Università degli Studi di Padova
Dipartimento di Scienze Statistiche

Corso di Laurea Magistrale in
Scienze Statistiche

Inferenza bayesiana su modelli mistura di Support Vector Machines

Relatore: Dott. Mauro Bernardi
Correlatore: Dott. Andrea Sottosanti
Dipartimento di Scienze Statistiche

Laureando: Emanuele Degani
Matricola: 1137781

Anno Accademico 2017/2018

Indice

Introduzione

Uno dei campi maggiormente prolifici della letteratura statistica negli ultimi anni è stato indubbiamente quello riguardante il *machine learning* e, più specificamente, l'insieme di tecniche, metodi e modelli per affrontare problemi di classificazione o regressione su insiemi di dati complessi; questo ha trovato vasta applicabilità in numerosi aspetti e problemi, ed è stato in grado di coniugare materie e campi fino a qualche decennio fa molto lontani tra loro utilizzando tecniche di stima ed algoritmi spesso estranei ai paradigmi inferenziali propri della statistica.

Proprio quest'ultimo aspetto ha reso necessario *(ri)portare* la formalizzazione di tecniche di stima, metodi di ottimizzazione numerica ed algoritmi propri del *data mining* in un contesto più familiare agli statistici, per arricchire le stime ottenute con gli strumenti classici dell'inferenza, quali ad esempio intervalli di credibilità/confidenza, test o bande di previsione; il prezzo da pagare per possedere una maggiore qualità informativa sui risultati ottenuti riguarda inevitabilmente la necessità di formalizzare adeguatamente un modello statistico che interpreti il modello di classificazione o regressione in un'ottica probabilistica, a partire dal quale svolgere l'inferenza.

Riguardo il paradigma inferenziale da utilizzare, in questo lavoro si propende per una scelta bayesiana: le motivazioni che vanno a favore di questa sono molteplici e legati principlamente all'opportunità di svolgere l'inferenza in maniera facile ed immediata una volta nota la distribuzione a posteriori del modello; per contro, uno degli aspetti generalmente problematici riguarda proprio la possibilità di ricavare esplicitamente la forma della distribuzione a posteriori, o quantomeno di saper generare da quest'ultima; questo risulta generalmente possibile quando la verosimiglianza del modello è coniugata con la distribuzione a priori, o più realmente quando esiste la possibilità di ricorrere a tecniche di *data augmentation* che consentano di implementare facilmente un algoritmo di Gibbs Sampling.

Nel lavoro proposto l'attenzione verrà posta sulla classe dei modelli *Support Vector Machines*, particolarmente noti per la loro immediata interpretazione geometrica e la facilità nell'implementare algoritmi di stima funzionanti anche su *dataset* dalle dimensioni spesso proibitive: in particolare, in questo lavoro si andrà a considerare una versione la cui funzione obbiettivo comprende un termine di penalizzazione sui coefficienti del modello, che consente di svolgere uno *shrinkage* dei parametri (penalizzazione Ridge) o una selezione delle variabili (penalizzazione Lasso) particolarmente utile per mitigare possibili casi di sovradattamento del modello ai dati.

La possibilità di simulare dalla distribuzione a posteriori del modello e tramite questa svolgere l'inferenza è garantita da uno schema di Gibbs Sampling che sfrutta una rappresentazione aumentata proposta da Polson e Scott (2011) per la pseudo-verosimiglianza del modello: quest'ultima viene ricavata andando ad operare un tecnicismo sulla funzione di perdita del modello cambiata di segno, che le consente di essere assimilata tramite la sua trasformata esponenziale ad una funzione di verosimiglianza non direttamente associabile ad un modello probabilistico, ma valida per l'inferenza.

L'idea sviluppata nel corso di questo lavoro nasce dall'intuizione secondo la quale molto spesso ci si trova a voler affrontare problemi supervisionati di classificazione su dataset in cui i regressori emergono da sottopopolazioni latenti, connotabili da particolari caratteristiche delle osservazioni che vi appartengono o forme di concentramento in *cluster* evidenti esplorativamente, il cui meccanismo generatore è riconducibile in linea teorica alla classe dei modelli mistura parametrici a numero finito di componenti. L'inferenza bayesiana su quest'ultima classe di modelli è particolarmente facile ed immediata, ammette una notevole varietà di schemi di *data augmentation* per la simulazione dalla distribuzione a posteriori e, all'interno di questi, permette di svolgere un opportuno *clustering* delle osservazioni rispetto alla forma distributiva scelta per le componenti del modello.

La classe di modelli proposta in questo lavoro ha l'ambizioso obbiettivo di riuscire a cogliere *cluster* di osservazioni riconducibili a sottopopolazioni latenti che hanno generato il fenomeno e, su ciascuno di questi, stimare una opportuna SVM a seconda delle caratteristiche dei regressori che lo compongono: la stima di questa classe di modelli è interamente conseguibile all'interno dell'impianto inferenziale bayesiano, unendo le tecniche di simulazioni dalla posteriori per le SVM con quelle per la simulazione dalla posteriori per un modello mistura, per il quale supponiamo sia valida la scelta di adattare a ciascuna

componente una distribuzione gaussiana multivariata.

Idealmente il risultato atteso è che se i dati in esame esibiscono caratteristiche di *clustering* sulle variabili considerate come regressori, andare a connotare quali osservazioni appartengono a ciascuno di questi e stimarvi una opportuna SVM possa semplificare il problema e migliorare l'abilità predittiva rispetto ad un generico modello SVM stimato sull'intero insieme di osservazioni: ci si aspetta che quest'ultimo possa rischiare di sovraparametrizzarsi per cogliere la forma di particolari gruppi di osservazioni o di come le modalità della variabile risposta siano disposte al loro interno, con una conseguente crescita del numero di parametri da stimare nel modello.

Un secondo aspetto da tenere in considerazione svolgendo un'applicazione inferenziale di tipo bayesiano a problemi di classificazione riguarda lo sviluppo di una serie di metodologie per la valutazione della qualità previsiva del modello, il più possibile assimilabili agli strumenti già diffusi nei contesti di *machine learning*: all'interno di questo lavoro viene dato ampio spazio di approfondimento al significato bayesiano di *classificazione* di una generica osservazione, e a come è possibile ottenerla per una generica nuova osservazione utilizzando la classe di modelli proposti.

Il lavoro è suddiviso in 5 Capitoli: il Capitolo 1 formalizza i modelli SVM per problemi di classificazione, congiuntamente alle tecniche che consentono di simulare dalla distribuzione a posteriori in un contesto di stima bayesiano; il Capitolo 2 formalizza la classe dei modelli mistura parametrici a numero finito di componenti, congiuntamente alle tecniche di stima ed alle relative problematiche da affrontare rispettivamente nei paradigmi inferenziali frequentista e bayesiano, con una maggiore attenzione a quest'ultimo; il Capitolo 3 formalizza la classe dei modelli mistura a componenti gaussiane multivariate per SVM proposta in questo lavoro, assieme alle tecniche di stima e ad alcuni esempi che aiutano a comprenderne l'utilità; il Capitolo 4 mostra l'applicazione ad un dataset simulato, confrontando la qualità predittiva di un modello SVM *standard* con quella dei modelli proposti; infine, il Capitolo 5 propone come conseguire alcune possibili estensioni della classe di modelli proposti, relativamente alla possibilità di trattare problemi di regressione, andare a modificare la distribuzione delle componenti della mistura sulla quale si svolge il *clustering* delle osservazioni e selezionare opportunamente il numero di cluster latenti (e rispettive SVM) nel modello.

Capitolo 1

Support Vector Machines

Le *support vector machines* (meglio note con il loro acronimo, *SVM*) compongono una classe di modelli per il *machine learning* supervisionato particolarmente diffusa ed apprezzata quando si dispone di insiemi di dati complessi.

In questo capitolo viene percorsa una breve panoramica dei modelli SVM per problemi di classificazione (Sezione 1.1), dando risalto ai fondamenti teorici che ne consentono la formalizzazione ed all'interpretazione geometrica che ne ha favorito il loro sviluppo. Successivamente viene formalizzata la versione bayesiana diffusa in letteratura (1.2), concentrando la trattazione sulle tecniche che consentono una simulazione della distribuzione a posteriori computazionalmente agevole.

1.1 SVM per problemi di classificazione

Le SVM sono state inizialmente battezzate con il nome di *support-vector networks* da Cortes e Vapnik (1995), che ne hanno proposto una prima versione dedicata a problemi di classificazione.

Assumiamo in particolare di disporre di n coppie di dati $\{(y_i, \mathbf{x}_i)\}_i^n$ appartenenti all'insieme di stima, con $\mathbf{x}_i \in \mathbb{R}^p$ e $y_i \in \{-1, 1\}$, dove y_i assume il valore -1 o 1 a seconda della codifica che viene adottata per la modalità assunta dalla variabile risposta: l'intuizione geometrica che ha consentito di sviluppare le SVM prevede l'individuazione dell'iperpiano definito dall'equazione $\left\{\mathbf{x} : f(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\beta} + \beta_0 = 0, \|\boldsymbol{\beta}\| = 1\right\}$ che realizza la migliore discri-

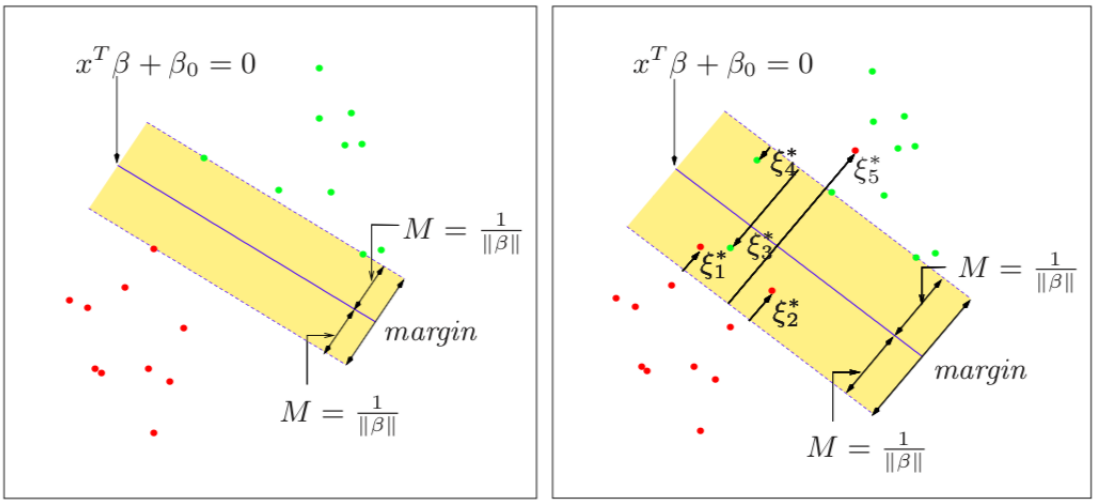


Figura 1.1: Rappresentazione geometrica in $\mathbb{R}^2$ di un SVM per la classificazione: nella figura a sinistra il caso di classi non-sovrapposte, a destra quello di classi sovrapposte. La figura è tratta da Hastie, Tibshirani e Friedman (2009).

minazione geometrica tra le osservazioni $\mathbf{x}_i$ appartenenti ad una o all'altra classe, ovvero tra i due ipersemispazi di $\mathbb{R}^p$ sui quali si posizionano le osservazioni, risolvendo il seguente problema di ottimizzazione:

$$\max_{\boldsymbol{\beta},\beta_0,\|\boldsymbol{\beta}\|=1} M \quad \text{s.v.} \quad y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq M, \quad i = 1, 2, \ldots, n \tag{1.1}$$

che può essere ugualmente riformulato nel seguente problema di ottimizzazione convessa eliminando il vincolo $\|\boldsymbol{\beta}\| = 1$ ed imponendo $M = 1/\|\boldsymbol{\beta}\|$:

$$\min_{\boldsymbol{\beta},\beta_0} \|\boldsymbol{\beta}\| \quad \text{s.v.} \quad y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq 1, \quad i = 1, 2, \ldots, n. \tag{1.2}$$

La Figura 1.1 fornisce una rappresentazione geometrica in $\mathbb{R}^2$ utile a comprendere il fondamento concettuale sul quale si basano questi modelli: concentrandoci per ora sull'immagine a sinistra, la linea azzurra individua l'iperpiano (retta) definito implicitamente dall'equazione $\mathbf{x}^T\hat{\boldsymbol{\beta}} + \hat{\beta}_0 = 0$ che realizza la migliore separazione dei due gruppi di osservazioni (punti rossi, in basso a sinistra, e verdi, in alto a destra), dove $(\hat{\boldsymbol{\beta}}, \hat{\beta}_0)$ è l'unica soluzione della (1.2). La banda colorata di *beige* rappresenta il più ampio *margine* individuato tra l'osservazione più vicina di ciascuna delle due classi, con un'ampiezza pari a $2/\|\hat{\boldsymbol{\beta}}\|$: risolvere il problema (1.2) significa pertanto individuare il vettore $\boldsymbol{\beta}$ a norma euclidea minore che massimizza l'ampiezza del margine tra i due gruppi di osservazioni.

Intuitivamente, un modello di questo tipo va alla ricerca della migliore separazione possibile tra i gruppi di osservazioni appartenenti alle due diverse modalità della variabile risposta: l'iperpiano che realizza il maggior divario definisce una chiara separazione dello spazio $\mathbb{R}^p$ in due ipersemispazi, e dunque la seguente regola di classificazione per la generica osservazione $\tilde{\mathbf{x}}$:

$$G(\tilde{\mathbf{x}}) = \text{sign}\left(\hat{f}(\tilde{\mathbf{x}})\right) = \text{sign}\left(\tilde{\mathbf{x}}^T\hat{\boldsymbol{\beta}} + \hat{\beta}_0\right). \tag{1.3}$$

Quanto appena descritto circa l'individuazione dell'iperpiano che realizza la separazione ottima tra le classi richiede comunque un'estensione ai casi più comuni e statisticamente più rilevanti, ovvero quando i due gruppi di osservazioni sono generalmente sovrapposti: in questo caso l'iperpiano che realizza la migliore separazione non è individuabile, ovvero i due gruppi di osservazioni non sono linearmente separabili.

Un modo per tenere conto di questi casi (i più diffusi) prevede la possibilità di ammettere che alcune osservazioni giacciano nell'ipersemispazio *sbagliato*: definiamo pertanto le variabili ausiliarie $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)$ che misurano la distanza di ciascuna osservazione dall'estremo del margine relativo al gruppo di appartenenza qualora questa giaccia all'interno della banda o nell'ipersemispazio sbagliato, e pari a 0 per tutte le osservazioni posizionate correttamente. Il problema di ottimizzazione definito dalla (1.1) può essere modificato per tenere conto di questa estensione nel seguente modo:

$$\max_{\boldsymbol{\beta},\beta_0,\|\boldsymbol{\beta}\|=1} M \quad \text{s.v.} \quad \begin{array}{l} y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq M(1-\xi_i), \\ \xi_i \geq 0, \quad \sum_{i=1}^{n} \xi_i \leq \gamma \end{array}, \quad i = 1, 2, \ldots, n. \tag{1.4}$$

Il valore ξ_i all'interno del vincolo è una quantità proporzionale a *quanto* l'osservazione $\mathbf{x}_i$ giace dal lato opposto dell'estremo del margine relativo al proprio gruppo di appartenenza: imponendo che $\sum_{i=1}^{n} \xi_i \leq \gamma$ per un prefissato valore di γ stiamo di fatto imponendo un limite superiore alla proporzione totale di osservazioni che giacciono nel lato opposto definito dal margine; dato che la scorretta classificazione avviene quando $\xi_i > 1$, il vincolo implicitamente impone che al più possano essere classificate erroneamente γ osservazioni dell'insieme di stima. Sempre facendo riferimento alla Figura 1.1, la cornice a destra illustra in $\mathbb{R}^2$ l'idea appena descritta.

Come già fatto per il caso di osservazioni linearmente separabili, la (1.4) può essere riscritta eliminando il vincolo $\|\boldsymbol{\beta}\| = 1$, giungendo al seguente problema di ottimizzazione:

$$\min_{\boldsymbol{\beta},\beta_0} \|\boldsymbol{\beta}\| \quad \text{s.v.} \quad \begin{array}{l} y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i, \\ \xi_i \geq 0, \quad \sum_{i=1}^{n} \xi_i \leq \gamma \end{array}, \quad i = 1, 2, \ldots, n. \tag{1.5}$$

La regola di classificazione coincide con la (1.3), ma rispetto al caso di osservazioni linearmente separabili cambia il problema di ottimizzazione da dover risolvere per ottenere le stime $(\hat{\boldsymbol{\beta}}, \hat{\beta}_0)$.

Un'ulteriore estensione riguarda la possibilità di allargare lo spazio $\mathbb{R}^p$ delle osservazioni $\mathbf{x}_i$ utilizzando espansioni di base del tipo $h_m(\mathbf{x}) : \mathbb{R}^p \longmapsto \mathbb{R}$, come ad esempio polinomi o *splines*: generalmente gli iperpiani separatori individuati in questi spazi aumentati si riflettono in separazioni non-lineari nello spazio $\mathbb{R}^p$ originale di partenza. Rimanendo volutamente fedeli alla letteratura del *machine learning*, definiamo *support-vector machines* per la classificazione tutti i modelli che accolgono un'estensione od una restrizione dello spazio di partenza dei regressori, talvolta giungendo a spazi estesi dalle dimensioni parecchio elevate o a spazi ristretti dalle dimensioni volutamente inferiori.

Definito $h(\mathbf{x}_i) = (h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \ldots, h_M(\mathbf{x}_i))$ il generico vettore delle covariate relative all'osservazione $\mathbf{x}_i$, con M che può essere minore, uguale maggiore a p, la procedura di stima è la stessa descritta nelle pagine precedenti: per questa ragione, nonostante ora si consideri $h(\mathbf{x}_i) \in \mathbb{R}^M$ anzichè $\mathbf{x}_i \in \mathbb{R}^p$, si parla generalmente ed in modo ambivalente, senza fare alcuna specifica distinzione, di *support-vector machines* in entrambi i casi: la regola di previsione sarà, ancora una volta, $G(\tilde{\mathbf{x}}) = \text{sign}\left(\hat{f}(\tilde{\mathbf{x}})\right) = \text{sign}\left(h(\tilde{\mathbf{x}})^T\hat{\boldsymbol{\beta}} + \hat{\beta}_0\right)$.

Seguendo quanto descritto da Hastie, Tibshirani e Friedman (2009) circa la risoluzione di problemi di ottimizzazione convessa, è possibile esprimere la soluzione del problema (1.5) come

$$\hat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i h(\mathbf{x}_i) \tag{1.6}$$

con $\hat{\alpha}_i \neq 0$ solo per le osservazioni i che rispettano la condizione $y_i\left(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0\right) = 1 - \xi_i$, note in letteratura con il nome di *support vectors* (da cui il nome *support vector machines*).

Questa espressione, inserita all'interno della funzione dell'iperpiano separatore $f(\mathbf{x}) = h(\mathbf{x})^T\boldsymbol{\beta} + \beta_0$ consente di esprimere la soluzione come

$$\hat{f}(\mathbf{x}) = \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\alpha}_i y_i \langle h(\mathbf{x}), h(\mathbf{x}_i)\rangle, \tag{1.7}$$

dove $\langle h(\mathbf{x}), h(\mathbf{x}_i)\rangle = h(\mathbf{x})^T h(\mathbf{x}_i)$ corrisponde al ben noto operatore *prodotto interno*.

Notando che le osservazioni $\mathbf{x}_i$ entrano all'interno della (1.7) solo tramite i prodotti interni $\langle\cdot,\cdot\rangle$, la specificazione delle funzioni di base $h(\mathbf{x})$ può avvenire tramite la definizione di opportune *funzioni kernel* del tipo $K(\mathbf{x}, \mathbf{x}') = \langle h(\mathbf{x}), h(\mathbf{x}')\rangle$. Le più diffuse ed utilizzate sono sintetizzate nella seguente tabella, per valori di c, κ_1 e κ_2 opportunamente scelti.

Tipologia	$K(\mathbf{x}, \mathbf{x}')$
Polinomiale	$(1 + \langle \mathbf{x}, \mathbf{x}'\rangle)^d$
Base radiale	$\exp(-\|\mathbf{x} - \mathbf{x}'\|^2/c)$
Sigmoide	$\tanh(\kappa_1\langle \mathbf{x}, \mathbf{x}'\rangle + \kappa_2)$

La (1.7) può pertanto essere riscritta, nel suo caso più generale, come

$$\hat{f}(\mathbf{x}) = \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\alpha}_i y_i K(\mathbf{x}, \mathbf{x}_i) \tag{1.8}$$

e la classificazione della generica osservazione $\tilde{\mathbf{x}}$ avverrà pertanto secondo la seguente regola di classificazione:

$$G(\tilde{\mathbf{x}}) = \text{sign}\left(\hat{f}(\tilde{\mathbf{x}})\right) = \text{sign}\left(\hat{\beta}_0 + \sum_{i=1}^{n} \hat{\alpha}_i y_i K(\tilde{\mathbf{x}}, \mathbf{x}_i)\right). \tag{1.9}$$

Di particolare interesse per il lavoro svolto in questa tesi è la rappresentazione delle SVM per la classificazione come metodi di penalizzazione, ovvero la possibilità di risolvere il problema di ottimizzazione (1.5) minimizzando rispetto a $\boldsymbol{\beta}$ e β_0 la seguente funzione obbiettivo

$$d^c(\boldsymbol{\beta}, \beta_0, \lambda) = \sum_{i=1}^{n} \max(1 - y_i f(\mathbf{x}_i), 0) + \frac{\lambda}{2}\|\boldsymbol{\beta}\|^2 \tag{1.10}$$

dove il primo addendo rappresenta la funzione di perdita *hinge loss* ed il secondo il termine di penalizzazione: fissando $\lambda = 1/(2\gamma)$, la soluzione del problema coincide con quella

per il (1.5). Questo ci consente di rappresentare le SVM come un problema di stima di funzioni penalizzate, dove i coefficienti di $f(\mathbf{x}) = \beta_0 + h(\mathbf{x})^T\boldsymbol{\beta}$ sono schiacciati verso lo zero. A seconda delle esigenze è possibile fare scelte alternative all'interno della classe delle penalizzazioni di tipo $\mathscr{L}_\alpha$, $\alpha \in (0, 2)$: un caso noto in letteratura riguarda la penalizzazione Lasso ($\alpha = 1$), particolarmente utile quando si intende svolgere una selezione delle variabili fissando a 0 alcuni dei parametri inclusi nella specificazione del modello.

1.2 Inferenza bayesiana su SVM per la classificazione

Un recente ambito di ricerca particolarmente prolifico riguarda la revisitazione dei principali metodi e modelli per il *machine learning* in ottica probabilistica, andando alla ricerca di opportune metodologie che ne consentano la formalizzazione in un contesto bayesiano.

Per quanto riguarda le SVM per problemi di classificazione, il lavoro di Polson e Scott (2011) va certamente in questa direzione, sviluppando una rappresentazione a variabili latenti di un'estensione della forma penalizzata richiamata nella (1.10) che ammette penalizzazioni di tipo $\mathscr{L}_\alpha$.

Si consideri nuovamente n coppie $\{(y_i, \mathbf{x}_i)\}_i^n$ appartenenti all'insieme di stima, con $y_i \in \{-1, 1\}$ e $\mathbf{x}_i = (1, x_{i1}, x_{i2}, \ldots, x_{i\,p-1}) \in \mathbb{R}^p$, dove per evitare di complicare la notazione assumiamo che $\mathbf{x}_i$ includa già l'intercetta ed eventuali espansioni di base del tipo $h(\mathbf{x}_i)$.

La stima di $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)$ avviene minimizzando la seguente funzione obbiettivo

$$d_\alpha^c(\boldsymbol{\beta}, \nu) = \sum_{i=1}^{n} \max(1 - y_i\mathbf{x}_i^T\boldsymbol{\beta}, 0) + \frac{1}{\nu^\alpha}\sum_{j=1}^{p}\left|\frac{\beta_j}{\sigma_j}\right|^\alpha; \tag{1.11}$$

rispetto alla (1.10), questa ammette maggiore flessibilità introducendo come già detto la possibilità di scegliere il grado di penalizzazione $\alpha \in (0, 2]$, e riscalando opportunamente ciascun addendo della penalizzazione per la *standard deviation* empirica σ_j del predittore $\mathbf{x}_j$ associato (con l'eccezione di $\sigma_1 = 1$ per il termine di intercetta).

La trasformata esponenziale applicata alla (1.11) cambiata di segno non apporta alcuna modifica sostanziale al problema di ottimizzazione, ma ci consente di interpretarlo in

un'ottica bayesiana:

$$\exp(-d^c_\alpha(\boldsymbol{\beta},\nu)) \propto \exp\left(-\sum_{i=1}^{n}\max(1-y_i\mathbf{x}_i^T\boldsymbol{\beta},0)\right)\exp\left(-\frac{1}{\nu^\alpha}\sum_{j=1}^{p}\left|\frac{\beta_j}{\sigma_j}\right|^\alpha\right)$$
$$\propto \underbrace{\left(\prod_{i=1}^{n}\exp\left(-\max(1-y_i\mathbf{x}_i^T\boldsymbol{\beta},0)\right)\right)}_{\text{Pseudo-Verosimiglianza}}\underbrace{\left(\prod_{j=1}^{p}\exp\left(-\frac{1}{\nu^\alpha}\left|\frac{\beta_j}{\sigma_j}\right|^\alpha\right)\right)}_{\text{Distribuzione a priori}}. \quad (1.12)$$

Minimizzare la (1.11) corrisponde quindi, in termini probabilistici, ad andare a massimizzare la pseudo-distribuzione a posteriori definita come

$$p(\boldsymbol{\beta}|\nu,\alpha,\mathbf{y}) \propto C_\alpha(\nu)\boldsymbol{\mathcal{L}}(\mathbf{y}|\boldsymbol{\beta})p(\boldsymbol{\beta}|\alpha,\nu), \quad (1.13)$$

ovvero ad individuarne il suo MaP. Quest'ultima può essere espressa come il prodotto di una costante di normalizzazione $C_\alpha(\nu)$, una distribuzione a priori $p(\boldsymbol{\beta}|\alpha,\nu) = \prod_{j=1}^{p} p(\beta_j|\alpha,\nu)$ su $\boldsymbol{\beta}$ ed una pseudo-verosimiglianza $\boldsymbol{\mathcal{L}}(\mathbf{y}|\boldsymbol{\beta}) = \prod_{i=1}^{n}\boldsymbol{\mathcal{L}}(y_i|\boldsymbol{\beta})$, come mostrato nella (1.12).

E' bene comunque sottolineare come la pseudo-verosimiglianza ricavata sia più un'artificio tecnico per svolgere l'inferenza che una vera e propria funzione di verosimiglianza: all'interno della classe dei modelli SVM non esiste infatti alcuna ipotesi probabilistica circa la distribuzione congiunta di $(y,\mathbf{x})$, proprio perchè il modello è formalizzato esclusivamente tramite considerazioni di natura geometrica.

Dal punto di vista bayesiano, la conoscenza della distribuzione a posteriori, e la possibilità di generare da essa, è quanto necessario per fare inferenza: nel caso in esame, e in realtà in tutti quelli in cui la distribuzione a priori non è coniugata alla famiglia di distribuzioni scelta per modellare il fenomeno di interesse, ottenere la distribuzione a posteriori è spesso complicato e richiede sforzi computazionali non indifferenti, così come la possibilità di generare dalla stessa. Quel che si usa comunemente fare in questi casi è implementare degli algoritmi in grado di simulare delle catene markoviane ergodiche (irriducibili, aperiodiche e ricorrenti positive) che evolvono secondo prefissati kernel di transizione fino a raggiungere la convergenza: una volta raggiunta, il tracciato della catena da lì in avanti, privato della componente di sequenzialità temporale, si assume generato dalla distribuzione a posteriori di interesse.

Di particolare interesse in questo contesto sono gli algoritmi di Gibbs Sampling (o di Gibbs-with-Metropolis Sampling), che all'interno dei metodi Markov Chain Monte Carlo

(MCMC) permettono una più rapida ed agevole generazione della transizione della catena sfruttando la possibilità di ricavare una forma distributiva nota e dalla quale è immediato generare per la *posterior full-conditional* di ciascun parametro, ovvero dalla distribuzione a posteriori di ciascun parametro condizionata a tutti gli altri ed ai dati a disposizione. In generale, a meno di precise e mirate scelte delle distribuzioni a priori, è raro riuscire ad individuare degli algoritmi di Gibbs Sampling per i quali le *full-conditional* si possono ricavare agilmente; nella maggior parte dei casi in cui è possibile simulare dalla distribuzione a posteriori tramite un Gibbs Sampler, è necessario ricorrere a tecniche di *data augmentation* per poterlo fare.

Il lavoro di Polson e Scott (2011) risulta particolarmente utile in questo senso, perchè fornisce una rappresentazione a variabili latenti che consente di ricavare un Gibbs Sampler con il quale simulare l'intera distribuzione a posteriori (1.13). In particolare, propongono una rappresentazione a variabili latenti per la verosimiglianza ed una per il termine di penalizzazione che agevolano la possibilità di ricavare le *full-conditionals* in forma chiusa.

Il primo risultato sviluppato nel lavoro riguarda la possibilità di esprimere l'i-esimo contributo alla pseudo-verosimiglianza $\mathcal{L}_i(y_i|\boldsymbol{\beta})$ come una mistura di posizione e scala di *kernel* gaussiani, introducendo una variabile latente λ_i che entra sia come parametro di posizione che come parametro di scala nel singolo termine, ammettendo la possibilità per $\boldsymbol{\beta}$ di apparire all'interno di una forma quadratica:

$$\begin{aligned}
\mathcal{L}_i(y_i|\boldsymbol{\beta}) &= \exp\left(-2\max(1 - y_i\mathbf{x}_i^T\boldsymbol{\beta}, 0)\right) \\
&= \int_0^\infty \frac{1}{\sqrt{2\pi\lambda_i}} \exp\left(-\frac{1}{2}\frac{(1 + \lambda_i - y_i\mathbf{x}_i^T\boldsymbol{\beta})}{\lambda_i}\right) d\lambda_i \\
&= \int_0^\infty \mathsf{N}\left(1 - y_i\mathbf{x}_i^T\boldsymbol{\beta}; -\lambda_i, \lambda_i\right)\, d\lambda_i
\end{aligned} \tag{1.14}$$

dove con $\mathsf{N}(y; \mu, \sigma^2)$ indichiamo impropriamente la densità di una v.c. $Y \sim \mathsf{N}(\mu, \sigma^2)$. La dimostrazione del risultato è contenuta all'interno di Polson e Scott (2011).

Il secondo risultato riportato all'interno del lavoro è dovuto a West (1987) e consente di scrivere il singolo addendo del termine di penalizzazione $p(\beta_j|\nu, \alpha)$, corrispondente ad una distribuzione gaussiana generalizzata (*exponential power*), come mistura di scala di *kernel* gaussiani, introducendo una variabile latente ω_j che entra come parametro di scala

nel singolo termine:

$$p(\beta_j|\nu,\alpha) = \frac{\alpha}{\nu\Gamma(\alpha^{-1})}\exp\left(-\left|\frac{\beta_j}{\nu\sigma_j}\right|^{\alpha}\right)$$
$$= \int_0^{\infty} \mathsf{N}(\beta_j; 0, \nu^2\omega_j\sigma_j^2)\, p(\omega_j|\alpha)\, d\omega_j. \tag{1.15}$$

A differenza del risultato precedente, qui si introduce una specifica distribuzione misturizzante per ω_j, che nel nostro caso corrisponde a $p(\omega_j|\alpha) \propto \omega_j^{-3/2}\,\mathsf{St}^+_{\alpha/2}(\omega_j^{-1})$, dove $\mathsf{St}^+_{\alpha/2}(\cdot)$ indica la densità di una v.c. *stabile* (spesso nota come *Levy* α*-stabile*) positiva di indice $\alpha/2$. Di norma, la famiglia delle distribuzioni stabili positive non ammette la possibilità di esprimere in maniera esplicita la funzione di densità (se non ricorrendo ad una trasformata di Fourier della sua funzione caratteristica), rendendola nella pratica difficile da maneggiare. Esistono però due casi particolari che consentono di superare questa problematica facilmente: il primo corrisponde alla penalizzazione di tipo Lasso ($\alpha = 1$), che ci consente di ricondurre la $\mathsf{St}^+_{1/2}(\cdot)$ ad una $\mathsf{Levy}\,(0,1)$, e dunque $\omega_j|\alpha \sim \mathsf{Exp}\,(1/2)$; il secondo corrisponde alla penalizzazione di tipo Ridge ($\alpha = 2$), per il quale la $p(\omega_j|\alpha)$ è degenere in 1.

I due risultati appena mostrati ci consentono di esprimere la pseudo-posteriori (1.13) come marginale di una pseudo-posteriori *aumentata* e comprensiva delle variabili latenti $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ e $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_p)$:

$$p(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\omega}|\mathbf{y}, \alpha, \nu) \propto \left(\prod_{i=1}^{n} \lambda_i^{-1/2}\right) \exp\left(-\frac{1}{2}\sum_{i=1}^{n}\frac{(1 - y_i\mathbf{x}_i^T\boldsymbol{\beta} - \lambda_i)^2}{\lambda_i}\right)$$
$$\times \left(\prod_{j=1}^{p} \omega_j^{-1/2} p(\omega_j|\alpha)\right) \exp\left(-\frac{1}{2\nu^2}\sum_{j=1}^{p}\frac{\beta_j^2}{\sigma_j^2\omega_j}\right) \tag{1.16}$$

ovvero, in forma compatta:

$$p(\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\omega}|\mathbf{y}, \alpha, \nu) \propto \mathsf{N_n}(\mathbf{1}_n - \boldsymbol{\lambda}; \tilde{\mathbf{X}}\boldsymbol{\beta}, \boldsymbol{\Lambda}) \times \mathsf{N_p}(\boldsymbol{\beta}; \mathbf{0}, \nu^2\boldsymbol{\Omega}^{1/2}\boldsymbol{\Sigma}\boldsymbol{\Omega}^{1/2}) \times \prod_{j=1}^{p} p(\omega_j|\alpha) \tag{1.17}$$

dove $\tilde{\mathbf{X}} = \mathsf{diag}(\mathbf{y})\mathbf{X}$, $\boldsymbol{\Lambda} = \mathsf{diag}(\boldsymbol{\lambda})$, $\boldsymbol{\Omega} = \mathsf{diag}(\boldsymbol{\omega})$ e $\boldsymbol{\Sigma} = \mathsf{diag}(\boldsymbol{\sigma^2})$.

A partire da questa rappresentazione, è possibile ottenere le *full-conditional* a posteriori per $\boldsymbol{\beta}$ e per le variabili latenti, utilizzabili per implementare un opportuno Gibbs Sampler per la simulazione della distribuzione a posteriori sul modello aumentato.

Full-conditional a posteriori per $\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}, \boldsymbol{\lambda}, \boldsymbol{\omega}$:

Per ricavare la distribuzione a posteriori $p(\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}, \boldsymbol{\lambda}, \boldsymbol{\omega})$ si osserva la forma della (1.17): escludendo i termini che non dipendono da $\boldsymbol{\beta}$, questa può essere ricondotta alla distribuzione a posteriori di un modello di regressione lineare ad errori eteroschedastici, con $\mathbf{1}_n - \boldsymbol{\lambda} \sim \mathsf{N_n}(\tilde{\mathbf{X}}\boldsymbol{\beta}, \boldsymbol{\Lambda})$ e distribuzione a priori sui coefficienti $\boldsymbol{\beta} \sim \mathsf{N_p}(\mathbf{0}, \nu^2\boldsymbol{\Omega}^{1/2}\boldsymbol{\Sigma}\boldsymbol{\Omega}^{1/2})$.

Ricorrendo ai risultati teorici circa la forma della distribuzione a posteriori per un modello di regressione lineare ad errori omoschedastici, la *full-conditional* diventa

$$\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}, \boldsymbol{\lambda}, \boldsymbol{\omega} \sim \mathsf{N_p}(\mathbf{b}, \mathbf{B}) \tag{1.18}$$

dove $\mathbf{B}^{-1} = \nu^{-2}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Omega}^{-1} + \tilde{\mathbf{X}}^T\boldsymbol{\Lambda}^{-1}\tilde{\mathbf{X}}$ e $\mathbf{b} = \mathbf{B}\tilde{\mathbf{X}}^T(\mathbf{1}+\boldsymbol{\lambda}^{-1})$, con $\boldsymbol{\lambda}^{-1} = (1/\lambda_1, 1/\lambda_2, \ldots, 1/\lambda_n)$.

Full-conditional a posteriori per $\lambda_i|y_i, \mathbf{x}_i, \boldsymbol{\beta}$:

Polson e Scott (2011) mostrano che la *full-conditional* per λ_i dipende solo da $\boldsymbol{\beta}$, y_i e $\mathbf{x}_i$, e corrisponde a

$$\lambda_i|\boldsymbol{\beta}, y_i \sim \mathsf{GenInvGaussian}\left(\frac{1}{2}, 1, (1 - y_i\mathbf{x}_i^T\boldsymbol{\beta})^2\right) \tag{1.19}$$

o, equivalentemente,

$$\lambda_i^{-1}|\boldsymbol{\beta}, y_i \sim \mathsf{InvGaussian}\left(\left|1 - y_i\mathbf{x}_i^T\boldsymbol{\beta}\right|^{-1}, 1\right). \tag{1.20}$$

Full-conditional a posteriori per $\omega_j|\beta_j, \nu, \boldsymbol{\Sigma}, \alpha$:

La *full-conditional* per ω_j è proporzionale all'integranda della (1.15), ma in generale questa risulta essere una distribuzione particolarmente complicata dalla quale generare, per le ragioni esposte in precedenza circa la difficoltà di scrivere esplicitamente la funzione di densità per una distribuzione stabile positiva. Nel caso Lasso ($\alpha = 1$) però, facendo riferimento al risultato secondo il quale $\omega_j|\alpha = 1 \sim \mathsf{Exp}\,(1/2)$, è possibile ricavare esplicitamente tale distribuzione:

$$\omega_j|\beta_j, \nu, \alpha = 1 \sim \mathsf{GenInvGaussian}\left(\frac{1}{2}, 1, \frac{\beta_j^2}{\nu^2\sigma_j^2}\right) \tag{1.21}$$

o, equivalentemente,

$$\omega_j^{-1}|\beta_j, \nu, \alpha = 1 \sim \mathsf{InvGaussian}\left(\frac{\nu\sigma_j}{|\beta_j|}, 1\right). \tag{1.22}$$

Ancor più immediato è il caso Ridge ($\alpha = 2$): in questo caso infatti la full-conditional risulta essere degenere in 1.

Full-conditional a posteriori per $\nu^{-1}|\boldsymbol{\beta}, \boldsymbol{\Sigma}, a_\nu, b_\nu$:

Polson e Scott (2011) hanno ammesso anche la possibilità di trattare ν^{-1} come variabile casuale, e dunque di simularla al pari degli altri parametri: la proposta è quella di fissare una distribuzione a priori $\nu^{-1} \sim \mathsf{Gamma}(a_\nu, b_\nu)$, che è coniugata alla distribuzione gaussiana generalizzata della (1.15) e va debitamente aggiunta alla (1.17), ottenendo a posteriori

$$\nu^{-1}|\boldsymbol{\beta} \sim \mathsf{Gamma}\left(a_\nu + \frac{p}{\alpha}, b_\nu + \sum_{j=1}^{p}\left|\frac{\beta_j}{\sigma_j}\right|\right). \tag{1.23}$$

L'alternativa alla scelta bayesiana di assegnare una distribuzione di probabilità al peso della penalizzazione ν^{-1} è quella di trattarlo come un iperparametro, dunque fisso per tutte le iterazioni del Gibbs Sampler. In questo caso, la selezione del suo valore ottimale avviene combinando il Gibbs Sampler per la simulazione della distribuzione a posteriori con una tecnica di *cross-validation*, individuando una opportuna misura di bontà di adattamento sulla quale svolgere i confronti; questo metodo, seppur più vicino alle usuali tecniche di *tuning* dei parametri diffuse nell'ambito del *data mining*, è tanto più computazionalmente oneroso quanto più si infittisce l'intervallo di valori sui quali si crossvalida.

Un'implementazione in codice del Gibbs Sampler per generare dalla distribuzione a posteriori è riportata nel Codice A.4: quest'ultimo è stato scritto per misture gaussiane multivariate di SVM per problemi di classificazione (trattate nel Capitolo 3) ma consente la stima bayesiana di SVM standard fissando $K = 1$.

