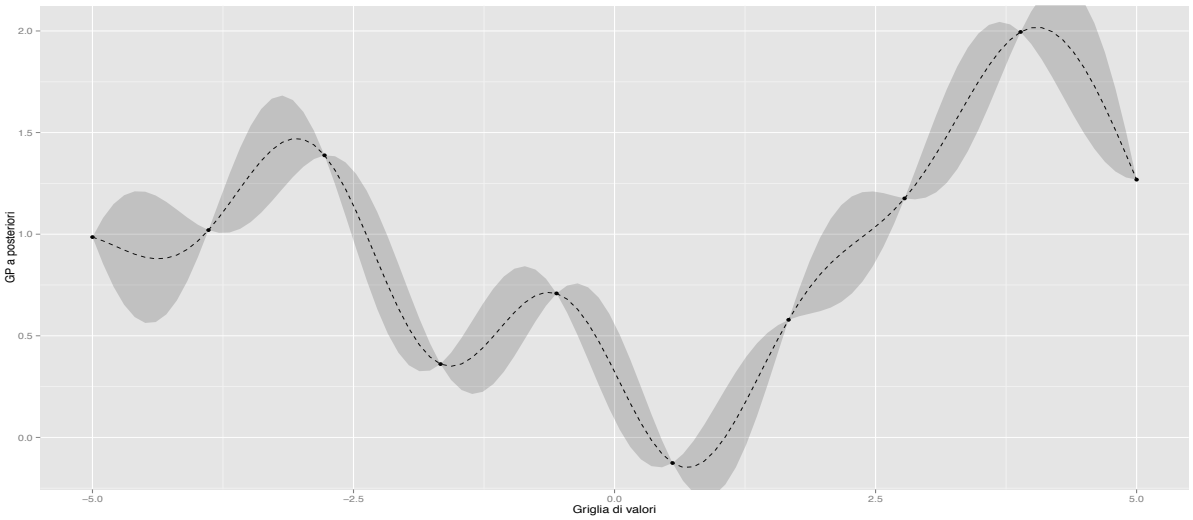
Per alcune particolari scelte della matrice di covarianza $C(x, x)$ un GP può essere ricondotto a modelli noti in letteratura. Si supponga, ad esempio

$$y = f(x) + \varepsilon, \quad f(x) = \Phi\beta, \tag{4.5}$$

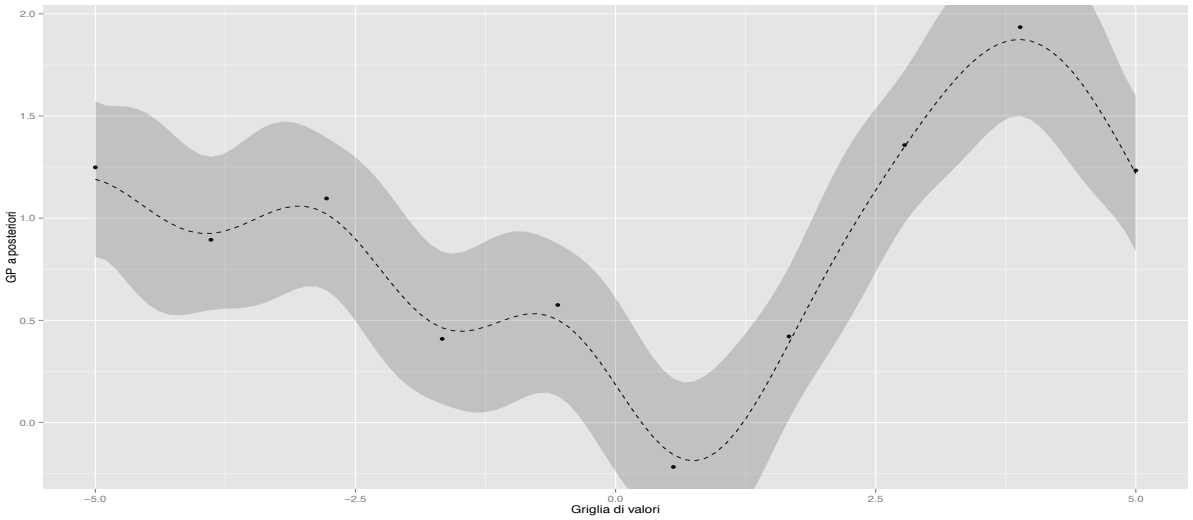
in cui $\beta \sim N_p(0, \Sigma_p)$. Allora la distribuzione a priori per $f(x)$ è

$$f(x) \sim N_n\left(0, \Phi\Sigma_p\Phi^T\right). \tag{4.6}$$

Quindi, imponendo $C(x, x) = \Phi(x)\Sigma_p\Phi(x)^T$, si può riconoscere la struttura di un GP, indotto dalle funzioni di base $\Phi(x)$. Questa classe di *funzioni nucleo* include i modelli



(a) Distribuzione a posteriori di un GP con osservazioni prive di errore aggiuntivo. La distribuzione a posteriori presenta variabilità solamente nei punti x^*. Nei punti iniziali x, invece, avviene una perfetta interpolazione.



(b) Distribuzione a posteriori di un GP in presenza di errore aggiuntivo. La linea si avvicina alle osservazioni ma non le interpola.

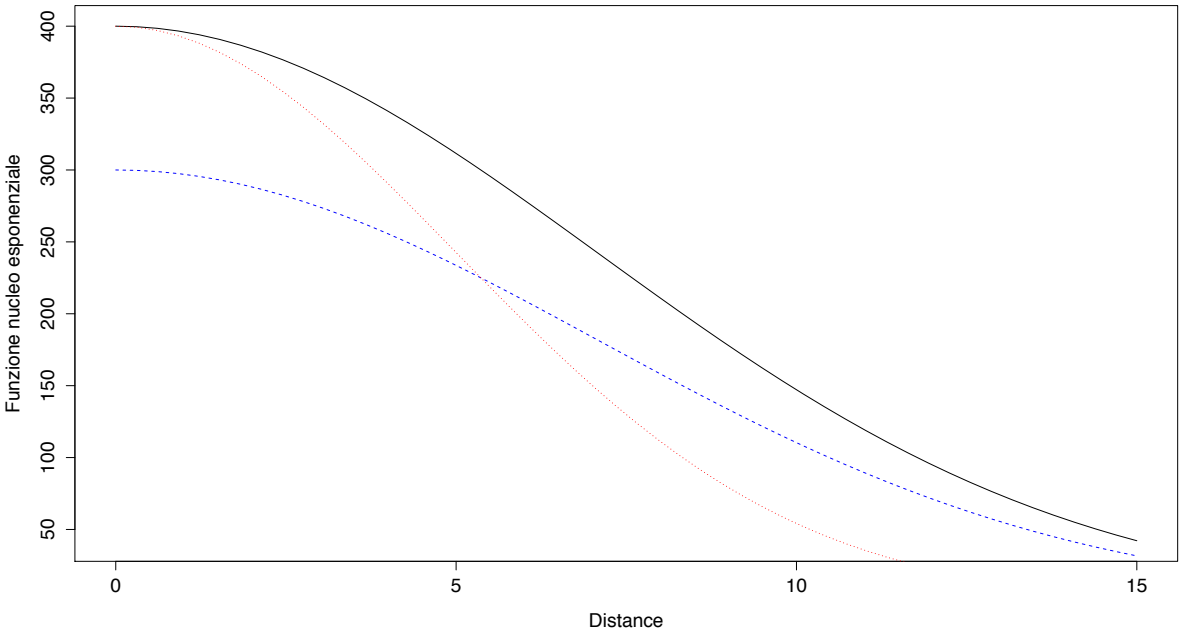
Figura 4.1: Confronto tra stima di un GP in assenza ed in presenza di un errore, serie simulata in cui si suppone che σ^2 sia un parametro noto.

lineari, regressioni polinomiali e le splines di regressione, che rappresentano quindi un caso particolare dei processi gaussiani. Una differente possibilità è data dalla seguente *funzione nucleo*, che favorisce cambiamenti repentini e meno "lisci":

$$C(x_i, x_j; \kappa) = \kappa_2 \exp\{-\kappa_1 |x_i - x_j|\} + \kappa_3 \delta_{ij}, \qquad \forall i, j, \tag{4.7}$$

in cui, come nella (2.4), la quantità δ_{ij} è una funzione indicatrice. Queste due *funzioni nucleo* godono della proprietà di *stazionarietà*, nel senso che dipendono unicamente dalla distanza $x_i - x_j$ e non dagli specifici valori che x_i, x_j assumono. Pertanto, possono essere rappresentate facilmente in un grafico bidimensionale. In Figura 4.2 è rappresentata la funzione covarianza (2.4) al variare di alcuni parametri.

Figura 4.2: Funzione nucleo esponenziale nell'intervallo $(0, 15)$ in cui i parametri κ sono pari a $\kappa = (0.01, 400)$, linea nera; $\kappa = (0.01, 300)$, linea blu; $\kappa = (0.02, 400)$ linea rossa. Il parametro κ_2 determina la variabilità complessiva, ma non modifica la forma della funzione, trattandosi di una costante moltiplicativa. Il parametro κ_1 invece determina quanto velocemente le osservazioni diventano incorrelate tra loro.



4.1.2 Inferenza in GP con parametri di disturbo

La varianza σ^2 ed il vettore di parametri κ presente nella matrice di covarianza, non possono essere trattati come costanti note nelle applicazioni reali. Almeno due strade sono percorribili: la prima rimpiazza ciascun parametro con una sia stima. Questa procedura

inferenziale è detta bayesiana empirica: la distribuzione a priori dipende, infatti, dai dati stessi. Nonostante ciò appaia in netta contraddizione con il paradigma bayesiano da un punto di vista "filosofico", esistono molti punti di contatto. Si veda, ad esempio, Petrone et al. (2014). Le stime sono generalmente ottenute tramite verosimiglianza marginale.

Nel secondo caso, invece, viene seguito un approccio completamente bayesiano e a ciascun parametro verrà associata una distribuzione a priori.

Verosimiglianza marginale

Un metodo per ottenere una stima per i parametri di interesse è tramite verosimiglianza marginale. La densità marginale di y è definita come

$$p(y \mid x) = \int p(y \mid f, x)p(f \mid x)df. \tag{4.8}$$

E' possibile provare che $p(y \mid x)$ è una normale con $y \sim N(0, C + \sigma^2 I_n)$. Pertanto, si può massimizzare equivalentemente la funzione di logverosimiglianza marginale associata al fine di ottenere una stima per il vettore (σ^2, κ). La funzione obiettivo è pertanto pari a

$$\log p(y \mid x) = -\frac{1}{2} y^T \left(C(\kappa) + \sigma^2 I_n\right)^{-1} y - \frac{1}{2} \log |C(\kappa) + \sigma^2 I_n|. \tag{4.9}$$

Per la massimizzazione di questa quantità è possibile ricorrere a algoritmi di massimizzazione numerica, ad esempio la funzione `nlminb` presente in R. Si noti che tale funzione presenta spesso dei punti di massimo locale. E' buona norma, pertanto, inizializzare la massimizzazione utilizzando vari e differenti punti di partenza.

Gibbs sampling

La distribuzione a posteriori dei parametri di disturbo è difficile da ricavare analiticamente. Per evitare il calcolo della costante di normalizzazione, nelle ultime decadi i metodi *Markov Chains Monte Carlo* (MCMC), sono diventati sempre più popolari. Per una trattazione generale delle tecniche MCMC, come l'algoritmo Metropolis-Hastings ed il Gibbs sampling si faccia riferimento a Casella & Robert (2010) oppure a Albert (2008). Qui di seguito verrà presentato un Gibbs sampling in cui, ove necessario, viene annidato un passo Metropolis. Si tratta pertanto di un algoritmo ibrido tra i due sopra citati. Si supponga che la distribuzione a priori per σ^2 sia tale che

$$\sigma^{-2} \sim Ga(a, b), \tag{4.10}$$

allora si ottiene una forma coniugata per la cosiddetta distribuzione *full conditional*, ovvero la distribuzione per il parametro di interesse condizionata a tutti gli altri e ai dati. Essa è necessaria per il *Gibbs sampling*. Si avrà che

$$\sigma^{-2} \mid - \sim Ga\left(a + \frac{n}{2}, b + \frac{1}{2}||y - f||^2\right),$$

in cui la notazione $\sigma^{-2} \mid -$ indica la distribuzione di σ^{-2} condizionatamente a tutte le altre quantità.

Per quel che riguarda invece gli eventuali parametri presenti nella matrice $C(x, x; \kappa)$, indipendentemente dalla distribuzione a priori $p(\kappa)$ specificata, non è possibile trovare una forma coniugata per la distribuzione *full conditional*, la quale risulta proporzionale a

$$p(\kappa \mid -) \propto p(\kappa)|C(\kappa)|^{-1/2} \exp\left\{-\frac{1}{2} f^T C(\kappa)^{-1} f\right\}. \tag{4.11}$$

Prendendo il logaritmo si ottiene:

$$\log p(\kappa \mid -) = c - \frac{1}{2}\log|C(\kappa)| - \frac{1}{2} f^T C(\kappa)^{-1} f + \log p(\kappa). \tag{4.12}$$

Sono possibili varie vie: in primo luogo, si può inserire un passo Metropolis: questa scelta non richiede particolari sforzi analitici, ma deve essere effettuata un'attenta calibrazione. Una seconda scelta consiste nel costruire un algoritmo di valori pseudocasuali, ad esempio il cosiddetto *accettazione rifiuto*, studiato ad hoc per il modello in questione. Una terza alternativa è seguire l'approccio presentato da Gilks et al. (1995) che risulta sostanzialmente automatico sotto opportune condizioni di regolarità.

Alcune difficoltà computazionali sorgono quando si usa una matrice di covarianza di tipo esponenziale: la scomposizione di Cholesky risulta numericamente instabile. Per questa ragione, come fatto in Dunson & Herring (2006), un'alternativa è data dalla funzione

$$C(x_i, x_j; \kappa) = \kappa_2 \exp\left\{-\kappa_1 (x_i - x_j)^2\right\} + \nu\kappa_2 I(x_i = x_j), \qquad \forall i, j. \tag{4.13}$$

dove $\nu > 0$ è una costante fissata. La distribuzione a priori $p(\kappa)$ è generalmente, ma non necessariamente, a componenti indipendenti. Nel caso della matrice definita dall'equazione (4.13), ad esempio si può supporre

$$p(\kappa) = p(\kappa_1)p(\kappa_2), \quad p(\kappa_i) \sim Ga(a_j, b_j), \quad i = 1, 2 \tag{4.14}$$

Il Gibbs sampling è quindi presentato sinteticamente nell'Algoritmo 4. Si noti come il costo computazionale cresce notevolmente, poichè la scomposizione di Cholesky viene effettuata a ciascun ciclo.

4.1.3 Regressione: i dati di telefonia mobile

Nel Capitolo 1 una fonte di informazione è stata trascurata. Le telefonate mensili per ciascuna *SIM* sono delle brevi serie storiche. Sembra legittimo aspettarsi una forma di dipendenza tra osservazioni di mensilità differenti. A scopo illustrativo, un'unica serie è stata analizzata nel dettaglio tramite un GP, utilizzando diverse metodologie. Trattandosi di dati di conteggio, un processo Gaussiano potrebbe sembrare inappropriato allo scopo. Formalmente parlando, il modello è specificato erroneamente; tuttavia gli effetti dell'errata

Algorithm 4 Gibbs sampling per un GP

1. Per $r = 1$ a R
   (a) Generare dalla distribuzione a posteriori del GP da $f(x) \mid - \sim N\left(\tilde{\mu}, \tilde{\Sigma}\right)$.
   (b) Generare la varianza globale da $\sigma^{-2} \mid - \sim Ga(a + \frac{n}{2}, b + \frac{1}{2}||y - f||^2)$.
   (c) Generare da $p(\kappa \mid -) \propto p(\kappa)|C(\kappa)|^{-1/2} \exp\left\{-\frac{1}{2} f^T C(\kappa)^{-1} f\right\}$, direttamente o tramite un passo Metropolis.

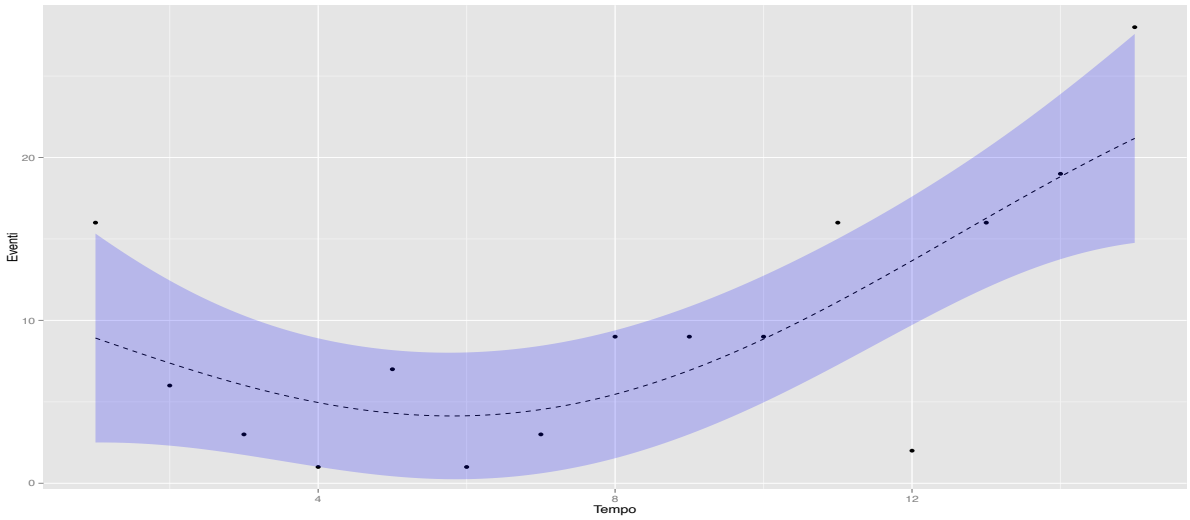
specificazione sono sembrati trascurabili, soprattutto quando il numero di eventi osservati è sufficientemente elevato.

Il nucleo utilizzato è quello descritto dall'equazione (4.13) mentre la stima dei parametri di disturbo è avvenuta sia tramite verosimiglianza marginale, in Figura 4.3(a), che tramite Gibbs sampling, in Figura 4.3(b). Si è posto $\nu = 0.00001$, ovvero un valore sufficientemente piccolo da essere trascurabile a livello interpretativo ma necessario per garantire la stabilità numerica, soprattutto nel Gibbs sampling.

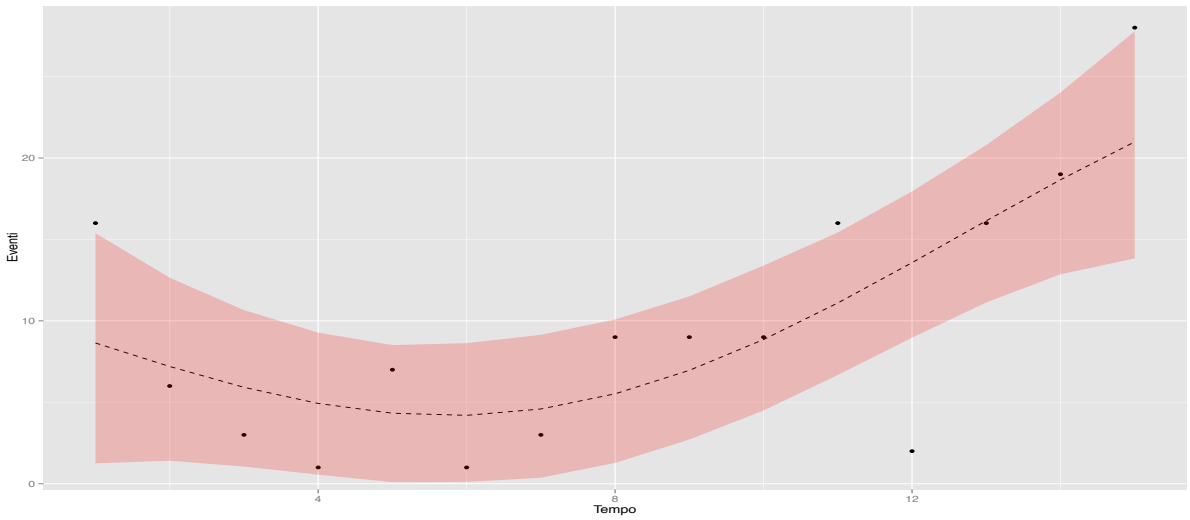
La stima per la verosimiglianza marginale ha richiesto che venissero considerati differenti punti iniziali a causa della presenza di massimi locali. Sono state utilizzate le funzioni `nlminb` e `optim` che hanno condotto allo stesso risultato pur essendo state inizializzate diversamente. Inoltre, poichè il vettore $\theta = (\kappa, \sigma^2)$ assume solamente valori strettamente positivi, è stata necessaria una riparametrizzazione per ragioni di stabilità numerica. La massimizzazione è avvenuta rispetto a $\psi = \exp\theta$.

La stima tramite Gibbs sampling è risultata computazionalmente più onerosa e ha richiesto qualche attenzione ulteriore. Per il vettore di parametri κ si è reso necessario un passo Metropolis, utilizzando una passeggiata casuale gaussiana come *proposal distribution*. La varianza di questa distribuzione è stata determinata a partire dalla matrice di informazione osservata relativa alla funzione di verosimiglianza marginale. Essa è stata calcolata numericamente tramite la funzione `hessian` del pacchetto R `numDeriv` di Gilbert & Varadhan (2015). Inoltre, le stime ottenute tramite verosimiglianza marginale sono state utilizzate per inizializzare la catena. Sono state generate 3000 osservazioni di cui 100 sono state eliminate come periodo di *burn-in*. Analizzando attentamente i *traceplots* non si notano elementi contro l'avvenuta convergenza della catena.

In Tabella 4.1 sono riportate le stime di ciascun parametro ottenuto tramite le due metodologie. Per il metodo bayesiano è stata utilizzata la media a posteriori come indicatore. Qualitativamente, non ci sono forti differenze tra i due metodi di stima. Mentre il primo risulta computazionalmente estremamente efficiente, il secondo è formalmente più coerente. Inoltre, in applicazioni più complesse, la verosimiglianza marginale può non essere facilmente calcolabile e pertanto capire come sviluppare un Gibbs sampling diventa



(a) Stima bayesiana empirica



(b) Stima puramente bayesiana

Figura 4.3: Dati di telefonia mobile: numero di telefonata mensili di un singolo cliente. Confronto tra *Gaussian Process* ottenuti tramite un metodo bayesiano empirico e un approccio puramente bayesiano. Le bande colorate identificano degli intervalli di credibilità al 95%, punto per punto.

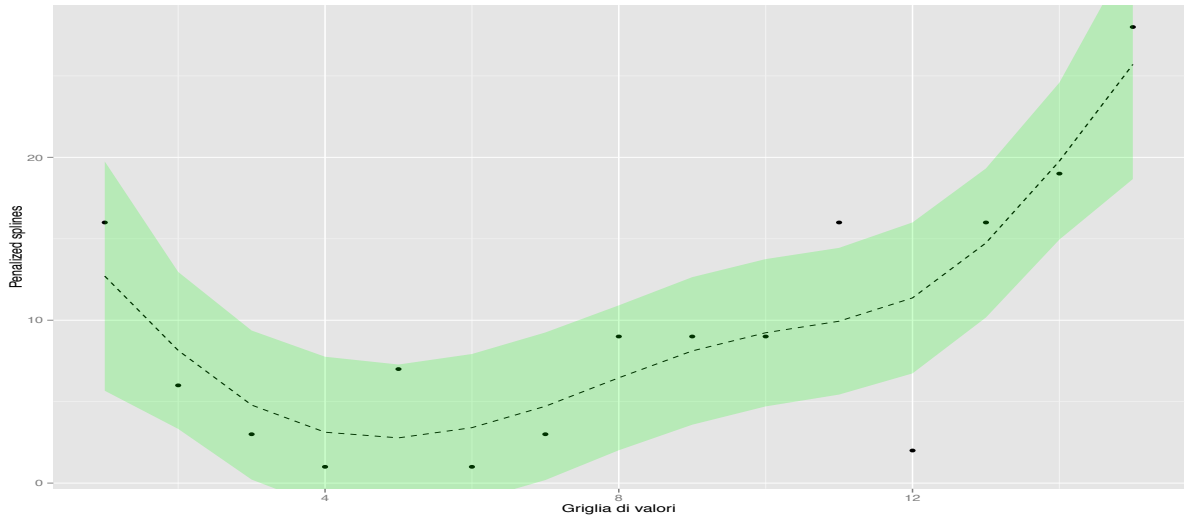
di fondamentale importanza.

Tabella 4.1: Stima dei parametri di disturbo tramite metodo bayesiano empirico e puramente bayesiano

	κ_1	κ_2	σ
Verosimiglianza Marginale	0.0074	367.1781	5.1625
Gibbs sampling	0.0085	428.0508	5.5104
Metodo frequentista: splines	-	-	3.5492

Infine, sia in Tabella 4.1 che in Figura 4.4 sono riportati i risultati di una stima con metodi frequentisti. E' stata adottata la tecnica nota come *smoothing spline*. Il parametro di lisciamento è stato selezionato automaticamente tramite convalida incrociata generalizzata. Si noti come la varianza residua di quest'ultimo modello sembri essere minore rispetto a modelli basati sul *GP*. Poiché essa è stata valutata nell'insieme di stima, non si tratta di un indicatore affidabile. Ad ogni modo, la differenza appare modesta rispetto ai *GP* a giudicare da una ispezione grafica.

Figura 4.4: Dati di telefonia mobile: numero di telefonata mensili di un singolo cliente. Lisciatore ottenuto tramite smoothing spline. Le bande colorate identificano degli intervalli di confidenza al 95%, punto per punto.



4.2 Processo di Dirichlet funzionale (*FDP*)

Si richiami ora struttura funzionale definita dall'equazione (3.1) e si supponga di voler utilizzare i processi gaussiani per affrontare il problema. Da un lato, si può assumere che

tutte le osservazioni funzionali provengano dallo stesso processo gaussiano. Dall'altro, si può ipotizzare invece che ciascuna osservazione funzionale debba essere modellata come un processo gaussiano a sé stante, al limite condividendo la medesima funzione di covarianza. Le due alternative appaiono estreme ed opposte: la prima sembra essere decisamente poco flessibile, mentre la seconda eccessivamente complessa.

Una soluzione intermedia può essere ottenuta seguendo un approccio bayesiano non parametrico, sfruttando le proprietà del DP. Ciò è stato fatto ad esempio in Scarpa & Dunson (2009) oppure in Dunson et al. (2008), seppure con alcune differenze rispetto a quanto verrà esposto ora. Si supponga che

$$x_i(t) = f_i(t) + \varepsilon_i, \quad \varepsilon_i \sim N\left(0, \sigma^2 I_n\right), \qquad t = 1, \dots, T, \quad i = 1, \dots, n, \tag{4.15}$$

$$f_i(t) \sim DP(\alpha, G_0), \qquad G_0 \sim GP\left(\mu, C(t, t; \kappa)\right). \tag{4.16}$$

La struttura indotta dalla (4.16) prende il nome di processo di Dirichlet funzionale (*Functional Dirichlet Process, FDP*), poiché la distribuzione di base del DP è in realtà un processo stocastico. In forma alternativa è possibile scrivere, tramite la rappresentazione *stick-breaking*,

$$f_i(t) = \sum_{h=1}^{\infty} \pi_h \delta_{\theta_h}, \qquad \theta_h \sim GP\left(\mu, C(t, t; \kappa)\right), \tag{4.17}$$

in cui i pesi π_h sono definiti come nelle equazioni (2.15)-(2.16).

Questa specificazione comporta una particolare forma di *clustering funzionale*, indotto dal modello stesso ed in particolare dalla scelta del DP come a priori. Si noti, inoltre, che questo tipo di costruzione include le due precedenti alternative come casi particolari, quantomeno approssimativamente. Infatti al crescere di α il numero di *cluster* indotti dal DP aumenta sino ad uguagliare il numero di osservazioni. Ciò implica che ciascuna traiettoria verrà trattata separatamente dalle altre. Nel caso opposto, ovvero per $\alpha \to 0$, si ottiene un unico *cluster* contenente tutte le osservazioni che darà luogo ad un'unica traiettoria, comune a tutte le osservazioni.

La specificazione del modello si conclude definendo la funzione nucleo $C(t, t; \kappa)$, ad esempio definita come nella (4.13) e specificando le distribuzioni a priori per i parametri rimanenti, ad esempio ponendo

$$\sigma^{-2} \sim \text{Ga}(a, b), \qquad \alpha \sim \text{Ga}(a_\alpha, b_\alpha), \tag{4.18}$$

$$\kappa_1 \sim \text{Ga}(a_\kappa, b_\kappa), \qquad \kappa_2 \sim \text{Ga}(a_\kappa, b_\kappa). \tag{4.19}$$

Si noti, in particolare, la presenza di una distribuzione a priori per α: a livello interpretativo, ciò implica che sono gli stessi dati, congiuntamente alla distribuzione a priori, a indurre il numero di *cluster*. La priori gamma induce una forma coniugata, come mostrato in Escobar & West (1995).