A titolo di esempio, utilizzando la libreria `penalizedSVM` sviluppata da Becker et al. (2009) che implementa la stima di SVM penalizzate andando a minimizzare la (1.11), è stato simulato un insieme di stima di esempio con $n = 500$ osservazioni ed un insieme di verifica con $n = 250$ osservazioni. Ciascuno è composto da una variabile risposta (bilanciata, sulla quale si intende svolgere la classificazione) ed altre 30 variabili esplicative, delle quali solo le prime 10 sono state generate in modo da essere *significanti* nel discriminare le due modalità

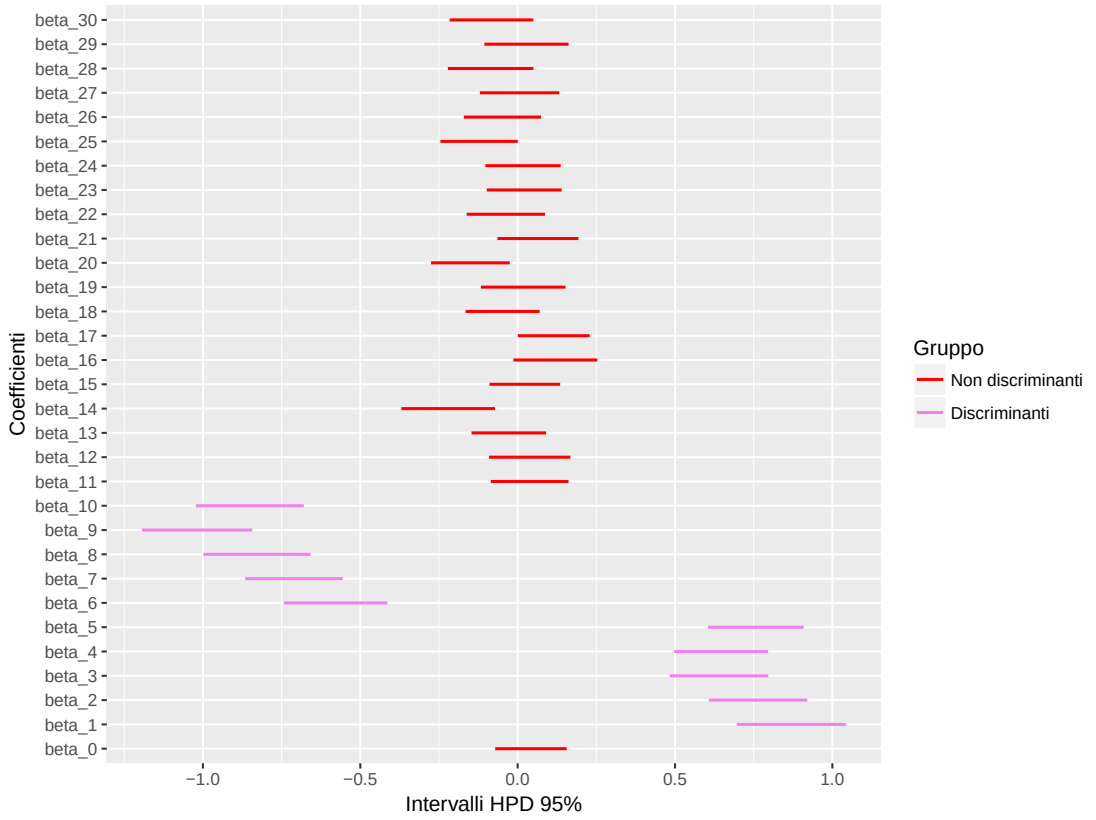


Figura 1.2: Intervalli HPD a livello 95% costruiti sulla distribuzione a posteriori di ciascun coefficiente β_j, $j = 0, 1, 2, \ldots, 30$.

della variabile risposta. Sull'insieme di stima è stato dunque implementato l'algoritmo di Gibbs Sampling su $R = 10^4$ iterazioni, scartando la prima metà di *burnin* ed ottenendo una stima puntuale bayesiana dei coefficienti $\hat{\beta}_j^B$ calcolando la mediana della distribuzione a posteriori (generata) per ciascun coefficiente (nei Capitoli successivi verranno mostrate tecniche più sofisticate per il calcolo della matrice di confusione in ambito bayesiano).

Sull'insieme di verifica è stata poi costruita la classica tabella di confusione, che ha riportato un tasso di corretta classificazione pari all'83.27%. Il codice eseguito per ottenere i risultati, interamente replicabile, è riportato nel Codice B.1. I traceplot delle catene, che non riporto per questioni di spazio, assicurano la convergenza della catena su tutte le componenti.

Il grafico riportato in Figura 1.2 conferma che l'algoritmo implementato converge su stime dei coefficienti *corrette*. Ciascun segmento rappresenta l'intervallo HPD a copertura

95% costruito sulla distribuzione a posteriori di ciascun β_j: in violetto sono riportati quelli per i coefficienti associati alle prime 10 variabili del dataset (il segmento in rosso più basso è relativo all'intercetta β_0), mentre quelli rossi sono associati alle restanti 20 variabili del dataset, generate in modo che non fossero in grado di discriminare la variabile risposta. Questi ultimi contengono quasi tutti lo 0 (i pochi che non lo contengono hanno estremi sinistri o destri che lo sfiorano), a significare che il loro contributo è pressochè irrilevante nel discriminare le due modalità della variabile risposta, come ci si doveva aspettare; i segmenti violetti, al contrario, sono tutti ben distanti dallo 0.

Globalmente il modello sembra comportarsi piuttosto bene e lascia ampio margine di miglioramento; per quanto riguarda l'inferenza, l'impostazione bayesiana ha il vantaggio di produrre una misura di incertezza sulle stime $\hat{\boldsymbol{\beta}}$ a differenza di tutti gli altri algoritmi di stima delle SVM, che ricavano solo stime puntuali.

Capitolo 2

Modelli mistura parametrici a numero finito di componenti

In questo capitolo vengono presentati i modelli parametrici a mistura finita e gli associati approcci inferenziali: nella Sezione 2.1 si fornisce una trattazione teorica generale per questa classe di modelli, mentre nelle sezioni successive vengono richiamate le procedure di inferenza individuate nel paradigma frequentista (Sezione 2.2) ed in quello bayesiano (Sezione 2.3): quest'ultima viene approfondita nelle Sottosezioni 2.3.1 e 2.3.2 con le più diffusa classi di modelli a mistura finita, quelle a componenti gaussiane rispettivamente univariate e multivariate.

2.1 Modelli parametrici a mistura finita

I modelli mistura a numero finito di componenti (detti anche *a mistura finita*, o più comunemente *misture*) assumono un ruolo particolarmente rilevante nell'ambito della modellazione statistica di fenomeni sconosciuti perchè consentono di approssimare l'ignota distribuzione dalla quale sono stati generati i dati tramite una combinazione lineare convessa di distribuzioni che si ipotizza caratterizzino sottopolazioni latenti del fenomeno di interesse; i modelli *parametrici* a mistura finita scelgono, in particolare, di assegnare a ciascuna componente della combinazione lineare una forma parametrica appropriata per la natura del fenomeno oggetto di studio. Una trattazione sufficientemente approfondita e

rigorosa di questa classe di modelli è contenuta nei capitoli iniziali del Frühwirth-Schnatter (2006): nelle prossime pagine ci si limiterà ad illustrarne le caratteristiche essenziali, concentrando l'attenzione sulle metodologie inferenziali appartenenti ai paradigmi frequentista e bayesiano, con un maggiore occhio di riguardo a quest'ultimo.

Si consideri una variabile casuale $\mathbf{Y}$ a valori in $\mathcal{Y} \subset \mathbb{R}^r$, discreta o continua, univariata ($r = 1$) o multivariata ($r > 1$): diremo che $\mathbf{Y}$ proviene da una distribuzione mistura finita con K componenti se la funzione di densità $p(\mathbf{y})$ associata può essere scritta come

$$p(\mathbf{y}) = \pi_1 p_1(\mathbf{y}) + \pi_2 p_2(\mathbf{y}) + \cdots + \pi_K p_K(\mathbf{y}) = \sum_{j=1}^{K} \pi_j p_j(\mathbf{y}) \tag{2.1}$$

dove $p_j(\mathbf{y})$ è la funzione di densità associata alla j-esima componente della mistura, K è il numero di componenti e $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_K)$ è il vettore dei pesi, a valori nel simplesso unitario $\mathcal{E}_K \subset (0,1)^K$ soggetto ai vincoli $\pi_j \geq 0, \forall j$ e $\pi_1 + \pi_2 + \cdots + \pi_K = 1$.

Nel caso delle misture parametriche, come già detto, si ipotizza che le componenti $p_j(\mathbf{y})$ siano indicizzate da un parametro $\boldsymbol{\theta} \in \Theta$: in particolare, nella maggior parte dei casi si assume che tutte le componenti della mistura provengano dalla stessa famiglia parametrica di distribuzioni con densità $p(\mathbf{y}|\boldsymbol{\theta})$, ovvero che $p_j(\mathbf{y}) \equiv p_j(\mathbf{y}|\boldsymbol{\theta}_j), \forall j$: questo permette di esprimere la funzione di densità della distribuzione mistura finita con K componenti come

$$p(\mathbf{y}) \equiv p(\mathbf{y}|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K, \boldsymbol{\pi}) = \pi_1 p_1(\mathbf{y}|\boldsymbol{\theta}_1) + \cdots + p_K(\mathbf{y}|\boldsymbol{\theta}_K) = \sum_{j=1}^{K} \pi_j p(\mathbf{y}|\boldsymbol{\theta}_j). \tag{2.2}$$

I modelli parametrici a mistura finita ammettono una notevole flessibilità, particolarmente consona quando la distribuzione empirica dei dati $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$ esibisce caratteristiche di multimodalità, asimmetria, eccessiva curtosi o forme particolari; di contro, però, le procedure di inferenza potrebbero essere particolarmente complicate e computazionalmente onerose. Un'ulteriore aspetto da tenere in considerazione riguarda la scelta del numero K di componenti della mistura: nonostante esistano metodi e criteri informativi per la scelta di un valore opportuno, questo viene generalmente fissato e supposto noto.

Una caratteristica di questa classe di modelli da tenere in considerazione quando si intende svolgere l'inferenza riguarda l'invarianza della funzione di verosimiglianza del modello sotto permutazioni delle sue componenti, più nota in letteratura con il termine di *label switching problem*, inizialmente studiata da Redner e Walker (1984) e sulla quale si è concentrata una buona dose di letteratura sia in ambito frequentista che bayesiano.

Definito $\mathcal{P}^K$ l'insieme di tutte le possibili $K!$ permutazioni delle etichette associate alle classi $\{1, 2, \ldots, K\}$ che compongono un generico modello mistura, e considerate due generiche e distinte permutazioni $\rho, \eta \in \mathcal{P}^K$, è immediato riconoscere che la proprietà commutativa sull'addizione e sulla moltiplicazione causano la sequente identità sulle funzioni di densità (o equivalentemente, di verosimiglianza) che descrivono il modello:

$$p(\mathbf{y}_1, \ldots, \mathbf{y}_n | \rho(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K, \boldsymbol{\pi})) = p(\mathbf{y}_1, \ldots, \mathbf{y}_n | \eta(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K, \boldsymbol{\pi})) \tag{2.3}$$

ovvero

$$\prod_{i=1}^{n} \left(\sum_{j=1}^{K} \pi_{\rho(j)} p(\mathbf{y}_i | \boldsymbol{\theta}_{\rho(j)}) \right) = \prod_{i=1}^{n} \left(\sum_{j=1}^{K} \pi_{\eta(j)} p(\mathbf{y}_i | \boldsymbol{\theta}_{\eta(j)}) \right) \tag{2.4}$$

dove $\rho\left(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K, \boldsymbol{\pi}\right) = \left(\boldsymbol{\theta}_{\rho(1)}, \ldots, \boldsymbol{\theta}_{\rho(K)}, \pi_{\rho(1)}, \ldots \pi_{\rho(K)}\right)$ e $\eta\left(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K, \boldsymbol{\pi}\right)$ rappresentano simbolicamente le due permutazioni del vettore dei parametri. In statistica, questa caratteristica rende un modello *non identificabile*: non esiste infatti alcuna mappa biunivoca del tipo $\varphi : \Theta \times \mathcal{E}_K \longrightarrow p(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\pi}) \in \mathcal{F}$, dove $\mathcal{F}$ rappresenta la famiglia di distribuzioni parametriche associate al generico modello mistura (salvo il caso banale $K = 1$): questo perchè scelta una generica $p(\mathbf{y}|\boldsymbol{\theta}^*, \boldsymbol{\pi}^*) \in \mathcal{F}$ l'antimmagine $\varphi^{-1}\left(p(\mathbf{y}|\boldsymbol{\theta}^*, \boldsymbol{\pi}^*)\right)$ identifica $K!$ diversi parametri nel dominio di partenza ottenibili come permutazioni di $(\boldsymbol{\theta}^*, \boldsymbol{\pi}^*)$. Ad esempio, scelta $p(y|\mu_1, \mu_2) = \sum_{j=1}^{2} \pi_j p(y|\mu_j, 1)$ è immediato notare che entrambe le coppie ordinate (μ_1, μ_2) e (μ_2, μ_1) identificano la stessa distribuzione mistura in $\mathcal{F}$.

La non-identificabilità del modello non limita comunque la possibilità di svolgere l'inferenza nei due classici schemi noti, quello frequentista e bayesiano, adottando gli opportuni accorgimenti che descrivo nelle prossime Sezioni.

2.2 Tecniche e problematiche dell'inferenza frequentista

Ci si ponga per il momento in un contesto di inferenza classico, facendo riferimento al paradigma frequentista: l'obbiettivo è quello di ottenere le stime di $\boldsymbol{\pi}$ e dei vettori che parametrizzano ciascuna componente della mistura $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K$ andando a massimizzare la funzione di verosimiglianza, che (supponendo di aver osservato un campione di osservazioni

iid $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$) assume la seguente forma:

$$\begin{aligned}\mathcal{L}\left(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K, \boldsymbol{\pi}; \{\mathbf{y}_i\}_{i=1}^n\right) &= \prod_{i=1}^{n} p(\mathbf{y}_i|\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K, \boldsymbol{\pi}) \\ &= \prod_{i=1}^{n} \left(\sum_{j=1}^{K} \pi_j p(\mathbf{y}_i|\boldsymbol{\theta}_j) \right). \end{aligned} \tag{2.5}$$

E' immediato notare la difficoltà nel ricavare la stima di massima verosimiglianza esplicita di $\boldsymbol{\vartheta} = (\boldsymbol{\pi}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$ definita come

$$\hat{\boldsymbol{\vartheta}}_{SMV} = \underset{\Theta^K \times \mathcal{E}}{\arg\max}\, \mathcal{L}(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K, \boldsymbol{\pi}; \mathbf{y}_1, \ldots, \mathbf{y}_n); \tag{2.6}$$

questa richiederebbe infatti il calcolo del vettore *score* per la somma di trasformate logaritmiche di altrettante somme (se consideriamo la funzione di log-verosimiglianza) e la successiva soluzione analitica delle equazioni di verosimiglianza. Allo stesso modo, è stato dimostrato che una massimizzazione numerica potrebbe risultare particolarmente onerosa in termini computazionali quando n, K e/o $p = \dim \Theta$ sono sufficientemente elevati, rischiando addirittura che l'algoritmo di ottimizzazione si aréni in mode locali e portando dunque a conclusioni inferenziali potenzialmente errate.

In alternativa, l'approccio più opportuno e maggiormente utilizzato per la stima di massima verosimiglianza dei modelli mistura utilizza il noto algoritmo EM (Expectation-Maximization) proposto da Dempster, Laird e Rubin (1977): questo, tramite l'introduzione di un'opportuna variabile latente che indica la sotto-popolazione di appartenenza per ciascuna $\mathbf{y}_i$, permette di ottenere una forma della log-verosimiglianza *aumentata* più agevole da trattare e da massimizzare. Senza entrare nei dettagli, possiamo sintetizzare la procedura dell'algoritmo nell'iterazione dei seguenti passaggi:

1. E(xpectation)-Step: si calcola il valore atteso della log-verosimiglianza aumentata, rispetto alla distribuzione della variabile latente condizionata alle stime di $\boldsymbol{\vartheta}$ ottenute al passo precedente ed ai dati;

2. M(aximization)-Step: si massimizza rispetto a $\boldsymbol{\vartheta}$ il valore atteso condizionato ottenuto al passo precedente;

fino a raggiungere la convergenza.

Globalmente sia la massimizzazione numerica che l'algoritmo EM permettono di ottenere stime di massima verosimiglianza (e di godere di tutte le buone proprietà asintotiche possedute dallo stimatore di massima verosimiglianza), a patto che la convergenza dell'algoritmo venga raggiunta: se nell'algoritmo EM questa è garantita sotto alcune condizioni piuttosto generali, algoritmi di massimizzazione numerica potrebbero invece soffrire dei problemi esposti poco fa, compromettendo l'efficienza numerica della procedura di stima e la validità delle conclusioni inferenziali. In generale, comunque, la stima di massima verosimiglianza nel contesto dei modelli parametrici a mistura finita può riscontrare alcune problematiche pratiche relative principalmente alla difficoltà di ricavare *standard error* associati alle stime ottenute direttamente dall'algoritmo EM, ed alla forma della verosimiglianza del modello quando le componenti provengono dalla famiglia delle distribuzioni normali, che è stato dimostrato essere non-limitata.

Uno dei punti di forza dello schema inferenziale frequentista, invece, risiede sicuramente nel fatto che il problema di *label switching* è immediatamente risolto in fase di ottimizzazione: la funzione di verosimiglianza sarà $K!$-modale, dove ciascuna moda giace in corrispondenza di una delle possibili $K!$ permutazioni della generica stima di massima verosimiglianza. A livello pratico, dunque, sarà sufficiente garantire la possibilità di identificare numericamente una delle $K!$ possibili mode della funzione per identificare la stima di massima verosimiglianza a meno di permutazioni interne.

2.3 Inferenza bayesiana sul modello mistura a numero finito di componenti

Una metodologia più opportuna per questa classe di modelli, che permette di tenere conto ed in certi casi di risolvere alcune delle problematiche evidenziate nelle righe precedenti, fa riferimento al paradigma inferenziale bayesiano.

Frühwirth-Schnatter (2006) elenca alcuni possibili vantaggi che si ottengono decidendo di seguire questo approccio: innanzitutto, la scelta di un'opportuna distribuzione a priori permette di introdurre un'effetto di lisciamento sulla funzione di verosimiglianza del modello mistura, riducendo il rischio di ottenere mode locali sulle quali l'algoritmo EM potrebbe andare ad arrestarsi; in secondo luogo, la metodologia bayesiana ammette la possibilità di svolgere inferenza senza particolari condizioni di regolarità da soddisfare, al contrario della

metodologia frequentista che richiede una numerosità campionaria sufficientemene elevata affinchè i risultati asintotici siano validi ed ammettano strumenti inferenziali adeguati; infine, la conoscenza della distribuzione a posteriori permette, da sola ed indipendentemente dall'algoritmo di stima utilizzato, di ottenere una misura di incertezza delle stime.

Riguardo la scelta della distribuzione a priori $p(\boldsymbol{\vartheta})$, va immediatamente sottolineato come non esista alcuna famiglia distributiva che realizzi il coniugio naturale con la forma della funzione di verosimiglianza (2.5): nonostante questo, però, l'avvento dei metodi MCMC ha consentito lo sviluppo di procedure per la simulazione dalla distribuzione a posteriori particolarmente apprezzati per la loro facilità ed immediatezza. In particolare, Dempster, Laird e Rubin (1977) hanno gettato le basi dello schema di *data augmentation* sul quale si basano la maggior parte dei metodi di MCMC e Gibbs Sampling per la simulazione dalla distribuzione a posteriori dei modelli mistura, mentre Diebolt e Robert (1994) sono stati tra i primi ad esplicitare i passaggi per un Gibbs Sampling sul modello mistura finito a componenti gaussiane, approfondito nella Sottosezione 2.3.1.

Come già visto in precedenza nella Sezione 2.2 in riferimento all'algoritmo EM, risulta particolarmente utile nel contesto dei modelli a mistura finita adottare una strategia di *data-augmentation*: in particolare, introduciamo nel modello a mistura finita una variabile latente Z_i associata a ciascuna osservazione $\mathbf{y}_i$ e contenente l'etichetta della sottopopolazione della mistura dalla quale è generata, ovvero $Z_i \in \{1, 2, \ldots, K\}$, $i = 1, 2, \ldots, n$. Una rappresentazione alternativa diffusa in letteratura, ma che non adottiamo in questo lavoro per evitare di complicare ulteriormente la notazione, prevede di considerare un'insieme di vettori latenti $\mathbf{Z}_i$, con $\mathbf{Z}_i \in \mathcal{E}_K$, dove $\mathcal{E}_K$ è la base canonica di $\mathbb{R}^K$; in questo modo, $\mathbf{Z}_i = \mathbf{e}_j$ indicherà che l'osservazione $\mathbf{y}_i$ proverrà dalla sottopopolazione j-esima.

Questa rappresentazione ci consente di interpretare il modello secondo una struttura gerarchica a variabili latenti, dove la distribuzione di ciascuna $\mathbf{y}_i$ dipende dall'indicatore latente Z_i, semplificandone la struttura a patto di introdurre un'ulteriore variabile.

Sfruttando le proprietà della probabilità condizionata, è possibile riscrivere la distribuzione congiunta delle osservazioni e delle variabili latenti nel seguente modo

$$p(\mathbf{y}_1, \ldots, \mathbf{y}_n, \mathbf{Z}|\boldsymbol{\vartheta}) = p(\mathbf{y}_1, \ldots, \mathbf{y}_n|\mathbf{Z}, \boldsymbol{\vartheta})\, p(\mathbf{Z}|\boldsymbol{\vartheta}) \tag{2.7}$$

evidenziando, separatamente, la distribuzione congiunta delle $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$, che nel primo

livello del modello sarà specificata condizionatamente a $\mathbf{Z}$

$$p(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n | \mathbf{Z}, \boldsymbol{\vartheta}) = \prod_{i=1}^{n} p(\mathbf{y}_i | Z_i, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K) = \prod_{i=1}^{n} p(\mathbf{y}_i | \boldsymbol{\theta}_{Z_i}) \tag{2.8}$$

e la distribuzione delle latenti, per le quali si assume l'indipendenza, specificata nel secondo livello come

$$p(\mathbf{Z}|\boldsymbol{\vartheta}) = p(\mathbf{Z}|\boldsymbol{\pi}) = \prod_{i=1}^{n} p(Z_i|\boldsymbol{\pi}), \tag{2.9}$$

con $p(Z_i|\boldsymbol{\pi}) = p(Z_i = j|\boldsymbol{\pi}) = \pi_j$, per $j = 1, 2, \ldots, K$. Di particolare interesse è la distribuzione congiunta delle variabili latenti, che può essere interpretata come un processo di *clustering* delle osservazioni per un numero prefissato di *cluster* K: il processo generativo dei dati sceglierà in prima battuta il *cluster* secondo la legge $p(Z_i|\boldsymbol{\pi})$, e in seconda battuta genererà $\mathbf{y}_i$ dalla distribuzione associata al *cluster* selezionato, $p(\mathbf{y}_i|\boldsymbol{\vartheta}_{Z_i})$.

L'obbiettivo dell'approccio bayesiano rimane comunque quello di individuare la distribuzione a posteriori $p(\boldsymbol{\vartheta}|\mathbf{y}_1, \ldots, \mathbf{y}_n)$, che tramite il Teorema di Bayes può essere scritta nella nota forma

$$p(\boldsymbol{\vartheta}|\mathbf{y}_1, \ldots, \mathbf{y}_n) \propto \mathcal{L}(\mathbf{y}_1, \ldots, \mathbf{y}_n|\boldsymbol{\vartheta})\, p(\boldsymbol{\vartheta}), \tag{2.10}$$

dove la verosimiglianza del modello può essere espressa al pari di quanto già visto nella (2.5), ma evidenziando opportunamente la dipendenza da $\boldsymbol{\vartheta}$:

$$\mathcal{L}(\mathbf{y}_1, \ldots, \mathbf{y}_n|\boldsymbol{\vartheta}) = \prod_{i=1}^{n} \left(\sum_{j=1}^{K} \pi_j p(\mathbf{y}_i|\boldsymbol{\theta}_j) \right). \tag{2.11}$$

L'introduzione dello schema di *data augmentation* risulta in questo caso particolarmente vantaggioso perchè consente di semplificare l'espressione della verosimiglianza, che può essere riscritta come

$$\mathcal{L}(\mathbf{y}_1, \ldots, \mathbf{y}_n, \mathbf{Z}|\boldsymbol{\vartheta}) = \prod_{i=1}^{n} \prod_{j=1}^{K} \left(\pi_j p\left(\mathbf{y}_i|\boldsymbol{\theta}_j\right)\right)^{\mathbb{1}(Z_i=j)} \tag{2.12}$$

$$= \left(\prod_{j=1}^{K} \prod_{i:Z_i=j} p(\mathbf{y}_i|\boldsymbol{\theta}_j) \right) \left(\prod_{j=1}^{K} \pi_j^{n_j} \right) \tag{2.13}$$

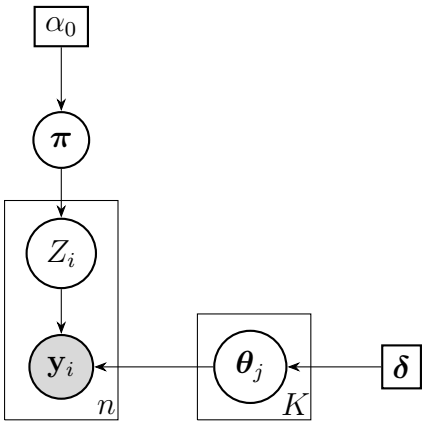


Figura 2.1: DAG del modello mistura nello schema inferenziale bayesiano. I nodi (cerchi) rappresentano variabili casuali, i rettangoli iperparametri delle distribuzioni a priori sulle variabili casuali, i rettangoli contenenti nodi vettori, con la dimensione riportata nell'angolo in basso a destra.

dove $n_j = \sum_{i=1}^{n} \mathbb{1}(Z_i = j)$, $j = 1, 2, \ldots, K$. Questa seconda formulazione risulta particolarmente utile in un contesto di inferenza bayesiano perchè permette di mostrare chiaramente quali componenti della verosimiglianza entrano nella composizione delle distribuzioni a posteriori per ciascun parametro.

In Figura 2.1 è riportato il *direct acyclic graph (DAG)* riferito al modello mistura bayesiano parametrico: una rappresentazione particolarmente diffusa nei contesti di modellazione bayesiana perchè permette di avere un'immediata intuizione delle relazioni di dipendenza ed indipendenza condizionata tra le variabili del modello.

La scelta della distribuzione a priori $p(\boldsymbol{\vartheta})$ per ciascuno dei parametri incluso nel modello, rimane la fase più delicata da compiere nella modellazione bayesiana, a maggior ragione quando l'obbiettivo è quello di ricavare un Gibbs Sampler in cui sia facile generare da ciascuna *full conditional* a posteriori; un'ipotesi comune e di buon senso che si fa nel contesto dei modelli parametrici bayesiani a mistura finita è quella di indipendenza tra l'insieme dei vettori $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K$ che indicizzano la distribuzione di ciascuna sottopopolazione e i pesi della mistura $\boldsymbol{\pi}$, ovvero che

$$p(\boldsymbol{\vartheta}) = p(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)\, p(\boldsymbol{\pi}). \tag{2.14}$$

Per quanto riguarda la scelta della priori su $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$, trattandosi di un vettore

a valori nel simplesso $\mathcal{E}^K$ è particolarmente appropriato assumere

$$\boldsymbol{\pi} \sim \mathsf{Dir}\left(\frac{\alpha_0}{K}, \ldots, \frac{\alpha_0}{K}\right) \tag{2.15}$$

ovvero una distribuzione Dirichlet simmetrica in cui i pesi a priori su ciascuna componente sono tutti uguali e pari a α_0/K: questo permette di rappresentare l'assenza di informazione a priori sul vantaggio di una componente rispetto ad un'altra, ammettendo inoltre la possibilità di parametrizzare l'intera distribuzione con un solo iperparametro α_0, detto *di concentrazione* perchè regola il grado di *piattezza* della distribuzione. La distribuzione a priori scelta è particolarmente consona in questo specifico caso perchè risulta coniugata naturale con la famiglia delle distribuzioni multinomiali/discrete che modellano $Z_i|\boldsymbol{\pi}$ (il secondo termine in cui è stata *fattorizzata* la verosimiglianza del modello aumentato nella (2.12)): vale infatti che, a posteriori,

$$\boldsymbol{\pi}|\mathbf{Z} \sim \mathsf{Dir}\left(\frac{\alpha_0}{K} + n_1, \ldots, \frac{\alpha_0}{K} + n_K\right). \tag{2.16}$$

Rispetto la scelta della distribuzione a priori per la variabile latente Z_i associata alla generica osservazione, è già stato detto che questa dipenderà inevitabilmente da $\boldsymbol{\pi}$ e corrisponde a

$$Z_i|\boldsymbol{\pi} \sim \mathsf{Discreta}\left(K, \boldsymbol{\pi}\right) \quad i = 1, 2, \ldots, n \tag{2.17}$$

stando ad indicare che $p(Z_i = j|\boldsymbol{\pi}) = \pi_j$. Questa distribuzione risulta essere coniugata con il primo fattore della (2.12): vale infatti che a posteriori

$$Z_i|\mathbf{y}_i, \boldsymbol{\pi}, \{\boldsymbol{\theta}_j\}_{j=1}^K \sim \mathsf{Discreta}\left(K, \boldsymbol{\pi}^*\right) \quad i = 1, 2, \ldots, n \tag{2.18}$$

con $\pi_j^* = p(Z_i = j|\mathbf{y}_i, \pi_j, \boldsymbol{\theta}_j) \propto p(Z_i = j)p(\mathbf{y}_i|\boldsymbol{\theta}_j) = \pi_j p(\mathbf{y}_i|\boldsymbol{\theta}_j)$ per $j = 1, 2, \ldots, K$. In altre parole, la distribuzione a posteriori per Z_i aggiorna la probabilità a priori π_j di assegnare l'osservazione $\mathbf{y}_i$ al cluster j-esimo con il valore della densità associata alla componente di quel cluster calcolata nell'osservazione stessa: se quest'ultima risulta essere bassa, ovvero se $\mathbf{y}_i$ risulta collocarsi sulle code della distribuzione associata alla componente j-esima, questo si ripercuoterà in una bassa probabilità a posteriori che quella osservazione possa appartenere al cluster j-esimo, e viceversa se risulta essere alta.

Riguardo la scelta della priori su $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K$, generalmente si ipotizza l'indipendenza reciproca su ciascun parametro

$$p(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K|\boldsymbol{\delta}) = \prod_{j=1}^{K} p(\boldsymbol{\theta}_j|\boldsymbol{\delta}) \tag{2.19}$$

e che queste la priori specificata su ciascuno sia identicamente distribuita a tutte le altre, con uguale e generico vettore di iperparametri $\boldsymbol{\delta}$.

Per la generica $p(\boldsymbol{\theta}_j)$ generalmente si individua la coniugata naturale alla distribuzione di $p(\mathbf{y}|\boldsymbol{\theta}_j)$: quest'ultima nella maggior parte dei casi unidimensionali corrisponde ad una distribuzione di Poisson quando si trattano misture per dati di conteggio (per la quale la distribuzione a priori Gamma è coniugata naturale) o ad una distribuzione Normale quando si trattano misture di dati continui (per la quale, a seconda della scelta di fissare o meno il parametro di sensibilità $1/\sigma^2$ corrisponde ad una priori Normale o Normale-GammaInversa coniugata naturale); nei casi multidimensionali, invece, generalmente corrisponde ad una distribuzione Normale Multivariata, per la quale a seconda della scelta di fissare o meno la matrice di precisione Σ^{-1} la priori coniugata naturale corrisponde alla Normale Multivariata o alla Normale Multivariata-Wishart Inversa.

In generale, comunque, è sempre possibile svolgere un'analisi coniugata per la verosimiglianza aumentata $p(\mathbf{y}_1, \ldots, \mathbf{y}_n, \mathbf{Z}|\boldsymbol{\vartheta})$ se le componenti della mistura provengono da una famiglia esponenziale, ovvero se le rispettive funzioni di probabilità/densità possono essere scritte nella nota forma

$$p(\mathbf{y}_i|\boldsymbol{\theta}_j) = \exp\left\{\phi(\boldsymbol{\theta}_j)^T u(\mathbf{y}_i) - g(\boldsymbol{\theta}_j) + c(\mathbf{y}_i)\right\}; \tag{2.20}$$

Se ciascuna componente della priori può essere scritta come

$$p(\boldsymbol{\theta}_j|\boldsymbol{\delta}) \propto \exp\left\{\phi(\boldsymbol{\theta}_j)^T a_0 - g(\boldsymbol{\theta}_j) b_0\right\} \tag{2.21}$$

con iperparametri $\boldsymbol{\delta} = (a_0, b_0)$, allora la distribuzione a posteriori aumentata per ciascuna componente della mistura assume la seguente forma

$$p(\boldsymbol{\theta}_j|\mathbf{Z}, \mathbf{y}_1, \ldots, \mathbf{y}_n) \propto \exp\left\{\phi(\boldsymbol{\theta}_j)^T a_j - g(\boldsymbol{\theta}_j) b_j\right\} \tag{2.22}$$

che corrisponde, ancora una volta, ad una densità appartenente alla famiglia esponenziale scelta per modellare la distribuzione a priori della singola $\boldsymbol{\theta}_j$, ma con iperparametri aggiornati e corrispondenti rispettivamente a

$$a_j = a_0 + \sum_{i\,:\,Z_i=j} u(\mathbf{y}_i) \quad \text{e} \quad b_j = b_0 + \sum_{i=1}^{n} \mathbb{1}(Z_i = j). \tag{2.23}$$

Come già detto, per i modelli mistura bayesiani a componenti finite esiste la possibilità di implementare facilmente un algoritmo di Gibbs Sampling per simulare dalla distribuzione

a posteriori: in particolare, definendo

$$\boldsymbol{\eta} = (\boldsymbol{\vartheta}, \mathbf{Z}) = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K, \boldsymbol{\pi}, \mathbf{Z}) \tag{2.24}$$

l'insieme delle variabili contenute nel modello, sarà necessario derivare la distribuzione *full-conditional* a posteriori per ciascuna variabile, ovvero $p(\eta_i|\boldsymbol{\eta}_{-i}, \mathbf{y}_1, \ldots, \mathbf{y}_n)$. Questa è generalmente un'operazione non immediata da svolgere, in particolar modo quando il numero di variabili contenute nel modello è ampio; esiste però la possibilità di utilizzare il DAG rappresentato in Figura 2.1 per individuare le relazioni di dipendenza tra le variabili. Nozioni di *bayesian graphical models* assicurano che le variabili condizionanti un determinato nodo del DAG siano solo ed esclusivamente quelle appartenenti al cosiddetto *markov blanket* del nodo stesso, ovvero l'insieme di tutti i suoi nodi genitori, nodi figli e nodi genitori dei nodi figli: ulteriori dettagli ed approfondimenti, non necessari per comprendere il contenuto di questa tesi, sono contenuti in Koller e Friedman (2009).

Con riferimento al modello mistura, questa utile proprietà ci permette di ricavare le seguenti *full-conditional* a posteriori.

Full-conditional a posteriori per $\boldsymbol{\theta}_j|\mathbf{y}_1, \ldots, \mathbf{y}_n, \mathbf{Z}$:

Nel caso in cui la distribuzione a priori scelta per $\boldsymbol{\theta}_j$ avesse una forma del tipo (2.21) ed il singolo contributo alla verosimiglianza aumentata della componente j-esima una forma riconducibile alla (2.20), è possibile ottenere una *full-conditional* a posteriori della forma specificata nella (2.22):

$$p(\boldsymbol{\theta}_j|\mathbf{y}_1, \ldots, \mathbf{y}_n, \mathbf{Z}) \propto \exp\left\{\phi(\boldsymbol{\theta}_j)^T a_j - g(\boldsymbol{\theta}_j) b_j\right\} \tag{2.25}$$

con

$$a_j = a_0 + \sum_{i\,:\,Z_i=j} u(\mathbf{y}_i) \quad \text{e} \quad b_j = b_0 + \sum_{i=1}^{n} \mathbb{1}(Z_i = j). \tag{2.26}$$

Questa, nella maggior parte dei casi, corrisponde alla funzione di densità di una distribuzione dalla quale è immediato e facile generare.

Full-conditional a posteriori per $\boldsymbol{\pi}|\mathbf{Z}$:

Dato che $\mathbf{Z}$ è l'unico termine che rientra nel *markov blanket* di $\boldsymbol{\pi}$, questa corrisponde esattamente alla (2.16), ovvero

$$\boldsymbol{\pi}|\mathbf{Z} \sim \mathsf{Dir}\left(\frac{\alpha_0}{K} + n_1, \ldots, \frac{\alpha_0}{K} + n_K\right) \tag{2.27}$$

con $n_j = \sum_{i=1}^{n} \mathbb{1}(Z_i = j)$, per $j = 1, 2, \ldots, K$.

Full-conditional a posteriori per $Z_i \mid \{\mathbf{y}_i\}_{i=1}^n, \mathbf{Z}_{-i}, \boldsymbol{\pi}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K$:

Relativamente a questa *full-conditional*, ricaviamo innanzitutto la forma più generale che include $\boldsymbol{\pi}$ nell'insieme delle variabili condizionanti:

$$\begin{aligned} p\left(Z_i = j \,\middle|\, \{\mathbf{y}_i\}_{i=1}^n, \mathbf{Z}_{-i}, \boldsymbol{\pi}, \{\boldsymbol{\theta}_j\}_{j=1}^K\right) &= p\left(Z_i = j \,\middle|\, \mathbf{Z}_{-i}, \boldsymbol{\pi}, \{\boldsymbol{\theta}_j\}_{j=1}^K\right)\, p\left(\{\mathbf{y}_i\}_{i=1}^n \,\middle|\, \mathbf{Z}, \boldsymbol{\pi}, \{\boldsymbol{\theta}_j\}_{j=1}^K\right) \\ &= p(Z_i = j|\boldsymbol{\pi})\, p\left(\mathbf{y}_i \,\middle|\, \{\mathbf{y}\}_{-i}, \mathbf{Z}, \{\boldsymbol{\theta}_j\}_{j=1}^K\right) \\ &= \pi_j\, p(\mathbf{y}_i|\boldsymbol{\theta}_j). \end{aligned} \tag{2.28}$$

Un'ulteriore semplificazione generalmente adottata e che consente di ottenere il cosiddetto *collapsed Gibbs Sampler* prevede la possibilità di marginalizzare $\boldsymbol{\pi}$ nella densità di $\mathbf{Z}$:

$$\begin{aligned} p(\mathbf{Z}) &= \int p(\mathbf{Z}, \boldsymbol{\pi})\, d\boldsymbol{\pi} = \int p(\mathbf{Z}|\boldsymbol{\pi})\, p(\boldsymbol{\pi})\, d\boldsymbol{\pi} \\ &= \int \prod_{j=1}^{K} \pi_j^{n_j} \frac{\Gamma(\alpha_0)}{\Gamma\left(\alpha_0/K\right)^K} \prod_{j=1}^{K} \pi_j^{\alpha_0/K-1}\, d\boldsymbol{\pi} \\ &= \frac{\Gamma(\alpha_0)}{\Gamma\left(\alpha_0/K\right)^K} \int \prod_{j=1}^{K} \pi_j^{\alpha_0/K+n_j-1}\, d\boldsymbol{\pi} \\ &= \frac{\Gamma(\alpha_0)}{\Gamma\left(\alpha_0/K\right)^K} \frac{\prod_{j=1}^{K} \Gamma(n_j + \alpha_0/K)}{\Gamma(\alpha_0 + n)} \\ &= \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_0 + n)} \prod_{j=1}^{K} \frac{\Gamma(n_j + \alpha_0/K)}{\Gamma(\alpha_0/K)}. \end{aligned} \tag{2.29}$$

A partire da questa, è possibile individuare la *full-conditional* a priori per $\mathbf{Z}$ come

$$\begin{aligned} p(Z_i = j|\mathbf{Z}_{-i}) &= \frac{p(Z_i = j, \mathbf{Z}_{-i})}{p(\mathbf{Z}_{-i})} \\ &= \frac{\cancel{\Gamma(\alpha_0)}}{\Gamma(\alpha_0 + n)} \prod_{j=1}^{K} \frac{\Gamma(n_j + \alpha_0/K)}{\cancel{\Gamma(\alpha_0/K)}} \Big/ \frac{\cancel{\Gamma(\alpha_0)}}{\Gamma(\alpha_0 + n - 1)} \prod_{j=1}^{K} \frac{\Gamma(n_{j,-i} + \alpha_0/K)}{\cancel{\Gamma(\alpha_0/K)}} \\ &= \frac{\Gamma(\alpha_0 + n - 1)}{\Gamma(\alpha_0 + n)} \frac{\Gamma(n_j + \alpha_0/K)}{\Gamma(n_{j,-i} + \alpha_0/K)} = \frac{n_{j,-i} + \alpha_0/K}{n + \alpha_0 - 1} \end{aligned} \tag{2.30}$$

dove $n_{j,-i} = \sum_{-i} \mathbb{1}(Z_i = j)$ e tutti i termini della produttoria escluso il j-esimo si semplificano perchè $n_j = n_{j,-i}$.

La marginalizzazione rispetto a $\boldsymbol{\pi}$ ci consente di ricavare una *full-conditional* a posteriori per Z_i all'interno della quale non compare $\boldsymbol{\pi}$:

$$\begin{aligned} p\left(Z_i = j \,\middle|\, \{\mathbf{y}_i\}_{i=1}^n, \mathbf{Z}_{-i}, \cancel{\boldsymbol{\pi}}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\right) &= p(Z_i = j|\mathbf{Z}_{-i})\, p(\mathbf{y}_i|\boldsymbol{\theta}_j) \\ &= \frac{n_{j,-i} + \alpha_0/K}{n + \alpha_0 - 1}\, p(\mathbf{y}_i|\boldsymbol{\theta}_j) \end{aligned} \tag{2.31}$$

A questo punto, un algoritmo *collapsed Gibbs Sampling* per simulare dalla distribuzione a posteriori del modello parametrico a mistura finita può essere implementato seguendo i passaggi riportati schematicamente in Algorithm 1: la particolarità della versione *collapsed* è che non viene generato il vettore $\boldsymbol{\pi}$ dalla *full-conditional* associata, essendo stato quest'ultimo marginalizzato. Nonostante questo, comunque, la catena delle proporzioni $\boldsymbol{\pi}$ può essere ricostruita *all'indietro* a partire dalla catena per la variabile latente $\mathbf{Z}$, considerando le frequenze relative per ciascuna classe e simulando secondo la (2.16).

Algorithm 1 Collapsed Gibbs Sampling per una mistura parametrica finita

Disponendo dei dati osservati $(\mathbf{y}_1, \ldots, \mathbf{y}_n)$, del vettore degli iperparametri $\boldsymbol{\delta} = (a_0, b_0)$ riferiti alla distribuzione a priori su ciascun $\boldsymbol{\theta}_j$ e dell'iperparametro di concentrazione α_0, assegniamo il vettore di valori iniziali per $\mathbf{Z}$ a $\mathbf{Z}^{(1)}$ e quelli per $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K$ a $\boldsymbol{\theta}_1^{(1)}, \ldots, \boldsymbol{\theta}_K^{(1)}$ rispettivamente.
Per $t = 2, 3, \ldots, R$ si eseguono iterativamente le seguenti operazioni salvando, man mano, i valori generati:

1. Generazione dalla *full-conditional* a posteriori per $\boldsymbol{\theta}_j$, $j = 1, 2, \ldots, K$:

$$\boldsymbol{\theta}_j^{(t)} | \mathbf{y}_1, \ldots, \mathbf{y}_n, \mathbf{Z}^{(t-1)} \sim \exp\left\{ \phi\left(\boldsymbol{\theta}_j\right)^T a_j - g\left(\boldsymbol{\theta}_j\right) b_j \right\}$$

   con

$$a_j = a_0 + \sum_{i\,:\, Z_i^{(t-1)} = j} u(\mathbf{y}_i) \quad \text{e} \quad b_j = b_0 + \sum_{i=1}^{n} \mathbb{1}\left(Z_i^{(t-1)} = j \right),$$

   purchè la distribuzione a priori per $\boldsymbol{\theta}_j$ e la generica $p(\mathbf{y}_i | \boldsymbol{\theta}_j)$ abbiano la forma, rispettivamente, di (2.21) e (2.20).