Una seconda possibilità consiste nello specificare il modello tramite una base di funzioni, come nell'equazione (3.2). Pertanto, complessivamente, si avrà che

$$x_i(t) = f_i(t) + \varepsilon_i, \quad \varepsilon_i \sim N\left(0, \sigma^2 I_n\right), \qquad t = 1, \ldots, T, \quad i = 1, \ldots, n, \tag{4.20}$$

$$f_i(t) = \sum_{m=1}^{M} \beta_{im}\phi_m(t), \qquad \beta_i \sim DP(\alpha, G_0), \tag{4.21}$$

in cui $\beta_i = (\beta_{i1}, \ldots, \beta_{iM})$, per una qualche distribuzione di base G_0. Si noti la somiglianza tra questa analisi funzionale ed il modello definito dalle equazioni (2.18)-(2.19) per dati gerarchici. L'indice i identifica individui differenti, mentre le informazioni specifiche a ciascun individuo sono indicizzate dall'indice di tempo t. Nonostante l'interpretazione sia ovviamente diversa, il modello è sostanzialmente il medesimo. Una possibile limitazione della specificazione (4.21) è data dalla necessità di ipotizzare una specifica base di funzioni $\{\phi_m(t)\}_{m=1}^{M}$, così come il numero di componenti M. Utilizzando invece un GP come distribuzione di base, il problema non si pone.

4.2.1 Distribuzione a posteriori del *FDP*

Come in tutte le procedure inferenziali bayesiane, si è interessati nel calcolo della distribuzione a posteriori per tutte le quantità di interesse. L'unica possibilità nel campo bayesiano parametrico, anche nel caso più semplice, è utilizzare tecniche di tipo MCMC. Una prima possibilità è sfruttare un'approssimazione teorica. Se si utilizzano le prime H componenti dello *stick breaking*, si ha che $f_i(t) = \sum_{h=1}^{H} \pi_h \delta_{\theta_h}$. La distribuzione a priori per i pesi finiti $\pi = (\pi_1, \ldots, \pi_H)$ tale che

$$\pi \sim \text{Dir}\left(\alpha/H, \ldots, \alpha/H\right), \tag{4.22}$$

ha ottime proprietà teoriche. In particolare, sfruttando il risultato mostrato in Ishwaran & Zarepour (2002), si ha che, per $H \to \infty$, l'intera procedura converge ad un *FDP*. Sfruttando questa approssimazione costruire un *Gibbs sampling* risulta quindi relativamente semplice.

Una seconda possibilità è basata sull'algoritmo proposto da Bush & MacEachern (1996) in cui il DP viene "eliminato" tramite marginalizzazione. Ad ogni passo MCMC vengono aggiornati solamente gli indicatori relativi al *cluster* di appartenenza e gli atomi funzionali θ_h. Una seconda proposta è costituita dal cosiddetto *slice sampler* introdotto in Walker (2007).

Infine, come proposto da Ishwaran & James (2001), è possibile costruire un algoritmo MCMC a partire dalla rappresentazione troncata dello stick-breaking, come nell'equazione (2.17). Questa procedura viene qui descritta nel dettaglio, poiché è quella che verrà utilizzata successivamente.

La generazione della distribuzione a posteriori per il *FDP* ha molti punti di contatto con l'Algoritmo 4. Come fatto in precedenza, è stato necessario utilizzare un passo *Metropolis within Gibbs*, qualora una forma semplice per la *full conditional* non fosse disponibile. Si suppone che le priori siano specificate dalle equazioni (4.18)-(4.19). Per semplificare la notazione, la matrice di covarianza verrà indicata con $C(\kappa)$: ciò non genera confusione poiché, in questo contesto, ciascun GP viene valutato unicamente nella griglia di punti $t = 1, \dots, T$ e non in un insieme differente.

Non essendo presente nessun software che svolgesse queste operazioni, questo algoritmo è stato completamente implementato ed il codice è riportato in Appendice. E' stato utilizzato il software R ed in particolare il pacchetto R `Rcpp` Eddelbuettel & Francois (2011), il quale permette una facile integrazione tra i linguaggi di programmazione R e C++. Il codice scritto in linguaggio C++ è risultato molto più rapido in alcuni casi, permettendo così una veloce esecuzione della simulazione. L'algoritmo è composto dai seguenti *step*:

1. ***Identificazione dei* cluster.** Ciascun individuo ha probabilità $\pi_i = (\pi_{i1}, \dots, \pi_{iH})$ di appartenere all'h-esimo *cluster*. L'indice S_i è la realizzazione di questa variabile casuale, identificando il *cluster* di appartenenza per l'i-esimo individuo. In formule, si ha che

$$S_i \mid - \sim \text{Multinom}(\pi_i). \tag{4.23}$$

   Il vettore delle probabilità è invece pari a

$$\pi_{ih} = \mathbb{P}\left(S_i = h \mid -\right) \propto \underbrace{V_h \prod_{l<h} (1 - V_l)}_{\text{Stick-breaking}} \overbrace{\prod_{t=1}^{T} \phi\left(y_i(t), \theta_h, \sigma^2\right)}^{\text{Verosimiglianza}}. \tag{4.24}$$

   Si noti che la precedente equazione identifica quantità proporzionali alle probabilità π_i. Poiché questi pesi sono in numero finito, la costante di normalizzazione può essere facilmente ottenuta prendendone la somma.

2. ***Probabilità dello* stick-breaking**. Sfruttando la coniugazione si ottiene che

$$V_h \mid - \overset{i.i.d}{\sim} \text{Beta}\left(1 + \sum_{i=1}^{n} I(S_i = h), \alpha + \sum_{i=1}^{n} I(S_i > h)\right), \tag{4.25}$$

   in cui $I(\cdot)$ è la funzione indicatrice. La quantità $n_h = \sum_{i=1}^{n} I(S_i = h)$ identifica il numero di elementi presenti nel h-esimo *cluster*. La quantità $\sum_{i=1}^{n} I(S_i > h)$ invece identifica quanti individui sono presenti nei *cluster* successivi all'h-esimo.

3. ***Atomi funzionali della mistura***. Gli atomi θ_h sono ottenibili nella maniera usuale, condizionatamente all'appartenenza al *cluster* h-esimo. Si ha che

$$\theta_h \mid - \sim N_T\left(\tilde{\mu}_h, \tilde{\Sigma}\right). \tag{4.26}$$

Il vettore θ_h ha T elementi, poiché valutato nella griglia $t = 1, \ldots, T$. Il t-esimo elemento verrà indicato con la notazione $\theta_h(t)$. I parametri a posteriori sono

$$\tilde{\mu}_h = C(\kappa)\left(C(\kappa) + \frac{1}{n_h}\sigma^2 I_T\right)^{-1}\bar{y}_h, \quad \tilde{\Sigma} = C(\kappa) - C(\kappa)\left(C(\kappa) + \frac{\sigma^2}{n_h} I_T\right)^{-1} C(\kappa),$$

in cui il vettore $\bar{y}_h$ rappresenta, istante per istante, la media delle osservazioni. Rispetto all'equazione (4.4) c'è una lieve differenza: sono disponibili molteplici osservazioni funzionali. Grazie alle proprietà dei modelli normali e dato che ciascuna osservazione è valutata nella medesima griglia di punti, l'atomo funzionale θ_h è esprimibile in forma compatta. La derivazione di questa *full conditional* è trattata implicitamente in (Hoff, 2009, Capitolo 7).

Si noti, inoltre, che la coppia (S_i, θ_h) permette di identificare la traiettoria a posteriori per l'i-esimo individuo $f_i(t) \mid -$, la quale è pari a $\theta_{S_i}(t)$.

4. ***Parametri della funzione di covarianza***. I parametri della matrice di covarianza $C(\kappa)$ non possiedono forma chiusa per la *full conditional*. La densità congiunta è proporzionale a

$$p(\kappa \mid -) \propto p(\kappa)|C(\kappa)|^{-H/2} \exp\left\{-\frac{1}{2}\sum_{h=1}^{H} \theta_h^T C(\kappa)^{-1}\theta_h\right\}, \tag{4.27}$$

in cui $\pi(\kappa)$ rappresenta la distribuzione a priori. Come notato anche nel Capitolo 2, la presenza di punti di massimo locale implica a volte la correlazione negativa tra κ_1 e κ_2. Una possibile *proposal distribution* è quindi una normale bivariata con matrice di covarianza opportunamente tarata, che tenga conto di tale correlazione. Sia κ_r il valore MCMC al passo r-esimo, il valore proposto è del tipo

$$\kappa^* \sim N_2\left(\kappa_r, \Sigma_\kappa\right).$$

L'algoritmo Metropolis prevede che $\kappa_{r+1} = \kappa^*$ con probabilità γ la quale, grazie anche alla simmetria della *proposal distribution*, è pari a

$$\gamma = \min\left\{1, \frac{p(\kappa^* \mid -)}{p(\kappa_r \mid -)}\right\}. \tag{4.28}$$

Il principale vantaggio dell'algoritmo Metropolis sta nel fatto che γ può essere calcolata anche senza conoscere la costante di normalizzazione di $p(\kappa \mid -)$.

5. ***Varianza complessiva***. Sfruttando la coniugazione indotta dalla priori gamma, si ottiene la *full conditional*

$$\sigma^{-2} \mid - \sim \text{Ga}\left(a + \frac{nT}{2}, b + \frac{1}{2}\sum_{i=1}^{n}\sum_{t=1}^{T}\left(y_i(t) - f_i(t)\right)^2\right). \tag{4.29}$$

6. ***Parametro di concentrazione*** α. Questo parametro possiede una *full conditional* forma chiusa grazie alla coniugazione con la a priori. Si ha che

$$\alpha \mid - \sim \text{Ga}\left(a_\alpha + H - 1, b_\alpha - \sum_{h=1}^{H-1}\log\left(1 - V_h\right)\right). \tag{4.30}$$

4.2.2 Simulazione

Per verificare la corretta implementazione dell'algoritmo, sono stati generati dei dati fittizi. Sono state estratte 10 osservazioni funzionali provenienti dallo stesso processo gaussiano, valutato in $t = 1, \ldots, 20$. Ciascuno di essi è stato quindi replicato 20 volte, aggiungendo un errore gaussiano con varianza σ^2, per un totale di $n = 200$ osservazioni funzionali. In questa maniera, sono stati creati 10 *cluster* funzionali latenti, con la stessa funzione covarianza esponenziale, definita dalla (4.13), in cui $\kappa = (0.01, 300)$, $\nu = 0$. Sono proposti due differenti scenari che differiscono per la differente specificazione della varianza σ^2 che genera rispettivamente un rumore debole o forte.

Nel primo scenario, come riportato in Tabella 4.2 il numero di *cluster* latenti viene identificato con ottima precisione così come gli altri parametri coinvolti. Nel secondo scenario invece, come illustrato nella Figura 4.5, il segnale è molto più debole e solamente alcuni gruppi vengono identificati con precisione. Il numero di gruppi latenti stimato dal modello è leggermente superiore al valore effettivo. Ad ogni modo, ciò appare coerente con quanto specificato: in assenza di informazione sono ammissibili risultati inferenziali imprecisi.

Come considerazione, tramite il pacchetto R `microbenchmark` di Mersmann (2014) sono state condotto delle analisi circa i tempi di esecuzione delle funzioni `clusterprob` (implementata tramite `Rcpp`), e `cluster.probabilities` (implementata in puro linguaggio R), le quali calcolano i pesi dell'equazione (4.24). I risultati sono riassunti nella Tabella 4.3: l'implementazione in C++ risulta, come anticipato, nettamente più rapida.

4.3 Applicazione del *FDP* ai dati di telefonia mobile

4.3.1 Stima del modello

Il modello per dati funzionali descritto sinora è stato, infine, applicato alle stesse $n = 2000$ osservazioni del Capitolo 3. L'asimmetria evidenziata nelle analisi descrittive

Tabella 4.2: Risultati della simulazione. I valori utilizzati per la simulazione sono riportati in tabella. La stima bayesiana utilizzata è la media a posteriori, mentre *S.E.* indica la deviazione standard a posteriori.

	Primo scenario			**Secondo scenario**		
	"Vero" valore	*Stima*	*S.E.*	*"Vero" valore*	*Stima*	*S.E.*
# *cluster*	10	10.13	0.44	10	13.17	2.03
σ	5	4.99	0.58	20	19.86	0.39
κ_1	0.01	0.0105	0.00125	0.01	0.0091	0.00198
κ_2	300	331.53	65.51	300	402.25	91.35
α	-	2.12	0.83	-	2.40	0.80

Figura 4.5: Risultati simulazione **secondo scenario**. Ciascuna linea rappresenta la traiettoria stimata per l'individuo i-esimo. I colori invece identificano i gruppi latenti utilizzati nella fase di simulazione, generalmente ignoti nelle applicazioni reali.

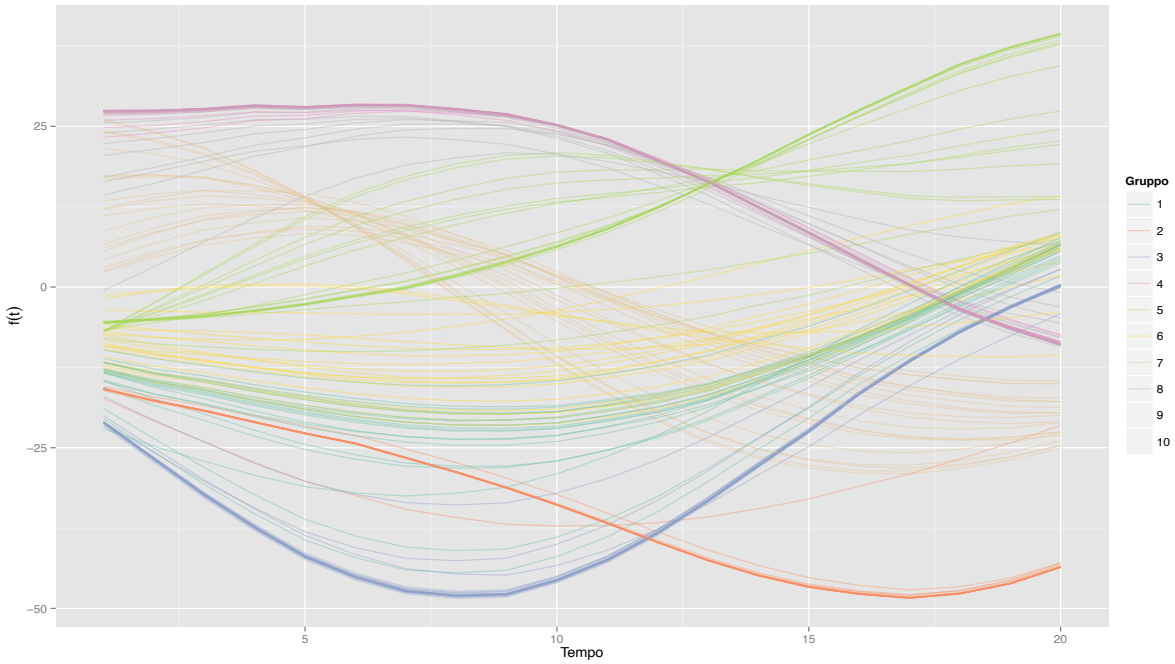


Tabella 4.3: Performance delle due funzioni `clusterprob` e `cluster.probabilities`: tempo di esecuzione espresso in millisecondi. La misurazione è stata effettuata 100 volte utilizzando il *dataset* simulato, utilizzando un processore 1.3 GHz Intel Core i5.