2. Sostituisco $\mathbf{Z}^{(t-1)}$ a $\mathbf{Z}^{(t)}$ per facilitare i calcoli nel passo 3.

3. Generazione dalla *full-conditional* a posteriori per Z_i, $i = 1, 2, \ldots, n$:

$$Z_i^{(t)} = j | \mathbf{y}_i, \mathbf{Z}_{-i}^{(t)}, \boldsymbol{\theta}_j^{(t)} \sim \frac{n_{j,-i}^{(t)} + \alpha_0 / K}{n + \alpha_0 - 1} \exp\left\{ \phi\left(\boldsymbol{\theta}_j^{(t)}\right)^T u(\mathbf{y}_i) - g\left(\boldsymbol{\theta}_j^{(t)}\right) + c(\mathbf{y}_i) \right\}$$

   per ciascun $j = 1, 2, \ldots, K$, dove l'apice (t) in $n_{j,-i}$ sta ad indicare che questa è calcolata su $\mathbf{Z}_{-i}^{(t)}$.

Si effettua un *burnin* iniziale e si restituiscono i risultati.

Una caratteristica da tenere in considerazione quando si svolge inferenza bayesiana sui modelli mistura e che spesso ne complica le fasi riguarda la gestione del problema di *label switching* già accennato in questo Capitolo. In un contesto bayesiano, questo si ripercuote essenzialmente nel campionamento dalla distribuzione a posteriori, che ora dovrà tenere in considerazione il fatto che le etichette delle componenti della mistura permutano a ciascuna iterazione del Gibbs Sampler. Questo problema è stato individuato parecchio recentemente nella letteratura bayesiana, essenzialmente quando sono stati resi disponibili metodi computazionali per svolgere simulazioni della distribuzione a posteriori: i principali contributi in merito sono dovuti a Frühwirth-Schnatter (2006) ed ai lavori di Celeux, Hurn e Robert (2000) e Jasra, Holmes e Stephens (2005).

Nei contesti di inferenza bayesiana una delle questioni più controverse e discusse riguarda la scelta della distribuzione a priori: nella maggior parte dei casi, quando non si possiede alcuna conoscenza a priori sui parametri, fisica o di natura teorica, la scelta ricade su distribuzioni a priori coniugate alla famiglia parametrica individuata per la modellazione dei dati che siano il più possibile *diffuse*. Nella fattispecie dei modelli mistura, un'ulteriore aspetto va specificato: per rendere le distribuzioni a priori sempre più ininfluenti sulla distribuzione a posteriori, la scelta più immediata ed in genere più comune è quella di porre delle distribuzioni a priori scambiabili sui parametri $\boldsymbol{\theta}_j$ appartenenti a ciascuna componente e distribuzioni a priori simmetriche sul vettore dei pesi $\boldsymbol{\pi}$. Questa scelta si ripercuote in una distribuzione a posteriori che risulta essere invariante su permutazioni delle etichette associate alle componenti, come già visto per la forma della verosimiglianza: in altre parole, la distribuzione a posteriori dei parametri sarà identica per ciascuna componente della mistura.

Per questa ragione ci si aspetta che algoritmi MCMC o di Gibbs Sampling, nel simulare dalla distribuzione a posteriori, riescano a percorrere la totalità dello spazio parametrico e in particolare tutte le $K!$ mode che compongono la distribuzione a posteriori: questo accade se durante l'evoluzione della catena la simmetria della distribuzione a posteriori riesce a forzare uno scambio di etichette sulle componenti, facendo in modo che la catena non si concentri solo attorno ad una delle $K!$ possibili configurazioni: in particolare, lo scambio di etichette sulle componenti e dunque l'intera esplorazione delle $K!$ diverse configurazioni sullo spazio parametrico è un requisito essenziale per assicurarsi la convergenza della catena alla distribuzione stazionaria (a posteriori) e quindi per poter utilizzare la catena simulata

per svolgere procedure di inferenza. Nei casi pratici, comunque, è bene sottolineare che il fenomeno di *label switching* non è sempre immediatamente colto dalle catene generate su algoritmi di Gibbs Sampling: in particolare, quando si considerano distribuzioni a posteriori le cui mode sono ben distanziate e diversificate l'una dall'altra, è possibile che si raggiunga una di queste e ci si fermi attorno a quest'ultima, senza esplorare le rimanenti $K!-1$ perchè troppo distanti.

In letteratura sono state diffuse numerose strategie per tenere conto di questa caratteristica: tra queste, oltre alla possibilità di specificare un vincolo di identificazione sul modello (ad esempio, in una mistura a due componenti gaussiane, imporre $\mu_1 < \mu_2$ per restringere lo spazio parametrico), strategia valida comunque solo nei casi univariati, Frühwirth-Schnatter (2006) ha proposto di introdurre al termine di ciascuna iterazione del Gibbs Sampler un passo in cui si permutano internamente tutti i parametri generati, di fatto scambiandone l'ordine in fase di *storing* dei valori simulati; questo permette, in concreto, di forzare la manifestazione esplicita del *label switching* anche nei casi descritti poco fa in cui le mode della posteriori sono talmente distanti l'una dall'altra che il Gibbs Sampler non riesce a percorrerle tutte quante.

Nei casi pratici, però, la convergenza del Gibbs Sampler (ovvero, l'esplorazione di tutte le $K!$ mode della posteriori) complica non poco l'inferenza e la possibilità di utilizzare direttamente ciascuna componente della catena per calcolare quantità di interesse: questo perchè il *label switching* mischia gli indici delle componenti della mistura e rende, nei fatti, la marginale dei parametri associati a ciascuna componente identica a tutte le altre $K-1$. Per risolvere questo inconveniente, generalmente si adottano delle procedure di trattamento della catena effettuando il *relabeling* di ciascuna componente fissando una delle possibili $K!$ permutazioni delle etichette, oppure dei metodi di clustering non-gerarchici quali, ad esempio, il k-medie. Solo successivamente è possibile calcolare quantità d'interesse riguardanti i singoli parametri che compongono la mistura, come ad esempio la media a posteriori: in caso contrario, quel che si otterrebbe andando a calcolare la media a posteriori di una singola componente della catena sarebbe la media di tutte le medie di ciascuna componente, rendendo vana l'inferenza. In tutti i casi in cui il Gibbs Sampler non riesce autonomamente ad esplorare tutte le mode della distribuzione a posteriori, invece, è possibile ugualmente ritenersi soddisfatti se dopo un buon numero di iterazioni ciascuna componente della catena si stabilizza attorno ad un valore preciso: in questo caso, nonostante sia nei fatti sbagliato

parlare di *convergenza* del Gibbs Sampler dato che quest'ultimo non esplora l'intera distribuzione a posteriori, possiamo comunque accontentarci di aver individuato e raggiunto la convergenza in una delle $K!$ diverse configurazioni della mistura; in particolare questo ci permette di ottenere agevolmente quantità di interesse o grafici per ciascun parametro incluso nel modello mistura, evitando la fase di post-trattamento altresì necessaria nei casi in cui si manifesta esplicitamente il fenomeno di *label switching* durante l'evoluzione della catena.

2.3.1 Esempio notevole: inferenza bayesiana su modelli a mistura finita gaussiani univariati e studio di simulazione

Nella gran parte dei casi, ovvero quando l'intenzione è quella di modellare dati di natura continua, la scelta della distribuzione per le componenti della mistura ricade generalmente al caso della famiglia gaussiana: questa garantisce un'enorme flessibilità ed una facilità operativa non indifferente rispetto ad altre famiglie distributive. In questa sottosezione formalizzo il modello mistura a componenti gaussiane per il caso univariato ($r = 1$), illustrandone il relativo *collapsed Gibbs Sampler* che consente di simulare dalla distribuzione a posteriori.

Considerato $\mathbf{y} = (y_1, \dots, y_n)$ il vettore delle osservazioni a disposizione, diremo che Y a valori in $\mathcal{Y} \subset \mathbb{R}$ proviene da una distribuzione mistura finita con K componenti gaussiane se la funzione di densità $p(y)$ associata può essere scritta come

$$p(y|\boldsymbol{\mu}, \mathbf{s}, \boldsymbol{\pi}) = \pi_1 \mathsf{N}(y; \mu_1, s_1^{-1}) + \cdots + \pi_K \mathsf{N}(y; \mu_K, s_K^{-1}) = \sum_{j=1}^{K} \pi_j \mathsf{N}(y; \mu_j, s_j^{-1}) \tag{2.32}$$

dove $\mathsf{N}(y; \mu_j, s_j)$ indica impropriamente la funzione di densità per una gaussiana a media μ_j e varianza $\sigma_j^2 = s_j^{-1}$ calcolata in y, con $\boldsymbol{\mu} = (\mu_1, \dots, \mu_K)$, $\mathbf{s} = (s_1, \dots, s_K)$, e $\boldsymbol{\pi} = (\pi_1, \dots, \pi_K)$. La scelta di parametrizzare la variabilità di ciascuna gaussiana tramite il parametro di precisione s_j anzichè quello di varianza σ_j^2 più comunemente usato è pressochè indifferente a livello concettuale, ma ci consente di alleggerire la notazione evitando di riportare ogni volta l'apice 2 per σ, soprattutto in espressioni complicate o quando si hanno ulteriori apici da scrivere, come si vedrà in seguito.

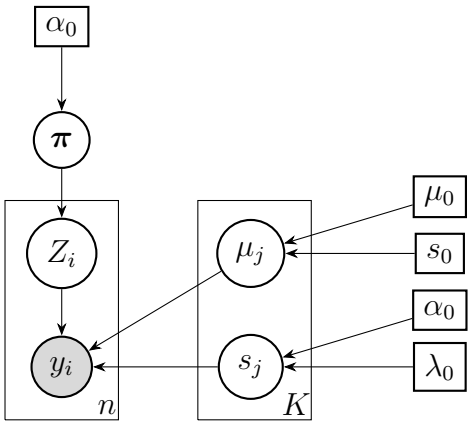


Figura 2.2: DAG del modello mistura gaussiano nello schema inferenziale bayesiano.

Per procedere con uno schema di inferenza bayesiano, sulla falsa riga di quanto già fatto nella Sezione 2.3, scegliamo le seguenti distribuzioni a priori:

$$\boldsymbol{\pi} \sim \mathsf{Dir}\left(\frac{\alpha_0}{K}, \ldots, \frac{\alpha_0}{K}\right), \tag{2.33}$$

$$Z_i|\boldsymbol{\pi} \sim \mathsf{Discreta}\left(K, \boldsymbol{\pi}\right) \quad i = 1, 2, \ldots, n \tag{2.34}$$

$$\mu_j \sim \mathsf{N}\left(\mu_0, s_0^{-1}\right) \quad j = 1, 2, \ldots, K, \tag{2.35}$$

$$s_j \sim \mathsf{Gamma}\left(\alpha_0, \lambda_0\right) \quad j = 1, 2, \ldots, K. \tag{2.36}$$

Una rappresentazione grafica del modello in considerazione è quella in Figura 2.2.

Ricalcando i passaggi illustrati al caso generale nella Sezione 2.3 per ricavare le forme delle *full-conditional* nei contesti coniugati, è possibile implementare un *collapsed Gibbs Sampler* per il modello mistura gaussiano univariato bayesiano seguendo i passi illustrati nell'Algoritmo 2 ed implementato nel Codice A.2.

A titolo dimostrativo, è stato generato un campione $\mathbf{x}$ di $n = 500$ osservazioni da una mistura di due gaussiane $X \sim 0.4\,\mathsf{N}(-4, 2) + 0.6\,\mathsf{N}(2, 4)$ e, su questo, è stato effettuato un breve studio di simulazione per verificare la convergenza dell'algoritmo di Gibbs Sampling ai veri valori, generando quattro catene di lunghezza 1000 inizializzate da punti distinti dello spazio parametrico. Il codice dello studio di simulazione, interamente replicabile, è disponibile nel Codice B.2.

Algorithm 2 Collapsed Gibbs Sampling per una mistura finita di gaussiane univariate

Disponendo dei dati osservati $\mathbf{y} = (y_1, \ldots, y_n)$, del vettore degli iperparametri (μ_0, s_0) riferiti alla priori su ciascun μ_j, del vettore degli iperparametri (α_0, λ_0) riferiti alla priori su ciascun s_j, e dell'iperparametro di concentrazione α_0 riferito alla priori su $\boldsymbol{\pi}$, assegniamo il vettore di valori iniziali per $\mathbf{Z}$ a $\mathbf{Z}^{(1)}$, quello per $\boldsymbol{\mu}$ a $\boldsymbol{\mu}^{(1)}$ e quello per $\mathbf{s}$ a $\mathbf{s}^{(1)}$.
Per $t = 2, 3, \ldots, R$ si eseguono iterativamente le seguenti operazioni salvando, man mano, i valori generati:

1. Generazione dalla *full-conditional* a posteriori per μ_j, $j = 1, 2, \ldots, K$:

$$\mu_j^{(t)} | \mathbf{y}, \mathbf{Z}^{(t-1)}, s_j^{(t-1)} \sim \mathsf{N}\left(\frac{\mu_0 s_0 + s_j^{(t-1)} \sum_{i\,:\, Z_i^{(t-1)}=j} y_i}{n_j^{(t-1)} s_j^{(t-1)} + s_0}, \frac{1}{n_j^{(t-1)} s_j^{(t-1)} + s_0} \right)$$

2. Generazione dalla *full-conditional* a posteriori per s_j, $j = 1, 2, \ldots, K$:

$$s_j^{(t)} | \mathbf{y}, \mathbf{Z}^{(t-1)}, \mu_j^{(t)} \sim \mathsf{Gamma}\left(\alpha_0 + \frac{n_j^{(t-1)}}{2},\ \lambda_0 + \sum_{i\,:\, Z_i^{(t-1)}=j} \left(y_i - \mu_j^{(t)}\right)^2 \Big/ 2 \right)$$

3. Sostituisco $\mathbf{Z}^{(t-1)}$ a $\mathbf{Z}^{(t)}$ per facilitare i calcoli nel passo 3.

4. Generazione dalla *full-conditional* a posteriori per Z_i, $i = 1, 2, \ldots, n$:

$$Z_i^{(t)} = j | y_i, \mathbf{Z}_{-i}^{(t)}, \mu_j^{(t)}, s_j^{(t)} \sim \frac{n_{j,-i}^{(t)} + \alpha_0/K}{n + \alpha_0 - 1} \mathsf{N}\left(y_i; \mu_j^{(t)}, \frac{1}{s_j^{(t)}} \right)$$

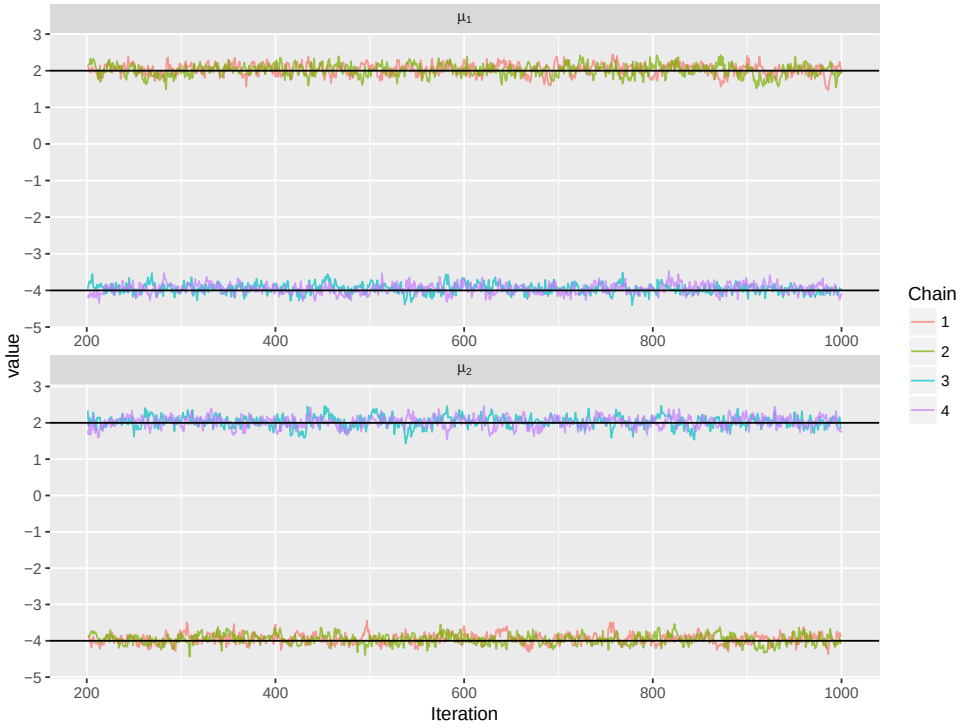
   per ciascun $j = 1, 2, \ldots, K$, dove anche in questo caso l'apice (t) in $n_{j,-i}$ sta ad indicare che questa è calcolata su $\mathbf{Z}_{-i}^{(t)}$.

Si effettua un *burnin* iniziale e si restituiscono i risultati.

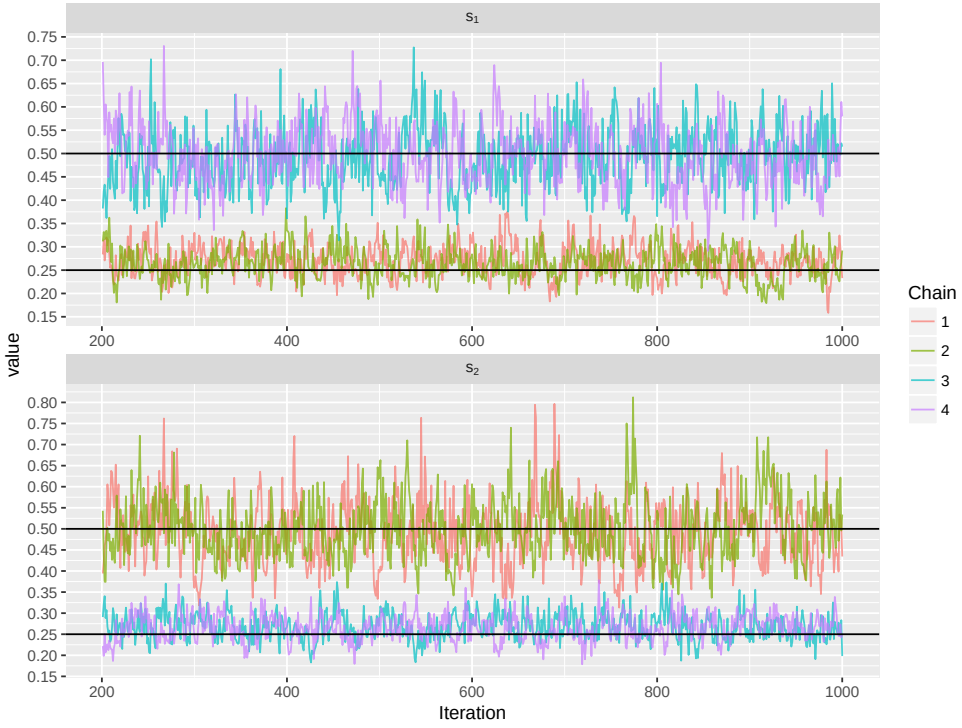
Osservando il *traceplot* dell'evoluzione delle catene in Figura 2.3, scartate del *burnin* iniziale corrispondente al primo 20% di valori simulati, si nota immediatamente come la convergenza venga raggiunta in tutte e quattro le catene, sia per i parametri μ_1, μ_2 che per i parametri s_1, s_2. In ciascun grafico (uno per ogni parametro) è tracciato con una linea orizzontale nera il vero valore del parametro dal quale è stata simulata la mistura; non deve trarre in inganno il fatto che le catene sembrino convergere su due valori distinti: questo è dovuto semplicemente all'interscambiabilità delle componenti della mistura, per la quale i parametri che la compongono sono stimabili a meno di permutazioni interne, e dunque ciascuna delle quattro catene decide casualmente quale *etichetta* della componente assegnare a μ_1 e quale a μ_2 (e, analogamente, a s_1 e s_2). Va ribadito nuovamente come, formalmente, non si possa parlare propriamente di *convergenza* della catena: quest'ultima in tutti e 4 i casi ha infatti esplorato solo una delle 2 possibili configurazioni del modello mistura che abbiamo specificato; nonostante questo però, ci accontentiamo di aver raggiunto la convergenza in una delle 2 configurazioni e dunque di conoscerne la distribuzione a posteriori a meno di permutazioni interne sulle componenti.

Una seconda serie di grafici particolarmente utile è quella in Figura 2.4 che compara le densità a posteriori per ciascuna componente, stimate in modo non-parametrico: queste, a meno di permutazioni interne sulle componenti della mistura, sono tutte concentrate attorno ai veri valori dai quali sono stati generati i dati. Quest'ultime consentono, tra le altre, la possibilità di ricavare intervalli HPD (*high posterior density*) per ciascuna componente, arricchendo la stima puntuale con una misura di *attendibilità a posteriori*.

I risultati della simulazione sono certamente confortanti circa il corretto *funzionamento* dell'algoritmo di generazione dalla distribuzione a posteriori: anche se non mostrato, le proporzioni delle classi simulate corrispondono a quelle teoriche, come è logico aspettarsi. Calcolare i classici indici di autocorrelazione o cross-correlazione tra le catene generate, o il Potential Scale Reduction Factor ($\hat{R}$) proposto da Gelman e Rubin (1992) risulta in questo caso particolarmente laborioso, a causa della *doppia-falsa* convergenza di ciascuna componente della catena, dovuta come già detto alla proprietà di interscambiabilità ed al fatto che ciascuna catena decide autonomamente a quale delle 2 configurazioni convergere, non riuscendo ad evidenziare esplicitamente il fenomeno di *label switching*; nonostante questo, possiamo accontentarci dei risultati grafici appena mostrati come primo strumento per verificare la convergenza dell'algoritmo alla distribuzione a posteriori della mistura.

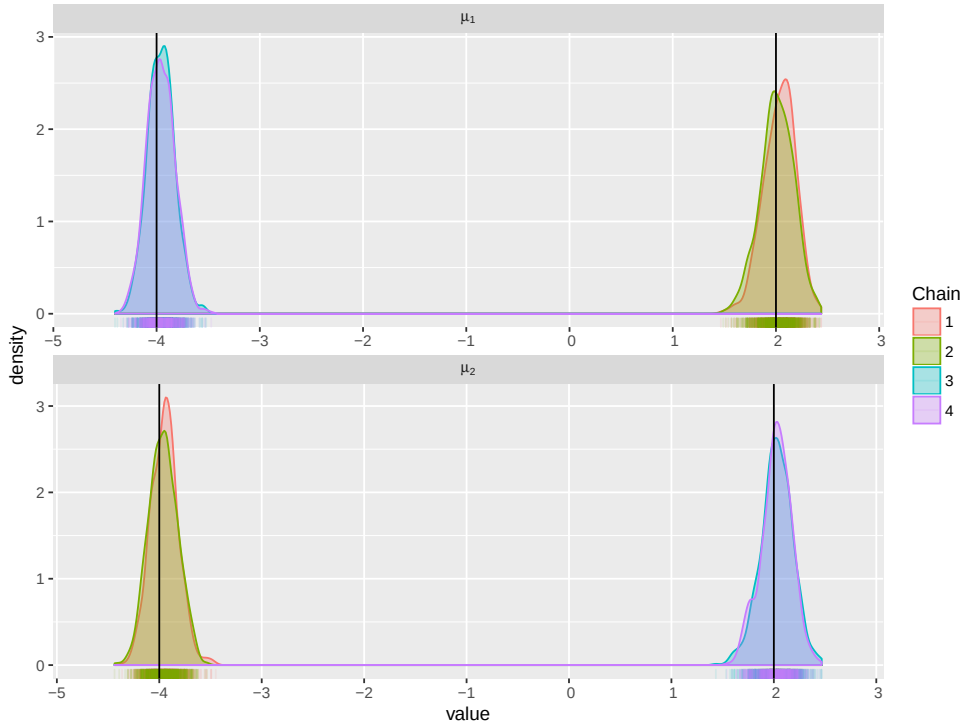


(a)

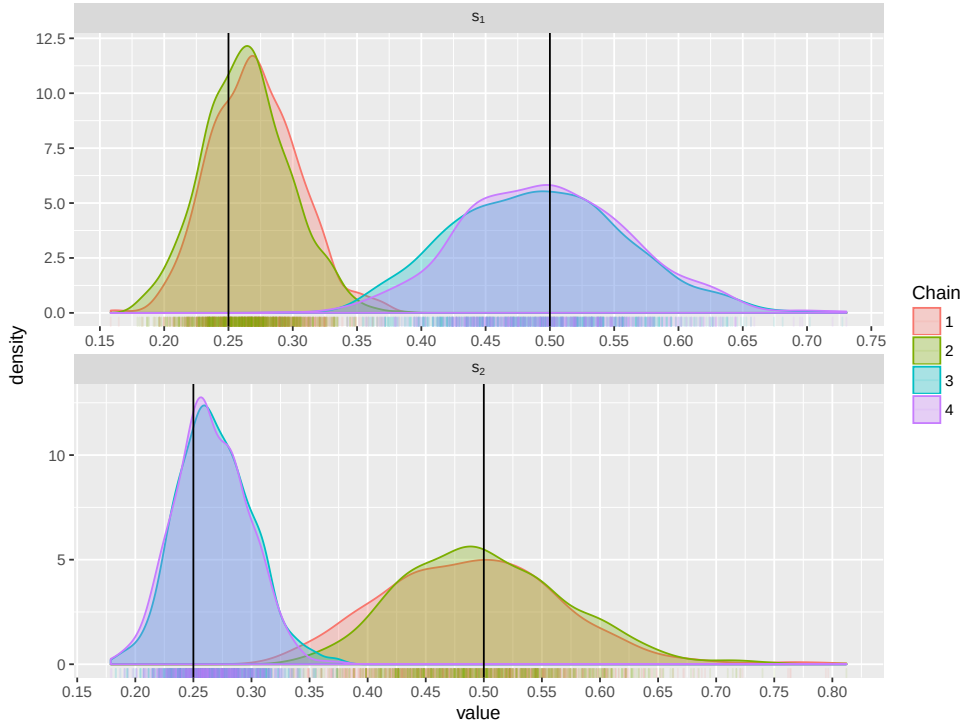


(b)

Figura 2.3: Traceplot dell'evoluzione per le componenti $\boldsymbol{\mu} = (\mu_1, \mu_2)$ (2.3(a)) e $\mathbf{s} = (s_1, s_2)$ (2.3(b)) delle quattro catene simulate.



(a)



(b)

Figura 2.4: Curve di densità per le componenti $\boldsymbol{\mu} = (\mu_1, \mu_2)$ (2.4(a)) e $\mathbf{s} = (s_1, s_2)$ (2.4(b)) delle quattro catene simulate.

2.3.2 Esempio notevole: inferenza bayesiana su modelli a mistura finita gaussiani multivariati e studio di simulazione

Un secondo esempio notevole che si è deciso di richiamare in questo lavoro riguarda lo schema inferenziale bayesiano per modelli a mistura finita, in cui le componenti sono distribuzioni gaussiane multivariate: questo non è nient'altro che una generalizzazione al caso $r > 1$ del modello affrontato nella Sottosezione 2.3.1, più verosimile al precedente nei casi applicativi in cui la generica osservazione y_i ha una natura multivariata.

La classe dei modelli a mistura finita con componenti gaussiane multivariate è di particolare interesse perchè gode della proprietà di approssimatore universale (Gelman et al. (2014)), secondo la quale è in grado di approssimare sufficientemente bene una qualunque $f(\mathbf{y})$ con un numero finito di componenti.

Considerato $(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)$ il vettore delle osservazioni a disposizione, diremo che $\mathbf{Y}$ a valori in $\mathcal{Y} \subset \mathbb{R}^r$ proviene da una distribuzione mistura finita con K componenti gaussiane multivariate (r-variate) se la funzione di densità $p(\mathbf{y})$ associata può essere scritta come

$$p(\mathbf{y}|\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K, \boldsymbol{\pi}) = \sum_{j=1}^{K} \pi_j \mathsf{N}_r\left(\mathbf{y}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right) \tag{2.37}$$

dove $\mathsf{N}_r\left(\mathbf{y}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)$ indica la funzione di densità per una gaussiana multivariata con vettore delle medie $\boldsymbol{\mu}_j$ e matrice di varianza-covarianza $\boldsymbol{\Sigma}_j$ calcolata in $\mathbf{y}$, e $\boldsymbol{\pi}$ l'usuale vettore contenente i pesi di ciascuna componente.

Uno schema inferenziale bayesiano per la stima dei parametri viene conseguito fissando le seguenti distribuzioni a priori:

$$\boldsymbol{\pi} \sim \mathsf{Dir}\left(\frac{\alpha_0}{K}, \ldots, \frac{\alpha_0}{K}\right), \tag{2.38}$$

$$Z_i|\boldsymbol{\pi} \sim \mathsf{Discreta}\left(K, \boldsymbol{\pi}\right) \quad i = 1, 2, \ldots, n, \tag{2.39}$$

$$(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \sim \mathsf{NIW}_r\left(\boldsymbol{\mu}_0, \kappa_0, \nu_0, \boldsymbol{\Sigma}_0\right) \quad j = 1, 2, \ldots, K \tag{2.40}$$

dove la notazione NIW_r sta ad indicare una distribuzione Gaussiana-Wishart Inversa. Formalmente, questa è definita come

$$\mathsf{NIW}_r\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}; \boldsymbol{\mu}_0, \kappa_0, \nu_0, \boldsymbol{\Sigma}_0\right) \equiv \mathsf{N}_r\left(\boldsymbol{\mu}; \boldsymbol{\mu}_0, \kappa_0^{-1}\boldsymbol{\Sigma}\right) \times \mathsf{IW}_r\left(\boldsymbol{\Sigma}; \boldsymbol{\Sigma}_0, \nu_0\right) \tag{2.41}$$

dove $\boldsymbol{\mu}_0$ è la media a priori per $\boldsymbol{\mu}$, κ_0 è il grado di *fiducia* che attribuiamo alla priori su $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}_0$ è proporzionale alla media a priori per $\boldsymbol{\Sigma}$ e ν_0, che rappresenta i gradi di libertà

per la distribuzione Wishart Inversa, rappresenta il grado di *fiducia* che attribuiamo alla priori su $\boldsymbol{\Sigma}$. Il vantaggio della scelta di una distribuzione a priori di questo tipo risiede nella coniugazione naturale garantita con la forma della verosimiglianza del modello, che comporta una distribuzione a posteriori:

$$(\boldsymbol{\mu}, \boldsymbol{\Sigma} | \mathbf{y}_1, \ldots, \mathbf{y}_n) \sim \mathsf{NIW}_r \left(\boldsymbol{\mu}_n, \kappa_n, \nu_n, \boldsymbol{\Sigma}_n\right) \tag{2.42}$$

dove l'aggiornamento degli iperparametri a posteriori avviene nel seguente modo (per i dettagli della derivazione, rimando al lavoro di Kamper (2013)):

$$\kappa_n = \kappa_0 + n; \tag{2.43}$$

$$\nu_n = \nu_0 + n; \tag{2.44}$$

$$\boldsymbol{\mu}_n = \kappa_n^{-1} \left(\kappa_0 \boldsymbol{\mu}_0 + \sum_{i=1}^{n} \mathbf{y}_i \right); \tag{2.45}$$

$$\boldsymbol{\Sigma}_n = \boldsymbol{\Sigma}_0 + \sum_{i=1}^{n} \mathbf{y}_i \mathbf{y}_i^T + \kappa_0 \boldsymbol{\mu}_0 \boldsymbol{\mu}_0^T - \kappa_n \boldsymbol{\mu}_n \boldsymbol{\mu}_n^T. \tag{2.46}$$

Il Gibbs Sampling per simulare dalla distribuzione a posteriori del modello mistura risulterà essere estremamente facilitato dal coniugio naturale tra la distribuzione a priori scelta e la forma della verosimiglianza del modello: si tratterà quindi unicamente di saper simulare da K distribuzioni NIW_r, una per ciascuna coppia $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, con iperparametri $\boldsymbol{\mu}_{n_j}$, κ_{n_j}, ν_{n_j} e $\boldsymbol{\Sigma}_{n_j}$ dove il pedice n_j sta ad indicare che queste vengono aggiornate solo ed unicamente con le $\mathbf{y}_i$ corrispondenti a $Z_i = j$, per $j = 1, 2, \ldots, K$. La generazione avverrà simulando $\boldsymbol{\Sigma}_j^{(t)} \sim \mathsf{IW}_r \left(\boldsymbol{\Sigma}_{n_j}, \nu_{n_j}\right)$ e successivamente $\boldsymbol{\mu}_j^{(t)} \sim \mathsf{N}_r \left(\boldsymbol{\mu}_{n_j}, \kappa_{n_j}^{-1} \boldsymbol{\Sigma}_j^{(t)}\right)$.

L'unico passo genuino di Gibbs Sampling riguarda l'aggiornamento del vettore di latenti $\mathbf{Z}$: questo avverrà, per $i = 1, 2, \ldots, n$ nella t-esima iterazione dell'algoritmo al solito modo:

$$Z_i^{(t)} = j | \mathbf{y}_i, \mathbf{Z}_{-i}^{(t)}, \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)} \sim \frac{n_{j,-i}^{(t)} + \alpha_0 / K}{n + \alpha_0 - 1} \mathsf{N}_r \left(\mathbf{y}_i; \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)}\right) \quad j = 1, 2, \ldots, K. \tag{2.47}$$

Un'implementazione dell'algoritmo è disponibile in Codice A.3: nonostante formalmente basti implementare solo ed unicamente il passo di aggiornamento di $\mathbf{Z}$, disponendo della coniugazione naturale tra distribuzione a priori e verosimiglianza del modello che consente di generare facilmente dalla distribuzione a posteriori, è stato incluso nell'algoritmo anche il salvataggio della catena dei valori simulati per $\boldsymbol{\mu}_1^{(t)}, \boldsymbol{\mu}_2^{(t)}, \ldots, \boldsymbol{\mu}_K^{(t)}$. Per quanto riguarda le matrici di varianza e covarianza, invece, data l'elevata complessità che comportano nello *storing* l'algoritmo restituisce solo le ultime $\boldsymbol{\Sigma}_1^{(R)}, \boldsymbol{\Sigma}_2^{(R)}, \ldots, \boldsymbol{\Sigma}_K^{(R)}$ simulate.

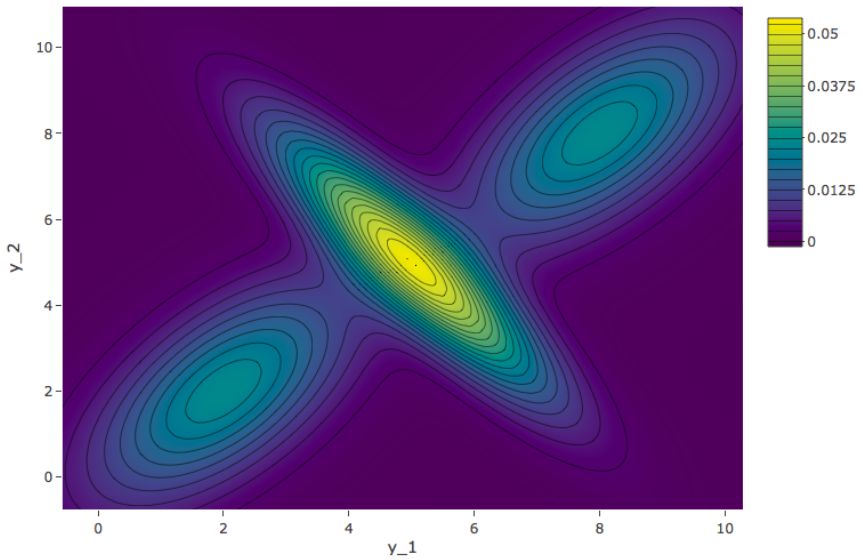


Figura 2.5: Curve di livello per la funzione di densità della mistura definita in Equazione (2.48).

Un esempio applicativo che aiuta a comprendere la natura del modello è di seguito riportato. Sono state generate $n = 2000$ osservazioni da una distribuzione mistura composta da $K = 3$ gaussiane bivariate

$$\mathbf{Y} \sim 0.3\,\mathsf{N}_2\left(\begin{bmatrix}2\\2\end{bmatrix}, \begin{bmatrix}3 & 1.5\\1.5 & 2\end{bmatrix}\right) + 0.4\,\mathsf{N}_2\left(\begin{bmatrix}5\\5\end{bmatrix}, \begin{bmatrix}4 & -2.5\\-2.5 & 2\end{bmatrix}\right) + 0.3\,\mathsf{N}_2\left(\begin{bmatrix}8\\8\end{bmatrix}, \begin{bmatrix}3 & 1.5\\1.5 & 2\end{bmatrix}\right) \tag{2.48}$$

rappresentata graficamente tramite le sue curve di livello in Figura 2.5.

La mistura è stata volutamente generata di modo che le 3 componenti fossero sufficientemente ben distinte l'una dall'altra, con quella centrale caratterizzata da una disparità di variabilità nelle due variabili, che la rende sufficientemente *stretta* ma con una *vetta* ben più alto rispetto a quello delle altre due componenti.

Il Gibbs Sampling per simulare dalla distribuzione a posteriori è stato iterato su $R = 1000$ iterazioni, scartando il primo 20% di burn-in, con un vettore di partenza $\mathbf{Z}_0$ in cui ciascuna latente associata all'osservazione è stata assegnata casualmente ad una delle 3 componenti della mistura. Il codice dell'intera simulazione è riportato in Codice B.3.

E' stato già detto come in quest'ultimo algoritmo, data la coniugazione naturale della distribuzione a priori per $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ con la forma della verosimiglianza, sia possibile simulare direttamente dalla distribuzione a posteriori per ciascuna componente della mistura senza

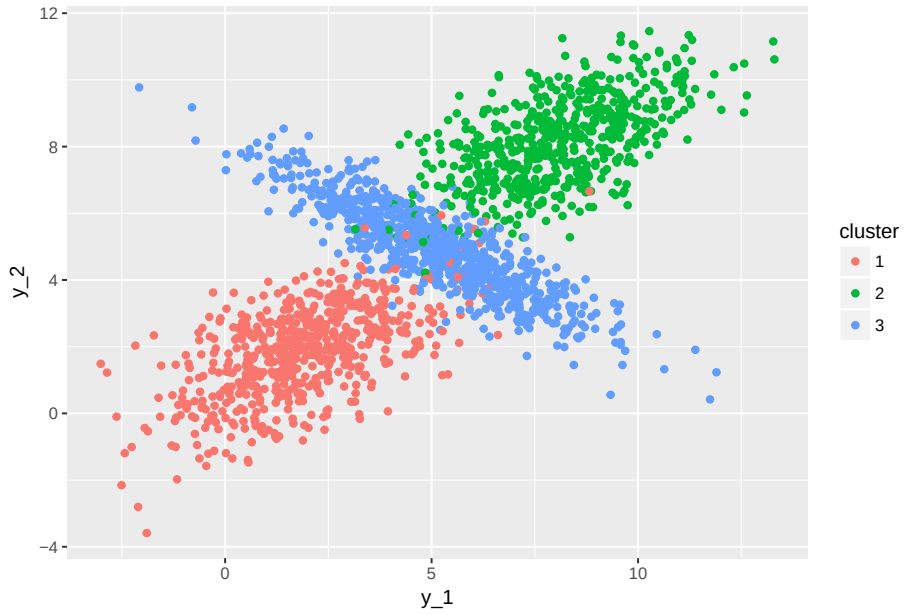


Figura 2.6: Clustering delle osservazioni $\mathbf{y}_i$ sulla base di $\mathbf{Z}^{(R)}$, $i = 1, 2, \ldots, n$.

ricorrere ad alcun passo del Gibbs Sampling: quest'ultimo risulta invece necessario per realizzare il *clustering* di ciascuna osservazione ad una delle 3 componenti della mistura.

In Figura 2.6 è stato rappresentato l'insieme delle osservazioni $\mathbf{y}_i = (y_{i1}, y_{i2})$ generate dalla distribuzione mistura definita nella (2.48), colorando ciascuna di queste con un colore diverso a seconda del cluster di appartenenza nel vettore $\mathbf{Z}^{(R)}$ corrispondente all'ultima iterazione del Gibbs Sampler: a meno delle osservazioni posizionate nelle *conche* tra le mode di una e dell'altra gaussiana, la convergenza sembra essere stata raggiunta in modo soddisfacente ed l'assegnazione di ciascuna osservazione alla rispettiva componente dalla quale è stata generata risulta essere complessivamente corretta.

Allo stesso modo, anche senza mostrare i risultati ottenuti, è garantita per la posteriori di ciascuna coppia $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ la corretta concentrazione attorno ai veri valori dai quali è stata generata la mistura. Valgono nuovamente i ragionamenti già fatti circa l'opportunità di tenere in considerazione la proprietà di scambiabilità per la distribuzione mistura, e di conseguenza il significato che attribuiamo al termine *convergenza* in questo specifico contesto: quest'ultima è raggiunta, ancora una volta, a meno di permutazioni interne sull'ordine delle etichette delle classi $j = 1, 2, \ldots, K$.

Capitolo 3

Modelli mistura a componenti gaussiane multivariate per Support Vector Machines

In questo capitolo viene presentata la classe dei modelli a mistura finita con componenti gaussiane multivariate per Support Vector Machines per problemi di classificazione: nella Sezione 3.1 vengono riportate le considerazioni che hanno spinto lo sviluppo di questa classe di modelli, formalizzata nella Sezione 3.2 congiuntamente alle tecniche e procedure inferenziali che ne consentono la stima in Sezione 3.3. In aggiunta, nella Sezione 3.4 vengono illustrati alcuni esempi applicativi su dataset simulati, assieme alle opportune tecniche che consentono di svolgere la classificazione.

3.1 Motivazione e contestualizzazione dei modelli

L'estensione delle Support Vector Machines che viene proposta in questo lavoro riguarda la formalizzazione di una classe di modelli che ne considera la mistura in un numero finito di componenti, specificandone un'opportuna forma parametrica, assieme alle relative tecniche che consentono un'approccio inferenziale di natura bayesiana. La motivazione che giustifica la potenziale utilità di questa classe di modelli è legata alla sempre più eterogenea natura dei problemi di classificazione che ci si trova ad affrontare: molto spesso i fenomeni che si intende studiare e per i quali si vuole proporre una modellazione adeguata sono particolarmente complessi e coinvolgono *pattern* latenti o sottopopolazioni con caratteristi-

che particolari, per le quali i principali modelli noti in letteratura potrebbero non riuscire a cogliere completamente l'intera natura del fenomeno, se non andando a complicare la struttura stessa del modello e rischiando di ricadere in casi di sovradattamento.

La possibilità di considerare una mistura finita di SVM tenta di affrontare queste specifiche problematiche partizionando il problema in *sotto-cluster* con caratteristiche simili, ed adattando su ciascuno di questi una opportuna SVM di classificazione: anzichè considerare un modello generale sull'intero insieme di dati a nostra disposizione, si considerano quindi delle SVM *esperte* ciascuna su un *cluster* latente, da combinare assieme assegnando a ciascuna uno specifico peso per ottenere un modello finale.

La prima formalizzazione che è stata perseguita nel lavoro che accompagna questa tesi, ma poi scartata, ricalca la possibilità di considerare per queste rappresentazioni una mistura finita, e su questa sviluppare le procedure di inferenza bayesiana viste nella Sezione 2.1. Un aspetto cruciale nella formalizzazione di un modello mistura riguarda la scelta della forma distributiva da assegnare a ciascuna componente: un modo che è sembrato inizialmente immediato e coerente con quanto già visto nella Sezione 1.2 è stato quello di modellare ciascuna componente con la pseudo-verosimiglianza, formalizzando un modello del tipo

$$\begin{aligned} p(y_i|\boldsymbol{\pi}, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K, \mathbf{x}_i) &= \sum_{j=1}^{K} \pi_j \, p(y_i|\boldsymbol{\beta}_j, \mathbf{x}_i) \\ &= \sum_{j=1}^{K} \pi_j \, \exp\left\{-2 \max\left(1 - y_i \mathbf{x}_i^T \boldsymbol{\beta}, 0\right)\right\}. \end{aligned} \tag{3.1}$$

Un approccio di questo tipo si è rilevato fallace nei casi applicativi per una serie di motivi. Il primo riguarda la natura della classe dei modelli SVM, che non presuppone alcuna formalizzazione di natura probabilistica circa la legge $p(y_i|\boldsymbol{\beta}, \mathbf{x}_i)$: sebbene l'*escamotage* di definire quest'ultima come la trasformata esponenziale della funzione di perdita cambiata di segno sia valida per formalizzare il modello in ottica bayesiana, ovvero per attribuire una forma alla verosimiglianza e dunque svolgere l'inferenza, risulta concettualmente sbagliato attribuirne il significato di *distribuzione per* $Y_i|\mathbf{x}_i, \boldsymbol{\beta}_j$ e dunque utilizzarla per modellare la componente probabilistica di ciascuna componente. Il secondo aspetto riguarda l'opportunità di modellare $Y_i|\mathbf{x}_i, \boldsymbol{\beta}_j$: operando questa scelta si andrebbe a forzare un *clustering* sulle modalità assunte dalla variabile risposta, e non su particolari conformazioni legate a possibili strutture latenti assunte dai regressori; trattandosi Y_i di una risposta a natura dicotomica, un modello così formalizzato andrebbe erroneamente a caratterizzare $K = 2$

cluster latenti, assegnando ciascuna $\mathbf{x}_i$ ad uno o all'altro cluster a seconda della modalità y_i associata, e non sulla base della natura di $\mathbf{x}_i$. La terza ragione riguarda il fine previsivo del modello; in particolare, non risulta immediato definire un metodo che sfrutti la struttura del modello per allocare una nuova osservazione $\tilde{\mathbf{x}}$ ad uno dei K cluster e dunque ottenere la classificazione dal classificatore associato alla SVM stimata sul cluster j-esimo: questo perchè la pseudo-verosimiglianza del modello contiene inevitabilmente $\tilde{y}_i$, ignoto.

Queste ragioni, supportate dall'evidenza empirica emersa in casi applicativi, hanno suggerito una diversa contestualizzazione del problema e, di conseguenza, una formalizzazione più opportuna per tenere conto delle motivazioni accennate all'inizio della Sezione.

3.2 Formalizzazione del modello

Si supponga di disporre di un insieme di dati $\{(y_i, \mathbf{x}_i)\}_{i=1}^n$, con generica $(y_i, \mathbf{x}_i) \in \{-1, 1\} \times \mathbb{R}^p$, dove y_i rappresenta la codifica della modalità assunta dalla variabile risposta e $\mathbf{x}_i$ rappresenta, al solito, il vettore delle covariate.

Riguardo il meccanismo probabilistico che ha generato i dati, si suppone che la generica coppia $(y_i, \mathbf{x}_i)$ provenga da un vettore casuale

$$(Y, \mathbf{X}) \sim f^0_{Y,\mathbf{X}}(y, \mathbf{x}) = f^0_Y(y|\mathbf{X} = \mathbf{x})\, f^0_{\mathbf{X}}(\mathbf{x}), \tag{3.2}$$

dove con $f^0_{\mathbf{X}}(\cdot)$ si denota la vera ed ignota legge che si suppone abbia generato le covariate e con $f^0_Y(y|\mathbf{X} = \mathbf{x})$ l'ignoto classificatore.

La formalizzazione della classe di modelli mistura finita a componenti gaussiane multivariate di SVM per problemi di classificazione avviene:

- Ipotizzando che la generica $\mathbf{x}_i$ (priva di eventuali funzioni di base relative ad espansioni kernel dello spazio dei regressori) sia stata generata dal vettore casuale $\mathbf{X} \sim f_{\mathbf{X}}(\mathbf{x})$, con $f_{\mathbf{X}}(\mathbf{x})$ definita come

$$f_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{X}}\left(\mathbf{x}; \{\boldsymbol{\mu}_j\}_{j=1}^K, \{\boldsymbol{\Sigma}_j\}_{j=1}^K\right) = \sum_{j=1}^K \pi_j \mathsf{N}_p(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \tag{3.3}$$

  che specifica incorrettamente la vera ed ignota $f^0_{\mathbf{X}}(\mathbf{x})$ o equivalentemente sfruttando la nota rappresentazione aumentata come

$$f_{\mathbf{X}}(\mathbf{x}) = \sum_{j=1}^K p_{\mathbf{X}}(\mathbf{x}|j)\, p_Z(Z = j), \tag{3.4}$$

dove $p_{\mathbf{X}}(\mathbf{x}|j) = \mathsf{N}_p(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ e $p_Z(Z = j) = \mathsf{Discreta}(j; K, \boldsymbol{\pi}) = \pi_j$, K fissato. In altre parole, si ipotizza che $\mathbf{x}$ provenga da una mistura finita di distribuzioni gaussiane multivariate;

- Definita la funzione di perdita *hinge loss* $\mathcal{L}_H\left(Y, \hat{G}(\mathbf{X})\right) = \max\left(1 - Y\mathbf{X}^T\boldsymbol{\beta}, 0\right)$ valida per un generico modello SVM per problemi di classificazione, si definisce la funzione di perdita *hinge loss aumentata* come

$$\mathcal{L}_H^{\text{augm}}\left(Y, \hat{G}(\mathbf{X}, Z)\right) = \max\left(1 - Y\mathbf{X}^T\boldsymbol{\beta}_Z, 0\right). \tag{3.5}$$

A questo punto è possibile ricavare l'*expected classification error (ECE)* (Hastie, Tibshirani e Friedman (2009), Sec 2.4) della *hinge loss aumentata* come

$$\begin{aligned}
ECE &= \mathbb{E}_{(Y,\mathbf{X},Z)}\left[\mathcal{L}_H^{\text{augm}}\left(Y, \hat{G}(\mathbf{X}, Z)\right)\right] && (3.6)\\
&= \mathbb{E}_{(\mathbf{X},Z)}\left[\mathbb{E}_Y\left[\mathcal{L}_H^{\text{augm}}\left(Y, \hat{G}(\mathbf{x}, z)\right)\right]\right] && (3.7)\\
&= \mathbb{E}_{(\mathbf{X},Z)}\left[\sum_{k\in\{-1,1\}} \mathcal{L}_H^{\text{augm}}\left(k, \hat{G}(\mathbf{x}, z)\right) \Pr(Y = k|\mathbf{x}, z)\right] && (3.8)\\
&= \int \sum_{j=1}^{K} \sum_{k\in\{-1,1\}} \mathcal{L}_H^{\text{augm}}\left(k, \hat{G}(\mathbf{x}, j)\right) \Pr(Y = k|\mathbf{x}, j)\, p_{\mathbf{X}}(\mathbf{x}|j)\, p_Z(Z = j)\; d\mathbf{x} && (3.9)\\
&= \int \sum_{j=1}^{K} \sum_{k\in\{-1,1\}} \mathcal{L}_H^{\text{augm}}\left(k, \hat{G}(\mathbf{x}, j)\right) \Pr(Y = k|\mathbf{x}, j)\, \mathsf{N}_p(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\, \pi_j\, d\mathbf{x}. && (3.10)
\end{aligned}$$

Il classificatore $\hat{G}(\mathbf{x}, z)$ *ottimale* per il modello proposto è quello che permette di minimizzare la *ECE*: questo è ricavabile andandola a minimizzare punto per punto

$$\hat{G}(\tilde{\mathbf{x}}, \tilde{z}) = \underset{g\in\{-1,1\}}{\arg\min} \sum_{k\in\{-1,1\}} \mathcal{L}_H(k, g) \Pr(Y = k|\mathbf{X} = \tilde{\mathbf{x}}, Z = \tilde{z}); \tag{3.11}$$

avendo marginalizzato $(\mathbf{x}, z)$, questo corrisponderà esattamente al classificatore associato alla z-esima SVM:

$$\hat{G}(\tilde{\mathbf{x}}, \tilde{z}) = \text{sign}\left(\tilde{\mathbf{x}}^T\boldsymbol{\beta}_{\tilde{z}}\right) \tag{3.12}$$

ed è quello che verrà utilizzato per classificare la generica $\tilde{\mathbf{x}}$ a patto di conoscere il suo cluster di appartenenza: maggiori dettagli circa quest'ultima problematica e l'operatività del classificatore in problemi concreti sono dati nella Sottosezione 3.4.

La scelta di ipotizzare una struttura probabilistica a sotto-popolazioni latenti sulla distribuzione che ha generato le $\mathbf{x}_i$ rappresenta la principale distinzione con la prima classe di modelli che è stata pensata, nella quale si andava ad operare un clustering includendo anche la modalità assunta dalla variabile risposta associata y_i; si ritiene che questo secondo metodo sia più *fedele* alle motivazioni espresse nella Sezione 3.1, facendo in modo che il *clustering* avvenga esclusivamente sulla base della conformazione che le $\mathbf{x}_i$ assumono sullo spazio dei regressori: la modalità assunta dalla variabile risposta y_i associata non entra dunque in alcun modo nella fase di *clustering* della struttura dei dati, ma verrà usata esclusivamente per la stima della SVM associata a ciascun cluster.

Implicitamente, la scelta di specificare (erroneamente) la vera ed ignota $f^0(\mathbf{x})$ che ha generato le $\mathbf{x}_i$ come una distribuzione mistura a componenti gaussiane sottintende tre ipotesi. La prima è che $\mathbf{x}_i \sim p_j(\mathbf{x})$, ovvero che ciascuna osservazione $\mathbf{x}_i$ proviene da una tra K possibili distribuzioni latenti, la j-esima $p_j(\mathbf{x})$, e che K sia un numero fissato a priori a seconda di possibili informazioni di natura teorica e/o esplorativa possedute dal ricercatore: questa ipotesi è in linea con le considerazioni già svolte circa la possibile esistenza di particolari conformazioni dei dati derivanti da sottopopolazioni latenti. La seconda ipotesi è che ciascuna $p_j(\mathbf{x})$ appartenga ad una famiglia di distribuzioni parametriche, che per comodità e senza perdita di generalità si assume essere la stessa per ciascuna componente, ovvero che $p_j(\mathbf{x}) \in \mathcal{F} = \{p(\mathbf{x}|\boldsymbol{\theta}_j) \mid \boldsymbol{\theta}_j \in \Theta\}$: questa ipotesi consente di lavorare all'interno del contesto inferenziale bayesiano parametrico, ma nulla vieta la possibilità di estendere lo schema inferenziale al caso bayesiano nonparametrico in cui $\dim(\Theta) = \infty$. La terza ipotesi, che attribuisce l'aggettivo *gaussiani multivariati* al nome per questa classe di modelli riguarda la scelta distributiva per ciascuna $p(\mathbf{x}|\boldsymbol{\theta}_j)$: ipotizziamo altresì che $\mathbf{X}|Z = j \sim \mathsf{N}_p(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$.

Questa terza ipotesi risulta essere, tra tutte, indubbiamente quella più stringente: si sta imponendo una particolare forma distributiva, sferica o tutt'alpiù ovaliode, a ciascuna popolazione latente; nonostante questa evidente restrizione, la distribuzione gaussiana multivariata è stata prediletta ad altre distribuzioni multivariate per via della sua più agevole maneggiabilità, e sempre facendo riferimento alla proprietà di approssimatore universale possieduta dalla mistura finita di gaussiane multivariate per la quale si suppone sia in grado

di approssimare sufficientemente bene la vera ed ignota $f^0(\mathbf{x})$. Il rilassamento di ciascuna delle tre ipotesi formulate viene trattato adeguatamente nel Capitolo 5.

Fatte queste dovute premesse, il modello che si intende formalizzare può essere più intuitivamente sintetizzato come un generico modello mistura a componenti finite gaussiane multivariate, una per ciascuna delle K sottopopolazioni latenti; sul cluster j-esimo si andrà a stimare una SVM *esperta* sulla conformazione di quel preciso cluster, utilizzando come insieme di stima tutte le coppie $(\mathbf{x}_i, y_i)$ per le quali, in fase di *clustering*, è stato assegnato $Z_i = j$.

3.3 Tecniche inferenziali nel paradigma bayesiano

Per quanto riguarda l'inferenza bayesiana sul modello, questa viene svolta richiamando le tecniche già descritte nella Sezione 1.2 e nella Sottosezione 2.3.2, rispettivamente per l'inferenza bayesiana su SVM per problemi di classificazione e su modelli mistura a componenti gaussiane multivariate. In particolare, per ciascuno dei parametri e delle variabili latenti incluse nel modello si specificano le seguenti distribuzioni a priori:

$$(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \sim \mathsf{NIW}_p(\boldsymbol{\mu}_0, \kappa_0, \nu_0, \boldsymbol{\Sigma}_0) \quad j = 1, 2, \ldots, K \tag{3.13}$$

$$\boldsymbol{\pi} \sim \mathsf{Dir}\left(\frac{\alpha_0}{K}, \ldots, \frac{\alpha_0}{K}\right) \tag{3.14}$$

$$Z_i|\boldsymbol{\pi} \sim \mathsf{Discreta}(K, \boldsymbol{\pi}) \quad i = 1, 2, \ldots, n \tag{3.15}$$

$$\nu_j^{-1} \sim \mathsf{Gamma}(a_\nu, b_\nu) \quad j = 1, 2, \ldots, K \tag{3.16}$$

$$\omega_{hj} \sim p(\omega_{hj}|\alpha) \propto \omega_{hj}^{-3/2} \mathsf{St}^+_{\alpha/2}\left(\omega_{hj}^{-1}\right) \quad h = 1, 2, \ldots, p \quad j = 1, 2, \ldots, K \tag{3.17}$$

$$\boldsymbol{\beta}_j|\nu_j, \boldsymbol{\omega}_j \sim \mathsf{N}_p\left(\mathbf{0}, \nu_j^2 \boldsymbol{\Omega}_j^{1/2} \boldsymbol{\Sigma} \boldsymbol{\Omega}_j^{1/2}\right) \quad j = 1, 2, \ldots, K \tag{3.18}$$

$$\lambda_i \sim 1 \quad \forall i = 1, 2, \ldots, n, \tag{3.19}$$

dove $\boldsymbol{\Omega}_j = \text{diag}(\boldsymbol{\omega}_j)$ per $j = 1, 2, \ldots, K$. Rispetto a quanto già visto nella Sezione 1.2, disponendo ora di K distinti $\boldsymbol{\beta}_j$ andrà specificata una struttura gerarchica a variabili latenti distinta per ciascun coefficiente: questo giustifica l'introduzione del pedice j.

Per quanto riguarda la forma della verosimiglianza del modello, facendo riferimento alla rappresentazione aumentata proposta nella (1.14) per il singolo contributo alla *pseudo-verosimiglianza* di una SVM per problemi di classificazione, e della (2.12) per il modello mistura aumentato, questa può essere scritta come

$$\prod_{i=1}^{n}\prod_{j=1}^{K}\left\{\pi_j \mathsf{N}\left(1-\lambda_i; y_i\mathbf{x}_i^T\boldsymbol{\beta}_j, \lambda_i\right)\mathsf{N}_p\left(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)\right\}^{\mathbb{1}(Z_i=j)}. \tag{3.20}$$

Il modello genuinamente bayesiano che deriverebbe dall'utilizzo di questa verosimiglianza richiederebbe anche l'utilizzo di y_i per l'aggiornamento della full-conditional di ciascuna latente Z_i come

$$p(Z_i = j|\mathbf{x}_i, \pi_j) \propto \pi_j \mathsf{N}_p\left(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)\mathsf{N}\left(1-\lambda_i; y_i\mathbf{x}_i^T\boldsymbol{\beta}_j, \lambda_i\right), \tag{3.21}$$

risollevando nuovamente le motivazioni che ci hanno spinto ad escludere l'utilizzo della versione aumentata per svolgere il *clustering*, secondo quanto già espresso dalla forma della verosimiglianza proposta nella 3.1.

La formalizzazione alternativa che viene proposta in questo Capitolo è sintetizzabile dal DAG in Figura 3.1. Questo diagramma consente di ricavare facilmente tutte le relazioni di dipendenza condizionata tra tutte le variabili aleatorie che compongono il modello: in particolare, la presenza di una freccia che collega un nodo ad un altro sottintende il fatto che nel ricavare la full-conditional per il nodo di partenza si utilizzano tutti i nodi appartenenti al suo *markov blanket*, secondo quanto già detto nel Capitolo 2. La freccia tratteggiata in Figura 3.1 simboleggia una relazione di dipendenza imposta dal modello che è stato formalizzato, per la quale però non si considera l'usuale paradigma di aggiornamento della full-conditional a posteriori per il nodo di partenza Z_i imposto dal suo *markov blanket*, che corrisponderebbe alla (3.21). La motivazione è dovuta alla necessità di voler utilizzare $\mathbf{Z}$ per identificare le coppie $(y_i, \mathbf{x}_i)$ con le quali eseguire l'aggiornamento della full-conditional per ciascun $\boldsymbol{\beta}_j$ e, allo stesso tempo, di voler escludere y_i dalle quantità condizionanti nella full-conditional per l'aggiornamento di ciascuna Z_i, per le motivazioni appena richiamate.

La scelta di *tratteggiare la freccia* che in Figura 3.1 punta dal nodo Z_i al nodo y_i è formalmente giustificabile seguendo un filone di letteratura emergente in ambito bayesiano,

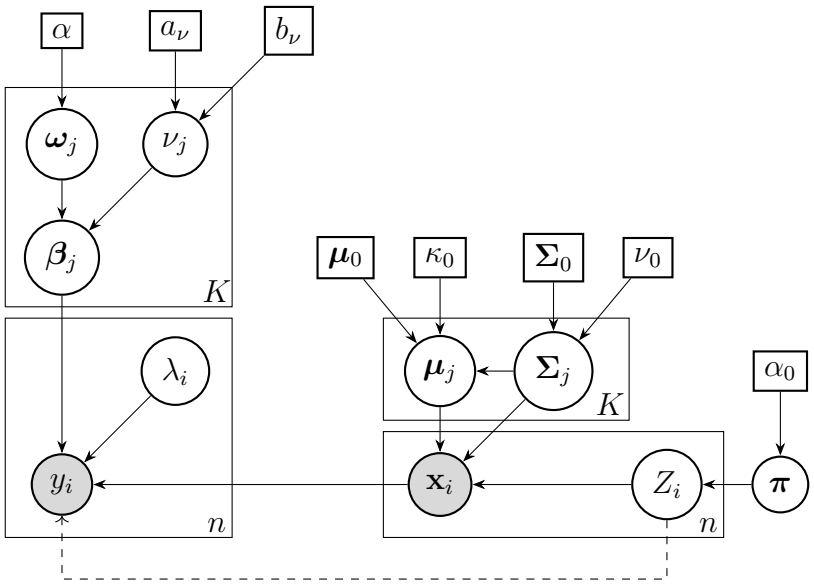


Figura 3.1: DAG del modello mistura gaussiano multivariato per SVM di classificazione.

che trova la sua sintesi nei lavori di Jacob et al. (2017) e di Liu, Bayarri e Berger (2009); seguendo l'approccio proposto, è consentito operare una *modularizzazione* del modello *congiunto* (ottenibile dal DAG in Figura 3.1 se la freccia fosse continua, utilizzando il classico aggiornamento dovuto al teorema di Bayes) in più modelli *condizionati*, i quali a loro volta non consentono di ricostruirne la versione *congiunta*. Jacob et al. (2017) non si limitano a giustificare teoricamente l'approccio basato sulla modularizzazione di un modello congiunto, bensì dimostrano anche che in alcuni casi, tra i quali può essere incluso quello specifico facente riferimento al modello che si intende formalizzare, l'approccio proposto è preferibile a quello *genuinamente bayesiano* basato sul modello *completo*.

Con riferimento al modello proposto, l'approccio di modularizzazione si compone di un modello *marginale*, che fa riferimento al passo di *clustering* riferito alla componente mistura e basato su

$$p\left(\mathbf{Z}, \{(\boldsymbol{\mu}, \boldsymbol{\Sigma})\}_{j=1}^{K} \,\middle|\, \{\mathbf{x}_i\}_{i=1}^{n}\right) \propto \mathcal{L}\left(\{\mathbf{x}_i\}_{i=1}^{n} \,\middle|\, \{(\boldsymbol{\mu}, \boldsymbol{\Sigma})\}_{j=1}^{K}, \mathbf{Z}\right) p(\mathbf{Z})\, p\left(\{(\boldsymbol{\mu}, \boldsymbol{\Sigma})\}_{j=1}^{K}\right), \quad (3.22)$$

e di un modello *condizionato* riferito al passo di stima di ciascuna SVM, basato su:

$$p\left(\{\boldsymbol{\beta}\}_{j=1}^{K}, \boldsymbol{\lambda}, \{\boldsymbol{\omega}\}_{j=1}^{K}, \boldsymbol{\nu} \,\middle|\, \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}\, \hat{\mathbf{Z}}\right) \propto \mathcal{L}\left(\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}, \hat{\mathbf{Z}} \,\middle|\, \{\boldsymbol{\beta}\}_{j=1}^{K}, \boldsymbol{\lambda}, \{\boldsymbol{\omega}\}_{j=1}^{K}, \boldsymbol{\nu}\right) \times p(\{\boldsymbol{\beta}\}_{j=1}^{K})\, p(\boldsymbol{\lambda})\, p(\{\boldsymbol{\omega}\}_{j=1}^{K})\, p(\boldsymbol{\nu}) \quad (3.23)$$

dove con $\hat{\mathbf{Z}}$ si indica il vettore delle etichette dei cluster stimati nel modello *marginale*. Un approccio simile è contenuto in Woodard, Crainiceanu e Ruppert (2013).

A partire da questi modelli si possono ricavare le *full-conditional* elencate di seguito, secondo quanto già visto in precedenza nella Sottosezione 2.3.2 per la componente del modello relativa alla mistura e nella Sezione 1.2 per la componente riguardante la SVM di classificazione: per questo secondo caso, trattando una mistura di K distinte SVM è stato necessario specificare nelle formule di aggiornamento degli iperparametri per le distribuzioni a posteriori dei parametri associati alla j-esima componente che queste avvengono considerando solo ed esclusivamente le osservazioni $\mathbf{x}_i$ (e dunque le associate y_i) clusterizzate nella j-esima componente.

$$(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \,|\, \{\mathbf{x}_i\}_{i=1}^n, \mathbf{Z} \sim \mathsf{NIW}_p\left(\boldsymbol{\mu}_{n_j}, \kappa_{n_j}, \nu_{n_j}, \boldsymbol{\Sigma}_{n_j}\right) \quad j = 1, 2, \ldots, K \tag{3.24}$$

$$\boldsymbol{\pi}|\mathbf{Z} \sim \mathsf{Dir}\left(\frac{\alpha_0}{K} + n_1, \ldots, \frac{\alpha_0}{K} + n_K\right) \tag{3.25}$$

$$Z_i|\mathbf{x}_i, \boldsymbol{\pi} \sim \mathsf{Discreta}\left(K, \boldsymbol{\pi}^*\right) \quad \text{con} \quad \pi_j^* \propto \pi_j \mathsf{N}_p\left(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right) \quad i = 1, 2, \ldots, n \tag{3.26}$$

$$\nu_j^{-1}|\boldsymbol{\beta}_j \sim \mathsf{Gamma}\left(a_\nu + p, b_\nu + \sum_{h=1}^{p}\left|\frac{\beta_{hj}}{\sigma_h}\right|\right) \quad j = 1, 2, \ldots, K \tag{3.27}$$

$$\omega_{hj}|\boldsymbol{\beta}_j, \nu_j, \alpha \sim \begin{cases} \mathsf{GenInvGauss}\left(\frac{1}{2}, 1, \frac{\beta_{hj}^2}{\nu_j^2 \sigma_j^2}\right) & \text{se} \quad \alpha = 1 \\ 1 & \text{se} \quad \alpha = 2 \end{cases} \quad \forall h = 1, 2, \ldots, p \quad j = 1, 2, \ldots, K \tag{3.28}$$

$$\boldsymbol{\beta}_j|\left\{(y_i, \mathbf{x}_i)\right\}_{i=1}^n, \mathbf{Z}, \boldsymbol{\omega}_j, \nu_j, \boldsymbol{\lambda} \sim \mathsf{N}_p\left(\mathbf{b}_j, \mathbf{B}_j\right) \quad j = 1, 2, \ldots, K \tag{3.29}$$

$$\lambda_i|\left(y_i, \mathbf{x}_i\right), Z_i, \{\boldsymbol{\beta}_j\}_{j=1}^K \sim \mathsf{GenInvGauss}\left(\frac{1}{2}, 1, \left(1 - y_i \mathbf{x}_i^T \boldsymbol{\beta}_{Z_i}\right)^2\right) \quad i = 1, 2, \ldots, K, \tag{3.30}$$

dove $\mathbf{B}_j^{-1} = \nu_j^{-2}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Omega}_j^{-1} + \tilde{\mathbf{X}}_j^T \boldsymbol{\Lambda}_j^{-1} \tilde{\mathbf{X}}_j$ e $\mathbf{b}_j = \mathbf{B}_j \tilde{\mathbf{X}}_j^T \left(\mathbf{1}_{n_j} + \boldsymbol{\lambda}_j^{-1}\right)$.

A livello intuitivo, il modello lavorerà quindi su due livelli distinti nonostante la stima avvenga congiuntamente secondo l'approccio modularizzato:

1. Livello di *clustering*: Si suppone $p\left(\mathbf{x} \,\middle|\, \boldsymbol{\pi}, \{\boldsymbol{\mu}_j\}_{j=1}^K, \{\boldsymbol{\Sigma}_j\}_{j=1}^K\right) = \sum_{j=1}^K \pi_j \mathsf{N}_p\left(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)$ e si provvede a svolgere l'inferenza sul modello, seguendo i passaggi già ampiamente descritti nella Sottosezione 2.3.2. Questo permetterà di ottenere il vettore $\mathbf{Z} = (Z_1, \ldots, Z_n)$ con le etichette del *cluster* al quale è stata assegnata ciascuna $\mathbf{x}_i$;

2. Livello di stima delle SVM: Si stimano K distinte SVM, utilizzando come insieme di stima per la j-esima tutte e solo le coppie $(\mathbf{x}_i, y_i)$ per cui $Z_i = j$ nel livello di *clustering*, seguendo le tecniche inferenziali già accennate nel Capitolo 1. La stima produrrà K distinti vettori di parametri $\boldsymbol{\beta}_j$, uno per ciascuna delle K diverse SVM.

Relativamente al *clustering* delle osservazioni $\mathbf{x}_i$, svolto a posteriori dalle *full-conditional* (3.24)-(3.26), va notato come l'unico interesse per la stima del modello mistura riguardi esclusivamente il vettore delle latenti $\mathbf{Z}$ contenente l'etichetta del *cluster* al quale è stata ricondotta ciascuna delle $\mathbf{x}_i$ dell'insieme di stima: in altre parole, ai fini della stima del modello non è di alcun interesse la stima dei parametri $\boldsymbol{\mu}_j$ e $\boldsymbol{\Sigma}_j$ che caratterizzano ciascuna delle K distribuzioni che compongono la mistura; questo aspetto si ripercuote nella procedura di Gibbs Sampling per la stima della mistura, per la quale il passo di simulazione di ciascuna $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ risulta fondamentalmente essere solo una fonte di maggior dispendio di risorse computazionali. Questa caratteristica del modello rende la scelta delle componenti gaussiane multivariate particolarmente adatta per il nostro scopo, approfittando della possibilità di ricavare analiticamente una forma chiusa per la marginalizzazione di $\boldsymbol{\mu}_j$ e $\boldsymbol{\Sigma}_j$ dalla funzione di densità che la caratterizza.

Seguendo le motivazioni esposte all'interno del lavoro di van Dyk e Park (2008) circa l'introduzione, dove possibile, di collassamenti sulle *full-conditional* a posteriori marginalizzando parametri ininfluenti, adottiamo una versione *collapsed* del Gibbs Sampler per la stima del modello mistura descritto nella Sottosezione 2.3.2, che ne garantisce una più rapida e precisa convergenza alla distribuzione stazionaria: questo avverrà, come già annunciato, andando a marginalizzare i parametri $\{\boldsymbol{\mu}_j\}_{j=1}^K$, $\{\boldsymbol{\Sigma}_j\}_{j=1}^K$ e $\boldsymbol{\pi}$ dalla *full-conditional* a posteriori per $Z_i = j$: la novità rispetto agli algoritmi di Gibbs Sampling utilizzati fino ad ora riguarda la marginalizzazione, oltre che dei pesi $\boldsymbol{\pi}$, anche delle K coppie di parametri $\{(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\}_{j=1}^K$.

La *full-conditional* per $Z_i = j$ può essere riscritta come:

$$p\left(Z_i = j \,\middle|\, \mathbf{Z}_{-i}, \{\mathbf{x}_i\}_{i=1}^n\right) \propto p\left(Z_i = j \,|\mathbf{Z}_{-i}\right) p\left(\{\mathbf{x}_i\}_{i=1}^n \,\middle|\, Z_i = j, \mathbf{Z}_{-i}\right)$$
$$\propto p\left(Z_i = j \,|\mathbf{Z}_{-i}\right)\; p\left(\mathbf{x}_i \,|\, \{\mathbf{x}_h\}_{-i}, \mathbf{Z}\right) \tag{3.31}$$

sfruttando le proprietà dei prodotti condizionati e scartando i coefficienti moltiplicativi non dipendenti da j. Il primo termine, $p\left(Z_i = j \,|\mathbf{Z}_{-i}\right)$, è stato già ricavato nella (2.30); il secondo termine, invece, può essere ulteriormente riscritto come $p\left(\mathbf{x}_i \,|\, \{\mathbf{x}_h\}_{j,-i}\right)$ intendendo che tutte le $\mathbf{x}_h$ per le quali $Z_h \neq j$ non influiscono sul condizionamento e possono dunque essere scartate: questo corrisponde esattamente alla predittiva a posteriori per la j-esima componente della mistura, calcolata in $\mathbf{x}_i$.

La scelta di modellare ciascuna componente della mistura con una distribuzione gaussiana multivariata risulta come già detto particolarmente consona per l'obbiettivo che intendiamo perseguire, perchè ammette la possibilità di ricavare una forma analitica per il secondo termine, riconducibile nella fattispecie alla densità di una t-student multivariata calcolata in $\mathbf{x}_i$. Applicando la proprietà fondamentale della probabilità condizionata, possiamo riscrivere questo termine come

$$p\left(\mathbf{x}_i \,|\, \{\mathbf{x}_h\}_{j,-i}\right) = \frac{p\left(\mathbf{x}_i, \{\mathbf{x}_h\}_{j,-i}\right)}{p\left(\{\mathbf{x}_h\}_{j,-i}\right)}. \tag{3.32}$$

Soffermandoci per un momento sul numeratore (considerazioni analoghe sono valide per il denominatore), definendo $\{\mathbf{x}\}_j$ l'insieme di $\mathbf{x}_i$ per cui $Z_i = j$, questo può essere scritto come

$$p\left(\{\mathbf{x}\}_j\right) = \int_{\boldsymbol{\mu}_j} \int_{\boldsymbol{\Sigma}_j} p\left(\{\mathbf{x}\}_j \,\middle|\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right) p\left(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)\; d\boldsymbol{\mu}_j\, d\boldsymbol{\Sigma}_j \tag{3.33}$$

dove si evidenzia come questo sia nient'altro che la costante di normalizzazione per la distribuzione a posteriori della coppia $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ costruita rispetto a tutte le $\mathbf{x}_i$ appartenenti al cluster j-esimo. Andando a calcolare esplicitamente la forma della verosimiglianza per la componente j-esima della mistura come $\prod_{h \in \{Z_h = j\}} p\left(\mathbf{x}_h \,\middle|\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)$ e sostituendo a $p(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ la densità di una $\mathsf{NIW}_p\left(\boldsymbol{\mu}_0, \kappa_0, \nu_0, \boldsymbol{\Sigma}_0\right)$ si ottiene

$$p\left(\{\mathbf{x}\}_j\right) = \frac{(2\pi)^{-p\, n_j/2}}{Z_{\mathsf{NIW}}\left(p, \kappa_0, \nu_0, \boldsymbol{\Sigma}_0\right)} \int_{\boldsymbol{\mu}_j} \int_{\boldsymbol{\Sigma}_j} |\boldsymbol{\Sigma}_j|^{-\frac{\nu_0 + n_j + p + 2}{2}} \times$$
$$\times \exp\left\{-\frac{\kappa_{n_j}}{2}\left(\boldsymbol{\mu}_j - \boldsymbol{\mu}_{n_j}\right)^T \boldsymbol{\Sigma}_j^{-1}\left(\boldsymbol{\mu}_j - \boldsymbol{\mu}_{n_j}\right) - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\Sigma}_{n_j}\right)\right\}\; d\boldsymbol{\mu}_j\, d\boldsymbol{\Sigma}_j \tag{3.34}$$

dove $Z_{\mathsf{NIW}}\left(p, \kappa_0, \nu_0, \boldsymbol{\Sigma}_0\right) = s^{(\nu_0+1)p/2}\pi^{p(p+1)/4}\kappa_0^{-p/2}|\boldsymbol{\Sigma}_0|^{-\nu_0/2}\prod_{k=1}^{p}\Gamma\left(\frac{\nu_0+1-k}{2}\right)$ è la costante di normalizzazione per la prior $\mathsf{NIW}_p\left(\boldsymbol{\mu}_0, \kappa_0, \nu_0, \boldsymbol{\Sigma}_0\right)$ p-dimensionale. All'interno dell'integrale compare invece il kernel di una $\mathsf{NIW}_p\left(\boldsymbol{\mu}_{n_j}, \kappa_{n_j}, \nu_{n_j}, \boldsymbol{\Sigma}_{n_j}\right)$ p-dimensionale, dove gli iperparametri sono aggiornati rispetto alle formule (2.43)-(2.46): questo a meno della sua costante di normalizzazione integra ad 1, essendo una densità, permettendo di facilitare ulteriormente l'espressione:

$$p\left(\{\mathbf{x}\}_j\right) = (2\pi)^{-p\,n_j/2}\frac{Z_{\mathsf{NIW}}\left(p, \kappa_{n_j}, \nu_{n_j}, \boldsymbol{\Sigma}_{n_j}\right)}{Z_{\mathsf{NIW}}\left(p, \kappa_0, \nu_0, \boldsymbol{\Sigma}_0\right)}. \tag{3.35}$$

Sostituendo questa espressione all'interno della (3.33), nel numeratore e nella corrispondente versione per il denominatore, otteniamo la seguente espressione:

$$\begin{aligned} p\left(\mathbf{x}_i \mid \{\mathbf{x}_h\}_{j,-i}\right) &= \frac{(2\pi)^{-p\,n_j/2}}{(2\pi)^{-p\,(n_j-1)/2}}\frac{Z_{\mathsf{NIW}}\left(p, \kappa_{n_j}, \nu_{n_j}, \boldsymbol{\Sigma}_{n_j}\right)\cancel{/Z_{\mathsf{NIW}}\left(p, \kappa_0, \nu_0, \boldsymbol{\Sigma}_0\right)}}{Z_{\mathsf{NIW}}\left(p, \kappa_{n_{j,-i}}, \nu_{n_{j,-i}}, \boldsymbol{\Sigma}_{n_{j,-i}}\right)\cancel{/Z_{\mathsf{NIW}}\left(p, \kappa_0, \nu_0, \boldsymbol{\Sigma}_0\right)}} \\ &= (2\pi)^{-p/2}\frac{Z_{\mathsf{NIW}}\left(p, \kappa_{n_j}, \nu_{n_j}, \boldsymbol{\Sigma}_{n_j}\right)}{Z_{\mathsf{NIW}}\left(p, \kappa_{n_{j,-i}}, \nu_{n_{j,-i}}, \boldsymbol{\Sigma}_{n_{j,-i}}\right)} \\ &= \pi^{-p/2}\frac{\kappa_{n_j}^{-p/2}|\boldsymbol{\Sigma}_{n_j}|^{\nu_{n_j}/2}\prod_{k=1}^{p}\Gamma\left(\frac{\nu_{n_j}+1-k}{2}\right)}{\kappa_{n_{j,-i}}^{-p/2}|\boldsymbol{\Sigma}_{n_{j,-i}}|^{\nu_{n_{j,-i}}/2}\prod_{k=1}^{p}\Gamma\left(\frac{\nu_{n_{j,-i}}+1-k}{2}\right)}. \end{aligned} \tag{3.36}$$

Questa, per quanto disponibile in forma chiusa, risulta spesso laboriosa da calcolare. Una valida alternativa è quella proposta da Murphy (2013) a pag. 135, richiamando il risultato dovuto a Iranmanesh et al. (2012) secondo il quale una generica distribuzione t-student multivariata è rappresentabile come mistura infinita di distribuzioni normali multivariate misturizzate secondo una distribuzione gaussiana-Wishart inversa:

$$p\left(\tilde{\mathbf{x}}\middle|\,\{\mathbf{x}_i\}_{i=1}^{n}\right) = \iint \mathsf{N}_p\left(\tilde{\mathbf{x}}|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)\mathsf{NIW}_p\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{\mu}_n, \kappa_n, \nu_n, \boldsymbol{\Sigma}_n\right)\,d\boldsymbol{\mu}d\boldsymbol{\Sigma} \tag{3.37}$$

$$= \mathsf{T}_{\nu_n-p+1}\left(\tilde{\mathbf{x}}; \boldsymbol{\mu}_n, \frac{\kappa_n+1}{\kappa_n\left(\nu_n-p+1\right)}\boldsymbol{\Sigma}_n\right) \tag{3.38}$$

che consente di utilizzare la seguente forma analitica, riconducibile alla funzione di densità di una distribuzione nota e facilmente calcolabile:

$$p\left(\mathbf{x}_i \mid \{\mathbf{x}_h\}_{j,-i}\right) = \mathsf{T}_{\nu_{n_{j,-i}}-p+1}\left(\mathbf{x}_i; \boldsymbol{\mu}_{n_{j,-i}}, \frac{\kappa_{n_{j,-i}}+1}{\kappa_{n_{j,-i}}(\nu_{n_{j,-i}}-p+1)}\boldsymbol{\Sigma}_{n_{j,-i}}\right). \tag{3.39}$$

In conclusione, la (3.31) può essere scritta come

$$p\left(Z_i = j \,\middle|\, \mathbf{Z}_{-i}, \{\mathbf{x}_i\}_{i=1}^n\right) \propto \frac{n_{j,-i} + \alpha_0/K}{n + \alpha_0 - 1} \, \mathsf{T}_{\nu_{n_{j,-i}} - p + 1}\left(\mathbf{x}_i; \boldsymbol{\mu}_{n_{j,-i}}, \frac{\kappa_{n_{j,-i}} + 1}{\kappa_{n_{j,-i}}(\nu_{n_{j,-i}} - p + 1)} \boldsymbol{\Sigma}_{n_{j,-i}}\right); \tag{3.40}$$

avendo marginalizzato tutte le coppie $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, il vettore $\mathbf{Z}$ con le etichette del *cluster* al quale è stata assegnata ciascuna $\mathbf{x}_i$ verrà simulato all'interno dell'algoritmo per la stima del modello unicamente utilizzando questa espressione, per ogni $i = 1, 2, \ldots, n$.

Lo pseudo codice che consente l'implementazione di un algoritmo di *collapsed gibbs sampling* per la simulazione dalla distribuzione a posteriori del modello è riportato nell'Algoritmo 3 ed implementato in Codice A.4.