	Minimo	*Primo Quartile*	*Media*	*Terzo Quartile*	*Massimo*
`cluster.probabilities`	43.18	48.03	56.71	64.76	84.88
`clusterprob`	2.29	2.31	2.88	3.37	4.80

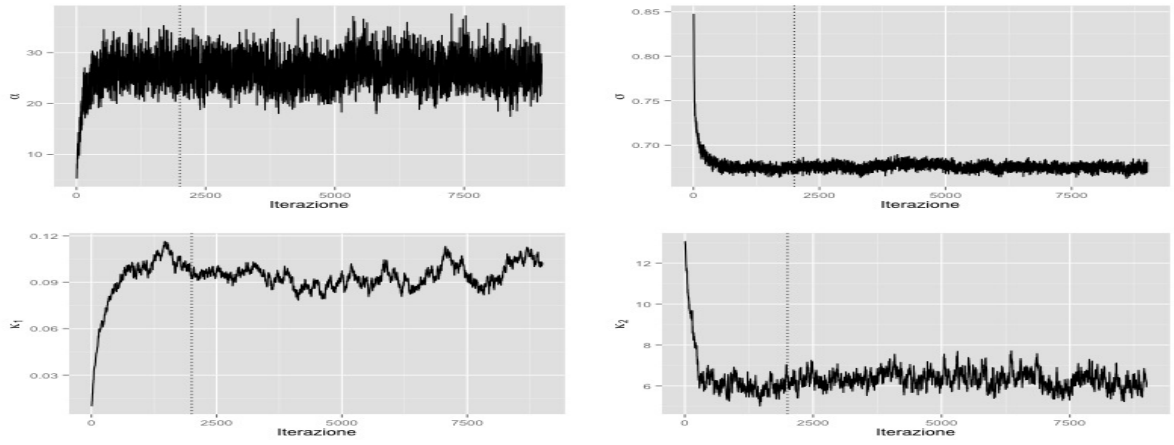
ha suggerito una trasformazione logaritmica dei dati, così da stabilizzare il processo di stima. In particolare i nuovi processi funzionali seguono l'equazione

$$x^*(t) = \log\left(x(t) + 1\right).$$

Si tratta di una trasformazione monotona crescente e pertanto è possibile riottenere i valori originari tramite la trasformata inversa. Rispetto all'approccio frequentista presentato nel Capitolo 3, il *FDP* consente una forma di *clustering funzionale* indotta dal modello stesso.

Le distribuzioni a priori per i parametri del modello sono le stesse utilizzate per l'analisi di simulazione. Gli iperparametri sono stati scelti tali da rendere "vaghe" e quindi poco informative le distribuzioni a priori. Sono state ottenute 9000 iterazioni di cui le prime 2000 sono state eliminate come periodo di *burn-in*. Non sembra esserci evidenza contro la convergenza della catena MCMC: i *traceplots* di alcune quantità rilevanti sono stati riportati in Figura 4.6, includendo il periodo di *burn-in*. Il *mixing* della catena, soprattutto per quel che concerne i parametri della funzione $C(\kappa)$, è perfezionabile. Si nota infatti un chiaro assestamento iniziale, ma in seguito le oscillazioni rimangono fortemente autocorrelate. Eventuali tarature della *proposal distribution* non hanno portato a miglioramenti apprezzabili. Ad ogni modo, si tratta di parametri di scarso interesse inferenziale e pertanto si è deciso di trascurare questo aspetto.

Figura 4.6: *Traceplots* per i parametri α, σ, κ. La linea tratteggiata denota il periodo di *burn-in*



4.3.2 *Clustering funzionale* e *label switching*

Come risultato della catena MCMC, si hanno a disposizione 1500 differenti raggruppamenti per le osservazioni funzionali. Ciascuna osservazione appartiene a *cluster* differenti i quali, oltretutto, cambiano di numero iterazione per iterazione. Come ulteriore complicazione, si manifesta il cosiddetto *label-switching* (Redner & Walker, 1984): ad esempio il "secondo" gruppo alla r-esima iterazione potrebbe essere il "terzo" alla $r+1$-esima. A

fini interpretativi, si è interessati ad avere una caratterizzazione univoca di tali *cluster*. Sebbene esistano metodi che consentono la "rietichettatura" (*relabeling*) dei *cluster* si è preferito optare per una soluzione più semplice, proposta da Medvedovic & Sivaganesan (2002). Viene definita la seguente misura di dissimilarità tra due osservazioni funzionali f_i, f_j e facendo riferimento alla notazione usata in precedenza si ha che

$$d(f_i, f_j) = \frac{1}{R}\sum_{r=1}^{R} I(S_{ir} \neq S_{jr}) = \frac{\text{Numero di volte in cui } S_i \neq S_j}{\text{Numero replicazioni MCMC}}, \tag{4.31}$$

in cui R rappresenta il numero di replicazioni Montecarlo dell'algoritmo MCMC. Quando il numero di osservazioni è elevato il calcolo di questa matrice è oneroso, poichè coinvolge $n(n+1)/2$ elementi. Pertanto, è stata implementata una efficiente funzione, chiamata `clusterdist`, in linguaggio C++ che svolge il calcolo tenendo conto della simmetria della matrice di dissimilarità D.

A partire dalla matrice D è quindi possibile utilizzare uno degli usuali algoritmi di *clustering*. Come in precedenza, è stato utilizzato l'algoritmo *PAM*. Sono stati quindi selezionati 30 *cluster*. Il numero è stato scelto in maniera abbastanza arbitraria: non eccessivamente elevato da compromettere l'interpretazione, né eccessivamente ridotto da falsificare le analisi.

4.3.3 Analisi dei risultati

In Tabella 4.4 sono stati riportati i valori a posteriori per le quantità di interesse. Il numero di *cluster* risulta elevato come conseguenza di un valore elevato per il parametro di concentrazione α.

Tabella 4.4: Risultati della catena MCMC per i principali parametri di interesse. HPD LI e HPD LS indicando rispettivamente il limite inferiore ed il limite superiore di un intervallo di credibilità HPD di livello 0.95

	Media a posteriori	*Mediana a posteriori*	*HPD LI*	*HPD LS*
# *cluster*	137.3	137	129	144
σ	0.675	0.675	0.668	0.682
κ_1	0.093	0.094	0.082	0.109
κ_2	6.32	6.3	5.500	7.094
α	26.46	26.38	21.469	32.312

A seguito dell'algoritmo di clustering, le funzioni sono state rappresentate in Figura 4.7. I differenti *cluster* sono molto ben caratterizzati: i numeri $5, 7, 10, \ldots$ sono traiettorie sostanzialmente costanti ma con differenti livelli. Invece i *cluster* con numero

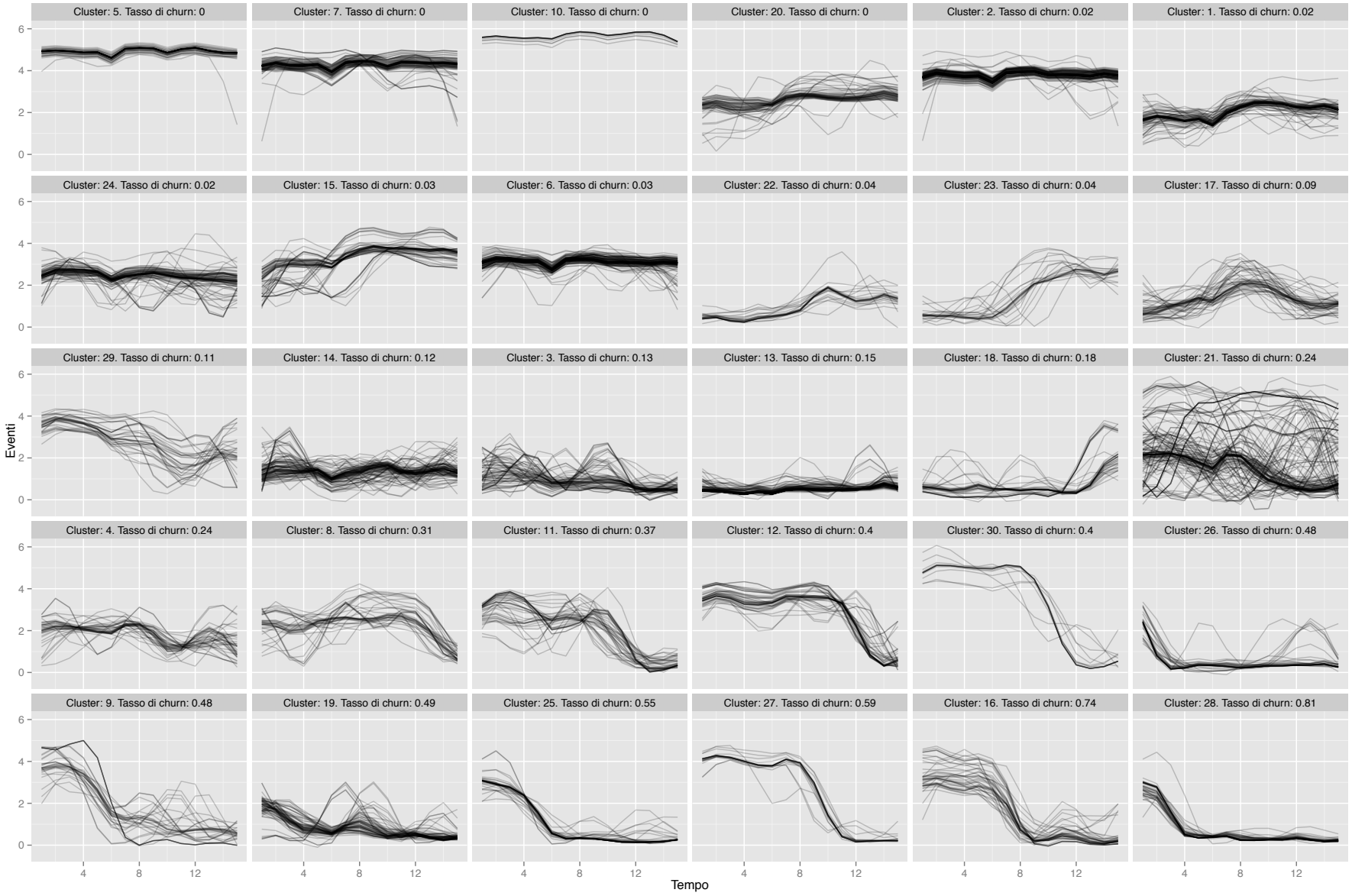
$19, 25, 27, \ldots$ manifestano traiettorie decrescenti, ma con differente livello di pendenza. I numeri $22, 23$ manifestano invece traiettorie crescenti.

Il *cluster* numero 21 è particolare: è l'unico che parrebbe essere non omogeneo al suo interno. Ciò è giustificato dal fatto che l'algoritmo *PAM* discrimina tra soli 30 gruppi a fronte dei circa 130 identificati dal modello bayesiano non parametrico. Sembra esserci infatti la tendenza a raggruppare nello stesso *cluster* osservazioni funzionali "orfane" di un proprio gruppo di appartenenza.

E' stata calcolata la proporzione di clienti in stato di *churn* per ciascun *cluster*. Come prevedibile, le traiettorie decrescenti manifestano un rischio di *churn* molto più elevato rispetto a quelle con trend elevato e costante. In alcuni casi, ad esempio nel *cluster* 26, sembra che il *churn* avvenga dopo un periodo di latenza, in cui il telefono viene utilizzato ad intermittenza, fino alla effettiva disattivazione. Questa tipologia di clienti può è sostanzialmente *silente*, fatta eccezione per alcune sporadiche telefonate. In una successiva analisi si potrebbe pertanto decidere di trattare questi clienti come in stato di "*churn*".

Molto diversa è invece la situazione del *cluster* 12. In questo gruppo di osservazioni, infatti, i clienti riducono drasticamente il numero di telefonate solamente nell'ultimo periodo. Questa tipologia di clienti, poiché solo di recente ha deciso di abbandonare la compagnia telefonica, è quella che più necessiterebbe di azioni di *retention*, ovvero azioni di marketing volte a convincere l'utente a rimanere fedele all'azienda.

Figura 4.7: Rappresentazione di 30 *cluster* ottenuti tramite la funzione `pam`, scala logaritmica. La sequenza dei gruppi è stata ordinata sulla base della percentuale di clienti inattivi.



Capitolo 5

Classificazione con predittore funzionale

5.1 Previsione del *churn*, nuove proposte

Nel Capitolo 1 si è tentato di prevedere il *churn* utilizzando ciascun valore della serie storica come variabile esplicativa. In questo Capitolo l'obiettivo dell'analisi è il medesimo, ma l'interdipendenza temporale di ciascuna osservazione è esplicitamente trattata. In primo luogo, verranno presentati approcci più tradizionali, mentre successivamente si farà largo uso dei risultati del Capitolo 4. In entrambi i casi, le osservazioni funzionali vengono riassunte tramite un qualche indicatore, nel tentativo di ricondursi a tecniche note in letteratura. La scelta di questo indicatore è cruciale a fini previsivi.

Rispetto al Capitolo 1 c'è un'importante differenza: è stato effettuato un ricampionamento degli insiemi di stima e di verifica ottenuti in precedenza, al fine di poter calcolare, o anche semplicemente salvare in memoria, alcuni risultati. Si pensi ad esempio alle *matrici di dissimilarità* ottenute in precedenza, le quali crescono con ordine $O(n^2)$. Per rendere meglio l'idea di quanto ciò sia computazionalmente gravoso, si consideri che una matrice di dimensione 20.000×20.000 occupa circa 3Gb di memoria. Le 2000 osservazioni dei Capitoli 3 e 4 provengono nella misura del 75% e 25% rispettivamente dall'insieme di stima e quello di verifica. Ad ogni modo, la ridotta numerosità campionaria non compromette la validità di un eventuale confronto tra i metodi presentati ora e quelli del Capitolo 1.

5.2 Regressione logistica funzionale

Viene qui ripreso l'approccio presentato in Ramsay & Silverman (2005, Capitolo 15). L'idea consiste nel considerare un particolare integrale dell'osservazione funzionale come indicatore riassuntivo. Si faccia anzitutto riferimento alla notazione della (3.1) e si consideri una sua *basis expansion* definita dalla (3.2), la quale può essere equivalentemente

espressa in forma matriciale come

$$f_i(t) = \Phi^T(t)\beta_i, \quad \text{oppure in forma compatta} \quad f(t) = \mathbf{C}\Phi(t), \tag{5.1}$$

in cui la i-esima riga della matrice $\mathbf{C}$ contiene i coefficienti β_i. Ci si vuole ricondurre al modello di regressione logistica, specificando opportunamente il predittore lineare η. Il modello prevede che esso sia pari a

$$\eta = \int_0^T f(t)\delta(t)dt, \qquad \delta(t) = \Phi_*^T(t)\beta_*, \tag{5.2}$$

in cui pertanto si introduce una nuova base di funzioni $\delta(t)$, il cui scopo è mettere in relazione la variabile scalare y_i con ciascuna osservazione funzionale. Nonostante questa forma sembri matematicamente poco trattabile, si osservi che sostituendo le quantità opportune si ottiene

$$\eta = \int_0^T \mathbf{C}\Phi(t)\Phi_*^T(t)\beta_* dt = \mathbf{C}\mathbf{J}\beta_*, \qquad \mathbf{J} = \int_0^T \Phi(t)\Phi_*^T(t)dt. \tag{5.3}$$

Pertanto, supponendo che i coefficienti della matrice $\mathbf{C}$ siano noti o quantomeno stimati in una fase precedente, il modello risulta lineare nei parametri e possono quindi essere utilizzati i metodi presentati nel Capitolo 1.