Algorithm 3 Collapsed Gibbs Sampling per una mistura finita gaussiana multivariata di SVM per problemi di classificazione

Disponendo del vettore $\mathbf{y} = (y_1, \ldots, y_n)$ contenente le modalità della variabile risposta codificate in $\{-1, 1\}$, della matrice $\mathbf{X}$ di dimensione $n \times (p+m)$ contenente le p colonne dei regressori ed intercetta, ed eventuali m colonne corrispondenti alle funzioni di base che permettono di includere kernel, del vettore degli iperparametri (a_ν, b_ν) riferiti alla priori su ciascun ν_j^{-1}, del vettore delle standard deviation σ_j per ciascuna variabile della matrice $\mathbf{X}$, del grado scelto per la penalizzazione $\alpha \in \{1, 2\}$ e dell'iperparametro di concentrazione α_0 riferito alla priori su $\boldsymbol{\pi}$, assegniamo i vettori di valori iniziali per $\boldsymbol{\beta}_j$ a $\boldsymbol{\beta}_j^{(1)}$, $j = 1, 2, \ldots, K$, quello per $\boldsymbol{\lambda}$ a $\boldsymbol{\lambda}^{(1)}$, quelli per $\boldsymbol{\omega}_j$ a $\boldsymbol{\omega}_j^{(1)}$, $j = 1, 2, \ldots, K$, quello per $\boldsymbol{\nu}$ a $\boldsymbol{\nu}^{(1)}$ e quello per $\mathbf{Z}$ a $\mathbf{Z}^{(1)}$.
Per $t = 2, 3, \ldots, R$ si eseguono iterativamente le seguenti operazioni salvando, man mano, i valori generati:

1. Sostituisco $\mathbf{Z}^{(t-1)}$ a $\mathbf{Z}^{(t)}$ per facilitare i calcoli nel passo 1.
2. Generazione dalla *full-conditional* a posteriori per Z_i, $i = 1, 2, \ldots, n$:

$$Z_i^{(t)} = j | \mathbf{X}, \mathbf{Z}_{-i}^{(t)} \propto \frac{n_{j,-i}^{(t)} + \alpha_0/K}{n + \alpha_0 - 1} \mathsf{T}_{\nu_{n_{j,-i}} - p + 1} \left(\mathbf{x}_i; \boldsymbol{\mu}_{n_{j,-i}}, \frac{\kappa_{n_{j,-i}} + 1}{\kappa_{n_{j,-i}} (\nu_{n_{j,-i}} - p + 1)} \boldsymbol{\Sigma}_{n_{j,-i}} \right)$$

   per $j = 1, 2, \ldots, K$. Il clustering è realizzato esclusivamente sulle p variabili *originali*.
3. Generazione dalla *full-conditional* a posteriori per λ_i, $i = 1, 2, \ldots, n$:

$$\left(\lambda_i^{(t)}\right)^{-1} \Big| \mathbf{y}, \mathbf{X}, \mathbf{Z}^{(t)}, \boldsymbol{\beta}_1^{(t-1)}, \ldots, \boldsymbol{\beta}_K^{(t-1)} \sim \mathsf{InvGaussian} \left(\left| 1 - y_i \mathbf{x}_i^T \boldsymbol{\beta}_{Z_i^{(t)}}^{(t-1)} \right|^{-1}, 1 \right)$$

4. Generazione dalla *full-conditional* a posteriori per $\boldsymbol{\beta}_j$, $j = 1, 2, \ldots, K$:

$$\boldsymbol{\beta}_j^{(t)} \Big| \mathbf{y}, \mathbf{X}, \mathbf{Z}^{(t)}, \boldsymbol{\lambda}^{(t)}, \boldsymbol{\omega}_j^{(t-1)} \sim \mathsf{N}_{p+m} \left(\mathbf{b}_j, \mathbf{B}_j \right)$$

   con $\mathbf{B}_j^{-1} = \left(\nu_j^{(t-1)}\right)^{-2} \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{\Omega}_j^{(t-1)}\right)^{-1} + \tilde{\mathbf{X}}_j^T \left(\boldsymbol{\Lambda}_j^{(t)}\right)^{-1} \tilde{\mathbf{X}}_j$ e $\mathbf{b}_j = \mathbf{B}_j \tilde{\mathbf{X}}_j^T \left(\mathbf{1}_{n_j^{(t)}} + \left(\boldsymbol{\lambda}_j^{(t)}\right)^{-1} \right)$, dove $\mathbf{1}_{n_j^{(t)}}$ è un vettore di 1 lungo $n_j^{(t)}$, $\boldsymbol{\lambda}_j^{(t)}$ e $\boldsymbol{\Lambda}_j^{(t)}$ sono rispettivamente un vettore ed una matrice diagonale contenenti tutti e solo i $\lambda_i^{(t)}$ tali per cui $i \in \left\{ i : Z_i^{(t)} = j \right\}$, $\tilde{\mathbf{X}}_j$ è una matrice le cui righe sono composte da tutti e solo i $y_i \mathbf{x}_i^T$ tali per cui $i \in \left\{ i : Z_i^{(t)} = j \right\}$, $\boldsymbol{\Omega}_j^{(t-1)} = \mathsf{diag}\left(\boldsymbol{\omega}_j^{(t-1)}\right)$, $n_j^{(t)} = \sum_{i=1}^n \mathbb{1}(Z_i^{(t)} = j)$ e $\boldsymbol{\Sigma} = \mathsf{diag}\left(\boldsymbol{\sigma^2}\right)$.
5. Generazione dalla *full-conditional* a posteriori per ν_j^{-1}, $j = 1, 2, \ldots, K$:

$$\left(\nu_j^{(t)}\right)^{-1} \Big| \boldsymbol{\beta}_j^{(t)}, \alpha = 1 \sim \mathsf{Gamma} \left(a_\nu + p, b_\nu + \sum_{h=1}^{p+m} \left| \frac{\beta_{hj}^{(t)}}{\sigma_h} \right| \right)$$

   dove $\beta_{hj}^{(t)}$ è la h-esima componente di $\boldsymbol{\beta}_j^{(t)}$. Se $\alpha = 2$, si tratta di elevare al quadrato il risultato appena ottenuto.
6. Generazione dalla *full-conditional* a posteriori per ω_{hj}, $j = 1, 2, \ldots, K$ e $h = 1, 2, \ldots, p+m$:

$$\left(\omega_{hj}^{(t)}\right)^{-1} \Big| \boldsymbol{\beta}_j^{(t)}, \nu_j^{(t-1)}, \alpha = 1 \sim \mathsf{InvGaussian} \left(\nu_j^{(t-1)} \frac{\sigma_h}{\left|\beta_{hj}^{(t)}\right|}, 1 \right)$$

   dove $\beta_{hj}^{(t)}$ è la h-esima componente di $\boldsymbol{\beta}_j^{(t)}$. Se $\alpha = 2$, invece, $\omega_{hj}^{(t)} = 1, \forall h$ e $\forall j$.

Si effettua un *burnin* iniziale e si restituiscono i risultati.

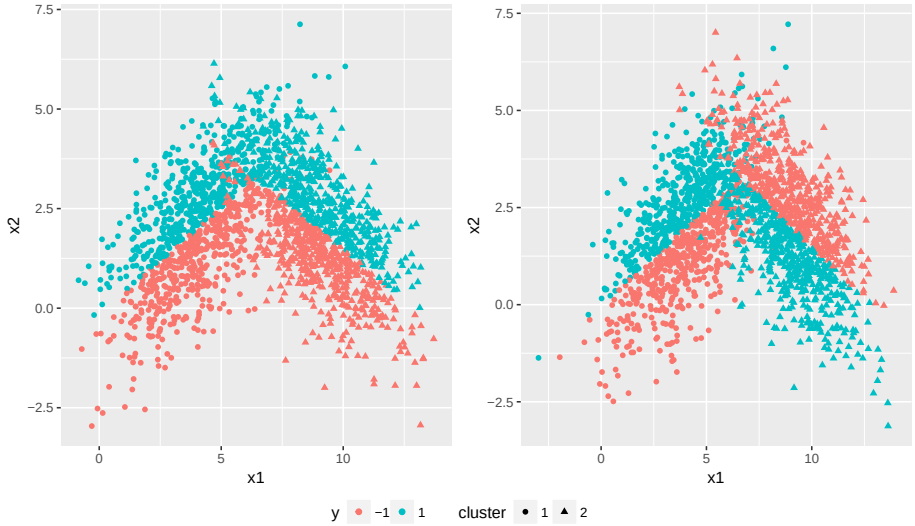


Figura 3.2: Dataset simulati per problemi di classificazione, $n = 2000$ e con proporzioni delle modalità assunte dalla variabile risposta bilanciate. I simboli dei punti rappresentano il cluster di provenienza, i colori l'etichetta associata alla modalità della variabile risposta.

3.4 Esempi di simulazione su problemi di classificazione

A titolo d'esempio, sono stati simulati due dataset (Figura 3.2) sui quali svolgere alcune simulazioni e testare l'efficacia del modello proposto. I dataset sono stati generati supponendo l'esistenza di 2 sotto-popolazioni latenti, ben visibili ad occhio considerando il simbolo utilizzato per rappresentare ciascuna osservazione sul piano. Il primo dataset, quello a sinistra, vuole rappresentare una situazione di studio in cui esiste una quasi-perfetta separazione lineare delle modalità all'interno di ciascun cluster: globalmente, però, la frontiera di separazione sembra avere più una forma *spezzata* o tutt'al più assimilabile ad una parabola rivolta verso il basso. Il secondo dataset, quello rappresentato a destra, rappresenta una situazione simile al primo dataset, con due cluster ben distinguibili ma in cui quello ha destra ha le modalità della variabile risposta *invertite* rispetto al primo. In Codice B.4 sono riportati i comandi utilizzati per generare i due dataset.

Ciascuna simulazione è stata svolta suddividendo il dataset in un insieme di stima (75% dei dati) ed uno di verifica (restante 25% dei dati), mantenendo su entrambi il bilanciamento originario delle modalità assunte dalla variabile risposta; la stima dei modelli è avvenuta

utilizzando il Gibbs Sampler riassunto in Algoritmo 3 sull'insieme di stima, eventuamente esteso con m funzioni di base rappresentanti eventuali kernel, su $R = 5000$ iterazioni e scartandone la prima metà di *burnin*. Riguardo la scelta degli iperparametri, questi sono stati scelti di modo che le distribuzioni a priori per $\boldsymbol{\pi}$, ν_j^{-1} e $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ siano il più possibile diffuse. In tutti i casi che verranno presentati, è stata scelta una penalizzazione di tipo Lasso ($\alpha = 1$) che oltre a svolgere uno *shrinkage* dei parametri permette di operare una selezione dei regressori inclusi all'interno del modello. In Codice B.5 sono riportati i comandi riferiti ai risultati nella 3.4.1, mentre in Codice B.6 quelli riferiti ai risultati della 3.4.2.

3.4.1 Primo dataset simulato

Inizialmente vengono presentati i risultati dei modelli stimati sul primo dataset rappresentato a sinistra in Figura 3.2: questo consente di approfondire le tecniche utilizzate per valutare la bontà nello svolgere la classificazione, fornendo un'immediata applicazione. Per questioni di spazio e maggiore facilità di esposizione dei risultati, ci si è limitati a presentare la simulazione svolta unicamente su una catena simulata, riportando graficamente gli aspetti che si è ritenuto più consono sottolineare ed illustrare. La convergenza delle catene simulate alla distribuzione stazionaria è comunque raggiunta anche con valori di partenza differenti.

SVM standard, kernel polinomiale lineare ($K = 1, d = 1$):

Il primo modello adattato al dataset simulato considera la struttura più semplice di tutte: una SVM per problemi di classificazione; per uniformarsi ai modelli stimati che verranno mostrati successivamente, indico questo caso con la sigla $(K = 1, d = 1)$ stando a significare che è stata stimata una mistura gaussiana multivariata di SVM per problemi di classificazione a $K = 1$ componenti (ovvero, una SVM standard) senza alcuna estensione dello spazio dei regressori (ovvero, considerando un kernel polinomiale di grado $d = 1$ comprendente solo le variabili x_1 e x_2, assieme all'intercetta). Osservando la rappresentazione grafica del dataset simulato ci si può già attendere che il modello stimato non sarà in grado di eseguire una buona classificazione, dato che le modalità assunte dalla variabile risposta non sono linearmente separabili.

In Figura 3.3 sono riportati i traceplot post-burnin rappresentanti l'evoluzione delle catene per i parametri associati a ciascuna covariata inclusa nel modello: queste presentano

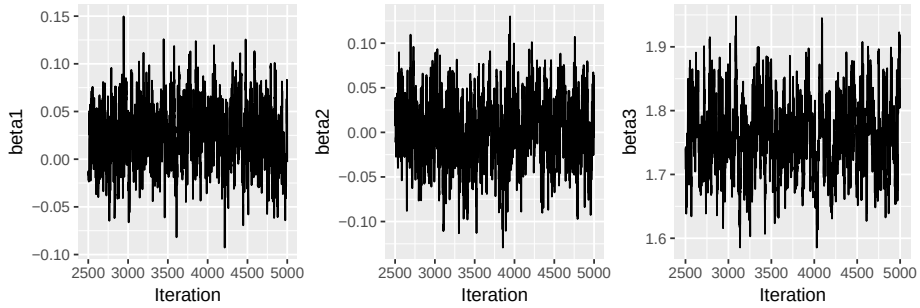


Figura 3.3: Traceplot delle catene per β_1, β_2 e β_3 nel modello SVM standard $K = 1, d = 1$.

un *mixing* piuttosto soddisfacente, e la convergenza sembra essere visibilmente raggiunta. Osservando più attentamente i valori sui quali si stabilizzano le catene, possiamo concludere che la penalizzazione di tipo Lasso ha di fatto escluso l'intercetta e la variabile x_1, valorizzando solo il contributo portato dalla variabile x_2. Questo risultato è in linea con quanto atteso: osservando nuovamente la rappresentazione nel dataset a sinistra di Figura 3.2, x_2 è l'unica covariata in grado di svolgere una (seppur minima) discriminazione tra le due aree colorate, ovvero tra le due modalità assunte della risposta; la variabile x_1, d'altra parte, da sola non apporta alcun significativo contributo alla discriminazione, e per questo motivo la penalizzazione Lasso porta correttamente la distribuzione a posteriori del coefficiente associato ad essa a concentrarsi attorno allo 0. Le statistiche riassuntive relative alle distribuzioni a posteriori marginali per ciascun parametro sono riportate in Tabella 3.1.

Coefficiente	Media	StdDev	$Q_{0.25}$	$Q_{0.75}$
Intercetta	0.026	0.032	0.004	0.048
x_1	0	0.038	-0.025	0.026
x_2	1.755	0.057	1.717	1.794

Tabella 3.1: Statistiche riassuntive (media aritmetica, *standard deviation*, primo e terzo quartile) della distribuzione a posteriori per ciascun coefficiente del modello $K = 1, d = 1$.

Nonostante i visibili limiti del modello preso in considerazione, un modo per valutarne l'adeguatezza nello svolgere una corretta classificazione della variabile risposta è quello di costruire la nota matrice di confusione, ottenuta andando a confrontare le vere modalità assunte dalla variabile risposta con quelle predette dal classificatore sull'insieme di verifica, per ciascuna delle osservazioni che gli appartengono. In un contesto bayesiano, la

costruzione del classificatore viene *complicata* (o arricchita, a seconda dei punti di vista) dal fatto che i parametri β_j non vengono stimati puntualmente, ma per ciascuno di questi si dispone di una distribuzione (marginale) a posteriori: di conseguenza, considerata una generica nuova osservazione $\tilde{\mathbf{x}}$ il classificatore permetterà di ottenere una distribuzione di classificazioni associata e non una singola classificazione, come si è abituati nei contesti più standard del *data mining*.

Data $p(\boldsymbol{\beta}|\mathbf{X}, \mathbf{y})$ la distribuzione a posteriori ottenuta per $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_{p+m})$ (in questo specifico caso, non avendo utilizzato alcuna espansione kernel, $m = 0$), è possibile ottenere la distribuzione delle classificazioni per la generica $\tilde{\mathbf{x}}$ tramite il funzionale $\text{sign}\left(\tilde{\mathbf{x}}^T\boldsymbol{\beta}\right)$ che connota il noto classificatore $G(\mathbf{x})$ per SVM di classificazione: questa risulterà essere una distribuzione discreta a valori in $\{-1, 1\}$, sulla quale generalmente si intende calcolare una misura riassuntiva quale, ad esempio, il voto di maggioranza. Quest'ultimo è nei fatti approssimabile dal corrispettivo empirico calcolato sulla sequenza della distribuzione a posteriori ottenuta dalla catena per $\boldsymbol{\beta}$:

$$\text{MajVote}_{p(\boldsymbol{\beta}|\mathbf{X},\mathbf{y})}\left[\text{sign}\left(\tilde{\mathbf{x}}^T\boldsymbol{\beta}\right)\right] \simeq \widehat{\text{MajVote}}\left[\left\{\text{sign}\left(\tilde{\mathbf{x}}^T\boldsymbol{\beta}^{(j)}\right)\right\}_{j=1}^{N_{iter}}\right]. \tag{3.41}$$

Nonostante la facilità computazionale di calcolo, questa misura riassuntiva in un certo modo impoverisce la distribuzione a posteriori delle classificazioni, assegnando una modalità piuttosto che l'altra alla generica $\tilde{\mathbf{x}}$ solo in base alla maggiore frequenza relativa assunta dall'una o dall'altra modalità sulla distribuzione empirica delle classificazioni e senza tenere conto, ad esempio, di misure di eterogeneità sulla distribuzione stessa.

Un metodo alternativo che si è ritenuto più consono utilizzare riguarda l'utilizzo della matrice di confusione, che abbiamo già detto essere un valido strumento per valutare l'adeguatezza del modello nel classificare correttamente: anzichè calcolare una matrice di confusione *finale* sulla base delle classificazioni ottenute per voto di maggioranza sulla distribuzione delle classificazioni per ciascuna nuova osservazione dell'insieme di verifica, prendiamo in considerazione l'intera distribuzione delle matrici di confusione calcolate su ciascun $\boldsymbol{\beta}^{(j)}$.

A livello teorico, questo è formalizzabile (senza complicare eccessivamente la notazione, ma dando un'intuizione generale del metodo) andando a considerare il funzionale *matrice di confusione* che associa a ciascuna terna $\left(\tilde{\mathbf{y}}, \tilde{\mathbf{X}}; \boldsymbol{\beta}\right)$ la rispettiva matrice di confusione $\text{ConfMat}\left(\tilde{\mathbf{y}}, \tilde{\mathbf{X}}; \boldsymbol{\beta}\right)$: questo è definito sulla generica coppia $\tilde{\mathbf{y}}, \tilde{\mathbf{X}}$ rappresentante l'insieme di

verifica e sul vettore $\boldsymbol{\beta}$ che entra all'interno del classificatore $G(\tilde{\mathbf{x}}) = \text{sign}\left(\tilde{\mathbf{x}}^T\boldsymbol{\beta}\right)$ di ciascuna osservazione, e produrrà la matrice di confusione associata. Dal punto di vista bayesiano, stante la distribuzione a posteriori $\boldsymbol{\beta}|\mathbf{X},\mathbf{y}$, il funzionale ConfMat $\left(\tilde{\mathbf{y}},\tilde{\mathbf{X}};\boldsymbol{\beta}\right)$ è anch'esso interpretabile come una vera e propria distribuzione, ottenuta come trasformazione di $\boldsymbol{\beta}$, sulla quale si intenderà calcolare una qualche misura riassuntiva. Una scelta adeguata in tal senso ricade nel valore atteso della distribuzione, intendendo quest'ultimo come la matrice composta dai quattro valori attesi calcolati su ciascuna delle quattro celle che compongono la sequenza di matrici di confusione: questo è approssimabile via Monte Carlo andando a considerare la media aritmetica delle matrici di confusione calcolate per ciascun $\boldsymbol{\beta}^{(j)}$ generato dal Gibbs Sampler:

$$\mathbb{E}_{p(\beta|\mathbf{X},\mathbf{y})}\left[\text{ConfMat}\left(\tilde{\mathbf{y}},\tilde{\mathbf{X}};\boldsymbol{\beta}\right)\right] \simeq \frac{1}{N_{iter}}\sum_{j=1}^{N_{iter}}\text{ConfMat}\left(\tilde{\mathbf{y}},\tilde{\mathbf{X}};\boldsymbol{\beta}^{(j)}\right) \tag{3.42}$$

dove il pedice sul simbolo di valore atteso sta ad indicare che quest'ultimo è calcolato rispetto alla distribuzione a posteriori $p(\boldsymbol{\beta}|\mathbf{X},\mathbf{y})$. Com'è ovvio, svolgere la media aritmetica della sequenza di N_{iter} matrici di confusione significa costruire una matrice *finale* in cui ciascuna delle quattro celle è ottenuta dalla media aritmetica delle frequenze contenute nella stessa cella su ciascuna delle N_{iter} matrici di confusione. A questo punto, si tratterà di andare a calcolare la j-esima matrice di confusione ConfMat $\left(\tilde{\mathbf{y}},\tilde{\mathbf{X}};\boldsymbol{\beta}^{(j)}\right)$ al solito modo; quest'ultima può essere ulteriormente vista come la somma di n_{test} matrici di confusione, una per ciascuna delle n_{test} osservazioni contenute nell'insieme di verifica: la generica ConfMat $\left(\tilde{y}_i,\tilde{\mathbf{x}}_i;\boldsymbol{\beta}^{(j)}\right)$ corrisponderà dunque ad una matrice quadrata con un 1 in corrispondenza della cella che rappresenta la coppia $(\tilde{y}_i, G(\tilde{\mathbf{x}}_i))$.

Assegneremo quindi un giudizio sulla capacità del modello di classificare, andando a valutare la matrice di confusione *finale* ottenuta come:

$$\begin{aligned}\widehat{\text{ConfMat}}\left(\tilde{\mathbf{y}},\tilde{\mathbf{X}};\{\boldsymbol{\beta}_j\}_{j=1}^{N_{iter}}\right) &= \frac{1}{N_{iter}}\sum_{j=1}^{N_{iter}}\text{ConfMat}\left(\tilde{\mathbf{y}},\tilde{\mathbf{X}};\boldsymbol{\beta}^{(j)}\right)\\ &= \frac{1}{N_{iter}}\sum_{j=1}^{N_{iter}}\sum_{i=1}^{n_{test}}\text{ConfMat}\left(\tilde{y}_i,\tilde{\mathbf{x}}_i;\boldsymbol{\beta}^{(j)}\right).\end{aligned} \tag{3.43}$$

In Figura 3.4 è riportata la sequenza dei valori per la distribuzione delle matrici di confusione in ciascuna delle quattro celle: fissando l'indice dell'iterazione j-esima, è possibile

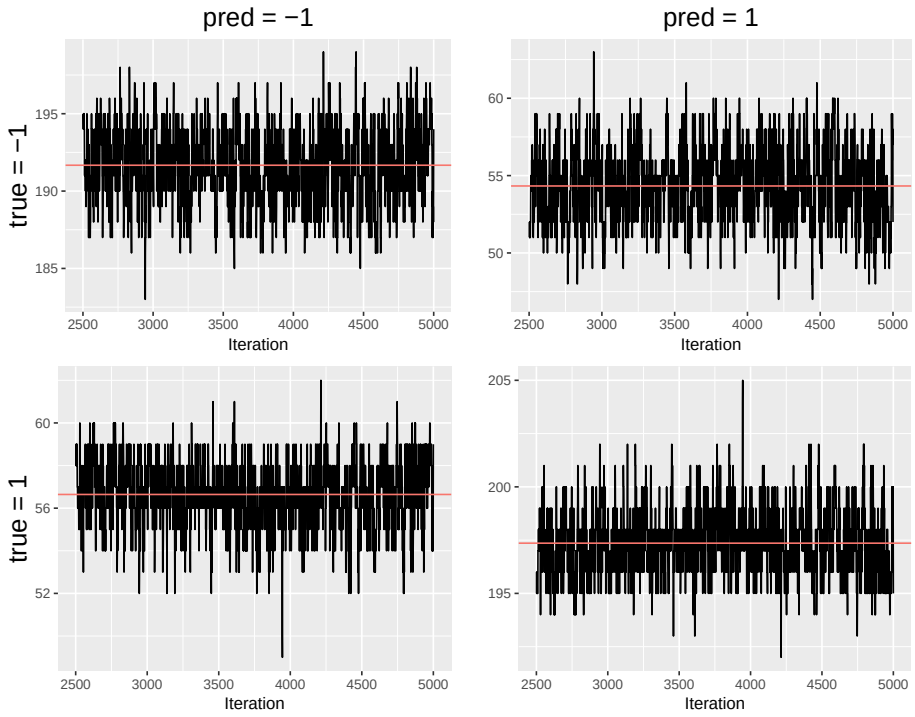


Figura 3.4: Traceplot per le quattro celle delle matrici di confusione valutate per la stima del modello SVM standard $K = 1, d = 1$. La linea orizzontale rossa definisce la media aritmetica della sequenza di valori per ciascuna cella.

ricavare graficamente il contenuto della ConfMat $\left(\tilde{\mathbf{y}}, \tilde{\mathbf{X}}; \boldsymbol{\beta}_j\right)$ andando ad osservare l'altezza raggiunta da ciascuna delle quattro sequenze. Il grafico di per se ha un'utilità limitata, ma consente di monitorare la stabilità delle matrici di confusione calcolate su ciascun $\boldsymbol{\beta}_j$. La matrice di confusione *finale* è riportata di seguito:

		Previsti	
		-1	1
Reali	-1	191.67 (2.43)	54.33 (2.43)
	1	56.64 (1.56)	197.36 (1.56)

Tabella 3.2: Matrice di confusione *finale* per il modello $K = 1, d = 1$. Tra parentesi è riportato lo scarto quadratico medio della catena di frequenze calcolate su ciascuna tabella di confusione, per ciascuna delle quattro celle.

Questa non contiene numeri interi perchè, come atteso, questi risultano dalla media aritmetica di frequenze. In ogni caso la lettura della matrice è invariata rispetto ai casi tradizionali: Il modello $K = 1, d = 1$ stimato commette un errore di classificazione complessivo pari a circa il 22%, certamente migliorabile. Le opzioni percorribili in tal senso sono sicuramente molteplici, ma di seguito ne vengono considerate 2 che permettono, distintamente, di toccare aspetti salienti del modello: la prima estensione corrisponde al caso $K = 1, d = 2$, la seconda al caso $K = 2, d = 1$.

SVM standard, kernel polinomiale quadratico ($K = 1, d = 2$):

Uno degli aspetti considerati nel modello precedente riguardava il limite tecnico nel riuscire a cogliere frontiere di separazione non-lineari: una possibile alternativa particolarmente consona in tal senso riguarda la possibilità di estendere lo spazio dei regressori considerando un kernel polinomiale di grado $d = 2$. Data la generica osservazione del dataset $\mathbf{x}_i = (x_{i1}, x_{i2})$, il kernel polinomiale di grado $d = 2$ è specificato come $K(\mathbf{x}_i, \mathbf{x}_i^*) = \left(1 + \mathbf{x}_i^T \mathbf{x}_i^*\right)^2$: svolgendo il prodotto ed il quadrato si ricavano le funzioni di base che lo compongono, corrispondenti a $h_1(\mathbf{x}) = 1$, $h_2(\mathbf{x}) = x_1$, $h_3(\mathbf{x}) = x_2$, $h_4(\mathbf{x}) = x_1^2$, $h_5(\mathbf{x}) = x_2^2$, $h_6(\mathbf{x}) = x_1 x_2$. Rispetto al modello precedente in cui $d = 1$, in questo caso si estende lo spazio dei regressori considerando il quadrato di entrambi e l'interazione: questo dovrebbe consentire la possibilità di cogliere meglio la forma della frontiera, che era stato notato avere una forma paraboloide. Le statistiche riassuntive per la distribuzione a posteriori di ciascun coefficiente sono riportate in Tabella 3.3.

Coefficienti	Media	StdDev	$Q_0.25$	$Q_0.75$
Intercetta	-1.446	0.103	-1.512	-1.373
x_1	-27.156	1.402	-28.114	-26.223
x_2	16.499	0.861	15.908	17.076
x_1^2	27.635	1.428	26.663	28.621
x_2^2	-8.554	0.464	-8.862	-8.265
$x_1 x_2$	0.557	0.271	0.364	0.748

Tabella 3.3: Statistiche riassuntive (media aritmetica, *standard deviation*, primo e terzo quartile) della distribuzione a posteriori per ciascun coefficiente del modello $K = 1, d = 2$.

A differenza del modello precedente, come atteso, tutte le variabili risultano fondamentali nel realizzare la discriminazione delle modalità assunte dalla variabile risposta: il merito è dovuto indubbiamente all'introduzione dell'interazione e dei termini quadratici, che riescono a cogliere meglio la forma della frontiera che distingue le due modalità.

La conferma che l'introduzione di un kernel polinomiale di grado $d = 2$ migliora sensibilmente i risultati della classificazione è data anche dalla matrice di confusione, ottenuta al pari della precedente e riportata in Tabella 3.4.

		Previsti	
		-1	1
Reali	-1	229.60 (1)	16.40 (1)
	1	23.10 (1.11)	230.90 (1.11)

Tabella 3.4: Matrice di confusione *finale* per il modello $K = 1, d = 2$. Tra parentesi è riportato lo scarto quadratico medio della catena di frequenze calcolate su ciascuna tabella di confusione, per ciascuna delle quattro celle.

Il tasso di errata classificazione si è notevolmente ridotto, arrivando a toccare circa il 7%: un risultato certamente migliore del modello precedente, come atteso. E' possibile tentare un'espansione polinomiale con un ulteriore grado per il kernel, ma questo oltre ad andare ad aumentare notevolmente il numero delle funzioni di base che ne consentono la rappresentazione, e dunque il numero di coefficienti da stimare, va oltre lo scopo di questi esempi.

Una caratteristica del dataset che non è ancora stata modellata riguarda invece la presenza di due cluster distinti, ben visibili osservando il grafico di sinistra in Figura 3.2; nonostante questo mostri esplicitamente a quale cluster appartiene ciascuna osservazione (essendo stato simulato, si dispone della conoscenza sul numero di sottopopolazioni latenti e sul cluster al quale appartiene ciascuna osservazione), è bene ricordare che in un contesto reale non si dispone di alcuna informazione sull'esistenza di eventuali cluster latenti, nè sul loro eventuale numero, nè su quali osservazioni appartengono all'uno o all'altro: nonostante ciò il suggerimento sull'esistenza di eventuali strutture latenti può comunque provenire da analisi esplorative condotte in fase di pre-analisi del dataset, o da conoscenze teoriche circa la natura del problema.

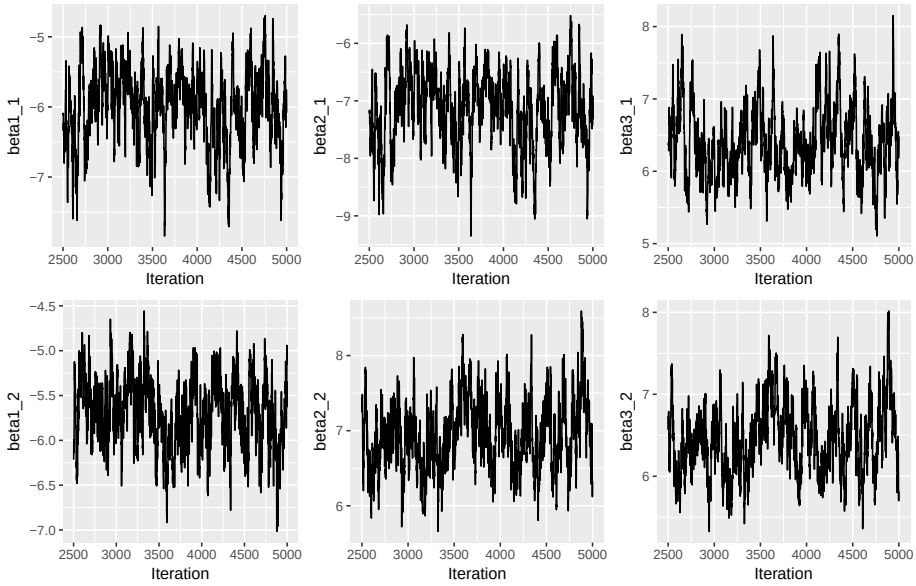


Figura 3.5: Traceplot delle catene per $\boldsymbol{\beta}_1$ (in alto) e $\boldsymbol{\beta}_2$ (in basso) nel modello $K = 2, d = 1$.

Mistura di gaussiane multivariate per SVM, kernel polinomiale lineare ($K = 2, d = 1$):

La prima estensione che si considera, dettata dalla natura del dataset, riguarda una mistura di $K = 2$ gaussiane multivariate: su ciascuno dei due cluster individuati si stimerà una SVM adattando un kernel polinomiale di grado $d = 1$ contente l'intercetta e le due variabili del dataset. Rispetto ai modelli SVM standard, la procedura di stima richiede l'assegnazione del vettore di partenza $\mathbf{Z}_0$ contenente le etichette dei cluster per ciascuna osservazione del training set: nonostante questa possa essere svolta con metodi di *clustering* come il k-medie, o tramite analisi esplorative, l'assegnazione di ciascuna osservazione ad uno dei $K = 2$ cluster è stata svolta in maniera totalmente casuale per evitare che la convergenza del Gibbs Sampling non venisse *facilitata*. I traceplot delle catene per i coefficienti $\boldsymbol{\beta}_j$ risultanti dal Gibbs Sampling sono riportati in Figura 3.5: questi risultano essere globalmente soddisfacenti, attestando una complessiva convergenza della catena in ciascuna delle 6 componenti, ma sembrano esibire una modesta autocorrelazione, risolvibile svolgendo un opportuno *thinning* delle catene: la motivazione è riconducibile al fatto che le osservazioni *borderline* continuano a cambiare assegnazione in ciascuna iterazione dell'algoritmo, non riuscendo quest'ultimo ad attribuire un'unico e definitivo *cluster* di appartenenza. I risultati della stima del modello sono sintetizzati in Tabella 3.5.

Coefficienti	Cluster	Media	StdDev	$Q_0.25$	$Q_0.75$
Intercetta	1	-5.993	0.536	-6.328	-5.621
	2	-5.694	0.383	-5.956	-5.425
x_1	1	-7.126	0.646	-7.549	-6.673
	2	6.944	0.456	6.620	7.242
x_2	1	6.405	0.490	6.053	6.706
	2	6.463	0.429	6.153	6.738

Tabella 3.5: Statistiche riassuntive (media aritmetica, *standard deviation*, primo e terzo quartile) della distribuzione a posteriori per ciascun coefficiente del modello $K = 2, d = 1$.

Il confronto con la Tabella 3.1 riferita al modello $K = 1, d = 1$ permette di notare alcuni aspetti: il primo, che andando a stimare una distinta SVM su ciascuno dei $K = 2$ *cluster* (individuati dal modello stesso), l'intercetta e la variabile x_1 risultano dare un contributo non-nullo alla costruzione della classificazione, a differenza del modello in cui questi due insiemi di dati venivano considerati assieme; il secondo, che il coefficiente associato alla variabile x_1 assume segno opposto per le SVM stimate nei due *cluster*: questo risultato è in linea con l'opposta *pendenza* delle linee di separazione nelle due classi in ciascuno dei due *cluster*, come evidente in Figura 3.2. Relativamente alla mistura, i pesi delle classi ottenuti come media aritmetica pesata delle proporzioni simulate dal Gibbs Sampler, risultano essere rispettivamente $\pi_1 = 0.49$ e $\pi_2 = 0.51$: rispetto alla scelta di attribuire un'ordinamento preciso all'interno delle componenti della mistura, e dunque di poter assegnare le etichette 1 e 2 ai parametri associati alle componenti della mistura, rimandiamo alle considerazioni già fatte circa l'interscambiabilità dei termini della mistura.

Lo strumento utilizzato per valutare la capacità del modello nello svolgere una corretta classificazione rimane, ancora una volta, la matrice di confusione *finale*, ottenuta al solito modo come media aritmetica della distribuzione delle matrici di confusioni calcolate per ciascuna delle N_{iter} iterazioni post-burnin ottenute dal Gibbs Sampler. Una doverosa precisazione rispetto al caso $K = 1$ riguarda invece il modo in cui si ottiene la classificazione della generica osservazione $\tilde{\mathbf{x}}$: il previsore $G(\tilde{\mathbf{x}}) = \text{sign}\left(\tilde{\mathbf{x}}^T \boldsymbol{\beta}\right)$ non sarà più utilizzabile perchè ora si dispone di K distinti $\boldsymbol{\beta}_j$ stimati, uno per ciascuna delle SVM stimate sui rispettivi *cluster* individuati dalla procedura di *clustering* sottostante la stima della mistura di gaussiane multivariate. Questa caratteristica, naturale conseguenza della maggiore flessibilità ammessa dal modello, complica la procedura di classificazione e rende necessa-

rio stabilire un criterio il più possibile legato alla struttura di quest'ultimo per assegnare ciascuna osservazione $\tilde{\mathbf{x}}$ ad uno dei K cluster, e dunque utilizzare il classificatore associato $G_j(\tilde{\mathbf{x}}) = \text{sign}\left(\tilde{\mathbf{x}}^T \boldsymbol{\beta}_j\right)$. Naturalmente, il problema si pone esclusivamente per tutte le $\tilde{\mathbf{x}}$ appartenenti all'insieme di verifica: per quelle che hanno contribuito alla stima del modello, l'intera catena dei vettori di clustering $\left\{\mathbf{Z}^{(t)}\right\}_{t=1}^{N_{iter}}$ proveniente dal Gibbs Sampler permette di conoscere a quale tra i K cluster è stata associata la i-esima osservazione nell'iterazione t-esima, e dunque quale tra i K classificatori utilizzare per svolgere una valutazione *in-sample* sulla qualità del modello stimato.

Per tutte le altre osservazioni, si tratterà dunque di assegnare ciascuna di queste ad uno dei K *cluster* che compongono la mistura, sulla base della composizione di quest'ultimi. Dal punto di vista teorico, si calcoleranno K distinte probabilità a posteriori

$$p\left(\tilde{Z} = j|\tilde{\mathbf{x}}\right) \propto p\left(\tilde{\mathbf{x}}|\tilde{Z} = j\right) p\left(\tilde{Z} = j\right) \quad j = 1, 2, \ldots, K \tag{3.44}$$

e si assegnerà $\tilde{\mathbf{x}}$ al cluster che raggiunge la più alta tra le K: questa operazione si svolgerà *condizionatamente* ai valori simulati dalla catena nella t-esima iterazione, volendo ottenere una distribuzione di classificazioni per ciascuna osservazione dell'insieme di verifica, che a sua volta consente di ricavare una distribuzione di matrici di confusione.

Il primo termine della (3.44) corrisponde alla densità della j-esima componente della mistura (una normale multivariata) calcolata in $\tilde{\mathbf{x}}$ e sulla base di $\boldsymbol{\mu}_j^{(t)}$ e $\boldsymbol{\Sigma}_j^{(t)}$: avendo però operato la scelta di marginalizzare entrambi, e non essendo dunque prontamente ottenibili dalla catena simulata, risulta più conveniente calcolare questa quantità utilizzando la predittiva a posteriori del cluster j-esimo:

$$p\left(\tilde{\mathbf{x}}|\tilde{Z} = j\right)^{(t)} = \mathsf{T}_{\nu_{n_{j,(t)}}-p+1}\left(\tilde{\mathbf{x}}; \boldsymbol{\mu}_{n_{j,(t)}}, \frac{\kappa_{n_{j,(t)}}+1}{\kappa_{n_{j,(t)}}\left(\nu_{n_{j,(t)}}-p+1\right)}\boldsymbol{\Sigma}_{n_{j,(t)}}\right) \quad j = 1, 2, \ldots, K \tag{3.45}$$

dove il pedice $n_{j,(t)}$ sta ad indicare che gli iperparametri vengono aggiornati secondo le formule (2.43)-(2.46) utilizzando esclusivamente le osservazioni appartenenti all'insieme di stima per le quali $Z_i^{(t)} = j$, ovvero che nell'iterazione t-esima del Gibbs Sampler sono state assegnate al cluster j-esimo.

Il secondo termine della (3.44) corrisponde invece al peso a posteriori di ciascun *cluster* all'interno della mistura: anche in questo caso i $\boldsymbol{\pi}^{(t)}$ non sono immediatamente disponibili

dalla catena simulata, essendo stati marginalizzati, ma possono essere facilmente recuperati andando a generare a posteriori

$$\boldsymbol{\pi}^{(t)} \sim \mathsf{Dir}\left(\alpha_0 + \sum_{i=1}^{n} \mathbb{1}(Z_i^{(t)} = 1), \ldots, \alpha_0 + \sum_{i=1}^{n} \mathbb{1}(Z_i^{(t)} = K)\right) \tag{3.46}$$

dalla usuale *full-conditional* che si utilizzerebbe se non si procedesse alla marginalizzazione dei pesi dalla mistura. In alternativa, se la simulazione diventa onerosa, è comunque concesso approssimare ciascun peso con la proporzione di $Z_i^{(t)} = j$. Il codice che implementa la funzione che permette di calcolare la distribuzione di matrici di confusione è riportato in Codice A.5.

Questa procedura permette, da sola, di assegnare una qualunque osservazione appartenente all'insieme di verifica ad uno dei *cluster* della mistura, utilizzando esclusivamente la struttura ipotizzata sul meccanismo generatore delle $\mathbf{x}$ e la conoscenza *a posteriori* (limitata alla iterazione t-esima). In fase di valutazione della bontà del modello, la (3.44) dovrà essere calcolata $N_{iter} \times n_{test} \times K$ volte, di modo da poter conoscere l'assegnazione delle n_{test} osservazioni appartenenti all'insieme di verifica per ciascuna delle N_{iter} iterazioni del Gibbs Sampler: questa procedura potrebbe rivelarsi computazionalmente intensiva quando la dimensione dell'insieme di verifica è particolarmente elevata.

La tabella di confusione *finale* è dunque riportata in Tabella 3.6, con un tasso di errata classificazione complessivo pari a circa il 4%.

		Previsti	
		-1	1
Reali	-1	236.58 (1.17)	9.41 (1.17)
	1	10.96 (2.09)	243.04 (2.09)

Tabella 3.6: Matrice di confusione *finale* per il modello $K = 2, d = 1$. Tra parentesi è riportato lo scarto quadratico medio della catena di frequenze calcolate su ciascuna tabella di confusione, per ciascuna delle quattro celle.

Il modello svolge quanto atteso, seppure su di un dataset elementare e simulato: andare a stimare una SVM distinta su ciascuna sottopopolazione latente, caratterizzabili quest'ultime da una mistura con $K = 2$ componenti gaussiane multivariate, permette senza

Figura 3.6: Clustering delle osservazioni $\mathbf{x}_i$ appartenenti all'insieme di stima sulla base di $\mathbf{Z}^{(N_{iter})}$, $i = 1, 2, \ldots, n$.

introdurre alcun tipo di kernel (e dunque appesantire il modello, pensando ai casi in cui $p >> 2$) di esibire una migliore capacità nello svolgere correttamente la classificazione.

Un'ulteriore conferma sulla corretta identificazione da parte del modello delle 2 distinte sottopopolazioni che hanno generato i dati è illustrata nello scatterplot rappresentato in Figura 3.6: questo rappresenta l'insieme di stima in cui ciascuna osservazione (punto) è stata colorata a seconda del *cluster* al quale è stata assegnata nell'ultimo vettore della catena $\mathbf{Z}^{(N_{iter})}$, esibendo un risultato in linea con quanto atteso.

Mistura di gaussiane multivariate per SVM, kernel polinomiale quadratico ($K = 2, d = 2$):

Per completezza rispetto all'analisi svolta sul primo dataset simulato, è stato stimato il modello a mistura finita di gaussiane multivariate per SVM introducendo un kernel polinomiale quadratico su ciascuna: a livello esplorativo, avendo individuato una chiara separazione lineare tra le modalità della variabile risposta su ciascun cluster, ci aspettiamo che l'utilizzo di un'espansione *kernel* polinomiale di secondo grado conduca ad un sovradattamento del modello all'insieme di stima, con un conseguente peggioramento dell'abilità previsiva sull'insieme di verifica.

Coefficienti	Cluster	Media	StdDev	$Q_0.25$	$Q_0.75$
Intercetta	1	-4.573	1.614	-5.682	-3.430
	2	-2.838	1.670	-4.044	-1.622
x_1	1	-12.085	6.185	-16.257	-7.816
	2	-27.486	2.037	-28.825	-26.198
x_2	1	19.100	4.357	16.097	22.165
	2	18.895	1.421	18.010	19.875
x_1^2	1	17.206	3.725	14.664	19.677
	2	27.186	4.207	24.225	30.107
x_2^2	1	-8.286	1.282	-9.177	-7.390
	2	-8.597	1.044	-9.337	-7.865
x_1x_2	1	-2.135	2.598	-4.014	-0.361
	2	-1.267	2.480	-3.009	0.572

Tabella 3.7: Statistiche riassuntive (media aritmetica, *standard deviation*, primo e terzo quartile) della distribuzione a posteriori per ciascun coefficiente del modello $K = 2, d = 2$.

		Previsti	
		-1	1
Reali	-1	231.43 (1.63)	14.56 (1.63)
	1	17.70 (3.19)	236.30 (3.19)

Tabella 3.8: Matrice di confusione *finale* per il modello $K = 2, d = 2$.

I risultati della stima sono riportati in Tabella 3.7, con un peso stimato delle due componenti che si attesta, anche in questo caso, attorno al 50% per entrambe. Si nota che rispetto al modello $K = 2, d = 1$ in questo caso tutti i coefficienti assumono lo stesso segno in entrambe le SVM stimate, con una evidente maggiore variabilità delle stime che le rendono meno *affidabili*. La tabella di confusione *finale* è riportata invece in Tabella 3.8, con un'tasso di errata classificazione pari al 6.5%.

Il modello, come previsto, esibisce un tasso di errata classificazione leggermente superiore al precedente, motivato quasi sicuramente dall'*overfitting* dovuto all'introduzione di un kernel polinomiale quadratico, non necessario per eseguire una adeguata discriminazione tra le modalità assunte dalla variabile risposta.

Un'ulteriore estensione del modello che è stata testata sul dataset, ma per la quale non vengono riportati nel dettaglio i risultati, riguarda la possibilità di includere $K > 2$ componenti della mistura, nonostante l'evidenza esplorativa e la conoscenza sul meccanismo che ha generato i dati suggeriscano sia sufficiente utilizzare 2 componenti. Un tentativo è stato fatto scegliendo $K = 3$ componenti, sia adottando un kernel polinomiale lineare ($d = 1$) che un kernel polinomiale quadratico ($d = 2$): in entrambi i casi il modello assegna un peso praticamente nullo ad una delle 3 componenti, di fatto *rigettando* l'ipotesi che possano esistere 3 sotto-popolazioni latenti nell'ignoto meccanismo che ha generato i dati. Complessivamente è possibile ritenersi soddisfatti dei risultati ottenuti dal modello a mistura finita di gaussiane multivariate per SVM proposto: questo consente di raggiungere un netto miglioramento nella qualità previsiva rispetto alla SVM standard, anche a parità di kernel utilizzato, seppur su un dataset di esempio simulato.

3.4.2 Secondo dataset simulato

Relativamente al secondo dataset simulato, illustrato graficamente a destra in Figura 3.2, si può notare come la struttura dei gruppi delle modalità assunte dalla variabile risposta sia leggermente più complicata del primo dataset analizzato: in questo caso ci aspettiamo che una SVM standard, da sola, non sia in grado di cogliere adeguatamente tutta la struttura del dataset senza ricorrere ad estensioni dello spazio dei regressori tramite kernel polinomiali di grado 2 o addirittura superiore, introducendo un elevato numero di parametri corrispondenti alle funzioni di base che consentono di implementarli; allo stesso modo ci si aspetta che un modello a mistura finita di gaussiane multivariate per SVM riesca a conseguire un risultato soddisfacente senza ricorrere ad espansioni tramite kernel. Rispetto all'analisi svolta sul primo dataset, per questioni di chiarezza espositiva e di spazio vengono riportati in seguito solo i risultati relativi a misure riassuntive sulla matrice di confusione *finale* calcolata per ciascuno dei modelli stimati: le metodologie utilizzate rispecchiano fedelmente quanto già descritto nelle pagine precedenti.

Relativamente alle misure utilizzate, in Tabella 3.9 sono riportati il tasso di errata classificazione (*missclassification error*, ottenuto come complemento ad 1 del tasso di corretta classificazione, definito a sua volta come rapporto tra la somma della diagonale della matrice di confusione *finale* e la somma di tutte le celle), *recall* (% di classificazioni positive

Modello	MisclassError	Recall	Precision
$K = 1, d = 1$	0.49	0.99	0.51
$K = 1, d = 2$	0.25	0.81	0.73
$K = 2, d = 1$	0.11	0.90	0.89
$K = 2, d = 2$	0.08	0.92	0.91

Tabella 3.9: Misure di *performance* sulle matrici di confusione *finali* calcolate su ciascuno dei 4 modelli stimati.

($y_i = +1$) sul totale delle osservazioni con vera $y_i = +1$) e *precision* (% di classificazioni positive ($y_i = +1$) sul totale delle osservazioni classificate con $y_i = +1$).

I modelli a mistura di gaussiane multivariate per SVM proposti esibiscono una netta e migliore qualità previsiva delle SVM standard, sia escludendo che considerando un'espansione kernel di tipo polinomiale sullo spazio dei regressori, in linea con quanto atteso. Relativamente agli ultimi due modelli stimati, l'introduzione di un kernel polinomiale di grado $d = 2$ sembra consentire al modello di adattarsi leggermente meglio ai dati senza soffrire di *overfitting*: questo potrebbe essere motivato dalla migliore capacità previsiva per le osservazioni situate laddove i due *cluster* si incontrano, in cui la frontiera di demarcazione tra le modalità assunte dalla variabile risposta potrebbe non avere un andamento del tutto lineare.

Globalmente è possibile ritenersi soddisfatti dei risultati ottenuti sui due dataset simulati: in entrambi, seppur costruiti *ad hoc* per testare ed avere la conferma che la classe di modelli proposti si comporti secondo le aspettative e fornisca un evidente miglioramento rispetto alle SVM standard di classificazione, i risultati sembrano dare sostegno alle motivazioni che hanno spinto alla formalizzazione di questa classe di modelli, già richiamate nella Sezione 3.1.

Capitolo 4

Applicazione su dataset simulato

In questo capitolo si propone un'applicazione della classe di modelli sviluppati nel Capitolo 3 ad un dataset simulato: la scelta di utilizzare quest'ultimo anzichè un dataset reale riguarda principalmente un'opportunità operativa nello svolgere la stima dei modelli con tempi computazionali ragionevoli dovuti alla presenza di poche variabili, oltre alla possibilità di sapere con certezza, conoscendo il meccanismo generatore dei dati, che questi presentano una struttura sufficientemente ragionevole e valida da motivare l'utilizzo dei modelli proposti.

Il dataset simulato contiene 3000 osservazioni generate da una mistura di $K = 4$ gaussiane multivariate trivariate ($p = 3$) equipesate con diversi vettori delle medie e matrici di covarianza scelti per fare in modo che i quattro *cluster* siano sufficientemente vicini tra loro da non essere immediatamente distinguibili a livello esplorativo. All'interno di ciascun cluster le modalità della variabile risposta sono state assegnate alle osservazioni ipotizzando la presenza di iperpiani con diverse pendenze che realizzano una discriminazione lineare tra i due gruppi. La scelta della dimensione $p = 3$ è motivata dall'esigenza di poter mostrare graficamente la natura del dataset generato e dei risultati ottenuti, oltre alla possibilità di estendere lo spazio dei regressori con opportuni kernel polinomiali in fase di stima delle SVM senza eccessivi sforzi computazionali. In Codice B.7 è riportato il listato di comandi per la creazione del dataset simulato.

In Figura 4.1 viene mostrato lo scatterplot tridimensionale del dataset generato, colorando ciascuna osservazione $\mathbf{x}_i = (x_{i1}, x_{i2}, x_{i3})$ con un colore diverso a seconda del cluster dal quale è stata generata. I simboli con i quali sono stati rappresentati i punti rappre-

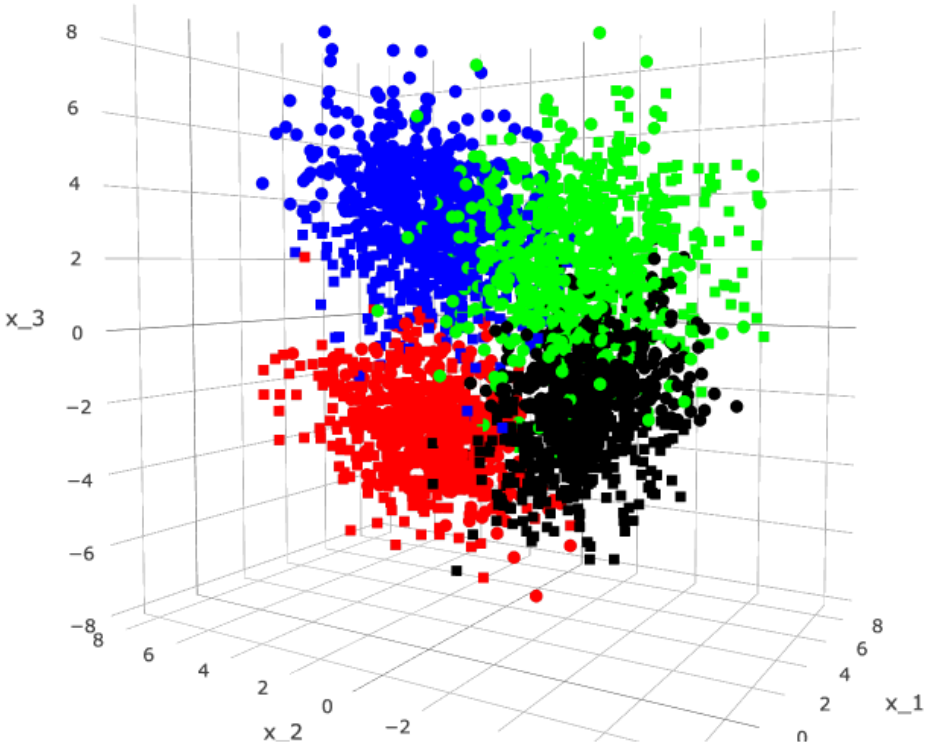


Figura 4.1: Scatterplot tridimensionale del dataset simulato. Ogni colore rappresenta la componente della mistura dalla quale è stata generata l'osservazione, mentre i simboli rappresentano rispettivamente le modalità -1 (quadrati) e +1 (cerchi) della variabile risposta associata.

sentano invece la modalità associata alla variabile risposta per l'osservazione in esame: un quadrato per le modalità $y_i = -1$ ed un cerchio per quelle $y_i = 1$. Globalmente il dataset può essere ritenuto sufficientemente valido per testare se il modello proposto nel Capitolo 3 è in grado di migliorare l'abilità predittiva rispetto ad una SVM di classificazione; ci si aspetta in particolare che il modello proposto sia in grado di svolgere le seguenti azioni:

- Identificare correttamente il numero K di sottopopolazioni latenti che hanno generato i dati, assieme alle proporzioni reali di ciascuna componente: questo avviene, nella pratica, andando a scegliere il K del modello che esibisce le migliori misure di *performance* sull'insieme di verifica;

- Svolgere un'adeguata selezione delle variabili tramite la penalizzazione Lasso su cia-

Modello	MisclassError	Recall	Precision	Proporzioni) (a meno di permutazioni)
$K = 1, d = 1$	0.46	0.53	0.54	—
$K = 1, d = 2$	0.28	0.68	0.73	—
$K = 2, d = 1$	0.28	0.73	0.70	(0.49, 0.51)
$K = 2, d = 2$	0.14	0.85	0.86	(0.48, 0.52)
$K = 3, d = 1$	0.22	0.78	0.77	(0.26, 0.27, 0.47)
$K = 3, d = 2$	0.14	0.86	0.90	(0.26, 0.27, 0.47)
$K = 4, d = 1$	0.14	0.84	0.87	(0.23, 0.25, 0.26, 0.26)
$K = 4, d = 2$	0.10	0.89	0.90	(0.24, 0.24, 0.26, 0.26)
$K = 5, d = 1$	0.14	0.85	0.87	(0.25, 0.24, 0.26, 0.25, 0)
$K = 5, d = 2$	0.10	0.89	0.90	(0, 0.24, 0.24, 0.25, 0.27)

Tabella 4.1: Misure di *performance* sulle matrici di confusione *finali* calcolate su ciascuno dei 5 modelli stimati. L'ultima colonna riporta le proporzioni empiriche sulle componenti della mistura stimate per tutti i modelli con $K > 1$.

scuna SVM stimata, andando nella pratica a concentrare le distribuzioni a posteriori sui parametri associati alle variabili irrilevanti attorno allo 0;

- Garantire un'adeguata convergenza delle catene simulate alla distribuzione a posteriori del modello;

- Eguagliare o migliorare i risultati predittivi che si raggiungerebbero ricorrendo ad espansioni kernel dello spazio dei regressori su un modello di SVM standard, evitando la stima di modelli eccessivamente parametrizzati.

L'applicazione è stata svolta andando a suddividere il dataset in un insieme di stima ed uno di verifica, mantenendo invariate le proporzioni delle modalità assunte dalla variabile risposta, composti rispettivamente dal 75% e dal restante 25% delle osservazioni. Per ciascun modello stimato, un algoritmo di Gibbs Sampling per la simulazione dalla distribuzione a posteriori è stato eseguito su 2000 iterazioni, scartando la prima metà di *burnin* ed andando a calcolare una matrice di confusione *finale* sull'insieme di verifica seguendo le metodologie già richiamate negli esempi applicativi illustrati nelle Sottosezioni 3.4.1 e 3.4.2 del Capitolo 3.

In Tabella 4.1 si riportano sinteticamente alcune misure di *performance* che consentono di esprimere giudizi in merito all'abilità predittiva di ciascun modello stimato. I risultati sembrano essere in linea con quanto atteso: la classe dei modelli proposti in questo lavoro sembra comportarsi meglio delle SVM standard (casi $K = 1$); in particolare tutti i modelli mistura considerati senza espansioni kernel dello spazio dei regressori (casi $d = 1$) si comportano decisamente meglio rispetto al modello $K = 1, d = 2$ che prevede l'introduzione di un kernel polinomiale quadratico e, dunque, un aumento del numero di parametri da stimare dovuto alle funzioni di base incluse nel modello.

Un secondo aspetto evidenziato riguarda le proporzioni stimate per le componenti della mistura: tenendo conto che ciascuna delle quattro vere componenti dalle quali sono stati generati i dati è equiprobabile (peso 0.25), per i casi $K = 2$ il modello sembra identificare le due componenti andando a raggruppare a due a due i quattro cluster; per i casi $K = 3$, invece, viene identificata una componente con peso vicino a 0.50, verosimilmente composta dai cluster verde e nero in Figura 4.1 che sono più vicini, e le due componenti rimanenti identificate nei cluster rosso e blu; infine, nei casi $K = 4$ il modello identifica correttamente le componenti (e rispettivi pesi) della *vera* mistura dalla quale sono stati generati i dati, come mostra la Figura 4.2 in cui a ciascuna osservazione dell'insieme di stima è associato un colore a seconda del cluster al quale è stata assegnata nell'ultimo aggiornamento del Gibbs Sampler relativo al vettore $\mathbf{Z}$: il fatto che l'ordine dei colori differisca da quello utilizzato in Figura 4.1 è dovuto all'interscambiabilità delle componenti della mistura in fase di stima del modello. Particolarmente interessanti sono i risultati mostrati per i modelli stimati con $K = 5$ componenti: andando ad osservare le proporzioni stimate, il modello assegna correttamente un peso nullo ad una delle cinque componenti, di fatto *rigettando* la presenza di un'ulteriore *cluster* latente sul meccanismo che ha generato i dati; questo aspetto è confermato anche dall'uguaglianza delle misure di *performance* calcolate sui modelli con $K = 4$ e $K = 5$ componenti.

La classe di modelli proposta suggerisce dunque un criterio operativo per capire quando arrestarsi nello sperimentare sequenzialmente modelli con un numero di componenti K sempre più elevato: ogniqualvolta si stimerà un modello con K^* componenti in cui una di queste risulta avere un peso nullo (o assimilabile allo 0), si sceglierà quello ottimale tra tutti quelli con $K < K^*$ componenti sulla base di una metrica di *performance* prefissata; questo metodo applicato al dataset in esame ci consente di identificare correttamente il modello

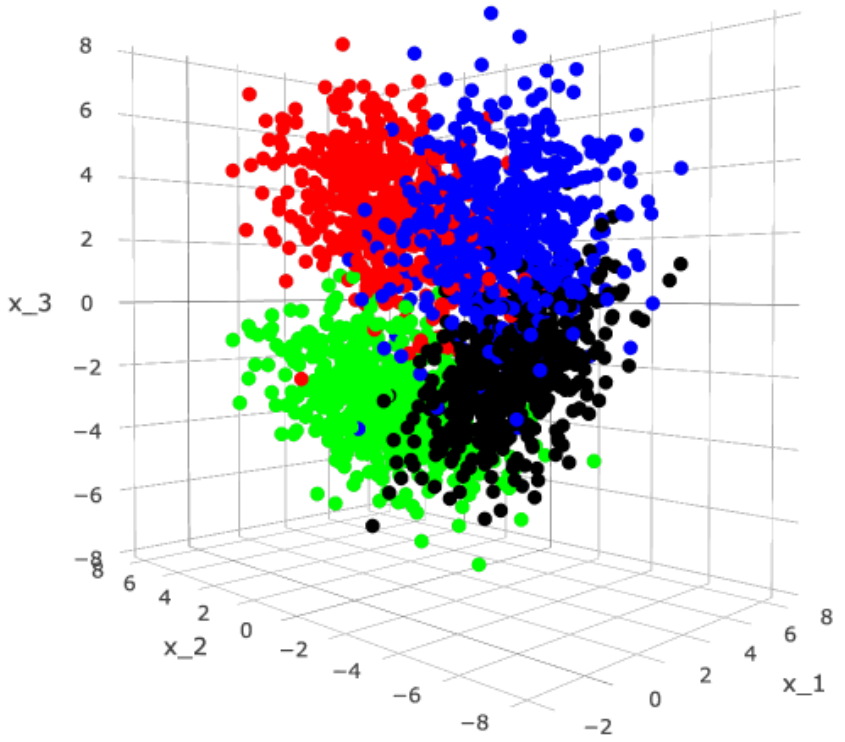


Figura 4.2: Scatterplot tridimensionale dell'insieme di stima utilizzato per stimare il modello al caso $K = 4, d = 2$. Ogni osservazione (punto) è colorata con un colore diverso a seconda del cluster al quale è stata assegnata nel vettore $\mathbf{Z}^{(N_{iter})}$ derivante dal Gibbs Sampler.

stimato al caso $K = 4, d = 2$ come ottimale, nonostante le metriche di performance siano pressochè identiche al modello stimato per il caso $K = 5, d = 2$.

Possibili ulteriori esempi di applicazione potrebbero essere svolti andando a complicare la struttura del dataset o perturbando la disposizione delle modalità della variabile risposta all'interno di ciascun cluster: in queste pagine ci si è limitati a mostrare come la classe di modelli proposta possa indubbiamente migliorare i risultati ottenuti da una SVM standard anche in dataset apparentemente elementari e *facili* come quello proposto.

Capitolo 5

Possibili estensioni e generalizzazioni

La classi di modelli presentata nel Capitolo 3 ammette numerose estensioni e generalizzazioni. Nelle prossime pagine si andranno ad approfondire nello specifico alcuni degli aspetti e delle scelte più critiche che hanno riguardato la formalizzazione dei modelli proposti, proponendo alcune possibili e valide estensioni percorribili.

5.1 Formalizzazione e tecniche inferenziali per problemi di regressione

Così come per la maggior parte dei modelli più diffusi nell'ambito del *machine learning*, nonostante le SVM siano state sviluppate e generalmente utilizzate in casi applicativi riguardanti problemi di classificazione, queste sono state estese per trattare problemi di regressione in cui la variabile risposta è di natura quantitativa.

Il problema da risolvere in questo contesto per ottenere le stime di β_0 e $\boldsymbol{\beta}$ della $f(\mathbf{x}) = \beta_0 + \mathbf{x}^T\boldsymbol{\beta}$ è simile a quanto già visto nella (1.10), ma in questo caso si utilizza la funzione di perdita ϵ-*insensitive loss* che ignora errori di ampiezza minore di ϵ; la stima del modello avverrà andando a minimizzare dunque la funzione di perdita penalizzata seguente:

$$d^r(\boldsymbol{\beta}, \beta_0, \lambda) = \sum_{i=1}^{n} \max(|y_i - f(\mathbf{x}_i)| - \epsilon, 0) + \frac{\lambda}{2}\|\boldsymbol{\beta}\|^2. \tag{5.1}$$

Valgono anche in questo contesto i ragionamenti già fatti circa la possibilità di aumentare lo spazio dei regressori con funzioni di base e trasformazioni delle variabili incluse nell'insieme di stima, e l'estendibilità del tipo di penalizzazione alla classe $\mathscr{L}_\alpha$.

Anche l'impianto sottostante la metodologia inferenziale per problemi di regressione su SVM in un'ottica bayesiana ricalca quanto già esposto nella Sezione 1.2 circa i problemi di classificazione; quel che cambia riguarda principalmente la funzione di perdita *ϵ-insensitive loss* inclusa nella (5.1), per la quale è opportuno individuare una nuova rappresentazione aumentata.

Si considerino dunque n coppie $\{(y_i, \mathbf{x}_i)\}_i^n$ appartenenti all'insieme di stima, con $y_i \in \mathbb{R}$ e $\mathbf{x}_i = (1, x_{i1}, x_{i2}, \ldots, x_{i\,p-1}) \in \mathbb{R}^p$, dove anche in questo caso per evitare complicazioni di notazione assumiamo che $\mathbf{x}_i$ includa già eventuali espansioni di base $h(\mathbf{x}_i)$.

La funzione obbiettivo da minimizzare per ottenere la stima di $\boldsymbol{\beta}$ è

$$d_\alpha^r(\boldsymbol{\beta}, \nu) = \sum_{i=1}^{n} \max\left(\left|y_i - \mathbf{x}_i^T\boldsymbol{\beta}\right| - \epsilon, 0\right) + \frac{1}{\nu^\alpha}\sum_{j=1}^{p}\left|\frac{\beta_j}{\sigma_j}\right|^\alpha \tag{5.2}$$

dove, anche per i problemi di regressione, si ammette maggiore flessibilità circa la scelta del grado di penalizzazione. Anche in questo caso è possibile interpretare il problema in ottica bayesiana operando una trasformazione esponenziale alla (5.2) cambiata di segno, rappresentando la funzione risultante in termini di pseudo-distribuzione a posteriori per $\boldsymbol{\beta}$:

$$p(\boldsymbol{\beta}|\nu, \alpha, \mathbf{y}) \propto \exp\left(-\sum_{i=1}^{n}\max(\left|y_i - \mathbf{x}_i^T\boldsymbol{\beta}\right| - \epsilon, 0)\right)\left(-\frac{1}{\nu^\alpha}\sum_{j=1}^{p}\left|\frac{\beta_j}{\sigma_j}\right|^\alpha\right) \tag{5.3}$$

$$\propto \underbrace{\left(\prod_{i=1}^{n}\exp(-\max(\left|y_i - \mathbf{x}_i^T\boldsymbol{\beta}\right| - \epsilon, 0))\right)}_{\text{Pseudo-Verosimiglianza}}\underbrace{\left(\prod_{j=1}^{p}\exp\left(-\frac{1}{\nu^\alpha}\left|\frac{\beta_j}{\sigma_j}\right|^\alpha\right)\right)}_{\text{Distribuzione a priori}}; \tag{5.4}$$

minimizzare la (5.2) corrisponderà nuovamente ad individuare il massimo della pseudo-distribuzione a posteriori definita come

$$p(\boldsymbol{\beta}|\nu, \alpha, \mathbf{y}) \propto C_\alpha(\nu)\mathcal{L}(\mathbf{y}|\boldsymbol{\beta})p(\boldsymbol{\beta}|\alpha, \nu) \tag{5.5}$$

in analogia a quanto fatto nel caso di problemi di classificazione per la (1.13).

Rispetto ai problemi di classificazione, l'implementazione di un Gibbs Sampler per la simulazione dalla distribuzione a posteriori necessita in questo caso di una nuova rappresentazione aumentata per la *ϵ-insensitive loss*: questa è stata proposta da Zhu et al. (2014)

e consente di scrivere l'i-esimo contributo alla pseudo-verosimiglianza $\mathcal{L}_i(y_i|\boldsymbol{\beta})$ come il prodotto di due misture di posizione e scala nelle quali $\boldsymbol{\beta}$ appare all'interno di una forma quadratica:

$$\begin{aligned}\mathcal{L}_i(y_i|\boldsymbol{\beta}) &= \exp\left(-2\max(|y_i - \mathbf{x}_i^T\boldsymbol{\beta}| - \epsilon, 0)\right) \\ &= \int_0^\infty \frac{1}{\sqrt{2\pi\lambda_i^-}} \exp\left(-\frac{1}{2}\frac{(\lambda_i^- + (y_i - \mathbf{x}_i^T\boldsymbol{\beta} - \epsilon))^2}{\lambda_i^-}\right) d\lambda_i^- \\ &\quad \times \int_0^\infty \frac{1}{\sqrt{2\pi\lambda_i^+}} \exp\left(-\frac{1}{2}\frac{(\lambda_i^+ - (y_i - \mathbf{x}_i^T\boldsymbol{\beta} + \epsilon))^2}{\lambda_i^+}\right) d\lambda_i^+ \\ &= \int_0^\infty \mathsf{N}(y_i - \mathbf{x}_i^T\boldsymbol{\beta} - \epsilon; -\lambda_i^-, \lambda_i^-) d\lambda_i^- \times \int_0^\infty \mathsf{N}(y_i - \mathbf{x}_i^T\boldsymbol{\beta} + \epsilon; \lambda_i^+, \lambda_i^+) d\lambda_i^+. \quad (5.6)\end{aligned}$$

Questo risultato, assieme alla possibilità di esprimere la distribuzione a priori sul singolo coefficiente β_j come una mistura di scala di distribuzioni normali (1.15) già vista nel Capitolo 1 per i problemi di classificazione, consente di esprimere la pseudo-posteriori (5.5) come marginale di una pseudo-posteriori *aumentata* e comprensiva delle variabili latenti $\boldsymbol{\lambda}^- = (\lambda_1^-, \lambda_2^-, \ldots, \lambda_n^-)$, $\boldsymbol{\lambda}^+ = (\lambda_1^+, \lambda_2^+, \ldots, \lambda_n^+)$ e $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_p)$:

$$\begin{aligned}p(\boldsymbol{\beta}, \boldsymbol{\lambda}^-, \boldsymbol{\lambda}^+, \boldsymbol{\omega}|\mathbf{y}, \alpha, \nu) \propto &\left(\prod_{i=1}^n (\lambda_i^-)^{-1/2}\right) \exp\left(-\frac{1}{2}\sum_{i=1}^n \frac{\left(\lambda_i^- + (y_i - \mathbf{x}_i^T\boldsymbol{\beta} - \epsilon)\right)^2}{\lambda_i^-}\right) \\ &\times \left(\prod_{i=1}^n (\lambda_i^+)^{-1/2}\right) \exp\left(-\frac{1}{2}\sum_{i=1}^n \frac{\left(\lambda_i^+ - (y_i - \mathbf{x}_i^T\boldsymbol{\beta} + \epsilon)\right)^2}{\lambda_i^+}\right) \\ &\times \left(\prod_{j=1}^p \omega_j^{-1/2} p(\omega_j|\alpha)\right) \exp\left(-\frac{1}{2\nu^2}\sum_{j=1}^p \frac{\beta_j^2}{\sigma_j^2\omega_j}\right) \quad (5.7)\end{aligned}$$

ovvero, in forma compatta,

$$\begin{aligned}p(\boldsymbol{\beta}, \boldsymbol{\lambda}^-, \boldsymbol{\lambda}^+, \boldsymbol{\omega}|\mathbf{y}, \alpha, \nu) \propto &\,\mathsf{N_n}(\mathbf{y} - \epsilon\mathbf{1}_n + \boldsymbol{\lambda}^-; \mathbf{X}\boldsymbol{\beta}, \boldsymbol{\Lambda}^-) \times \mathsf{N_n}(\mathbf{y} + \epsilon\mathbf{1}_n - \boldsymbol{\lambda}^+; \mathbf{X}\boldsymbol{\beta}, \boldsymbol{\Lambda}^+) \\ &\times \mathsf{N_p}(\boldsymbol{\beta}; \mathbf{0}, \nu^2\boldsymbol{\Omega}^{1/2}\boldsymbol{\Sigma}\boldsymbol{\Omega}^{1/2}) \times \prod_{j=1}^p p(\omega_j|\alpha) \quad (5.8)\end{aligned}$$

dove $\boldsymbol{\Lambda}^- = \mathsf{diag}(\boldsymbol{\lambda}^-)$ e $\boldsymbol{\Lambda}^+ = \mathsf{diag}(\boldsymbol{\lambda}^+)$.

A partire da questa rappresentazione, è possibile ricavare agilmente le seguenti *full-conditional* a posteriori per $\boldsymbol{\beta}$ e per le variabili latenti ed implementare un opportuno schema di Gibbs Sampling:

Full-conditional a posteriori per $\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}, \boldsymbol{\lambda}^-, \boldsymbol{\lambda}^+, \boldsymbol{\omega}, \boldsymbol{\Sigma}, \epsilon$:

I passaggi per ricavare la distribuzione a posteriori $p(\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}, \boldsymbol{\lambda}^-, \boldsymbol{\lambda}^+, \boldsymbol{\omega}, \boldsymbol{\Sigma}, \epsilon)$ richiamano quanto già visto nella Sezione 1.2 per l'inferenza bayesiana su SVM per problemi di classificazione. In particolare Zhu et al. (2014) ricavano la forma della *full-conditional* a posteriori seguente:

$$\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}, \boldsymbol{\lambda}^-, \boldsymbol{\lambda}^+, \boldsymbol{\omega}, \boldsymbol{\Sigma}, \epsilon \sim \mathsf{N_p}(\mathbf{b}, \mathbf{B}) \tag{5.9}$$

dove $\mathbf{B}^{-1} = \nu^{-2}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Omega}^{-1} + \mathbf{X}^T\tilde{\boldsymbol{\Lambda}}\mathbf{X}$ e $\mathbf{b} = \mathbf{B}\mathbf{X}^T\left(\boldsymbol{\Lambda}^-(\mathbf{y} - \epsilon\mathbf{1}) + \boldsymbol{\Lambda}^+(\mathbf{y} + \epsilon\mathbf{1})\right)$, con $\tilde{\boldsymbol{\Lambda}} = (\boldsymbol{\Lambda}^-)^{-1} + (\boldsymbol{\Lambda}^+)^{-1}$.

Full-conditional a posteriori per $\lambda_i^-|y_i, \mathbf{x}_i, \boldsymbol{\beta}, \epsilon$ e $\lambda_i^+|y_i, \mathbf{x}_i, \boldsymbol{\beta}, \epsilon$:

Zhu et al. (2014) mostrano che le *full-conditional* a posteriori per λ_i^- e λ_i^+ dipendono solo da $\boldsymbol{\beta}$, e corrispondono rispettivamente a

$$\lambda_i^-|y_i, \mathbf{x}_i, \boldsymbol{\beta}, \epsilon \sim \mathsf{GenInvGaussian}\left(\frac{1}{2}, 1, (y_i - \mathbf{x}_i^T\boldsymbol{\beta} - \epsilon)^2\right) \tag{5.10}$$

$$\lambda_i^+|y_i, \mathbf{x}_i, \boldsymbol{\beta}, \epsilon \sim \mathsf{GenInvGaussian}\left(\frac{1}{2}, 1, (y_i - \mathbf{x}_i^T\boldsymbol{\beta} + \epsilon)^2\right) \tag{5.11}$$

o, equivalentemente,

$$(\lambda_i^-)^{-1}|y_i, \mathbf{x}_i, \boldsymbol{\beta}, \epsilon \sim \mathsf{InvGaussian}\left(\left|y_i - \mathbf{x}_i^T\boldsymbol{\beta} - \epsilon\right|^{-1}, 1\right) \tag{5.12}$$

$$(\lambda_i^+)^{-1}|y_i, \mathbf{x}_i, \boldsymbol{\beta}, \epsilon \sim \mathsf{InvGaussian}\left(\left|y_i - \mathbf{x}_i^T\boldsymbol{\beta} + \epsilon\right|^{-1}, 1\right). \tag{5.13}$$

Full-conditional a posteriori per $\omega_j|\beta_j, \nu, \boldsymbol{\Sigma}, \alpha$:

Per la *full-conditional* di ω_j, valgono i ragionamenti già fatti nella Sezione 1.2 circa la difficoltà di operare con la distribuzione misturizzante stabile positiva. L'unico caso *trattabile* riguarda la penalizzazione Lasso ($\alpha = 1$), per la quale è possibile ricavare esplicitamente tale distribuzione:

$$\omega_j|\beta_j, \nu, \boldsymbol{\Sigma}, \alpha = 1 \sim \mathsf{GenInvGaussian}\left(\frac{1}{2}, 1, \frac{\beta_j^2}{\nu^2\sigma_j^2}\right) \tag{5.14}$$

o, equivalentemente,

$$\omega_j^{-1}|\beta_j, \nu, \boldsymbol{\Sigma}, \alpha = 1 \sim \mathsf{InvGaussian}\left(\frac{\nu\sigma_j}{|\beta_j|}, 1\right). \tag{5.15}$$

Anche in questo caso, per la penalizzazione Ridge ($\alpha = 2$) la full-conditional risulta essere degenere in 1.

Full-conditional a posteriori per $\nu^{-1}|\boldsymbol{\beta}, \boldsymbol{\Sigma}, a_\nu, b_\nu$:

L'inferenza bayesiana sulle SVM per problemi di regressione, così come per i problemi di classificazione, consente di trattare ν^{-1} come variabile casuale, al pari degli altri parametri: la proposta anche in questo caso è quella di fissare una distribuzione a priori $\nu^{-1} \sim \mathsf{Gamma}(a_\nu, b_\nu)$, da aggiungere opportunamente alla (5.8), che induce una distribuzione a posteriori

$$\nu^{-1}|\boldsymbol{\beta}, \boldsymbol{\Sigma}, a_\nu, b_\nu, \alpha \sim \mathsf{Gamma}\left(a_\nu + p, b_\nu + \sum_{j=1}^{p}\left|\frac{\beta_j}{\sigma_j}\right|\right). \tag{5.16}$$

Analogamente a quanto già affrontato nel Capitolo 3 per problemi di classificazione, è possibile formalizzare la classe di modelli a mistura finita di gaussiane multivariate anche su SVM per problemi di regressione. Le ragioni che ne motivano lo sviluppo esposte nella Sezione 3.1, così come la formalizzazione del modello esposto nella Sezione 3.2 e la metodologia inferenziale percorsa nella Sezione 3.3 per problemi di classificazione sono immediatamente estendibili al caso della regressione sfruttando la rappresentazione aumentata per la funzione di perdita ϵ-*insentitive loss* esposta nella (5.6) e le conseguenti forme delle full-conditional appena richiamate.

Anzichè ripercorrere nuovamente quanto fatto nelle Sezioni 3.2 e 3.3 adattandolo al caso dei problemi di regressione, trattandosi nella pratica di esigui aggiustamenti e modifiche sulla base di quanto appena richiamato, per questioni di spazio viene riportato solo lo pseudocodice riguardante l'algoritmo di Gibbs Sampling per la simulazione dalla distribuzione a posteriori del modello, i cui passi principali sono riportati in Algoritmo 4 ed implementati operativamente in Codice A.6. Valgono anche questo caso i ragionamenti descritti nella Sezione 3.4 per quanto riguarda la verifica della bontà predittiva del modello: trattandosi ora di problemi di classificazione si dovrà andare alla ricerca di un nuovo strumento (diverso dalla tabella di confusione, valida solo per problemi di classificazione) che consenta di misurare l'abilità predittiva e di confrontarla con quella di altri modelli.

Algorithm 4 Collapsed Gibbs Sampling per una mistura finita gaussiana multivariata di SVM per problemi di regressione

Disponendo del vettore $\mathbf{y} = (y_1, \ldots, y_n)$ contenente i valori variabile risposta $y_i \in \mathbf{R}$, della matrice $\mathbf{X}$ di dimensione $n \times (p+m)$ contenente le p colonne dei regressori ed intercetta, ed eventuali m colonne corrispondenti alle funzioni di base che permettono di includere kernel, del vettore degli iperparametri (a_ν, b_ν) riferiti alla priori su ciascun ν_j^{-1}, del vettore delle standard deviation σ_j per ciascuna variabile della matrice $\mathbf{X}$, del grado scelto per la penalizzazione $\alpha \in \{1, 2\}$, dell'iperparametro di concentrazione α_0 riferito alla priori su $\boldsymbol{\pi}$ e del margine ϵ, assegniamo i vettori di valori iniziali per $\boldsymbol{\beta}_j$ a $\boldsymbol{\beta}_j^{(1)}$, $j = 1, 2, \ldots, K$, quello per $\boldsymbol{\lambda}^-$ a $(\boldsymbol{\lambda}^-)^{(1)}$, quello per $\boldsymbol{\lambda}^+$ a $(\boldsymbol{\lambda}^+)^{(1)}$, quelli per $\boldsymbol{\omega}_j$ a $\boldsymbol{\omega}_j^{(1)}$, $j = 1, 2, \ldots, K$, quello per $\boldsymbol{\nu}$ a $\boldsymbol{\nu}^{(1)}$ e quello per $\mathbf{Z}$ a $\mathbf{Z}^{(1)}$.
Per $t = 2, 3, \ldots, R$ si eseguono iterativamente le seguenti operazioni salvando, man mano, i valori generati:

1. Sostituisco $\mathbf{Z}^{(t-1)}$ a $\mathbf{Z}^{(t)}$ per facilitare i calcoli nel passo 1.

2. Generazione dalla *full-conditional* a posteriori per Z_i, $i = 1, 2, \ldots, n$:

$$Z_i^{(t)} = j | \mathbf{X}, \mathbf{Z}_{-i}^{(t)} \propto \frac{n_{j,-i}^{(t)} + \alpha_0/K}{n + \alpha_0 - 1} \, \mathsf{T}_{\nu_{n_{j,-i}} - p + 1} \left(\mathbf{x}_i; \boldsymbol{\mu}_{n_{j,-i}}, \frac{\kappa_{n_{j,-i}} + 1}{\kappa_{n_{j,-i}} \left(\nu_{n_{j,-i}} - p + 1 \right)} \boldsymbol{\Sigma}_{n_{j,-i}} \right) \quad j = 1, 2, \ldots, K.$$

3. Generazione dalla *full-conditional* a posteriori per λ_i^- e λ_i^+, $i = 1, 2, \ldots, n$:

$$\left((\lambda_i^-)^{(t)} \right)^{-1} \Big| \, \mathbf{y}, \mathbf{X}, \mathbf{Z}^{(t)}, \boldsymbol{\beta}_1^{(t-1)}, \ldots, \boldsymbol{\beta}_K^{(t-1)} \sim \mathsf{InvGaussian} \left(\left| y_i - \mathbf{x}_i^T \boldsymbol{\beta}_{Z_i^{(t)}}^{(t-1)} - \epsilon \right|^{-1}, 1 \right)$$

$$\left((\lambda_i^+)^{(t)} \right)^{-1} \Big| \, \mathbf{y}, \mathbf{X}, \mathbf{Z}^{(t)}, \boldsymbol{\beta}_1^{(t-1)}, \ldots, \boldsymbol{\beta}_K^{(t-1)} \sim \mathsf{InvGaussian} \left(\left| y_i - \mathbf{x}_i^T \boldsymbol{\beta}_{Z_i^{(t)}}^{(t-1)} + \epsilon \right|^{-1}, 1 \right)$$

4. Generazione dalla *full-conditional* a posteriori per $\boldsymbol{\beta}_j$, $j = 1, 2, \ldots, K$:

$$\boldsymbol{\beta}_j^{(t)} \Big| \, \mathbf{y}, \mathbf{X}, \mathbf{Z}^{(t)}, \boldsymbol{\lambda}^{(t-1)}, \boldsymbol{\omega}_j^{(t-1)} \sim \mathsf{N}_p \left(\mathbf{b}_j, \mathbf{B}_j \right)$$

con $\mathbf{B}_j^{-1} = \left(\nu_j^{(t-1)} \right)^{-2} \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{\Omega}_j^{(t-1)} \right)^{-1} + \mathbf{X}_j^T \tilde{\boldsymbol{\Lambda}}_j^{(t)} \mathbf{X}_j$, $\tilde{\boldsymbol{\Lambda}}_j^{(t)} = \left(\left(\boldsymbol{\Lambda}^- \right)^{(t)} \right)^{-1} + \left(\left(\boldsymbol{\Lambda}^+ \right)^{(t)} \right)^{-1}$,
$\mathbf{b}_j = \mathbf{B}_j \mathbf{X}_j^T \left(\left((\boldsymbol{\Lambda}_j^-)^{(t)} \right)^{-1} \left(\mathbf{y}_j - \epsilon \mathbf{1}_{n_j^{(t)}} \right) + \left((\boldsymbol{\Lambda}_j^+)^{(t)} \right)^{-1} \left(\mathbf{y}_j + \epsilon \mathbf{1}_{n_j^{(t)}} \right) \right)$, dove $\mathbf{1}_{n_j^{(t)}}$ è un vettore di 1 lungo $n_j^{(t)}$, $\left(\boldsymbol{\Lambda}_j^- \right)^{(t)}$ e $\left(\boldsymbol{\Lambda}_j^+ \right)^{(t)}$ sono rispettivamente due matrici diagonali contenenti tutti e solo i $(\lambda_i^-)^{(t)}$ e $(\lambda_i^+)^{(t)}$ tali per cui $i \in \left\{ i : Z_i^{(t)} = j \right\}$, $\boldsymbol{\Omega}_j^{(t-1)} = \mathsf{diag} \left(\boldsymbol{\omega}_j^{(t-1)} \right)$ e $\boldsymbol{\Sigma} = \mathsf{diag} \left(\boldsymbol{\sigma}^{\mathbf{2}} \right)$

5. Generazione dalla *full-conditional* a posteriori per ν_j^{-1}, $j = 1, 2, \ldots, K$:

$$\left(\nu_j^{(t)} \right)^{-1} \Big| \, \boldsymbol{\beta}_j^{(t)}, \alpha = 1 \sim \mathsf{Gamma} \left(a_\nu + p, b_\nu + \sum_{h=1}^{p+m} \left| \frac{\beta_{hj}^{(t)}}{\sigma_h} \right| \right)$$

dove $\beta_{hj}^{(t)}$ è la h-esima componente di $\boldsymbol{\beta}_j^{(t)}$. Se $\alpha = 2$, si tratta di elevare al quadrato il risultato appena ottenuto.