Questo tipo di approccio è stato quindi applicati ai dati di telefonia mobile grazie alla funzione `fregre.glm` del già citato pacchetto R `fda.usc`. Sono state specificate due basi di funzioni di tipo *B-splines* per $\delta(t)$ e per $f(t)$. I risultati sono infine riportati in Tabella 5.1.

5.3 Modello ad effetti casuali

5.3.1 Specificazione del modello

Seguendo motivazioni di carattere bayesiano, viene qui suggerita una procedura che ricorda per alcuni aspetti quella presentata in Dunson et al. (2008). Nel Capitolo 4 ciascuna osservazione funzionale veniva associata ad un indicatore di *cluster* S_i il quale variava a ciascuna iterazione del *Gibbs sampling*. Si propone qui una procedura in due passi in cui anzitutto vengono definiti i *cluster funzionali* e, condizionatamente ad essi, si ottiene una stima per la probabilità di *churn*. In formule, si ha che

$$\mathbb{P}\left(Y_i = 1 \mid S_i = h\right) = \text{logit}^{-1}(\mu_h), \qquad \mu_h \sim F, \quad h = 1, 2, \dots \tag{5.4}$$

E' pertanto di interesse valutare la distribuzione a posteriori $\mu_h \mid y$. Sono quindi necessari due ulteriori passi di simulazione:

1. **Aggiornamento effetto di gruppo**. Si simula un valore dalla distribuzione condizionata
$$\mu_h \mid y, S \sim F^*, \qquad h = 1, 2, \ldots, \tag{5.5}$$
la cui distribuzione dipende, ovviamente, da F. Un caso particolare è analizzato nel paragrafo successivo.

2. **Generazione degli indicatori di *cluster***. Si simulano gli indicatori di *cluster* dalla loro distribuzione $p(S)$, ovvero la distribuzione a posteriori ottenuta nel Capitolo 4. Si noti, tuttavia, che questi non vengono generati dalla distribuzione *full conditional* $S \mid -$. Questa modifica è giustificata quasi esclusivamente da considerazioni di tipo operativo: si vuole evitare che i gruppi dipendano dalla variabile dipendente y ma, al tempo stesso, si vuole mantenere la variabilità che la procedura del Capitolo 4 comporta.

Questa procedura può essere vista come un'*approssimazione della distribuzione a posteriori* per μ nel caso in cui $p(S) \approx p(S \mid y)$, ovvero nel caso in cui le osservazioni y siano poco informative circa la composizione dei gruppi. In questo caso la distribuzione S che deriva dal Capitolo 4 è fortemente informativa riguardo la composizione dei gruppi e sovrasta l'informazione derivante da y. Ulteriore conferma di questo fatto consiste nel confronto con metodi di tipo frequentista, che porgono risultati simili.

Anche nel caso in cui ciò non accadesse, si sta implicitamente forzando una struttura di questo tipo. Ciò è comunque coerente da un punto di vista bayesiano, se si suppone che $p(S)$ sia una generica distribuzione a priori estremamente informativa, tale da non essere di fatto influenzata dalle osservazioni.

5.3.2 *Polya-Gamma data augmentation*

Si supponga ora che la distribuzione a priori per μ_h sia della forma

$$\mu_h \overset{i.i.d.}{\sim} N\left(\mu_0, \sigma_\mu^2\right), \quad h = 1, 2, \ldots, \qquad \sigma_\mu^{-2} \sim \mathrm{Ga}(a_\mu, b_\mu), \quad \mu_0 \sim N(0, \sigma_0^2). \tag{5.6}$$

La generazione della distribuzione a posteriori per μ_h è non banale e richiede a sua volta metodi di tipo MCMC. Trattandosi di un modello logistico tradizionalmente questo implicava uno algoritmo Metropolis-Hastings. Un'alternativa è costituita dalla proposta di Albert & Chib (1993), basata sul concetto di *data augmentation*, il quale permette di ricondursi ad un più elegante *Gibbs sampling*. L'idea alla base consiste nell'introduzione di "dati fittizi". Il *data augmentation step* è solamente uno "trucco" matematico: non viene realmente introdotta informazione aggiuntiva. Il processo latente verrà quindi trattato come una variabile casuale e la sua generazione sarà parte della catena MCMC.

L'idea suggerita in Polson et al. (2013) consiste quindi nell'introduzione di un vettore latente ω, avente distribuzione marginale Polya-Gamma. Essa risulta avere ottime proprietà qualora la distribuzione a priori per i parametri della regressione, in un modello logistico, sia normale. Per una recente applicazione di questa procedura in presenza di effetti casuali si veda ad esempio Durante et al. (2014). Essa è definita come

Definizione 5 (Distribuzione Polya-Gamma) *Una variabile casuale X ha distribuzione Polya-Gamma con parametri $b > 0, c \in \mathbb{R}$, denotata con $X \sim PG(b, c)$ se*

$$X = \frac{1}{2\pi^2} \sum_{k=1}^{\infty} \frac{g_k}{(k - 1/2)^2 + c^2/(4\pi^2)}, \tag{5.7}$$

in cui $g_k \sim Ga(b, 1)$ sono variabili casuali indipendenti.

Si presenta qui una lieve modifica dell'algoritmo originale, in cui è presente un parametro aggiuntivo, il quale consente l'interpretazione del modello in termini di *effetti casuali*. Per semplicità di notazione, il modello viene riscritto in forma matriciale, come segue

$$\mathbb{P}\left(Y = 1 \mid S = \mathbf{h}\right) = \text{logit}^{-1}\left(\eta\right), \quad \eta = D\mu \qquad \mu \sim N\left(\mu_{\mathbf{0}}, \mathbf{P}\right), \tag{5.8}$$

in cui la matrice D è costituita da variabili *dummies* che indicano l'appartenenza a ciascun *cluster* mentre $\mu = (\mu_1, \mu_2, \dots)$ e $\mathbf{P} = \sigma_\mu^2 I_n$. Si noti che sebbene il numero di *cluster* sia variabile a ciascuna iterazione, esso è comunque un numero finito. Per il modello in questione, la distribuzione a posteriori per $\mu \mid -$ si ottiene attraverso i seguenti passaggi

1. **Aggiornamento della variabile latente** ω. Sfruttando la coniugazione indotta dalla PG, si ottiene che

   $$\omega_i \mid - \sim PG(1, d_i^T \mu), \qquad i = 1, \dots, n \tag{5.9}$$

   in cui d_i rappresenta la i-esima riga della matrice D.

2. **Aggiornamento dell'effetto casuale** μ. Anche in questo caso, si ottiene la distribuzione *full conditional*

   $$\mu \mid - \sim N\left(\mu_\omega, \Sigma_\omega\right), \qquad \Sigma_\omega = (D^T \Omega D + \mathbf{P}^{-1})^{-1}, \quad \mu_\omega = \Sigma_\omega (D^T z + \mathbf{P}^{-1}\mu_0), \tag{5.10}$$

   in cui $\Omega = \text{diag}(\omega_1, \dots, \omega_n)$ mentre l'i-esimo elemento del vettore z è pari a $z_i = y_i - 1/2$.

3. **Aggiornamento della varianza di** μ. Sfruttando la coniugazione normale-gamma, si ottiene che

   $$\sigma_\mu^{-2} \mid - \sim \text{Ga}\left(a_\mu + \dim(\mu)/2, b_\mu + \mu^T \mu/2\right), \tag{5.11}$$

   in cui $\dim(\mu)$ rappresenta la dimensione del vettore μ la quale, come detto in precedenza, è variabile.

4. **Aggiornamento predittore lineare "di base"** μ_0. Si supponga che l'iperparametro $\sigma_0^{-2} = 0$, ovvero che la distribuzioni a priori sia impropria. Sfruttando la coniugazione e si ottiene ugualmente una *full conditional* propria tale che

$$\mu_0 \mid - \sim N\left(\frac{1}{\dim(\mu)} \sum_{h=1}^{\dim(\mu)} \mu_h, \frac{\sigma_\mu^2}{\dim(\mu)}\right) \tag{5.12}$$

L'intera procedura risulta quindi efficace a condizione di disporre di un algoritmo per generare valori casuali da una distribuzione PG. Non si tratta di un'operazione banale ed infatti essa è trattata nel dettaglio in Polson et al. (2013), i quali, come materiale supplementare, mettono a disposizione il pacchetto R `BayesLogit`, che svolge il compito. La catena MCMC generata in tal modo è inoltre uniformemente ergodica, come mostrato in Choi & Hobert (2013).

5.3.3 I dati di telefonia mobile

Il modello bayesiano descritto è stato quindi applicato ai dati di telefonia mobile. Solamente le 1500 osservazioni dell'insieme di stima sono state utilizzate per ottenere la distribuzione a posteriori $\mu_h \mid y$. Una previsione per le osservazioni appartenenti all'insieme di verifica è invece ottenibile associando, a ciascuna iterazione del Gibbs, l'effetto casuale corrispondente al *cluster* di appartenenza. Qualora questo non fosse disponibile, è stata utilizzata la media globale.

In Figura 5.1 è riportato un confronto tra i predittori lineari del modello ad effetti casuali e quello ottenuto dal modello di regressione logistica frequentista del Capitolo 1. Come si evince dal grafico, i risultati sono molto correlati e giungono a risposte simili. Nonostante il paragone sia, da un punto di vista "filosofico", inappropriato, è comunque confortante vedere come metodologie estremamente differenti giungano a risultati tra loro coerenti. Esistono, tuttavia, delle differenze: in Figura 5.2 viene messa in evidenza la natura discreta della probabilità di *churn*. A ciascuna iterazione, infatti, esiste un numero finito di probabilità di *churn*, creando talvolta degli "ammassamenti" riscontrabili anche in Figura 5.1. In secondo luogo, la struttura ad effetti casuali induce il ben noto effetto di *shrinkage*: le stime a posteriori tendono ad allinearsi intorno alla media complessiva μ_0. Osservando i valori negli assi della Figura 5.1, si può notare come nel modello ad effetti casuali i valori siano molto meno "estremi" rispetto al modello di regressione logistica. Il sovradattamento è molto mitigato dalla presenza del parametro σ_μ^2, che determina l'ammontare dello *shrinkage*, esattamente come avviene nella *ridge regression* (Hastie et al., 2001, Capitolo 3).

In Tabella 5.1 sono stati riportati gli indicatori di bontà di adattamento valutati nell'insieme di verifica.

Figura 5.1: Predittore lineare di un modello logistico frequentista e del modello ad effetti casuali, per un totale di 2000 osservazioni, appartenenti indistintamente all'insieme di stima e di verifica

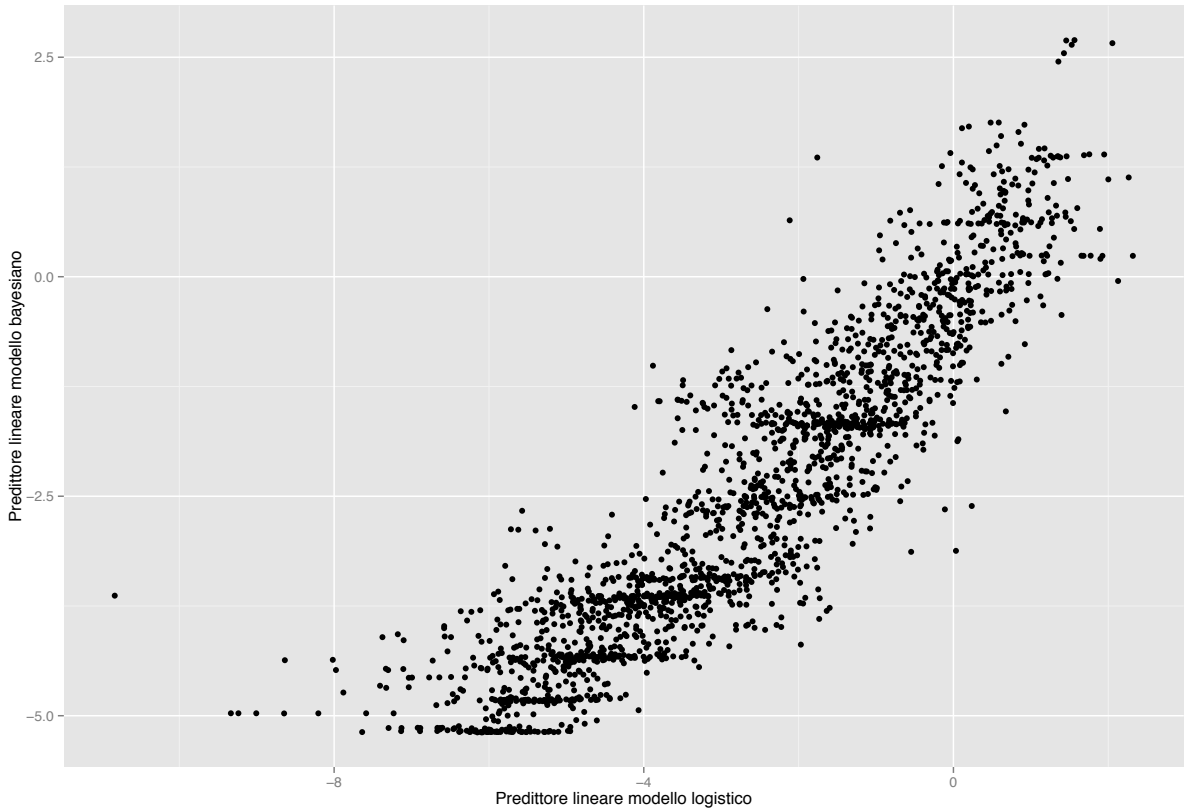
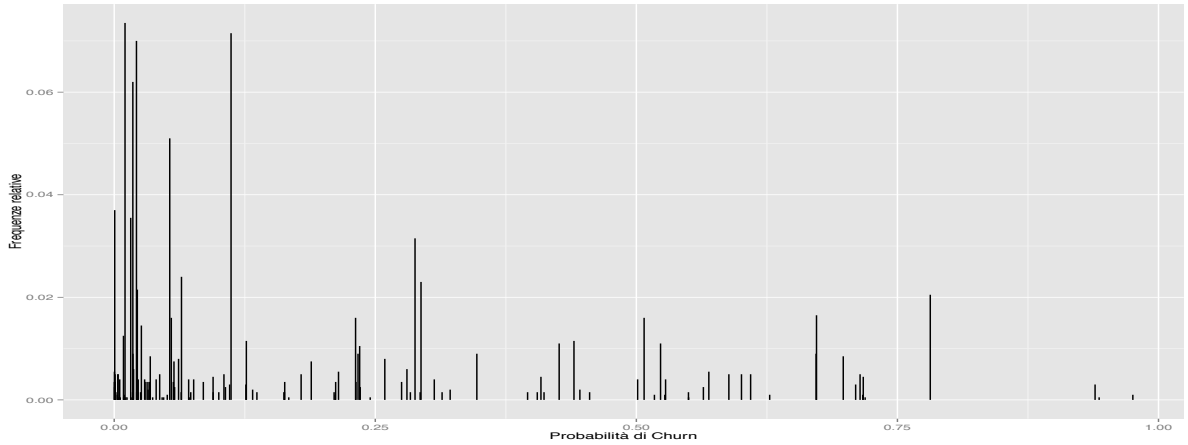


Figura 5.2: Probabilità di *churn* alla i-esima iterazione: la natura discreta delle osservazioni è messa in evidenza. In ordinata c'è la frequenza relativa di ciascun gruppo.



5.4 Algoritmo *k-nearest-neighbor*

In questo paragrafo, viene presentato un terzo approccio per la previsione del *churn* chiamato *k-nearest-neighbor* (Hastie et al., 2001, Capitolo 13). L'idea è molto semplice e utilizza il concetto di "distanza" tra le variabili esplicative, generalmente quella euclidea nel caso in cui queste assumano valori reali. La stima della probabilità di *churn* per un generico elemento y_i avviene considerando i primi k elementi più vicini e prendendone la media. Qualora vi fossero dei pareggi, essi vengono estratti casualmente fino al raggiungimento di k elementi. In formule si ha che

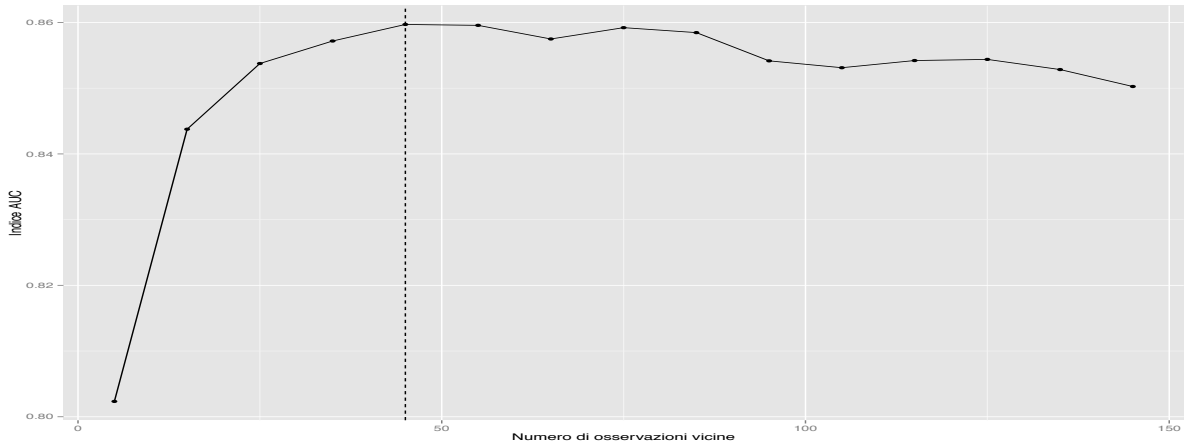
$$\hat{\pi}_i = \frac{1}{k} \sum_{i=1}^{k} y_{(i)}, \tag{5.13}$$

in cui $y_{(1)}, \ldots, y_{(k)}$ rispetto ad una misura di distanza $d(f_i, f_j)$. Nei problemi di classificazione viene generalmente utilizzato il "voto di maggioranza" per stabilire in quale classe allocare ciascuna osservazione. Nel caso di regressione binaria, tuttavia, ciò è equivalente a verificare se $\hat{\pi}_i > 1/2$.

Nel caso specifico, la matrice di distanza utilizzata è quella definita dall'equazione (4.31). Nonostante essa provenga da procedure di tipo bayesiano, si noti come l'intera procedura di *clustering funzionale* possa essere interpretata come una mera riorganizzazione delle variabili esplicative, dal momento che la variabile y_i non è stata coinvolta.

Il parametro k ha il ruolo di parametro di regolazione e deve essere selezionato in maniera opportuna. Si è deciso quindi di ottenerlo tramite convalida incrociata, utilizzando 10 gruppi. In Figura 5.3 è riportato l'indice AUC, al variare di k, sotto convalida incrociata. E' stato messo in evidenza il punto di massimo ad indicare il numero k ottimale.

Figura 5.3: Selezione del numero di *neighbor* ottimale utilizzando la convalida incrociata



Utilizzando $k = 45$ è stato quindi ottenuta una previsione nell'insieme di verifica: in Tabella 5.1 sono disponibili alcuni indicatori. Per questa fase, non è stato utilizzato alcun

pacchetto R. Le funzioni per la stima `knn` così come la procedura per selezione di k `knn.cv` sono state implementate nel software R e sono riportate in Appendice.

5.5 Confronto complessivo

Come riportato nella Tabella 5.1, il modello che consente la migliore previsione sembra essere, nuovamente, il *gradient boosting*. I modelli di regressione logistica, così come quello funzionale descritto nel primo paragrafo di questo Capitolo, sono stati utilizzati come *benchmark*, ovvero un termine di paragone, una soglia al di sotto del quale è difficile scendere. Il primo, infatti, trascurava ogni possibile interdipendenza tra le osservazioni, mentre il secondo richiederebbe un serio raffinamento nella scelta delle basi di funzioni.

I modelli che prendono le mosse dal *clustering funzionale* di tipo bayesiano non parametrico hanno infatti capacità previsive maggiori, come in effetti ci si aspetterebbe. Il metodo *k-nearest-neighbor* sembra essere leggermente migliore rispetto a quello ad effetti casuali. Ciò è in parte dettato dalla struttura non parametrica di tale metodo, che consente maggiore flessibilità. Si tratta, ad ogni modo, di differenze minime.

Il successo del *gradient boosting* è dovuto al fatto che esso implica il coinvolgimento di termini di interazione. Pertanto, in una qualche misura, l'interdipendenza tra le osservazioni viene modellata, seppure non in maniera esplicita. Ciò tuttavia non implica che esso sia il modello "migliore" in termini assoluti: esso non consente in alcuna maniera di intuire i differenti comportamenti dei clienti, di caratterizzarli e pertanto comprendere più approfonditamente le ragioni del *churn*.

Tabella 5.1: Confronto degli indici AUC, tasso di errata classificazione, falsi positivi, falsi negativi

	AUC	Errore complessivo	Falsi Positivi	Falsi Negativi
Gradient boosting (Cap. 1)	0.904	0.126	0.290	0.099
Logistico (Cap. 1)	0.849	0.184	0.500	0.176
Logistico funzionale	0.839	0.183	0.466	0.175
k-nearest-neighbor	0.885	0.146	0.176	0.144
Logistico effetti casuali	0.880	0.148	0.286	0.133

Appendice A

Codice utilizzato

A.1 Funzioni globali

Codice A.1: Funzioni inserite nel file `functions.R`

```r
# Caricamento librerie aggiuntive
require(ggplot2)
require(earth)
require(gbm)
require(ROCR)

# Calcolo indice AUC
Auc <- function(y,p.hat) {
  if(NCOL(p.hat)==1) return(performance(prediction(p.hat,y), measure="auc",x.measure="fpr")@y.values[[1]])
  auc<-apply(p.hat,2,function(x) performance(prediction(x,y), measure="auc",x.measure="fpr")@y.values[[1]] )
  names(auc) <- colnames(p.hat)
  auc
}

# Tasso di errata classificazione
MErr <- function(class.verifica, class.stima) {
  n<- length(class.verifica)
  1- sum(diag(table(class.verifica,class.stima)))/n
}

Err <- function(class.verifica, class.stima) {
  if(NCOL(class.stima)==1) return(MErr(class.verifica,class.stima))
  apply(p.hat,2,function(x) MErr(x,y.verifica) )
}

# Rappresentazione curva ROC
Roc <- function(y,p.hat,add=FALSE,col="black") {
  plot(performance(prediction(p.hat,y), measure="tpr", x.measure="fpr"),add=add,col=col)
  abline(c(0,1), lty="dashed")
}

# Rappresentazione curva Lift
Lift <- function(y,p.hat,add=FALSE,col="black") {
  plot(performance(prediction(p.hat,y), measure="lift", x.measure="rpp"),add=add,col=col)
  abline(h=1, lty="dashed")
```

```
}

# Calcolo Falsi Positivi
FPR  <- function(y,p.hat,cutoff=0.5) {
  n <- table(p.hat > cutoff,y)
  FPR=n[2,1] / (n[2,1] + n[2,2])
}

# Calcolo Falsi Negativi
FNR  <- function(y,p.hat,cutoff=0.5) {
  n <- table(p.hat > cutoff,y)
  FNR=n[1,2] / (n[1,2] + n[1,1])
}
```

A.2 Previsione del Churn

Codice A.2: Insieme di stima, insieme di verifica

```
set.seed(494); source("functions.R")
n <- NROW(data)

# Percentuale di dati utilizzati per l'insieme di stima
percentage <- 0.75

id.stima    <- sample(1:n,round(percentage*n))
id.verifica <- setdiff(1:n,id.stima)

id.train <- sample(1:length(id.stima),round(0.75*length(id.stima)))
id.validation <- setdiff(1:length(id.stima),id.train)

y.stima    <- factor(data$Churn[id.stima])
y.verifica <- factor(data$Churn[id.verifica])

X.stima    <- data[id.stima,-which(names(data)=="Churn")]
X.verifica <- data[id.verifica,-which(names(data)=="Churn")]

matrix.stima <- model.matrix(~.,data=X.stima)[,-1]
matrix.verifica <- model.matrix(~.,data=X.verifica)[,-1]
```

Codice A.3: Regressione logistica e selezione *stepwise*

```
# Regressione logistica - Modello Completo
m.bin <- glm(y.stima ~., data=X.stima, family="binomial")
# Regressione logistica - Modello con selezione stepwise
m.bin.reduced <- step(m.bin,direction = "both",trace = TRUE)

# Previsione
p.bin.reduced <- predict(m.bin.reduced,newdata=X.verifica, type="response")
```

Codice A.4: Modello MARS

```
# Modello MARS
m.mars1 <- earth(y.stima ~. , data= X.stima, glm=list(family=binomial),degree=1, nfold=0)
```

```
# Previsione
p.mars1 <- predict(m.mars1, newdata=X.verifica, type="response")
```

Codice A.5: Gradient boosting

```
# Parametro di apprendimento
shrinkage = 0.15

# Modello gradient boosting
m.gb <- gbm(as.numeric(y.stima)-1~.,data=X.stima, n.trees=300,interaction.depth=5, distribution="bernoulli",shrinkage=
     shrinkage,train.fraction=0.75)

# Selezione della iterazione ottimale nell'insieme di validazione
best.iter <- gbm.perf(m.gb,method="test")

# Previsione
p.gb <- predict(m.gb, newdata=X.verifica,n.trees=best.iter,type="response")
```

Codice A.6: Regressione logistica bayesiana

```
# Regressione logistica bayesiana
require(BayesLogit)
m.bayes.logit <- logit(y = as.numeric(y.stima)-1,X=model.matrix(m.bin.reduced),samp = 3000)
X.reduced.verifica <- model.matrix(~ X2004.11.01 + X2004.12.01 + X2005.01.01 +  X2005.07.01 + X2005.12.01 + X2006
     .01.01,data=X.verifica)

# Previsione
posterior.p <- plogis(X.reduced.verifica%*%t(m.bayes.logit$beta))
p.bayes.logit <- apply(posterior.p,1,mean)
```

Codice A.7: Confronto complessivo

```
cutoff <- 0.5

p.hat<- data.frame(Binomiale.red=p.bin.reduced,
                   Binomiale.bayes=p.bayes.logit,
                   MARS1=as.numeric(p.mars1),
                   Gradient_boosting=p.gb
                   )

Error <- numeric(NCOL(p.hat))
FP <- numeric(NCOL(p.hat))
FN <- numeric(NCOL(p.hat))

for(i in 1:NCOL(p.hat)) {
        table <- as.matrix(table(y.verifica, p.hat[,i]>cutoff))
        Error[i] <- 1 - sum(diag(table))/sum(table)
        FP[i] <- FPR(y.verifica,p.hat[,i],cutoff)
        FN[i] <- FNR(y.verifica,p.hat[,i],cutoff)
}

Sommario <- as.matrix(cbind(Auc(y.verifica,p.hat),Error,FP,FN))
colnames(Sommario) <- c("AUC","Errore","Falsi Positivi","Falsi Negativi")
```

A.3 Processi gaussiani

Codice A.8: Caricamento librerie necessarie e dati

```
# Caricamento librerie necessarie
require(fields)
require(mvtnorm)
require(numDeriv)

set.seed(160491)

# Caricamento dati - Definizione quantita interesse
data <- read.table("datasetspread.csv", sep=";",header=TRUE)
data <- subset(data, select=-c(ID,Churn,Causale))

y <- as.numeric(data[48,])
n <- 15
x <- 1:n
distance <- rdist(x,x)^2
```

Codice A.9: Stima processo gaussiano tramite verosimiglianza marginale

```
# Funzione per la verosimiglianza marginale
lmarginal <- function(theta,y,distance){
  C <- theta[2]*exp(-theta[1]*distance)
  diag(C) <- diag(C) + theta[3]^2
  L <- t(chol(C))
  alpha <- backsolve(t(L),forwardsolve(L,y))
  lC <- 2*sum(log(diag(L)))
  -0.5 * lC - 0.5*t(y)%*%alpha
}

# Differenti algoritmi di massimizzazione
nlminb(start=c(1,1,1), function(theta) -lmarginal(theta,y,distance), lower=c(1e-4,1e-4,1e-4))$objective
nlminb(start=c(0.1,0.1,0.1), function(theta) -lmarginal(theta,y,distance), lower=c(1e-4,1e-4,1e-4))$objective
nlminb(start=c(-1,-1,-1), function(theta) -lmarginal(exp(theta),y,distance))$objective
optim(par=c(1,1,1),function(theta) -lmarginal(exp(theta),y,distance))$value
optim(par=c(0,0,0),function(theta) -lmarginal(exp(theta),y,distance))$value

# Punto di massimo
theta.hat <- exp(optim(par=c(0,0,0),function(theta) -lmarginal(exp(theta),y,distance))$par)

kappa1 <- theta.hat[1]
kappa2 <- theta.hat[2]
sigma  <- theta.hat[3]

# Nuova griglia di punti
x.star <- seq(min(x),max(x),length=200);

# Matrici utili
C.prior <- kappa2*exp(-kappa1*rdist(x.star,x.star)^2)
C.mix   <- kappa2*exp(-kappa1*rdist(x,x.star)^2)
C.train <- kappa2*exp(-kappa1*rdist(x,x)^2)

# Scomposizione di Cholesky - Algoritmo efficiente
L     <- t(chol(C.train + sigma^2*diag(n)))
alpha <- backsolve(t(L),forwardsolve(L,y))
```

```
v     <- forwardsolve(L,C.mix)

# Media a posteriori e Varianza a posteriori
mu.tilde    <- t(C.mix)%*%alpha
Sigma.tilde <- C.prior - t(v)%*%v

# Intervallo di credibilita
se <- qnorm(0.975)*sqrt(pmax(diag(Sigma.tilde),0))

# Rappresentazione grafica
ggplot(data=NULL,aes(x=x.star,y=mu.tilde)) + geom_line(linetype="dashed") + xlab("Tempo") + ylab("Eventi") + geom_
     point(aes(x=x,y=y)) + geom_ribbon(ymax=mu.tilde+se,ymin=mu.tilde-se,alpha=.2,fill="blue")
```