6. Generazione dalla *full-conditional* a posteriori per ω_{hj}, $j = 1, 2, \ldots, K$ e $h = 1, 2, \ldots, p+m$:

$$\left(\omega_{hj}^{(t)} \right)^{-1} \Big| \, \boldsymbol{\beta}_j^{(t)}, \nu_j^{(t-1)}, \alpha = 1 \sim \mathsf{InvGaussian} \left(\nu_j^{(t-1)} \frac{\sigma_h}{\left| \beta_{hj}^{(t)} \right|}, 1 \right)$$

dove $\beta_{hj}^{(t)}$ è la h-esima componente di $\boldsymbol{\beta}_j^{(t)}$. Se $\alpha = 2$, invece, $\omega_{hj}^{(t)} = 1, \forall h$ e $\forall j$.

Si effettua un *burnin* iniziale e si restituiscono i risultati.

5.2 Misture di t-student multivariate per SVM

Nel Capitolo 3 è già stato ampiamente discusso come la scelta di modellare ciascuna componente della mistura con una distribuzione gaussiana multivariata comporti alcuni vantaggi notevoli: dal punto di vista concettuale, questa scelta permette di rifarsi (quantomeno teoricamente) alla proprietà di approssimatore universale della mistura; dal punto di vista pratico, questa scelta consente di implementare uno schema di Gibbs Sampling per la simulazione dalla distribuzione a posteriori particolarmente efficiente dovuto alla coniugazione naturale tra la priori NIW_r e la distribuzione gaussiana multivariata, seguendo le formule (2.43)-(2.46) per l'aggiornamento degli iperparametri.

Allo stesso modo, la distribuzione gaussiana multivariata potrebbe però non essere sufficientemente flessibile per cogliere adeguatamente le sottopopolazioni latenti del fenomeno di interesse, a maggior ragione quando queste non hanno forme ovalidi o sono caratterizzate da possibili osservazioni influenti che vanno ad influire pesantemente sul processo di *clustering*, alterandone le conclusioni.

Molte strade alternative possono essere percorse, e non è detto che una classe di famiglie distributive sia sempre migliore di altre: la scelta potrebbe dipendere dal contesto applicativo al quale si fa riferimento, da possibili analisi esplorative condotte *ex-ante* o da conoscenze teoriche. Riguardo lo spettro di scelte possibili, un'immediata generalizzazione riguarda la classe delle famiglie distributive ellittiche tra le quali svettano, per fama e notorietà, le distribuzioni *skew*-gaussiane multivariate (Azzalini (2014)) che consentono di modellare adeguatamente eventuali asimmetrie presenti nei dati; nonostante questa abilità le renda particolarmente attrattive per la classe di modelli sviluppata, è ampiamente risaputo come l'inferenza sui parametri non sia del tutto immediata e richieda tecnicismi non indifferenti dovuti dalla necessità di andare a fissare una distribuzione a priori sul parametro di *skewness*, scelta non immediata e che è stato dimostrato potrebbe complicare l'inferenza dalla distribuzione a posteriori a causa di possibili irregolarità sulla forma della verosimiglianza. In ogni caso, per la finalità applicativa che si sta considerando Frühwirth-Schnatter e Pyne (2010) hanno sviluppato uno schema di Gibbs Sampling che consente di stimare misture finite di distribuzioni *skew*-gaussiane sfruttando una rappresentazione stocastica della distribuzione *skew*-gaussiana in termini di modello ad effetti casuali gaussiani troncati, immediatamente implementabile nella classe di modelli sviluppata in questo

lavoro.

Un'alternativa più facilmente percorribile riguarda la possibilità di considerare la famiglia distributiva delle t-student multivariate: questa è caratterizzata generalmente da code *più pesanti* che potrebbero essere in grado di gestire adeguatamente e in maniera più *robusta* la presenza di osservazioni influenti, come descritto poco fa.

Uno schema di Gibbs Sampling particolarmente conveniente per realizzare l'inferenza bayesiana su di una mistura finita a componenti t-student multivariate è contenuto in Frühwirth-Schnatter (2006) e si basa sulla possibilità di rappresentare una generica distribuzione t-student multivariata con ν gradi di libertà come mistura di scala infinita di distribuzioni gaussiane multivariate, nella quale si fissa come distribuzione misturizzante una Gamma:

$$\mathbf{Y}_i|\gamma_i \sim \mathsf{N}_r\left(\boldsymbol{\mu}, \gamma_i^{-1}\boldsymbol{\Sigma}\right), \quad \gamma_i \sim \mathsf{Gamma}\left(\frac{\nu}{2}, \frac{\nu}{2}\right) \quad i = 1, 2, \ldots, n \tag{5.17}$$

ovvero, per la generica componente j-esima della mistura:

$$\mathbf{Y}_i|Z_i = j, \gamma_i \sim \mathsf{N}_r\left(\boldsymbol{\mu}_j, \gamma_i^{-1}\boldsymbol{\Sigma}_j\right), \quad \gamma_i \sim \mathsf{Gamma}\left(\frac{\nu_j}{2}, \frac{\nu_j}{2}\right) \quad i = 1, 2, \ldots, n \tag{5.18}$$

per $j = 1, 2, \ldots, K$. Questa rappresentazione aumentata ci consente di interpretare una mistura di t-student multivariate come una mistura di gaussiane multivariate in cui, condizionatamente alla generica j-esima componente, tutte le osservazioni appartenenti hanno una comune $\boldsymbol{\mu}_j$ ma una diversa eterogeneità catturata dalla latente γ_i, che va a moltiplicare la comune $\boldsymbol{\Sigma}_j$.

Relativamente alla stima della mistura finita di t-student multivariate, la rappresentazione aumentata (5.18) permette di rifarsi all'inferenza già descritta nella Sottosezione 2.3.2 per le misture finite di gaussiane multivariate, andando ad operare alcune lievi modifiche sulle equazioni di aggiornamento degli iperparametri della NIW_r:

$$\kappa^*_{n_j} = \kappa_0 + n^*_j \quad j = 1, 2, \ldots, K \tag{5.19}$$

$$\nu^*_{n_j} = \nu_0 + n^*_j \quad j = 1, 2, \ldots, K \tag{5.20}$$

$$\boldsymbol{\mu}_{n^*_j} = \kappa^{-1}_{n^*_j}\left(\kappa_0\boldsymbol{\mu}_0 + \sum_{i:Z_i=j}\gamma_i\mathbf{y}_i\right) \quad j = 1, 2, \ldots, K \tag{5.21}$$

$$\boldsymbol{\Sigma}_{n^*_j} = \boldsymbol{\Sigma}_0 + \sum_{i:Z_i=j}\gamma_i\mathbf{y}_i\mathbf{y}_i^T + \kappa_0\boldsymbol{\mu}_0\boldsymbol{\mu}_0^T - \kappa_{n^*_j}\boldsymbol{\mu}_{n^*_j}\boldsymbol{\mu}_{n^*_j}^T \quad j = 1, 2, \ldots, K; \tag{5.22}$$

con $n_j^* = \sum_{i=1}^{n} \mathbb{1}(Z_i = j)\,\gamma_i$, ed introducendo un ulteriore passo di simulazione dalla full-conditional per γ_i, che risulta essere:

$$\gamma_i|\boldsymbol{\mu}_{Z_i}, \boldsymbol{\Sigma}_{Z_i}, \nu_{Z_i}, \mathbf{y}_i \sim \mathsf{Gamma}\left(\frac{1}{2}(\nu_{Z_i} + r), \frac{1}{2}\left(\nu_{Z_i} + d_M(\mathbf{y}_i; \boldsymbol{\mu}_{Z_i}, \boldsymbol{\Sigma}_{Z_i})^2\right)\right) \quad \forall i \qquad (5.23)$$

dove $d_M(\mathbf{y}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \|\mathbf{y} - \boldsymbol{\mu}\|_{\boldsymbol{\Sigma}}$ denota l'usuale distanza di Mahalanobis. Si dà per scontato che il simbolo ν non crei confusione nel contraddistinguere i gradi di libertà della IW nelle equazioni (5.19)-(5.22) dai gradi di libertà della distribuzione t-student per ciascuna componente della mistura.

A questo punto, gli algoritmi di stima Algoritmo 3 ed Algoritmo 4 per i modelli proposti in questo lavoro possono essere adattati al caso di misture di t-student multivariate anzichè gaussiane multivariate sostituendo i passi 1. e 2. con i seguenti:

1. Generazione dalla *full-conditional* a posteriori per $(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, $j = 1, 2, \ldots, K$:

$$\left(\boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)}\right) \sim \mathsf{NIW}\left(\boldsymbol{\mu}_{n_j^*}, \kappa_{n_j^*}, \nu_{n_j^*}, \boldsymbol{\Sigma}_{n_j^*}\right)$$

   dove $\boldsymbol{\mu}_{n_j^*}$, $\kappa_{n_j^*}$, $\nu_{n_j^*}$ e $\boldsymbol{\Sigma}_{n_j^*}$ sono calcolate secondo le *nuove* formule di aggiornamento (5.19)-(5.22) rispetto a $\boldsymbol{\gamma}^{(t-1)}$.

2. Generazione dalla *full-conditional* a posteriori per γ_i, $i = 1, 2, \ldots, n$:

$$\gamma_i^{(t)}|\boldsymbol{\mu}_{Z_i^{(t-1)}}^{(t)}, \boldsymbol{\Sigma}_{Z_i^{(t-1)}}^{(t)}, \nu_{Z_i^{(t-1)}}, \mathbf{y}_i \sim \mathsf{Gamma}\left(\frac{\nu_{Z_i^{(t-1)}} + p}{2}, \frac{\nu_{Z_i^{(t-1)}} + d_M\left(\mathbf{y}_i; \boldsymbol{\mu}_{Z_i^{(t-1)}}^{(t)}, \boldsymbol{\Sigma}_{Z_i^{(t-1)}}^{(t)}\right)^2}{2}\right)$$

3. Generazione dalla *full-conditional* a posteriori per Z_i, $i = 1, 2, \ldots, n$:

$$Z_i^{(t)} = j|\mathbf{X}, \mathbf{Z}_{-i}^{(t)}, \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)}, \gamma_i^{(t)} \propto \frac{n_{j,-i}^{(t)} + \alpha_0/K}{n + \alpha_0 - 1}\,\mathsf{N}_p\left(\mathbf{x}_i; \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)}/\gamma_i^{(t)}\right)$$

   per $j = 1, 2, \ldots, K$. Il clustering è realizzato esclusivamente sulle p variabili *originali*. In alternativa è possibile far riferimento direttamente alla densità della t-student multivariata senza ricorrere alla versione aumentata, e dunque definire

$$Z_i^{(t)} = j|\mathbf{X}, \mathbf{Z}_{-i}^{(t)}, \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)} \propto \frac{n_{j,-i}^{(t)} + \alpha_0/K}{n + \alpha_0 - 1}\,\mathsf{T}_{\nu_j}\left(\mathbf{x}_i; \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)}\right)$$

   per $j = 1, 2, \ldots, K$. Ci si aspetta che questa seconda formula assicuri un *mixing* migliore delle catene, per via della marginalizzazione di γ_i.

L'estensione del modello alla mistura di t-student multivariate, come mostrato, è nella pratica immediata: esistono però alcune lievi differenze che potrebbero portare a preferire l'una o l'altra famiglia di distribuzioni in fase di implementazione.

La prima differenza riguarda la marginalizzazione richiamata nella (3.37) che consente di evitare il passo di simulazione dalla distribuzione a posteriori per $\left(\boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)}\right)$ negli Algoritmi 3 e 4: questo stratagemma non è nuovamente implementabile perchè le coppie $\left(\boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)}\right)$ sono necessarie per il calcolo degli iperparametri della full-conditional per la latente $\gamma_i^{(t)}$ riportata nel nuovo passo 2.

La seconda differenza riguarda la scelta dei gradi di libertà ν_j per ciascuna componente della mistura t-student: fino ad ora questi sono stati considerati fissi e scelti *a priori* in fase di definizione del modello. Frühwirth-Schnatter e Pyne (2010) suggeriscono la possibilità di ammettere ulteriore flessibilità al modello andando a trattarli al pari degli altri parametri ignoti: la scelta della priori $p(\nu_j)$ dovrà essere svolta in modo accurato onde evitare il raggiungimento di distribuzioni a posteriori improprie, e all'interno del loro lavoro suggeriscono alcune delle scelte più diffuse in letteratura. Indipendentemente dalla scelta, non esiste la possibilità di ricavare esplicitamente una *full-conditional* per la distribuzione a posteriori: viene dunque suggerito di calcolare $\nu_j^{(t)}$ come media della distribuzione a posteriori marginale

$$p\left(\nu_j | \mathbf{Z}, \{\mathbf{x}\}_{i=1}^n\right) \propto p(\nu_j) \prod_{i:Z_i=j} \mathsf{T}_{\nu_j}\left(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right) \tag{5.24}$$

per $j = 1, 2, \ldots, K$, approssimando quest'ultima con un opportuno passo di Metropolis-Hasting da includere tra i passi 2 e 3 riportati nella pagina precedente.

5.3 Selezione del numero K di componenti

Una delle questioni più dibattute nella letteratura dei modelli mistura riguarda la scelta del numero K di componenti che la compongono: questo fino ad ora è sempre stato supposto noto e fissato, ma nella realtà è considerabile come un parametro ignoto tanto quanto gli altri.

Utilizzare un approccio inferenziale di tipo bayesiano per la stima di K risulta particolarmente complesso, basti pensare alla necessità di dover specificare una opportuna

distribuzione a priori; esistono però alcune tecniche che permettono una selezione del numero adeguato di componenti della mistura basate principalmente su analisi esplorative dei dati che si intendono modellare, conoscenze teoriche sul fenomeno d'interesse o sul confronto di indici informativi calcolati su modelli stimati al variare di K, costruiti andando a penalizzare la verosimiglianza del modello con il numero di componenti introdotte (AIC e BIC tra i più comuni). Nel caso applicativo preso in esame in questo lavoro, una valida tecnica peraltro già utilizzata è quella di andare a selezionare il K ottimale come quello che consente di raggiungere la migliore qualità previsiva, oppure nell'andare ad esaminare le proporzioni stimate per $\boldsymbol{\pi}$ ed individuare se per $K > K^*$ ad alcune componenti viene assegnato un peso pressochè nullo: in un certo senso K risulta essere interpretabile come un parametro di regolazione del modello, da scegliere opportunamente per evitare che una mistura con *troppe* componenti vada a sovradattarsi all'insieme di stima, per il quale rimangono valide tutte le tecniche di selezione note nei contesti di *data mining*.

Tutti questi casi, validi ed utilizzabili, si scontrano però con una questione di praticità e di dispensio di risorse computazionali non indifferente, dovuta alla necessità di stimare più modelli al variare del numero K di componenti: idealmente si vorrebbe individuare un metodo di stima che èsuli dalla selezione di K per poter essere implementato, e che restituisca il numero di componenti ottimale. Nelle prossime righe vengono riportati due possibili approcci che vanno in questa direzione: questi possono essere concepiti come ulteriori sviluppi della classe di modelli proposta, ma andando ben oltre gli scopi di questo lavoro ci si limita a dare un'intuizione sul loro funzionamento.

Una prima tipologia fa riferimento alla classe di algoritmi *Reversible Jump MCMC* Green (1995) che consentono di simulare in modo efficiente dalla distribuzione a posteriori definita su spazi a dimensione variabile: questa metodologia di stima, della quale si da solo un accenno in queste righe, per quanto computazionalmente intensiva e dall'implementazione non particolarmente immediata consente congiuntamente di simulare dalla distribuzione a posteriori della mistura e di *saltare* da un modello all'altro sulla base del numero K di componenti che lo compongono. Un'applicazione di questa classe di algoritmi per la stima di una mistura di distribuzione gaussiane è fornita in Papastamoulis e Iliopoulos (2009).

Una seconda valida e più interessante alternativa riguarda invece la possibilità di considerare misture a numero di componenti infinitamente numerabile, meglio note in letteratura col nome di modelli *DPMM (Dirichlet Process Mixture Model)*, che si pongono in un ambito

inferenziale bayesiano nonparametrico. Una trattazione di questa classe di modelli va ben oltre gli scopi di questo lavoro; ci si limita pertanto a ricordarne l'esistenza includendola tra le possibili estensioni della classe di modelli proposta, nonchè la possibilità di simulare in modo efficiente dalla relativa distribuzione a posteriori implementanto gli algoritmi di Collapsed Gibbs Sampling proposti da Neal (2000).

Conclusioni

Il lavoro svolto in questa tesi ed i risultati ottenuti suggeriscono la possibilità di esaminare e studiare possibili estensioni della classe di modelli proposti, tra quelle illustrate brevemente nel Capitolo 5 ma non solo. Parte dei commenti sui risultati ottenuti sono stati già fatti nelle opportune sezioni: di seguito vengono invece affrontati con spirito critico alcuni degli evidenti limiti della classe dei modelli proposti e delle scelte operate in questo lavoro.

Il primo grande limite sul quale potrebbe scontrarsi la classe di modelli proposti è di natura computazionale: nonostante la stima possa avvenire per qualunque dimensione campionaria e numero di regressori, l'utilizzo applicativo su dataset dalle dimensioni medio-grandi richiede tempistiche di calcolo non indifferenti e che, indipendentemente dal risultato ottenuto, ne potrebbero scoraggiare l'utilizzo.

Un secondo grande limite riguarda l'applicabilità di questa classe di modelli: per come sono stati formalizzati, richiedono che il dataset sul quale si intende utilizzarli esibisca delle possibili sottopopolazioni latenti dalla forma il più possibile riconducibile a quella della distribuzione scelta per le componenti della mistura; anche a patto di saper catturare correttamente la presenza di cluster latenti con una forma precisa, non è comunque detto che all'interno di ciascuno di questi le modalità della variabile risposta siano più facilmente separabili rispetto che nell'andare a considerare il dataset nel suo insieme, evitando di ricorrere all'utilizzo di espansioni kernel dello spazio dei regressori dalle dimensioni complicate. In particolare, la scelta di imporre una precisa forma distributiva alla distribuzione mistura, per quanto motivata da necessità pratiche, risulta essere parecchio stringente ogniqualvolta si intende applicare questa classe di modelli a dataset in cui i dati esibiscono forme non-gaussiane, ovvero la maggior parte: una necessaria estensione riguarda senza ombra di dubbio la possibilità di considerare forme non parametriche sulla distribuzione mistura

che si intende utilizzare per riconoscere la presenza di eventuali sottopopolazioni latenti, congiuntamente a tecniche che ne consentano la stima in un contesto bayesiano.

Una possibile critica che può essere mossa verso la classe di modelli proposta riguarda la scelta di svolgere congiuntamente la stima della distribuzione mistura e delle SVM su ciascuna componente identificata; a livello intuitivo avrebbe infatti più senso che il modello potesse operare in due passi distinti: il primo, quello di *clustering*, nel quale si va ad attribuire a ciascuna osservazione l'appartenenza ad una delle K componenti della mistura; il secondo, da eseguire solo quando è stato raggiunto un certo livello di stabilità nel primo, nel quale si procede alla stima di una SVM per ciascuno dei cluster identificati. La stima congiunta è motivata in questo lavoro da due precise ragioni, ugualmente criticabili e meritevoli di ulteriori riflessioni: la prima riguarda l'impossibilità di svolgere una procedura di stima bayesiana in due passi sequenziali; la seconda riguarda la difficoltà nel trovare un metodo automatico che individui quando la catena ha raggiunto un certo livello di stabilità nella convergenza alla distribuzione a posteriori per il modello mistura. La stima congiunta permette inoltre di gestire il fenomeno di *label switching* già accennato nel Capitolo 2: se questo si manifesta in fase di stima della componente mistura, l'algoritmo proposto adegua la stima delle SVM di conseguenza.

Un'ultima considerazione riguarda l'opportunità di individuare un insieme di metodologie che consentano l'inferenza su classi di modelli originariamente sviluppate per problemi di regressione o classificazione, in cui la stima dei parametri avviene generalmente con algoritmi iterativi o di ottimizzazione numerica che prescindono da assunzioni probabilistiche sul modello generatore dei dati: in particolare, andare a *forzare* la possibilità di ricavare una pseudo-verosimiglianza per il modello che consente di riportare le tecniche di stima all'interno dei confini classici dell'approccio inferenziale bayesiano (o frequentista) potrebbe compromettere la validità stessa dell'inferenza; indipendentemente dai risultati ottenuti, sarà sempre bene tenere a mente che ogni possibile conclusione inferenziale proviene dalla scelta di *forzare* la presenza di un preciso e ben definito modello probabilistico, derivante da un tecnicismo operato sulla funzione di perdita del modello e non da considerazioni di natura prettamente statistica, sull'ignoto meccanismo generatore dei dati.

Appendice A

Codice degli algoritmi sviluppati

In questa Appendice vengono riportati i codici relativi agli algoritmi di Gibbs Sampling per la simulazione dalle distribuzioni a posteriori dei modelli formalizzati in questo lavoro. Il linguaggio utilizzato è C++, integrato dalla libreria *Armadillo* che consente tra le altre una più facile gestione di *array* multidimensionali e di operazioni di algebra lineare: questo permette di ottenere un vantaggio numerico non indifferente nella velocità di esecuzione degli algoritmi, ed in particolare dei cicli inclusi nelle funzioni rispetto al linguaggio R, strettamente necessario per svolgere le stime su dataset con dimensioni non banali in tempi ragionevoli. In ambiente R sono comunque state rilasciate le librerie `Rcpp` e l'integrazione `RcppArmadillo` che consentono di importare una qualunque funzione scritta in linguaggio C++ tramite il comando `sourceRcpp()`, *mascherando* il fatto che internamente l'oggetto `function` in R associato richiama la funzione scritta in C++, esegue le istruzioni in ambiente C++ e restituisce i risultati in oggetti direttamente interpretabili da R. Per maggiori informazioni tecniche sulle librerie utilizzate e sui vantaggi computazionali ottenibili dalla programmazione in *Rcpp/RcppArmadillo* si fa riferimento a Eddelbuettel (2013); una buona fonte documentativa riguardante la libreria *armadillo* è disponibile all'indirizzo `http://arma.sourceforge.net/docs.html`, mentre i primi passi su *Rcpp* sono stati svolti tenendo come riferimento l'ottima guida introduttiva disponibile all'indirizzo `https://teuder.github.io/rcpp4everyone_en/`.

Codice A.1: Funzioni di supporto per routine non implementate in Rcpp/Armadillo

```
// funzione d'appoggio per calcolare tabella di frequenza
ivec arma_table(ivec z, const int K){
  ivec out(K, fill::zeros);
  for(int i = 0; i < z.n_rows; ++i) out(z(i)-1) = out(z(i)-1) + 1;
  return out;
}

// funzione di appoggio che velocizza il calcolo della distanza di mahalanobis
vec mahalanobis_dist(mat x, rowvec center, mat cov){
  mat x_cen;
  x_cen.copy_size(x);
  for (int i=0; i < x.n_rows; i++) x_cen.row(i) = x.row(i) - center;
  return sum((x_cen * cov.i()) % x_cen, 1);
}

// funzione d'appoggio per calcolare densita' della normale multivariata
double ldmvnnorm(vec xstar, vec mu, mat Sigma){
  double out = 0;
  out = -0.5 * xstar.n_elem * log(2 * datum::pi) - 0.5 * log(det(Sigma)) - 0.5 * as_scalar(
      mahalanobis_dist(xstar.t(), mu.t(), Sigma));
  return out;
}

// funzione d'appoggio per generare da una Wishart
mat rwishart(int df, const mat S){
  mat Z(S.n_rows,S.n_rows);
  for(int i = 0; i < S.n_rows; i++) Z(i,i) = sqrt(R::rchisq(df-i));

  for(int j = 0; j < S.n_rows; j++){
    for(int i = j+1; i < S.n_rows; i++) Z(i,j) = R::rnorm(0,1);
  }
  mat C = trimatl(Z).t() * chol(S);

  return C.t()*C;
}

// funzione d'appoggio per generare da una gaussiana multivariata
vec rmvnnorm(vec mu, mat sigma){
```

```
  int p = sigma.n_cols;
  return chol(sigma, "lower") * randn(p) + mu;
}

// funzione di appoggio che calcola la densita' di una t-student multiv:
double ldmvnt(vec xstar, vec mu, mat Sigma, double df){
  const double p = xstar.n_elem;
  double out = 0;
  out = lgamma(0.5 * (df + p)) - lgamma(0.5 * df) - 0.5 * p * log(df * datum::pi) - 0.5 * log(
      det(Sigma)) -(df + p)/2 * log((1 + as_scalar(mahalanobis_dist(xstar.t(), mu.t(), Sigma))/
      df));
  return out;
}

// funzione di appoggio per normalizzare i log-pesi
vec norm_lweights(vec lweights){
  double maxlweights = max(lweights);
  vec out(lweights.n_elem);
  for(int j = 0; j < lweights.n_elem; j++) out(j) = exp(lweights(j) - maxlweights);
  out = out/sum(out);
  return out;
}
```

Codice A.2: Collapsed Gibbs Sampler per una mistura finita di gaussiane univariate

```
#include <RcppArmadilloExtensions/sample.h>
// [[Rcpp::depends(RcppArmadillo)]]
using namespace Rcpp;
using namespace arma;
using namespace std;

// [[Rcpp::export]]
List gibbs_mixture_univargauss(int R, vec mu0, vec s0, ivec z0, vec xoss){

  ////////////////////////////////////////////
  /// Collapsed Gibbs Sampler per mistura a K ///
  /// componenti gaussiane univariate.       ///
  ////////////////////////////////////////////

  // iperparametri (priori diffuse)
```

```
const double mu_0 = 0;
const double s_0 = 0.01;
const double alpha_0 = 0.01;
const double lambda_0 = 0.01;
const double alpha0 = 1;

const int K = mu0.size();
const int n = xoss.size();

// vettore con sequenza delle classi
vec sequence(K);
for(int l = 0; l < K; ++l) sequence(l) = l + 1;

// vettore delle probabilita'
vec prob(K, fill::zeros);

// matrici per le catene ed inizializzazione
mat mu(K, R, fill::zeros);
mat s(K, R, fill::zeros);
imat z(n, R, fill::zeros);

mu.col(0) = mu0;
s.col(0) = s0;
z.col(0) = z0;

for(int i = 1; i < R; ++i){ // iterations
  // frequenza di ciascuna classe in z_(i-1)
  ivec z_last = z.col(i-1);
  ivec nfreq = arma_table(z_last, K);

  for(int j = 0; j < K; ++j){
    // allocazione di mu
    double mu_tilde = (s_0 * mu_0 + s(j, i-1) * accu(xoss(find(z_last==(j+1))))/(nfreq(j) * s
        (j, i-1) + s_0);
    double s2_tilde = 1/(nfreq(j) * s(j, i-1) + s_0);
    mu(j, i) = Rf_rnorm(mu_tilde, sqrt(s2_tilde));

    // allocazione di s
    double alpha_tilde = alpha_0 + nfreq(j)/2.0;
```

```
      double lambda_tilde = lambda_0 + accu(pow(xoss(find(z_last==(j+1))) - mu(j, i), 2.0))/2.0;
      s(j, i) = Rf_rgamma(alpha_tilde, 1/lambda_tilde);
    }

    z.col(i) = z.col(i-1);
    // allocazione di z_h
    for(int h = 0; h < n; h++){
      prob.zeros();
      ivec z_rmv_h = z.col(i);
      z_rmv_h.shed_row(h); // elimino osservazione h-esima
      ivec freq_menoh = arma_table(z_rmv_h, K);

      for(int j = 0; j < K; j++){
        prob(j) = log((freq_menoh(j) + alpha0/K)/(n + alpha0 - 1)) + Rf_dnorm4(xoss(h), mu(j, i)
            , 1/sqrt(s(j, i)), true);
      }
      prob = exp(prob);
      z(h, i) = Rcpp::RcppArmadillo::sample(sequence, 1, false, prob)(0);
    }
  }
  return(List::create(Named("mu") = mu, Named("s") = s, Named("z") = z));
}
```

Codice A.3: Collapsed Gibbs Sampler per una mistura finita di gaussiane multivariate

```
#include <RcppArmadilloExtensions/sample.h>
//[[Rcpp::depends(RcppArmadillo)]]
using namespace Rcpp;
using namespace arma;
using namespace std;

// [[Rcpp::export]]
List gibbs_mixture_multivargauss(int R, int K, ivec z0, mat Xoss){

  ////////////////////////////////////////////
  /// Collapsed Gibbs Sampler per mistura a K ///
  /// componenti gaussiane multivariate.      ///
  ////////////////////////////////////////////

  const int n = Xoss.n_rows;
```

```
const int p = Xoss.n_cols;

// iperparametri (priori diffuse)
const vec m_0(p, fill::zeros);
const mat S_0(p, p, fill::eye);
const double ka_0 = 1.0;
const double nu_0 = p + 1.0;
const double alpha0 = 1.0;

// vettore con sequenza delle classi
vec sequence(K);
for(int l = 0; l < K; ++l) sequence(l) = l + 1;

// vettore delle probabilita'
vec probs(K, fill::zeros);

// matrici per le catene ed inizializzazione
cube Mu(p, K, R, fill::zeros);
imat z(n, R, fill::zeros);
// per Sigma non salvo l'intera catena delle matrici generate
// ma solo le ultime K
cube Sigma(p, p, K, fill::zeros);

z.col(0) = z0;

for(int i = 1; i < R; ++i){ // iterations
  // frequenza di ciascuna classe in z_(i-1)
  ivec z_prev = z.col(i-1);
  ivec nfreq = arma_table(z_prev, K);

  for(int j = 0; j < K; ++j){
    double ka_n = ka_0 + nfreq(j);
    double nu_n = nu_0 + nfreq(j);
    vec m_n = (ka_0 * m_0 + sum(Xoss.rows(find(z_prev == j + 1)), 0).t())/ka_n;
    mat S_n = S_0 + Xoss.rows(find(z_prev == j + 1)).t() * Xoss.rows(find(z_prev == j + 1)) +
        ka_0 * m_0 * m_0.t() - ka_n * m_n * m_n.t();

    // Simulazione di Sigma_j da Wishart inversa
    Sigma.slice(j) = inv(rwishart(nu_n, inv(S_n)));
```

```
      // Simulazione di Mu_j da gaussiana multivariata
      Mu.slice(i).col(j) = rmvnnorm(m_n, 1/ka_n * Sigma.slice(j));
    }

    z.col(i) = z_prev;
    for(int h = 0; h < n; ++h){
      probs.zeros();
      ivec z_rmv_h = z.col(i);
      z_rmv_h.shed_row(h); // elimino osservazione h-esima
      mat Xoss_rmv_h = Xoss;
      Xoss_rmv_h.shed_row(h); // elimino osservazione h-esima
      // vettore delle frequenze di ciascuna classe su z_(-h)
      ivec freq_menoh = arma_table(z_rmv_h, K);

      // vettore delle probabilita' per ciascuna classe
      for(int j = 0; j < K; j++){
        probs(j) = log((freq_menoh(j) + alpha0/K)/(n + alpha0 - 1.0)) + ldmvnorm(Xoss.row(h).t
            (), Mu.slice(i).col(j), Sigma.slice(j));
      }
      probs = exp(probs);
      z(h, i) = Rcpp::RcppArmadillo::sample(sequence, 1, true, probs)(0);
    }

  }
  return(List::create(Named("Mu") = Mu, Named("S") = Sigma, Named("z") = z));
}
```

Codice A.4: Collapsed Gibbs Sampler per una mistura finita di gaussiane multivariate su SVM per problemi di classificazione

```
#include <RcppArmadillo.h>
#include <RcppArmadilloExtensions/sample.h>
#include <Rmath.h>
//[[Rcpp::depends(RcppArmadillo)]]
using namespace Rcpp;
using namespace arma;
using namespace std;

// [[Rcpp::export]]
```

```
List gibbs_svm_mixture_multivgauss_classification(const int R, mat beta0, vec lambda_inv0, mat
    omega_inv0, vec nu_inv0, ivec z0, const double alpha, mat X, vec y, const int nvars) {

  ////////////////////////////////////////////////
  /// Collapsed Gibbs Sampler per mistura a K   ///
  /// componenti gaussiane multivariate di SVM  ///
  /// per classificazione penalizzata.          ///
  ////////////////////////////////////////////////

  const int K = beta0.n_cols;
  const int p = beta0.n_rows;
  const int n = y.n_elem;

  // iperparametri (priori diffuse)
  const double a_nu = 0.1;
  const double b_nu = 0.1;
  const double alpha0 = 1.0;
  const double ka_0 = 1.0;
  const double v_0 = nvars + 1.0 + 0.001;
  const vec m_0(nvars, fill::zeros);
  const mat S_0(nvars, nvars, fill::eye);

  // X_tilde
  mat Xtilde = diagmat(y) * X;
  // Sigma
  mat diagSigma = var(X);
  diagSigma(0) = 1.0;
  mat Sigma = diagmat(diagSigma);
  mat invSigma = inv(Sigma); // Sigma inversa

  // sequenza dei cluster della mistura
  ivec sequence(K);
  for(int l = 0; l < K; ++l) sequence(l) = l + 1.0;
  // vettore delle probabilita'
  vec probs(K, fill::zeros);
  vec logprobs(K, fill::zeros);

  // tensori e matrici dei risultati ed inizializzazione
  cube beta(p, K, R, fill::zeros);
```

```
mat lambda_inv(n, R, fill::zeros);
cube omega_inv(p, K, R, fill::zeros);
mat nu_inv(K, R, fill::zeros);
imat z(n, R, fill::zeros);

beta.slice(0) = beta0;
lambda_inv.col(0) = lambda_inv0;
omega_inv.slice(0) = omega_inv0;
nu_inv.col(0) = nu_inv0;
z.col(0) = z0;

for(int i = 1; i < R; ++i){
  // vettori di supporto
  mat beta_prev = beta.slice(i-1);
  vec lambda_inv_prev = lambda_inv.col(i-1);
  mat omega_inv_prev = omega_inv.slice(i-1);
  vec nu_inv_prev = nu_inv.col(i-1);
  ////////////////////// allocazione di z_i ////////////////////////
  z.col(i) = z.col(i-1);
  for(int h = 0; h < n; ++h){
    probs.zeros();
    logprobs.zeros();
    ivec z_rmv_h = z.col(i);
    z_rmv_h.shed_row(h); // elimino osservazione h-esima
    mat X_rmv_h = X;
    X_rmv_h = X_rmv_h.cols(1, nvars); // elimino intercetta + kernel per clusterizzare solo
        sui dati originali
    X_rmv_h.shed_row(h); // elimino osservazione h-esima
    // vettore delle frequenze di ciascuna classe su z_(-h)
    ivec freq_menoh = arma_table(z_rmv_h, K);

    // vettore delle log-probabilita' per ciascuna classe
    for(int j = 0; j < K; j++){
      double ka_n = ka_0 + freq_menoh(j);
      double v_n = v_0 + freq_menoh(j);
      vec m_n = (ka_0 * m_0 + sum(X_rmv_h.rows(find(z_rmv_h == j + 1)), 0).t())/ka_n;
      mat S_n = S_0 + X_rmv_h.rows(find(z_rmv_h == j + 1)).t() * X_rmv_h.rows(find(z_rmv_h ==
          j + 1)) + ka_0 * m_0 * m_0.t() - ka_n * m_n * m_n.t();
      mat Sigma_t = (ka_n + 1.0)/(ka_n * (v_n - nvars + 1.0)) * S_n;
```

```
    double df_t = v_n - nvars + 1.0;
    logprobs(j) = log((freq_menoh(j) + alpha0/K)/(n + alpha0 - 1.0)) + ldmvnt(X.row(h).cols
        (1, nvars).t(), m_n, Sigma_t, df_t);
  }

  probs = norm_lweights(logprobs);
  z(h, i) = Rcpp::RcppArmadillo::sample(sequence, 1, true, probs)(0);
}
ivec z_i = z.col(i);
ivec nfreq = arma_table(z_i, K);
////////////////////// allocazione di lambda_inv_i ////////////////////////
for(int h = 0; h < n; ++h){
  double mu_lambda_inv = 1/abs(1 - dot(Xtilde.row(h), beta_prev.col(z_i(h) - 1))); // mu
  lambda_inv(h, i) = rinvgauss(1, mu_lambda_inv, 1)(0);
}
vec lambda_inv_i = lambda_inv.col(i);
////////////////////// allocazione dei beta_j ////////////////////////
for(int j = 0; j < K; ++j){
  mat invLambda_j = diagmat(lambda_inv_i(find(z_i == (j + 1)))); // Lambda_j inversa
  mat invOmega_j = diagmat(omega_inv_prev.col(j)); // Omega_j inversa
  // B matrice di varianza/covarianza
  mat Binv(p, p, fill::zeros);
  Binv = pow(as_scalar(nu_inv_prev(j)), 2.0) * invSigma * invOmega_j + Xtilde.rows(find(z_i
      == (j + 1))).t() * invLambda_j * Xtilde.rows(find(z_i == (j + 1)));
  mat B = inv(Binv);
  // b vettore delle medie
  vec b(p, fill::zeros);
  vec one = ones(nfreq(j));
  b = B * Xtilde.rows(find(z_i == (j + 1))).t() * (one + lambda_inv_i(find(z_i == (j + 1))))
      ;
  beta.slice(i).col(j) = rmvnnorm(b, B);
}
mat beta_i = beta.slice(i);
////////////////////// allocazione di omega_inv_j ////////////////////////
if(alpha == 2){ // penalizzazione ridge: omega degenere in 1
  omega_inv.slice(i).fill(1);
} else {
  for(int j = 0; j < K; ++j){
    for(int h = 0; h < p; ++h){
```

```
          double mu_omega_inv = as_scalar(1/nu_inv_prev(j)) * as_scalar(sqrt(Sigma(h, h))/abs(
              beta_i(h, j))); // mu
            omega_inv(h, j, i) = rinvgauss(1, mu_omega_inv, 1)(0);
        }
      }
    }
    ////////////////////// allocazione di nu_inv_j ////////////////////
    for(int j = 0; j < K; ++j){
      double rate_nuinv = b_nu + accu(abs(beta_i.col(j))/sqrt(Sigma.diag())); // rate
      nu_inv(j, i) = Rf_rgamma(a_nu + p, 1/rate_nuinv);
    }

  }
  if(alpha == 2){
    return(List::create(Named("beta") = beta, Named("lambda_inv") = lambda_inv, Named("nu_inv")
        = pow(nu_inv, alpha), Named("z") = z);
  } else {
    return(List::create(Named("beta") = beta, Named("lambda_inv") = lambda_inv, Named("omega_inv
        ") = omega_inv, Named("nu_inv") = nu_inv, Named("z") = z);
  }
}
```

Codice A.5: Funzione per il calcolo della distribuzione di matrici di confusione

```
conftable_bayesian <- function(X.test.std, y.test, X.train.std, beta, burnin, nvars, z=rep(1,
    nrow(X.train.std))){
  ## X.test.std matrice n x p
  ## y.test vettore n x 1
  ## beta array p x K x R-burnin
  ## z matrice n x R-burnin
  p <- dim(beta)[1]
  K <- dim(beta)[2]
  n_iter <- dim(beta)[3]
  n <- length(y.test)
  conftab_array <- array(0, dim = c(2, 2, n_iter - burnin))

  m_0 <- rep(0, nvars)
  S_0 <- diag(1, nvars)
  ka_0 <- 1
  nu_0 <- nvars + 1 + 0.001
```

```
if(dim(beta)[2] == 1){
  ## caso K = 1: SVM semplice
  for(i in 1:(n_iter - burnin)){
    y_pred <- rep(-1, n)
    y_pred[X.test.std %*% beta[, K, i + burnin] > 0] <- 1
    conftab_array[,, i] <- table(factor(y.test, levels = c(-1, 1)), factor(y_pred, levels = c
        (-1, 1)))
  }
} else {
  ## caso K > 1: SVM Mixture
  X.train.std.vars <- X.train.std[, 2:(nvars + 1)]
  X.test.std.vars <- X.test.std[, 2:(nvars + 1)]
  for(i in 1:(n_iter - burnin)){
    if(i %% 100 == 0) print(round(i/(n_iter - burnin), 2))
    clust_iter <- rep(NA, n)

    ka_n <- rep(NA, K)
    nu_n <- rep(NA, K)
    m_n <- matrix(NA, nrow=nvars, ncol=K)
    S_n <- list(NA)

    for(k in 1:K){
      ka_n[k] <- ka_0 + sum(z[, i + burnin] == k)
      nu_n[k] <- nu_0 + sum(z[, i + burnin] == k)
      m_n[,k] <- (ka_0 * m_0 + apply(X.train.std.vars[z[, i + burnin] == k, ], 2, sum))/ka_n[k
          ]
      S_n[[k]] <- as.matrix(S_0 + t(X.train.std.vars[z[, i + burnin] == k, ]) %*% X.train.std.
          vars[z[, i + burnin] == k, ] +ka_0 * m_0 %*% t(m_0) - ka_n[k] * m_n[,k] %*% t(m_n[,
          k]))
    }

    for(h in 1:n){
      log_pred_post_h <- rep(NA, K)
      for(k in 1:K){
        log_pred_post_h[k] <- dmvt(X.test.std.vars[h, ], m_n[, k], (ka_n[k] + 1)/(ka_n[k] * (
            nu_n[k] - nvars + 1)) * S_n[[k]],nu_n[k] - nvars + 1, log = T) + log(sum(z[, i +
            burnin] == k)/n)
      }
```

```
        clust_iter[h] <- which.max(log_pred_post_h)
      }

      for(k in 1:K){
        conftab_array[,, i] <- conftab_array[,, i] + table(factor(y.test[clust_iter == k],
            levels=c(-1, 1)), factor(sign(X.test.std[clust_iter == k,] %*% beta[, k, i + burnin
            ]), levels=c(-1, 1)))
      }
    }
  }

  conftab_out <- matrix(NA, ncol=2, nrow=2)
  rownames(conftab_out) <- c("y=-1", "y=1")
  colnames(conftab_out) <- c("pred=-1", "pred=1")
  for(h in 1:2){
    for(k in 1:2){
      conftab_out[h, k] <- mean(conftab_array[h, k, ])
    }
  }
  return(list(conftab_out=conftab_out, conftab_array=conftab_array))
}
```