Codice A.10: *Gibbs sampling* per il processo gaussiano

```
# Logaritmo della a priori per parametro kappa
lprior <- function(kappa,a,b){
  dgamma(kappa,a,b,log=TRUE)
}

# Log-fullconditional per parametri kappa
lpost <- function(kappa,f,distance,a,b,jitter=0.001){
  if(any(kappa<0)) return(-Inf)
        C       <- exp(-distance*kappa[1])*kappa[2]
        diag(C) <- diag(C) + jitter*kappa[2]
        L       <- t(chol(C))
        alpha   <- backsolve(t(L),forwardsolve(L,f))
        lC      <- 2*sum(log(diag(L)))
  -0.5 * lC - 0.5*t(f)%*%alpha + lprior(kappa[1],a,b) + lprior(kappa[2],a,b)
}

# Costante nu
jitter <- 0.00001

# Iperparametri
mu.0     <- rep(0,length(x.star))
a  <- b  <- 1e-6
a0 <- b0 <- 1e-7

# Numero di replicazione MonteCarlo
R <- 3000

# Inizializzazione dei parametri
kappa  <- c(theta.hat[1],theta.hat[2])
sigma  <- theta.hat[3]

# Taratura della proposal distribution
eps<- sqrt(diag(solve(hessian(function(theta) -lmarginal(theta,y,distance),theta.hat))))[1:2]

# Vettori vuoti per l'output
f.out <- matrix(0,R,n)
sigma.out <- rep(0,R)
k.out <- matrix(0,R,2)
accept <- c(0,0)

for(r in 1:R) {

  # Aggiornamento funzione nucleo
```

```
  C.prior              <-   exp(-distance*kappa1)*kappa2
  diag(C.prior) <- diag(C.prior) + jitter*kappa2
  C.train              <- C.prior
  C.mix                <- C.prior

  # Scomposizione di Cholesky
  L     <- t(chol(C.train + sigma^2*diag(n)))
  alpha <- backsolve(t(L),forwardsolve(L,y))
  v     <- forwardsolve(L,C.mix)

  # Parametri del GP
  mu.tilde    <- t(C.mix)%*%alpha
  Sigma.tilde <- C.prior - t(v)%*%v

  # Estrazione dalla full Conditional di un GP
  f <- c(rmvnorm(1,mu.tilde,Sigma.tilde,method="svd"))
  residuals <- y - f

  # Full conditional per Varianza complessiva
  sigma <- sqrt(1/rgamma(1, a + n/2, b + sum(residuals^2)/2))
  Sigma <- matrix(0,n,n); diag(Sigma) <- sigma^2

  # Step Metropolis
  kappa.star <- kappa

  for(j in 1:2){
  kappa.star[j] <- rnorm(1,kappa[j],eps[j])
  alpha <- min(1,exp(lpost(kappa.star,f,distance,a0,b0,jitter=jitter)-
                     lpost(kappa,f,distance,a0,b0,jitter=jitter)))
  if(runif(1) < alpha) {kappa[j] <- kappa.star[j]; accept[j] <- accept[j] +1}
  }

  # Output
  f.out[r,] <- f
  sigma.out[r] <- sigma^2
  k.out[r,] <- kappa
}

burn.in <- 1:100

# Traceplots
par(mfrow=c(3,1))
plot(sqrt(sigma.out),main="Traceplot degli errori standard",ylab=expression(sigma),type="l")
plot(k.out[,1],main="Traceplot di kappa 1",ylab=expression(kappa[1]),type="l")
plot(k.out[,2],main="Traceplot di kappa 2",ylab=expression(kappa[2]),type="l")
par(mfrow=c(1,1))

# Stima a posteriori
f.mean <- apply(f.out[-burn.in,],2,mean)
theta.sim <- c(apply(k.out,2,mean),mean(sqrt(sigma.out[-burn.in])))
ymin   <- apply(f.out[-burn.in,],2,function(x) quantile(x,0.025))
ymax   <- apply(f.out[-burn.in,],2,function(x) quantile(x,0.975))

# Rappresentazione grafica
ggplot(data=NULL,aes(x=x,y=f.mean)) + geom_line(linetype="dashed") + xlab("Tempo") + ylab("Eventi") + geom_point(aes(x
     =x,y=y)) + geom_ribbon(ymax=ymax,ymin=ymin,alpha=.2,fill="red")
```

Codice A.11: *Smoothing splines*

```
require(mgcv)
# Calcolo modello smoothing splines
m.gam <- gam(y ~ s(x,bs="cr"))
fit <- predict(m.gam)

# Intervallo di confidenza
se <- qnorm(0.975)*predict(m.gam,se.fit=TRUE)$se.fit

# Rappresentazione grafica
ggplot(data=NULL,aes(x=x,y=fit)) + geom_line(linetype="dashed") + xlab("Griglia di valori") + ylab("Smoothing splines"
     ) + geom_point(aes(x=x,y=y)) + geom_ribbon(ymax=fit+se,ymin=fit-se,alpha=.2,fill="green")
```

A.4 Clustering funzionale

Codice A.12: Operazioni preliminari

```
# Caricamento librerie utili
require(fda)
require(fda.usc)
require(cluster)

# Selezione delle prime 2000 osservazioni
data <- X.stima;  data$ID <- rownames(data)
data <- data[1:2000,]
colnames(data) <- c(1:15,"ID")
```

Codice A.13: Lisciamento dei dati funzionali

```
# Trasformazione dati in formato "fdata"
data.fd <- fdata(as.matrix(data))

# Selezione dei parametri "lambda" e il numero di M di basi
lambda <- 2^seq(-2, 9, length.out = 30)
nb <- seq(4,18 , by = 1)
out0 <- min.basis(data.fd, lambda = lambda, numbasis = nb,verbose = TRUE)

# Salvataggio dati
data.smooth.fd <- out0$fdata.est
```

Codice A.14: Clustering funzionale

```
# Definizione numero di cluster
K <- 7

# Algoritmo PAM
cluster1 <- pam(data.smooth.fd$data,K)

# Rappresentazione grafica
data.plot1 <- data.frame(data.smooth.fd$data[cluster1$id.med,])
data.plot1$ID <- rownames(data.smooth.fd$data[cluster1$id.med,])
colnames(data.plot1) <- c(1:15,"ID")
data.plot1 <- melt(data.plot1)
data.plot1$variable <- as.numeric(data.plot1$variable)
ggplot(data=data.plot1,aes(x=variable,y=value,col=ID)) + geom_line(size=1.6) + xlab("Tempo") + ylab("Eventi")
```

A.5 *Functional Dirichlet Process*

Codice A.15: Funzione in C++ chiamata `clusterprob.cpp`

```
#include <RcppArmadillo.h>

using namespace Rcpp;
using namespace arma;

mat clusterprob(mat y, vec v,  mat theta, double sigma) {
   int n = y.n_rows;
   int H = theta.n_rows;
   int T = y.n_cols;
   mat out(n,H);
   vec stick(H);
   vec cumv(H);
   double Likelihood;

   cumv[0] = 1-v[0];
   stick[0]= v[0];
   // Primo step  Stick-breaking
   for(int h =1; h < H; h++) {
     cumv[h] = cumv[h-1]*(1-v[h]);
     }
   // Secondo step Stick-breaking
   for (int h = 1; h < H; h++) {
     stick[h]= v[h] * cumv[h-1];
     }

   // Verosimiglianza
    for (int i = 0; i < n; i++) {
      for (int h = 0; h < H; h++) {
        Likelihood=0;
        for(int t = 0; t < T; t++) {
          Likelihood = Likelihood + R::dnorm(y(i,t),theta(h,t),sigma,TRUE);
        }
        out(i,h)= stick[h]*exp(Likelihood);
      }
    }
    return out;
}
```

Codice A.16: Funzione in C++ chiamata `clusterdist`

```
#include <RcppArmadillo.h>

using namespace Rcpp;
using namespace arma;

mat clusterdist(mat cluster) {
  int R = cluster.n_rows;
  int n = cluster.n_cols;
  mat out(n,n);
  for(int i = 0; i < n; i++) {
    for(int j = 0; j < n; j++) {
      out(i,j) = 0;
    }
```

```
  }

  for(int r = 0; r < R; r++) {
    for(int i = 0; i < n; i++) {
      for(int j = (i+1); j < n; j++) {
        if(cluster(r,i)!=cluster(r,j)){
          out(i,j) = out(i,j) + 1;
          out(j,i) = out(i,j);
        }
      }
    }
  }

  return out;
}
```

Codice A.17: Step 1 *Gibbs sampling*

```
cluster.probabilities <- function(data,v,sigma2,theta){

  n <- NROW(y)
  H <- NROW(theta)
  out <- matrix(0,n,H); stick<-rep(0,H)
  Likelihood <- rep(0,H)
  se <- sqrt(sigma2)
  cumv <- v
  stick[1] <- v[1]
  for(h in 2:H) stick[h]<- v[h]*cumv[h-1]
  # Ciascuna riga e' un individuo
  for(i in 1:n) {
    for(h in 1:H){
    Likelihood[h] <- exp(sum(dnorm(data[i,],theta[h,],se,log=TRUE)))
    out[i,h]      <- stick[h]*Likelihood[h]
    }
  }
  out <- out / apply(out,1,sum) # Normalizzazione
  out
}

cluster.allocation <- function(pi) {
  require(Hmisc)
  as.numeric(as.factor(rMultinom(pi,1)))
}
```

Codice A.18: Step 2 *Gibbs sampling*

```
stickbreaking.weights <- function(cluster,alpha,H) {
  nc <- rep(0,H)
  v  <- rep(0,H); v[H]  <- 1
  for (h in 1:H) nc[h]<- length(cluster[cluster==h])
  for (h in 1:(H-1)) {
    v[h] <- rbeta(1,1 + nc[h],alpha + sum(nc[(h+1):H]))
  }
  v
}
```

Codice A.19: Step 3 *Gibbs sampling*

```
atoms <- function(y, Ckappa,cluster,sigma,H,mu.prior){

  t <- NCOL(Ckappa)
  theta<- matrix(0,H,t)

  for (h in 1:H) {
    y.h<- y[cluster==h,]
    nh <- sum(cluster==h)
    if(nh==0) {mu.tilde=rep(0,t); Sigma.tilde = Ckappa}
    if(nh>0){
      if(nh ==1) {y.bar <- y.h} else y.bar <- colMeans(y.h);

      L     <- t(chol(Ckappa + sigma^2/nh*diag(t)))
      v     <- forwardsolve(L,Ckappa)
      Sigma.tilde <- Ckappa - t(v)%*%v
      alpha <- backsolve(t(L),forwardsolve(L,y.bar))
      mu.tilde    <- t(Ckappa)%*%alpha
    }
    theta[h,]   <- c(rmvnorm(1,mu.tilde,Sigma.tilde,method="svd"))
  }
  theta
}
```

Codice A.20: Step 4 *Gibbs sampling*

```
# Matrice di Covarianza
Covariance <- function(kappa,distance,jitter) {
  C <- exp(-distance*kappa[1])*kappa[2]
  diag(C) <- diag(C) + jitter*kappa[2]
  C
}

lprior <- function(kappa,a,b){
  sum(dgamma(kappa,a,b,log=TRUE))
}

# Log-Full conditional
lpost <- function(kappa,theta,distance,a,b,jitter){
  if(any(kappa<0)) return(-Inf)
  H <- NROW(theta)
  C <- Covariance(kappa,distance,jitter)
  L <- t(chol(C))
  quadratic <- apply(theta,1, function(x) t(x)%*%backsolve(t(L),forwardsolve(L,x)))
  quadratic <- sum(quadratic)
  lC <- sum(log(diag(L)))
  -H *lC - 0.5*quadratic + lprior(kappa,a,b)
}

hyperprior.kappa.mvt <- function(kappa,theta,distance,eps,a0,b0,jitter){
# Metropolis Step
  kappa.star <- kappa
  accept <- 0
  kappa.star <- c(rmvnorm(1,kappa,eps))
  alpha <- min(1,exp(lpost(kappa.star,theta,distance,a0,b0,jitter=jitter)-
                     lpost(kappa,theta,distance,a0,b0,jitter=jitter)))
  if(runif(1) < alpha) {kappa <- kappa.star; accept <- 1}
  list(kappa=kappa, accept=accept)
}
```

Codice A.21: Step 5 *Gibbs sampling*

```
global.se <- function(y,theta,cluster,a,b){
  residuals <- y - theta[cluster,]
  sqrt(1/rgamma(1, a + length(residuals)/2, b + sum(residuals^2)/2))
}
```

Codice A.22: Step 6 *Gibbs sampling*

```
cluster.alpha <- function(v,a,b) {
  H <- length(v)
  rgamma(1,H-1,b - sum(log(1 - v[-H])))
}
```

Codice A.23: Inizializzazione *Gibbs sampling*

```
jitter <- 0.001
# Limite superiore per la mistura
H <- 200

# Costanti fissate
t <- 1:NCOL(data)
distance <- rdist(t,t)^2
n <- NROW(data)

# Definizione quantita a priori
a0 <- b0 <- 0.01  # Iperparametri per kappa
a  <- b  <- 1e-6  # Iperparametri varianza complessiva
mu.prior <- as.numeric(colMeans(data))

# Taratura Step Metropolis
mu.eps   <- c(0.101,6.0629)

eps.sd1  <- 0.001;
eps.sd2  <- 0.15;

eps.rho  <- -0.65

eps      <- diag(c(eps.sd1,eps.sd2))^2; eps[1,2]<- eps[2,1] <- eps.sd1*eps.sd2*eps.rho
accept   <- 0

# Inizializzazione
theta   <- matrix(0,H,NCOL(data))
mu0     <- rep(0,NCOL(data))
sigma   <- 1000;
alpha   <- 10#log(n)
v       <- rep(0.5,H) # Pesi stick-breaking
v[H]    <- 1          # Troncamento ad H termini
kappa   <- c(0.12, 5.5)
mu <- matrix(0,NROW(data),NCOL(data))
cluster.dist <- matrix(0,n,n)
beta <- rep(0,H); eta <- rep(0,n)

# Numero replicazioni MonteCarlo
R <- 9000
```

```
# Output
theta.out   <- array(0,dim=c(R,NROW(theta),NCOL(data)))
mu.out      <- array(0,dim=c(R,NROW(data),NCOL(data)))
kappa.out   <- matrix(0,R,2)
sigma.out   <- rep(0,R)            # Varianza complessiva
ns.out      <- rep(0,R)            # Numero di cluster
eta.out     <- matrix(0,R,n)
cluster.out <- matrix(0,R,n)
alpha.out   <- rep(0,R)
```

Codice A.24: *Gibbs sampling*

```
# Gibbs sampling
for(r in 1:R){

# Step 1
pi <- clusterprob(data,v,theta,sigma)
pi <- pi/apply(pi,1,sum)

#cluster.probabilities(data,v,sigma^2,theta)

# Step 1.5
cluster <- cluster.allocation(pi)

# Step 2
v <- stickbreaking.weights(cluster,alpha,H)

# Step 2.5 - Updating th covariance function
Ckappa <- Covariance(kappa,distance,jitter)

# Step3
theta <- atoms(data, Ckappa,cluster,sigma,H,mu.prior)

# Step 4. Updating Covariance parameter
update <- hyperprior.kappa.mvt(kappa,theta,distance,eps,a0,b0,jitter)
kappa  <- update$kappa
accept <- accept + update$accept

# Step 5
sigma <- global.se(data,theta,cluster,a,b)

# Step 6
alpha <- cluster.alpha(v,a,b)

# Step 7

  # Sistemazione dataset
  X    <- model.matrix(y ~ as.factor(cluster)-1)
  X.train <- X[1:1500,]
  X.test  <- X[1501:2000,]

  nbin <- 1
  p.random <- NCOL(X.train)
  z <- nbin*y.train - nbin/2

  # Altri parametri
  tau <- rgamma(1, a + p.random/2, b + crossprod(beta)/2);
  bmu <- rep(rnorm(1,mean(beta), sqrt(1/(tau*p.random))),p.random)
```

```
  #Polya-Gamma Update
  Omega <- matrix(0,n.train,n.train); P <- matrix(0,p.random,p.random)
  diag(Omega) <- rpg.devroye(num=n.train,n=nbin,z=eta)
  diag(P)<- tau
  V <- solve(t(X.train)%*%Omega%*%X.train + P)
  M <- V%*%(t(X.train)%*%z + P%*%bmu)
  beta     <- c(rmvnorm(1,mean = M,sigma = V))
  eta      <- c(X.train%*%beta)
  eta.test <- c(X.test%*%beta)

# Salvataggio Informazioni
sigma.out[r]   <- sigma
ns.out[r]      <- length(unique(cluster))
theta.out[r,,] <- theta
#mu.out[r,,]    <- theta[cluster,]
kappa.out[r,]  <- kappa
cluster.out[r,]<- cluster
eta.out[r,]      <- eta
eta.test.out[r,] <- eta.test
alpha.out[r]   <- alpha
mu0.out[r]     <- bmu[1]
tau.out[r]        <- tau
print(r) # Il numero dell'iterazione viene stampato a schermo. Puo' causare rallentamenti
}
```

Codice A.25: Clustering

```
cluster.dist <- as.dist(clusterdist(cluster.out[-burn.in,]))/NROW(cluster.out[-burn.in,])
N.cluster <- 30
cluster1 <- pam(cluster.dist,N.cluster)
```

A.6 Algoritmo *k-nearest-neighbor*

Codice A.26: Funzioni per il calcolo dell'Algoritmo *k-nearest-neighbor* e per la *cross-validation*

```
# TRUE della variabile type e' la stima.
knn <- function(y,type,distance,k) {
  ytrain <- y[type]
  dist   <- distance[!type,type]
  apply(dist,1,function(x) mean(ytrain[order(x)[1:k]]))
}

knn.cv <- function (y, distance, ngroup = n,k) {
  call <- match.call()
  n <- length(y)
  ngroup <- trunc(ngroup)
  if (ngroup < 2) {
    stop("ngroup deve essere maggiore o uguale a 2")
  }
  if (ngroup > n) {
    stop("ngroup deve essere inferiore o uguale al numero di osservazioni")
  }
  if (ngroup == n) {
    groups <- 1:n
    leave.out <- 1
```

```
  }
  if (ngroup < n) {
    leave.out <- trunc(n/ngroup)
    o <- sample(1:n)
    groups <- vector("list", ngroup)
    for (j in 1:(ngroup - 1)) {
      jj <- (1 + (j - 1) * leave.out)
      groups[[j]] <- (o[jj:(jj + leave.out - 1)])
    }
    groups[[ngroup]] <- o[(1 + (ngroup - 1) * leave.out):n]
  }

  # I gruppi sono il test set
  cv.fit <- rep(NA, n)
  for (j in 1:ngroup) {
    type <- rep(TRUE,n)
    type[sort(groups[[j]])] <- FALSE
    cv.fit[sort(groups[[j]])] <- knn(y,type,distance,k)
  }
  if (leave.out == 1)
    groups <- NULL
  return(cv.fit)
}
```

Ringraziamenti

Escludendo rari e geniali casi (Gauss, 1799), chiamiamoli pure *outliers*, il contributo portato da una tesi di laurea è talmente modesto da passare al dimenticatoio nell'arco di pochi giorni. Questo manoscritto, comunque venga valutato, credo non farà eccezione. Scrivere una tesi di laurea è tuttavia un traguardo importante: la comunità scientifica potrà forse stupirsene, ma l'obiettivo, per chiunque scriva, è già stato raggiunto.

Essa è, in primo luogo, la chiusura di un percorso. Se continuassimo a procedere negli studi senza interruzioni, rischieremmo di dimenticare le motivazioni iniziali. Al contrario, la consolidazione di un risultato favorisce la sosta, un eventuale cambio di direzione e l'acquisizione di nuovi stimoli. Avendo la possibilità di scrivere in libertà le proprie idee, gli interessi individuali si delineano con naturalezza. La tesi laurea, inoltre, consente di approfondire tematiche poco conosciute, e per quanto piccole siano le scoperte che deriveranno al termine della ricerca, l'atto di indagare con curiosità è di per sé affascinante.

La tesi è infine un buon momento per ringraziare. Non si arriva ad una laurea da soli e tantomeno facendo affidamento sulle proprie sole forze. I primi che meritano un grande grazie sono i miei genitori. Quasi ironicamente, dopo un'intera adolescenza (prolungatasi anche oltre il necessario), trascorsa a contraddirli, sono i primi che sento di voler ringraziare, per avermi lasciato scegliere ciò che volevo ed in piena libertà, per avermi educato, per avermi economicamente sostenuto permettendomi di costruirmi un futuro. Per avermi sempre voluto bene. Per estensione, un forte abbraccio va a tutta la mia famiglia, alle mie sorelline, e ai miei nonni, che sono stati un esempio sia umano che professionale.

Desidero poi ringraziare il mio relatore il prof. Bruno Scarpa, per l'aiuto ricevuto nella stesura di questa tesi, l'enorme disponibilità, per i consigli, le critiche ed ogni discussione. Per avermi spronato a dare il meglio ed avermi indicato un buon modo per vivere l'Università. E' anche "colpa" sua se mi sono rimasto così tanto affascinato dagli studi.

Infine gli amici, che sono tanti, troppi da elencare. A tutti loro va reso il merito di avermi sopportato nei miei modi di fare, di avermi rallegrato serata dopo serata, chi davanti ad un fuoco, chi davanti ad una Ceres, chi offendendo Sansa Stark.

Grazie a tutti.

Bibliografia

Abramowitz, M. & Stegun, I. A. (1972). *Handbook of Mathematical Functions*. National Bureasu of Standards Applied Mathematics, Series 55.

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory*, B. N. Petrov & F. Csaki, eds. Budapest: Akadémiai Kiado.

Albert, J. (2008). *Bayesian Computation with R*. Springer.

Albert, J. H. & Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association* .

Azzalini, A. (2008). *Inferenza Statistica*. Springer.

Azzalini, A. & Scarpa, B. (2011). *Data Analysis and Data Mining*. Oxford University Press.

Berger, J. (2006). The case for objective bayesian analysis. *Bayesian Analysis* **1**, 385–402.

Berry, M. J. A. & Linoff, G. S. (2004). *Data mining techniques. For marketing, sales, and customer relationship management*. Wiley.

Blackwell, D. & MacQueen, J. B. (1973). Ferguson distributions via polya urn schemes. *The Annals of Statistics* **1**, 353–355.

Breiman, L. (2001). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science* **16**, 199–231.

Bush, C. A. & MacEachern, S. N. (1996). A semiparametric bayesian model for randomised block design. *Biometrika* **83**, 275–285.

Canale, A. (2012). *Bayesian nonparametric models for count data with applications to customer base management*. Ph.D. thesis, Università degli studi di Padova.

Carlin, B. P. & Louis, T. A. (2008). *Bayesian Methods for Data Analysis*. Chapman and Hall.

Casella, G. & Robert, C. P. (2010). *Introducing Monte Carlo methods with R*. Springer.

Choi, H. M. & Hobert, J. P. (2013). The polya-gamma gibbs sampler for bayesian logistic regression is uniformly ergodic. *Electronic Journal of Stattistics* **7**, 2054–2064.

Diaconis, P. & Freedman, D. (1986). On the consistency of bayes estimates. *The Annals of Statistics* **14**, 1–26.

Dunson, D. B. (2009). Bayesian nonparametric hierarchical modeling. *Biometrical Journal* **51**, 273–284.

Dunson, D. B. & Herring, A. H. (2006). Semiparametric bayesian latent trajectory models. Tech. rep., Duke University.

Dunson, D. B., Herring, A. H. & Siega-RIz, A. M. (2008). Bayesian inference on changes in response densities over predictor clusters. *Journal of the American Statistical Association* **103**, 1508–1517.

Durante, D., Shah, I. & Torelli, N. (2014). Bayesian nonparametric modeling of contraceptive use in india. Submitted to the Annals of Applied Statistics.

Eddelbuettel, D. & Francois, R. (2011). Rcpp: Seamless r and c++ integration. *Journal of Statistical Software* **40**, 1–18.

Efron, B. (2012). Bayesian inference and the parametric bootstrap. *Annals of Applied Statistics* **6**, 1971–1997.

Escobar, M. D. & West, M. (1995). Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association* **90**, 577–588.

Febrero-Bande, M. & Oviedo de la Fuente, M. (2012). Statistical computing in functional data analysis: The R package fda.usc. *Journal of Statistical Software* **51**, 1–28.

Ferguson, T. S. (1973). A bayesian analysis of some nonparametric problems. *The Annals of Statistics* **1**, 209–230.

Ferguson, T. S. (1974). Prior distributions on spaces of probability measures. *The Annals of Statistics* **2**, 615–629.

Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* **55**, 119–139.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics* **19**, 1–67.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis* **38**, 367–378.

Gauss, C. F. (1799). *Demonstratio nova theorematis omnem functionem algebraicam rationalem integram unius variabilis in factores reales primi vel secundi gradus resolvi posse*. Master's thesis.

Gelman, A. & Hill, J. (2006). *Data Analysis Using Regression and Multilevel / Hierarchical Models*. Cambridge University Press.

Genz, A., Bretz, F., Miwa, T., Mi, X., Leisch, F., Scheipl, F. & Hothorn, T. (2015). *mvtnorm: Multivariate Normal and t Distributions*. R package version 1.0-3.

Gilbert, P. & Varadhan, R. (2015). *numDeriv: Accurate Numerical Derivatives*. R package version 2014.2-1.

Gilks, W., Best, N. G. & Tan, K. (1995). Adaptive rejection metropolis sampling within gibbs sampling. *Journal of the Royal Statistical Society* **44**, 455–472.

Hastie, T., Tibshirani, R. & Friedman, J. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The Annals of Statistics* **28**, 337–407.

Hastie, T., Tibshirani, R. & Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer.

Hjort, N. L., Holmes, C., Muller, P. & Walker, S. (2010). *Bayesian nonparametrics*. Cambridge University Press.

Hoff, P. (2009). *A First Course In Bayesian Statistical Methods*. Springer.

Ishwaran, H. & James, L. F. (2001). Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association* **96**, 161–173.

Ishwaran, H. & Zarepour, M. (2002). Exact and approximate sum representations for the dirichlet process. *The Canadian Journal of Statistics / La Revue Canadienne de Statistique* **30**, 269–283.

Jacques, J. & Preda, C. (2014). Functional data clustering: a survey. *Advances in Data Analysis and Classification* **8**, 231–255.

Kaufman, L. & Rousseeuw, P. (1987). Clustering by means of medoids. *Statistical Data Analysis Based on the L1-Norm and Related Methods* , 405–416.

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M. & Hornik, K. (2015). *cluster: Cluster Analysis Basics and Extensions*. R package version 2.0.3.

McCullagh, P. & Nelder, J. A. (1989). *Generalized linear models*. Chapman and Hall.

Medvedovic, M. & Sivaganesan, S. (2002). Bayesian infinite mixture model based clustering of gene expression profiles. *Bioinformatics* **18**, 1194–1206.

Mersmann, O. (2014). *microbenchmark: Accurate Timing Functions*. R package version 1.4-2.

Milborrow, S. (2015). *earth: Multivariate Adaptive Regression Splines*. R package version 4.4.1.

Muliere, P. & Tardella, L. (1998). Approximating distributions of random functionals of ferguson-dirichlet priors. *Canadian Journal of Statistics* **9**, 249–265.

Newton, M. & Raftery, A. E. (1994). Approximate bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society, Series B* , 3–48.

Nychka, D., Furrer, R. & Sain, S. (2015). *fields: Tools for Spatial Data*. R package version 8.2-1.

Petrone, S., Rousseau, J. & Scricciolo, C. (2014). Bayes and empirical bayes: do they merge? *Biometrika* **101**, 285–302.

Pitman, J. (1996). Some developments of the blackwell-macqueen urn scheme. *Statistics, probability and game theory* **30**, 245–267.

Polson, N. G., Scott, J. G. & Windle, J. (2013). Bayesian inference for logistic models using pólya–gamma latent variables. *Journal of the American Statistical Association* **108**, 1339–1349.

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ramsay, J. O., Hooker, G. & Graves, S. (2009). *Functional Data Analysis with R and MATLAB*. Springer.

Ramsay, J. O. & Silverman, B. W. (2005). *Functional Data Analysis*. Springer.

Rao, C. R. (2007). Has statistics a future? if so in what form? *Journal of Indian Society of Agricultural Statistics* **61**, 95–108.

Rasmussen, C. & Williams, C. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Redner, R. A. & Walker, H. F. (1984). Mixture densities, maximum likelihood and the em algorithm. *SIAM review* **26**, 195–239.

Ridgeway, G. et al. (2015). *gbm: Generalized Boosted Regression Models*. R package version 2.1.1.

Scarpa, B. & Dunson, D. B. (2009). Bayesian hierarchical functional data analysis via contaminated informative priors. *Biometrics* **65**, 772–780.

Sethuraman, J. (1994). A constructive definition of dirichlet priors. *Statistica Sinica* **4**, 639–650.

Walker, S. G. (2007). Sampling the dirichlet mixture model with slices. *Communications in Statistics – Simulation and Computation* **36**, 45–54.

Wickham, H. & Francois, R. (2015). *dplyr: A Grammar of Data Manipulation*. R package version 0.4.2.