Codice A.6: Collapsed Gibbs Sampler per una mistura finita di gaussiane multivariate su SVM per problemi di regressione

```
#include <RcppArmadillo.h>
#include <RcppArmadilloExtensions/sample.h>
#include <Rmath.h>
//[[Rcpp::depends(RcppArmadillo)]]
using namespace Rcpp;
using namespace arma;
using namespace std;

// [[Rcpp::export]]
List gibbs_svm_mixture_multivgauss_regression(const int R, mat beta0, vec lambdaneg_inv0, vec
    lambdapos_inv0, mat omega_inv0, vec nu_inv0, ivec z0, const double alpha, double eps, mat X
    , vec y, const int nvars) {

  ////////////////////////////////////////////////
  /// Collapsed Gibbs Sampler per mistura a K   ///
```

```
/// componenti gaussiane multivariate di SVM  ///
/// per regressione penalizzata.              ///
////////////////////////////////////////////////

const int K = beta0.n_cols;
const int p = beta0.n_rows;
const int n = y.n_elem;

// iperparametri (priori diffuse)
const double a_nu = 0.1;
const double b_nu = 0.1;
const double alpha0 = 1.0;

const double ka_0 = 1.0;
const double v_0 = nvars + 1.0 + 0.001;
const vec m_0(nvars, fill::zeros);
const mat S_0(nvars, nvars, fill::eye);

// X_tilde
mat Xtilde = diagmat(y) * X;
// Sigma
mat diagSigma = var(X);
diagSigma(0) = 1.0;
mat Sigma = diagmat(diagSigma);
mat invSigma = inv(Sigma); // Sigma inversa

// sequenza dei cluster della mistura
ivec sequence(K);
for(int l = 0; l < K; ++l) sequence(l) = l + 1.0;
// vettore delle probabilita'
vec probs(K, fill::zeros);
vec logprobs(K, fill::zeros);

// tensori e matrici dei risultati ed inizializzazione
cube beta(p, K, R, fill::zeros);
mat lambdaneg_inv(n, R, fill::zeros);
mat lambdapos_inv(n, R, fill::zeros);
cube omega_inv(p, K, R, fill::zeros);
mat nu_inv(K, R, fill::zeros);
```

```
imat z(n, R, fill::zeros);

beta.slice(0) = beta0;
lambdaneg_inv.col(0) = lambdaneg_inv0;
lambdapos_inv.col(0) = lambdapos_inv0;
omega_inv.slice(0) = omega_inv0;
nu_inv.col(0) = nu_inv0;
z.col(0) = z0;

for(int i = 1; i < R; ++i){
  // vettori di supporto
  mat beta_prev = beta.slice(i-1);
  vec lambdaneg_inv_prev = lambdaneg_inv.col(i-1);
  vec lambdapos_inv_prev = lambdapos_inv.col(i-1);
  mat omega_inv_prev = omega_inv.slice(i-1);
  vec nu_inv_prev = nu_inv.col(i-1);
  ////////////////////// allocazione di z_i ////////////////////////
  z.col(i) = z.col(i-1);
  for(int h = 0; h < n; ++h){
    probs.zeros();
    logprobs.zeros();
    ivec z_rmv_h = z.col(i);
    z_rmv_h.shed_row(h); // elimino osservazione h-esima
    mat X_rmv_h = X;
    X_rmv_h = X_rmv_h.cols(1, nvars); // elimino intercetta + kernel per clusterizzare solo
        sui dati originali
    X_rmv_h.shed_row(h); // elimino osservazione h-esima
    // vettore delle frequenze di ciascuna classe su z_(-h)
    ivec freq_menoh = arma_table(z_rmv_h, K);
    // vettore delle probabilita' per ciascuna classe
    for(int j = 0; j < K; j++){
      double ka_n = ka_0 + freq_menoh(j);
      double v_n = v_0 + freq_menoh(j);
      vec m_n = (ka_0 * m_0 + sum(X_rmv_h.rows(find(z_rmv_h == j + 1)), 0).t())/ka_n;
      mat S_n = S_0 + X_rmv_h.rows(find(z_rmv_h == j + 1)).t() * X_rmv_h.rows(find(z_rmv_h ==
          j + 1)) + ka_0 * m_0 * m_0.t() - ka_n * m_n * m_n.t();
      mat Sigma_t = (ka_n + 1.0)/(ka_n * (v_n - nvars + 1.0)) * S_n;
      double df_t = v_n - nvars + 1.0;
      logprobs(j) = log((freq_menoh(j) + alpha0/K)/(n + alpha0 - 1.0)) + ldmvnt(X.row(h).cols
```

```
        (1, nvars).t(), m_n, Sigma_t, df_t);
  }
  probs = norm_lweights(logprobs);
  z(h, i) = Rcpp::RcppArmadillo::sample(sequence, 1, true, probs)(0);
}
ivec z_i = z.col(i);
ivec nfreq = arma_table(z_i, K);
////////////////////// allocazione di lambda_inv_i //////////////////////
for(int h = 0; h < n; ++h){
  double mu_lambdaneg_inv = 1/abs(y(h) - dot(X.row(h), beta_prev.col(z_i(h) - 1)) - eps); //
      mu
  lambdaneg_inv(h, i) = rinvgauss(1, mu_lambdaneg_inv, 1)(0);
}
vec lambdaneg_inv_i = lambdaneg_inv.col(i);

for(int h = 0; h < n; ++h){
  double mu_lambdapos_inv = 1/abs(y(h) - dot(X.row(h), beta_prev.col(z_i(h) - 1)) + eps); //
      mu
  lambdapos_inv(h, i) = rinvgauss(1, mu_lambdapos_inv, 1)(0);
}
vec lambdapos_inv_i = lambdapos_inv.col(i);
////////////////////// allocazione dei beta_j //////////////////////
for(int j = 0; j < K; ++j){
  mat invLambdaneg_j = diagmat(lambdaneg_inv_i(find(z_i == (j + 1)))); // Lambdaneg_j
      inversa
  mat invLambdapos_j = diagmat(lambdapos_inv_i(find(z_i == (j + 1)))); // Lambdapos_j
      inversa
  mat Lambdatilde_j = invLambdaneg_j + invLambdapos_j; // Lambdatilde
  mat invOmega_j = diagmat(omega_inv_prev.col(j)); // Omega_j inversa
  // B matrice di varianza/covarianza
  mat Binv(p, p, fill::zeros);
  Binv = pow(as_scalar(nu_inv_prev(j)), 2.0) * invSigma * invOmega_j +  X.rows(find(z_i == (
      j + 1))).t() * Lambdatilde_j * X.rows(find(z_i == (j + 1)));
  mat B = inv(Binv);
  // b vettore delle medie
  vec b(p, fill::zeros);
  vec one = ones(nfreq(j));
  b = B * X.rows(find(z_i == (j + 1))).t() * (invLambdaneg_j * (y(find(z_i == (j + 1))) -
      eps * one) + invLambdapos_j * (y(find(z_i == (j + 1))) + eps * one));
```

```
      beta.slice(i).col(j) = rmvnnorm(b, B);
    }
    mat beta_i = beta.slice(i);
    ////////////////////// allocazione di omega_inv_j ////////////////////
    if(alpha == 2){ // penalizzazione ridge: omega degenere in 1
      omega_inv.slice(i).fill(1);
    } else {
      for(int j = 0; j < K; ++j){
        for(int h = 0; h < p; ++h){
          double mu_omega_inv = as_scalar(1/nu_inv_prev(j)) * as_scalar(sqrt(Sigma(h, h))/abs(
              beta_i(h, j))); // mu
          omega_inv(h, j, i) = rinvgauss(1, mu_omega_inv, 1)(0);
        }
      }
    }
    ////////////////////// allocazione di nu_inv_j ////////////////////
    for(int j = 0; j < K; ++j){
      double rate_nuinv = b_nu + accu(abs(beta_i.col(j))/sqrt(Sigma.diag())); // rate
      nu_inv(j, i) = Rf_rgamma(a_nu + p, 1/rate_nuinv);
    }
  }
  if(alpha == 2){
    return(List::create(Named("beta") = beta, Named("lambdaneg_inv") = lambdaneg_inv, Named("
        lambdapos_inv") = lambdapos_inv, Named("nu_inv") = pow(nu_inv, alpha), Named("z") = z))
        ;
  } else {
    return(List::create(Named("beta") = beta, Named("lambdaneg_inv") = lambdaneg_inv, Named("
        lambdapos_inv") = lambdapos_inv, Named("omega_inv") = omega_inv, Named("nu_inv") =
        nu_inv, Named("z") = z));
  }
}
```

Appendice B

Codice delle simulazioni condotte

Codice B.1: Esempio di applicazione del Gibbs Sampling su SVM bayesiana di classificazione

```
set.seed(12345)
library(penalizedSVM)

dati.train <- sim.data(n=500, ng = 30, nsg = 10, seed = 123)
dati.test <- sim.data(n=250, ng = 30, nsg = 10, seed = 124)

y.train <- dati.train$y
y.test <- dati.test$y
X.train <- as.matrix(model.matrix(~t(dati.train$x)))
X.test <- as.matrix(model.matrix(~t(dati.test$x)))
X.train.std <- cbind(X.train[,1], scale(X.train[,-1]))
X.test.std <- cbind(X.test[,1], scale(X.test[,-1]))

R <- 10^4
## valori di partenza della catena
beta0 <- rep(1, 31)
lambda_inv0 <- rep(1, nrow(X.train.std))
omega_inv0 <- rep(1, ncol(X.train.std))
nu_inv0 <- 1

sim <- gibbs_svm_mixture_multivgauss_classification(R, beta0, lambda_inv0, omega_inv0, nu_inv0,
    z0, 1, X.train.std, y.train, ncol(X.train))
betapost <- apply(sim$beta[,-(1:(R/2))], 1, median)
```

```
round(betapost, 5)
table(sign(X.train.std%*%betapost), y.train)
table(sign(X.test.std%*%betapost), y.test)

## grafici degli intervalli hpd sui beta
ludata <- data.frame(lowIC=rep(NA, 31), uppIC=rep(NA, 31), col=factor(c(1, rep(2, 10), rep(1,
    20))), beta=0:30)
library(TeachingDemos)
for(i in 1:nrow(ludata)) ludata[i, 1:2] <- emp.hpd(sim$beta[i, -(1:(R/2))], conf = 0.95)
library(ggplot2)
ggplot(ludata, aes(x = lowIC, xend = uppIC, y = beta, yend = beta)) +
geom_segment(aes(color = col, position="stack"), size = 0.75) +
scale_color_manual(name="Gruppo", labels=c("Non discriminanti", "Discriminanti"), values=c("red"
    , "violet")) +
scale_y_discrete(limits = 0:30, labels=sprintf("beta_%s", 0:30)) +
labs(y="Coefficienti", x="Intervalli HPD 95%")
```

Codice B.2: Studio di simulazione su una mistura di gaussiane univariate

```
set.seed(12345)
R <- 10^3
K <- 2
n <- 5*10^2
mu0 <- c(0, 0)
s0 <- c(1, 1)
z0 <- sample(1:K, size = n, replace = T)
## simulazione delle xoss:
ind <- sample(1:K, size = n, replace = T, prob = c(0.4, 0.6))
xoss <- rnorm(n, mean = c(-4, 2)[ind], sd = c(sqrt(2), sqrt(4))[ind])
Rcpp::sourceCpp("gibbs_mixture_univargauss.cpp") ## richiamo la funzione per fare gibbs sampler
sim <- gibbs_mixture_univargauss(R, mu0, s0, z0, xoss)

## Simulazione su 4 catene
library(plyr)
library(ggmcmc)
library(coda)
nsim <- 4
res_sim <- list()
mu0_matrix <- cbind(c(0, 0), c(-10, 10), c(5, 5), c(-5, -10))
s0_matrix <- cbind(c(1, 1), c(0.5, 0.5), c(3, 3), c(0.5, 3))
```

```
for(i in 1:nsim){
sample(1:K, size = n, replace = T)
res_sim[[i]] <- gibbs_mixture_univargauss(R, mu0_matrix[,i], s0_matrix[,i], z0, xoss)
}
```

Codice B.3: Studio di simulazione su una mistura di gaussiane multivariate

```
library(mvtnorm)
set.seed(12345)
R <- 10^3
K <- 3
n <- 2*10^3

## simulazione di Xoss
pi <- c(0.3, 0.3, 0.4)
id_oss <- sample(1:K, n, replace = T, prob = pi)

mu_1 <- c(2, 2)
mu_2 <- c(8, 8)
mu_3 <- c(5, 5)
Sigma_1 <- matrix(c(3, 1.5, 1.5, 2), byrow=T, nrow=2, ncol=2)
Sigma_2 <- matrix(c(3, 1.5, 1.5, 2), byrow=T, nrow=2, ncol=2)
Sigma_3 <- matrix(c(4, -2.5, -2.5, 2), byrow=T, nrow=2, ncol=2)

Xoss <- matrix(NA, nrow=n, ncol=2)
Xoss[id_oss == 1, ] <- rmvnorm(sum(id_oss == 1), mu_1, Sigma_1)
Xoss[id_oss == 2, ] <- rmvnorm(sum(id_oss == 2), mu_2, Sigma_2)
Xoss[id_oss == 3, ] <- rmvnorm(sum(id_oss == 3), mu_3, Sigma_3)

## stima
Rcpp::sourceCpp("gibbs_mixture_multivargauss.cpp") ## richiamo la funzione per fare gibbs
    sampler

z0 <- sample(1:K, n, replace = T)
sim <- gibbs_mixture_multivargauss(R, K, z0, Xoss)
```

Codice B.4: Codice per la creazione dei dataset simulati utilizzati nella Sezione 3.4

```
library(mvtnorm)
set.seed(12345)
```

```
# Esempio 1 ------------------------------------------------------------

n <- 2*10^3
data <- matrix(0, nrow = n, ncol = 4)
data <- data.frame(data)
names(data) <- c("y", "x1", "x2", "cluster")
data$cluster[1:(n/2)] <- 1
data$cluster[-(1:(n/2))] <- 2
data$y <- -1
data[1:(n/2), 2:3] <- rmvnorm(n = n/2, mean = c(4, 2), sigma = matrix(c(3,1.5,1.5,2), 2))
data[-(1:(n/2)),2:3] <- rmvnorm(n = n/2, mean = c(9, 2), sigma = matrix(c(3,-1.5,-1.5,2), 2))
model1 <- lm(x2~x1, data=data[1:(n/2),])
data$y[1:(n/2)][which(data$x2[1:(n/2)] > model1$coefficients[1] + model1$coefficients[2] * data$
    x1[1:(n/2)])] <- 1
model2 <- lm(x2~x1,data=data[-(1:(n/2)),])
data$y[-(1:(n/2))][which(data$x2[-(1:(n/2))] > model2$coefficients[1] + model2$coefficients[2] *
     data$x1[-(1:(n/2))])] <- 1

datasim_v1 <- data[,-4]
save(datasim_v1, file = "2cluster_classification_simulated_v1.Rdata")

# Esempio 2 ------------------------------------------------------------

n <- 2*10^3
data2 <- matrix(0, nrow = n, ncol = 4)
data2 <- data.frame(data)
names(data2) <- c("y", "x1", "x2", "cluster")
data2$cluster[1:(n/2)] <- 1
data2$cluster[-(1:(n/2))] <- 2
data2$y <- -1
data2[1:(n/2), 2:3] <- rmvnorm(n = n/2, mean = c(4, 2), sigma = matrix(c(3,1.5,1.5,2), 2))
data2[-(1:(n/2)),2:3] <- rmvnorm(n = n/2, mean = c(9, 2), sigma = matrix(c(3,-1.5,-1.5,2), 2))
model1 <- lm(x2~x1, data=data2[1:(n/2),])
data2$y[1:(n/2)][which(data2$x2[1:(n/2)] > model1$coefficients[1] + model1$coefficients[2] *
    data2$x1[1:(n/2)])] <- 1
model2 <- lm(x2~x1,data=data2[-(1:(n/2)),])
data2$y[-(1:(n/2))][which(data2$x2[-(1:(n/2))] < model2$coefficients[1] + model2$coefficients[2]
     * data2$x1[-(1:(n/2))])] <- 1
```

```
datasim_v2 <- data2[,-4]
save(datasim_v2, file = "2cluster_classification_simulated_v2.Rdata")
```

Codice B.5: Codice per la stima dei modelli in Sottosezione 3.4.1

```
set.seed(12345)
library(mvtnorm)
library(gtools)

load("2cluster_classification_simulated_v1.Rdata")
Rcpp::sourceCpp("gibbs_SVM_Mixture_MultivGauss_classification.cpp")
y <- datasim_v1$y
X <- as.matrix(datasim_v1[,-1])

## training e test set bilanciati rispetto all'originale
ind_train <- c(sample(which(y == 1), round(0.75 * sum(y == 1))),
sample(which(y == -1), round(0.75 * sum(y == -1))))
ind_test <- setdiff(1:nrow(X), ind_train)
X.train <- X[ind_train,]
X.test <- X[ind_test,]
y.train <- y[ind_train]
y.test <- y[ind_test]

############## K = 1, kernel polinomiale d = 1 #############
R <- 5*10^3
K <- 1
X_k1_d1.train <- as.matrix(cbind(rep(1, nrow(X.train)), X.train))
X_k1_d1.test <- as.matrix(cbind(rep(1, nrow(X.test)), X.test))
X.std_k1_d1.train <- cbind(X_k1_d1.train[,1], scale(X_k1_d1.train[, -1]))
X.std_k1_d1.test <- cbind(X_k1_d1.test[,1], scale(X_k1_d1.test[,-1]))

beta0 <- matrix(0, nrow=ncol(X.std_k1_d1.train), ncol=K)
lambda_inv0 <- rep(1, nrow(X.std_k1_d1.train))
omega_inv0 <- matrix(1, nrow=ncol(X.std_k1_d1.train), ncol=K)
nu_inv0 <- rep(1, K)
z0 <- sample(1:K, size = nrow(X.std_k1_d1.train), replace = T)
sim_k1_d1 <- gibbs_svm_mixture_multivgauss_classification(R, beta0, lambda_inv0, omega_inv0, nu_
    inv0, z0, 1, X.std_k1_d1.train, y.train, ncol(X.train))
```

```
cftab_k1_d1 <- conftable_bayesian(X.std_k1_d1.test, y.test, X.std_k1_d1.train, sim_k1_d1$beta, R
    /2, ncol(X.train), sim_k1_d1$z)

############## K = 1, kernel polinomiale d = 2 #############
R <- 5*10^3
K <- 1
X_k1_d2.train <- as.matrix(cbind(rep(1, nrow(X.train)), X.train, X.train^2, X.train[,2]*X.train
    [,1]))
X_k1_d2.test <- as.matrix(cbind(rep(1, nrow(X.test)), X.test, X.test^2, X.test[,2]*X.test[,1]))
X.std_k1_d2.train <- cbind(X_k1_d2.train[,1], scale(X_k1_d2.train[,-1]))
X.std_k1_d2.test <- cbind(X_k1_d2.test[,1], scale(X_k1_d2.test[,-1]))

beta0 <- matrix(0, nrow=ncol(X.std_k1_d2.train), ncol=K)
lambda_inv0 <- rep(1, nrow(X.std_k1_d2.train))
lambda_inv0 <- matrix(1, nrow=nrow(X.std_k1_d2.train), ncol=K)
omega_inv0 <- matrix(1, nrow=ncol(X.std_k1_d2.train), ncol=K)
nu_inv0 <- rep(1, K)
z0 <- sample(1:K, size = nrow(X.std_k1_d2.train), replace = T)
sim_k1_d2 <- gibbs_svm_mixture_multivgauss_classification(R, beta0, lambda_inv0, omega_inv0, nu_
    inv0, z0, 1, X.std_k1_d2.train, y.train, ncol(X.train))

cftab_k1_d2 <- conftable_bayesian(X.std_k1_d2.test, y.test, X.std_k1_d2.train, sim_k1_d2$beta, R
    /2, ncol(X.train), sim_k1_d2$z)

############## K = 2, kernel polinomiale d = 1 #############
R <- 5*10^3
K <- 2
X_k2_d1.train <- as.matrix(cbind(rep(1, nrow(X.train)), X.train))
X_k2_d1.test <- as.matrix(cbind(rep(1, nrow(X.test)), X.test))
X.std_k2_d1.train <- cbind(X_k2_d1.train[,1], scale(X_k2_d1.train[,-1]))
X.std_k2_d1.test <- cbind(X_k2_d1.test[,1], scale(X_k2_d1.test[,-1]))

beta0 <- matrix(0, nrow=ncol(X.std_k2_d1.train), ncol=K)
lambda_inv0 <- rep(1, nrow(X.std_k2_d1.train))
omega_inv0 <- matrix(1, nrow=ncol(X.std_k2_d1.train), ncol=K)
nu_inv0 <- rep(1, K)
z0 <- sample(1:K, size = nrow(X.std_k2_d1.train), replace = T)
sim_k2_d1 <- gibbs_svm_mixture_multivgauss_classification(R, beta0, lambda_inv0, omega_inv0, nu_
    inv0, z0, 1, X.std_k2_d1.train, y.train, ncol(X.train))
```

```
cftab_k2_d1 <- conftable_bayesian(X.std_k2_d1.test, y.test, X.std_k2_d1.train, sim_k2_d1$beta, R
    /2, ncol(X.train), sim_k2_d1$z)

############## K = 2, kernel polinomiale d = 2 #############
R <- 5*10^3
K <- 2
X_k2_d2.train <- as.matrix(cbind(rep(1, nrow(X.train)), X.train, X.train^2, X.train[,2]*X.train
    [,1]))
X_k2_d2.test <- as.matrix(cbind(rep(1, nrow(X.test)), X.test, X.test^2, X.test[,2]*X.test[,1]))
X.std_k2_d2.train <- cbind(X_k2_d2.train[,1], scale(X_k2_d2.train[,-1]))
X.std_k2_d2.test <- cbind(X_k2_d2.test[,1], scale(X_k2_d2.test[,-1]))

beta0 <- matrix(0, nrow=ncol(X.std_k2_d2.train), ncol=K)
lambda_inv0 <- rep(1, nrow(X.std_k2_d2.train))
omega_inv0 <- matrix(1, nrow=ncol(X.std_k2_d2.train), ncol=K)
nu_inv0 <- rep(1, K)
z0 <- sample(1:K, size = nrow(X.std_k2_d2.train), replace = T)
sim_k2_d2 <- gibbs_svm_mixture_multivgauss_classification(R, beta0, lambda_inv0, omega_inv0, nu_
    inv0, z0, 1, X.std_k2_d2.train, y.train, ncol(X.train))

cftab_k2_d2 <- conftable_bayesian(X.std_k2_d2.test, y.test, X.std_k2_d2.train, sim_k2_d2$beta, R
    /2, ncol(X.train), sim_k2_d2$z)
```

Codice B.6: Codice per la stima dei modelli in Sottosezione 3.4.2

```
set.seed(12345)
library(mvtnorm)
library(gtools)

load("2cluster_classification_simulated_v2.Rdata")
Rcpp::sourceCpp("gibbs_SVM_Mixture_MultivGauss_classification.cpp")
y <- datasim_v2$y
X <- as.matrix(datasim_v2[,-1])

## training e test set bilanciati rispetto all'originale
ind_train <- c(sample(which(y == 1), round(0.75 * sum(y == 1))),
sample(which(y == -1), round(0.75 * sum(y == -1))))
ind_test <- setdiff(1:nrow(X), ind_train)
X.train <- X[ind_train,]
```

```
X.test <- X[ind_test,]
y.train <- y[ind_train]
y.test <- y[ind_test]

############## K = 1, kernel polinomiale d = 1 #############
R <- 5*10^3
K <- 1
X_k1_d1.train <- as.matrix(cbind(rep(1, nrow(X.train)), X.train))
X_k1_d1.test <- as.matrix(cbind(rep(1, nrow(X.test)), X.test))
X.std_k1_d1.train <- cbind(X_k1_d1.train[,1], scale(X_k1_d1.train[, -1]))
X.std_k1_d1.test <- cbind(X_k1_d1.test[,1], scale(X_k1_d1.test[,-1]))

beta0 <- matrix(0, nrow=ncol(X.std_k1_d1.train), ncol=K)
lambda_inv0 <- rep(1, nrow(X.std_k1_d1.train))
omega_inv0 <- matrix(1, nrow=ncol(X.std_k1_d1.train), ncol=K)
nu_inv0 <- rep(1, K)
z0 <- sample(1:K, size = nrow(X.std_k1_d1.train), replace = T)
sim_k1_d1 <- gibbs_svm_mixture_multivgauss_classification(R, beta0, lambda_inv0, omega_inv0, nu_
    inv0, z0, 1, X.std_k1_d1.train, y.train, ncol(X.train))

cftab_k1_d1 <- conftable_bayesian(X.std_k1_d1.test, y.test, X.std_k1_d1.train, sim_k1_d1$beta, R
    /2, ncol(X.train), sim_k1_d1$z)

############## K = 1, kernel polinomiale d = 2 #############
R <- 5*10^3
K <- 1
X_k1_d2.train <- as.matrix(cbind(rep(1, nrow(X.train)), X.train, X.train^2, X.train[,2]*X.train
    [,1]))
X_k1_d2.test <- as.matrix(cbind(rep(1, nrow(X.test)), X.test, X.test^2, X.test[,2]*X.test[,1]))
X.std_k1_d2.train <- cbind(X_k1_d2.train[,1], scale(X_k1_d2.train[,-1]))
X.std_k1_d2.test <- cbind(X_k1_d2.test[,1], scale(X_k1_d2.test[,-1]))

beta0 <- matrix(0, nrow=ncol(X.std_k1_d2.train), ncol=K)
lambda_inv0 <- rep(1, nrow(X.std_k1_d2.train))
lambda_inv0 <- matrix(1, nrow=nrow(X.std_k1_d2.train), ncol=K)
omega_inv0 <- matrix(1, nrow=ncol(X.std_k1_d2.train), ncol=K)
nu_inv0 <- rep(1, K)
z0 <- sample(1:K, size = nrow(X.std_k1_d2.train), replace = T)
sim_k1_d2 <- gibbs_svm_mixture_multivgauss_classification(R, beta0, lambda_inv0, omega_inv0, nu_
```

```
    inv0, z0, 1, X.std_k1_d2.train, y.train, ncol(X.train))

cftab_k1_d2 <- conftable_bayesian(X.std_k1_d2.test, y.test, X.std_k1_d2.train, sim_k1_d2$beta, R
    /2, ncol(X.train), sim_k1_d2$z)

############## K = 2, kernel polinomiale d = 1 #############
R <- 5*10^3
K <- 2
X_k2_d1.train <- as.matrix(cbind(rep(1, nrow(X.train)), X.train))
X_k2_d1.test <- as.matrix(cbind(rep(1, nrow(X.test)), X.test))
X.std_k2_d1.train <- cbind(X_k2_d1.train[,1], scale(X_k2_d1.train[,-1]))
X.std_k2_d1.test <- cbind(X_k2_d1.test[,1], scale(X_k2_d1.test[,-1]))

beta0 <- matrix(0, nrow=ncol(X.std_k2_d1.train), ncol=K)
lambda_inv0 <- rep(1, nrow(X.std_k2_d1.train))
omega_inv0 <- matrix(1, nrow=ncol(X.std_k2_d1.train), ncol=K)
nu_inv0 <- rep(1, K)
z0 <- sample(1:K, size = nrow(X.std_k2_d1.train), replace = T)
sim_k2_d1 <- gibbs_svm_mixture_multivgauss_classification(R, beta0, lambda_inv0, omega_inv0, nu_
    inv0, z0, 1, X.std_k2_d1.train, y.train, ncol(X.train))

cftab_k2_d1 <- conftable_bayesian(X.std_k2_d1.test, y.test, X.std_k2_d1.train, sim_k2_d1$beta, R
    /2, ncol(X.train), sim_k2_d1$z)

############## K = 2, kernel polinomiale d = 2 #############
R <- 5*10^3
K <- 2
X_k2_d2.train <- as.matrix(cbind(rep(1, nrow(X.train)), X.train, X.train^2, X.train[,2]*X.train
    [,1]))
X_k2_d2.test <- as.matrix(cbind(rep(1, nrow(X.test)), X.test, X.test^2, X.test[,2]*X.test[,1]))
X.std_k2_d2.train <- cbind(X_k2_d2.train[,1], scale(X_k2_d2.train[,-1]))
X.std_k2_d2.test <- cbind(X_k2_d2.test[,1], scale(X_k2_d2.test[,-1]))

beta0 <- matrix(0, nrow=ncol(X.std_k2_d2.train), ncol=K)
lambda_inv0 <- rep(1, nrow(X.std_k2_d2.train))
omega_inv0 <- matrix(1, nrow=ncol(X.std_k2_d2.train), ncol=K)
nu_inv0 <- rep(1, K)
z0 <- sample(1:K, size = nrow(X.std_k2_d2.train), replace = T)
sim_k2_d2 <- gibbs_svm_mixture_multivgauss_classification(R, beta0, lambda_inv0, omega_inv0, nu_
```

```
    inv0, z0, 1, X.std_k2_d2.train, y.train, ncol(X.train))

cftab_k2_d2 <- conftable_bayesian(X.std_k2_d2.test, y.test, X.std_k2_d2.train, sim_k2_d2$beta, R
    /2, ncol(X.train), sim_k2_d2$z)
```

Codice B.7: Codice per la creazione del dataset simulato utilizzato nel Capitolo 4

```
library(mvtnorm)

set.seed(12345)
n <- 3000
p <- 3
K <- 4
dati <- matrix(NA, nrow=n, ncol=(p+1+1))
dati <- as.data.frame(dati)
colnames(dati) <- c("y", "x_1", "x_2", "x_3", "cluster")
dati$cluster[1:(n/K)] <- 1
dati$cluster[(n/K+1):(2*n/K)] <- 2
dati$cluster[(2*n/K+1):(3*n/K)] <- 3
dati$cluster[(3*n/K+1):n] <- 4
dati$y <- -1

Mu1 <- c(3, 2, 3)
Mu2 <- c(3, 2, -3)
Mu3 <- c(3, -3, 2)
Mu4 <- c(3, -3, -2)
Sigma1 <- matrix(c(3, 0.5, -0.5, 0.5, 3, 0.5, -0.5, 0.5, 3), nrow=p, ncol=p, byrow=T)
Sigma2 <- matrix(c(2, -0.5, -0.5, -0.5, 2, 0.5, -0.5, 0.5, 2), nrow=p, ncol=p, byrow=T)
Sigma3 <- matrix(c(3, 0.5, -0.5, 0.5, 3, -0.5, -0.5, -0.5, 3), nrow=p, ncol=p, byrow=T)
Sigma4 <- matrix(c(2, 0.5, 0.5, 0.5, 2, -0.5, 0.5, -0.5, 2), nrow=p, ncol=p, byrow=T)

dati[1:(n/K), 2:4] <- rmvnorm(n/4, Mu1, Sigma1)
dati[(n/K+1):(2*n/K), 2:4] <- rmvnorm(n/4, Mu2, Sigma2)
dati[(2*n/K+1):(3*n/K), 2:4] <- rmvnorm(n/4, Mu3, Sigma3)
dati[(3*n/K+1):n, 2:4] <- rmvnorm(n/4, Mu4, Sigma4)

model1 <- lm(x_3 ~ x_1 + I(x_1^2), data=dati[1:(n/K), ])
dati$y[1:(n/K)][which(dati$x_3[1:(n/K)] > model1$coefficients[1] + model1$coefficients[2] * dati
    $x_1[1:(n/K)] + model1$coefficients[3] * dati$x_1[1:(n/K)]^2)] <- 1
```

```
model2 <- lm(x_1 ~ x_2 + I(x_3 * x_2), data=dati[(n/K+1):(2*n/K), ])
dati$y[(n/K+1):(2*n/K)][which(dati$x_1[(n/K+1):(2*n/K)] > model2$coefficients[1] + model2$
    coefficients[2] * dati$x_2[(n/K+1):(2*n/K)] + model2$coefficients[3] * dati$x_2[(n/K+1):(2*
    n/K)] * dati$x_3[(n/K+1):(2*n/K)])] <- 1

model3 <- lm(x_1 ~ x_2 + x_3, data=dati[(2*n/K+1):(3*n/K), ])
dati$y[(2*n/K+1):(3*n/K)][which(dati$x_1[(2*n/K+1):(3*n/K)] < model3$coefficients[1] + model3$
    coefficients[2] * dati$x_2[(2*n/K+1):(3*n/K)] + model3$coefficients[3] * dati$x_3[(2*n/K+1)
    :(3*n/K)])] <- 1

model4 <- lm(x_3 ~ 1, data=dati[(3*n/K+1):n, ])
dati$y[(3*n/K+1):n][which(dati$x_3[(3*n/K+1):n] > model4$coefficients[1])] <- 1

datasim_4cluster <- dati[,-5]
save(datasim_4cluster, file = "4cluster_classification_simulated.Rdata")
```

Bibliografia

Azzalini, Adelchi (2014). *The skew-normal and related families*. Vol. 3. Institute of Mathematical Statistics (IMS) Monographs. With the collaboration of Antonella Capitanio. Cambridge University Press, Cambridge, pp. viii+262. ISBN: 978-1-107-02927-9.

Becker, Natalia et al. (2009). «penalizedSVM: a R-package for feature selection SVM classification». In: *Bioinformatics* 25.13, pp. 1711–1712. DOI: 10.1093/bioinformatics/btp286. URL: http://dx.doi.org/10.1093/bioinformatics/btp286.

Celeux, Gilles, Merrilee Hurn e Christian P. Robert (2000). «Computational and inferential difficulties with mixture posterior distributions». In: *J. Amer. Statist. Assoc.* 95.451, pp. 957–970. ISSN: 0162-1459. DOI: 10.2307/2669477. URL: https://doi.org/10.2307/2669477.

Cortes, Corinna e Vladimir Vapnik (1995). «Support-vector networks». en. In: *Machine Learning* 20.3, pp. 273–297. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/BF00994018. URL: http://link.springer.com/10.1007/BF00994018.

Dempster, A. P., N. M. Laird e D. B. Rubin (1977). «Maximum likelihood from incomplete data via the EM algorithm». In: *J. Roy. Statist. Soc. Ser. B* 39.1. With discussion, pp. 1–38. ISSN: 0035-9246.

Diebolt, Jean e Christian P. Robert (1994). «Estimation of finite mixture distributions through Bayesian sampling». In: *J. Roy. Statist. Soc. Ser. B* 56.2, pp. 363–375. ISSN: 0035-9246.

Eddelbuettel, Dirk (2013). *Seamless R and C++ Integration with Rcpp*. ISBN 978-1-4614-6867-7. New York: Springer. DOI: 10.1007/978-1-4614-6868-4.

Frühwirth-Schnatter, Sylvia (2006). *Finite mixture and Markov switching models*. Springer Series in Statistics. Springer, New York, pp. xx+492. ISBN: 978-0-387-32909-3; 0-387-32909-9.

Frühwirth-Schnatter, Sylvia e Saumyadipta Pyne (2010). «Bayesian inference for finite mixtures of univariate and multivariate skew-normal and skew-t distributions». In: *Biostatistics* 11.2, pp. 317–336. DOI: 10.1093/biostatistics/kxp062. URL: http://dx.doi.org/10.1093/biostatistics/kxp062.

Gelman, Andrew e Donald B. Rubin (1992). «Inference from Iterative Simulation Using Multiple Sequences». In: *Statistical Science* 7.4, pp. 457–472. ISSN: 08834237. URL: http://www.jstor.org/stable/2246093.

Gelman, Andrew et al. (2014). *Bayesian data analysis*. Third. Texts in Statistical Science Series. CRC Press, Boca Raton, FL, pp. xiv+661. ISBN: 978-1-4398-4095-5.

Green, Peter J. (1995). «Reversible jump Markov chain Monte Carlo computation and Bayesian model determination». In: *Biometrika* 82.4, pp. 711–732. ISSN: 0006-3444. DOI: 10.1093/biomet/82.4.711. URL: https://doi.org/10.1093/biomet/82.4.711.

Hastie, Trevor, Robert Tibshirani e Jerome Friedman (2009). *The elements of statistical learning*. Second. Springer Series in Statistics. Data mining, inference, and prediction. Springer, New York, pp. xxii+745. ISBN: 978-0-387-84857-0. DOI: 10.1007/978-0-387-84858-7. URL: https://doi.org/10.1007/978-0-387-84858-7.

Iranmanesh, A. et al. (2012). «A new mixture representation for multivariate t». In: *J. Multivariate Anal.* 107, pp. 227–231. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2012.01.013. URL: https://doi.org/10.1016/j.jmva.2012.01.013.

Jacob, P. E. et al. (2017). «Better together? Statistical learning in models made of modules». In: *ArXiv e-prints*. arXiv: 1708.08719 [stat.ME].

Jasra, A., C. C. Holmes e D. A. Stephens (2005). «Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modeling». In: *Statist. Sci.* 20.1, pp. 50–67. ISSN: 0883-4237. DOI: 10.1214/088342305000000016. URL: https://doi.org/10.1214/088342305000000016.

Kamper, Herman (2013). *Gibbs sampling for fitting finite and infinite gaussian mixture models*. URL: http://www.kamperh.com/notes/kamper_bayesgmm13.pdf.

Koller, Daphne e Nir Friedman (2009). *Probabilistic graphical models*. Adaptive Computation and Machine Learning. Principles and techniques. MIT Press, Cambridge, MA, pp. xxxvi+1231. ISBN: 978-0-262-01319-2.

Liu, F., M. J. Bayarri e J. O. Berger (2009). «Modularization in Bayesian analysis, with emphasis on analysis of computer models». In: *Bayesian Anal.* 4.1, pp. 119–150. ISSN: 1936-0975. DOI: 10.1214/09-BA404. URL: https://doi.org/10.1214/09-BA404.

Murphy, Kevin P. (2013). *Machine learning : a probabilistic perspective*. 1ª ed. MIT Press. ISBN: 0262018020.

Neal, Radford M. (2000). «Markov chain sampling methods for Dirichlet process mixture models». In: *J. Comput. Graph. Statist.* 9.2, pp. 249–265. ISSN: 1061-8600. DOI: 10.2307/1390653. URL: https://doi.org/10.2307/1390653.

Papastamoulis, Panagiotis e George Iliopoulos (2009). «Reversible jump MCMC in mixtures of normal distributions with the same component means». In: *Comput. Statist. Data Anal.* 53.4, pp. 900–911. ISSN: 0167-9473. DOI: 10.1016/j.csda.2008.10.022. URL: https://doi.org/10.1016/j.csda.2008.10.022.

Polson, Nicholas G. e Steven L. Scott (2011). «Data augmentation for support vector machines». In: *Bayesian Anal.* 6.1, pp. 1–23. ISSN: 1936-0975. DOI: 10.1214/11-BA601. URL: https://doi.org/10.1214/11-BA601.

Redner, Richard A. e Homer F. Walker (1984). «Mixture densities, maximum likelihood and the EM algorithm». In: *SIAM Rev.* 26.2, pp. 195–239. ISSN: 0036-1445. DOI: 10.1137/1026034. URL: https://doi.org/10.1137/1026034.

van Dyk, David A. e Taeyoung Park (2008). «Partially collapsed Gibbs samplers: theory and methods». In: *J. Amer. Statist. Assoc.* 103.482, pp. 790–796. ISSN: 0162-1459. DOI: 10.1198/016214508000000409. URL: https://doi.org/10.1198/016214508000000409.

West, Mike (1987). «On scale mixtures of normal distributions». In: *Biometrika* 74.3, pp. 646–648. ISSN: 0006-3444. DOI: 10.1093/biomet/74.3.646. URL: https://doi.org/10.1093/biomet/74.3.646.

Woodard, Dawn B., Ciprian Crainiceanu e David Ruppert (2013). «Hierarchical adaptive regression kernels for regression with functional predictors». In: *J. Comput. Graph. Statist.* 22.4, pp. 777–800. ISSN: 1061-8600. DOI: 10.1080/10618600.2012.694765. URL: https://doi.org/10.1080/10618600.2012.694765.

Zhu, Jun et al. (2014). «Gibbs max-margin topic models with data augmentation». In: *J. Mach. Learn. Res.* 15, pp. 1073–1110. ISSN: 1532-4435.

Ringraziamenti

Trovarsi alle battute conclusive di un percorso di studi, quello universitario, lascia indubbiamente spazio a qualche momento di riflessione su quanto l'università e tutto ciò che le gravita attorno abbia influenzato ed arricchito la mia crescita personale negli ultimi cinque anni. Ci sono alcune persone che desidero ringraziare e nominare, per aver contribuito in un modo e nell'altro ad aiutarmi nell'arrivare al termine di questo percorso.

Il ringraziamento più grande e doveroso va ai miei genitori, senza i quali il percorso universitario sarebbe probabilmente rimasto solo una possibilità remota e mai realmente intrapresa: non solo per avermi sempre spinto a dare il meglio di me stesso, ma per avermi insegnato a farlo anche e soprattutto quando si tratta di compiere dei sacrifici. Per avermi concesso l'opportunità di trascorrere in un'altra città l'intero periodo universitario senza che mi potesse mai mancare nulla; per la cieca ed incondizionata fiducia in tutte le scelte che ho deciso di compiere riguardanti il mio percorso universitario. Un ringraziamento che voglio inevitabilmente estendere a mia sorella ed ai miei parenti.

Un secondo ringraziamento va al prof. Bernardi, per l'infinita disponibilità nell'aiutarmi a svolgere questo lavoro e per il tempo dedicatomi in questi mesi, e ad Andrea per avermi aiutato con alcuni aspetti tecnici che mi avrebbero altrimenti richiesto un'infinita quantità in più di tempo.

Ai miei amici di Brescia va un ringraziamento sincero per esserci sempre stati, nonostante l'università in questi anni ci abbia spesso catapultati in differenti città del Nord Italia e lo studio mi abbia talvolta richiesto di dover rinunciare ad una serata assieme.

Alle compagne ed ai compagni di Studenti Per - Udu Padova va un generale ringraziamento per questi cinque anni di duro lavoro, sempre ed incondizionatamente dalla stessa parte. Per l'immensa soddisfazione di aver vinto queste benedette elezioni universitarie, che solo chi ha inseguito questo sogno in tutti questi anni di duro ed infinito lavoro può

realmente capire ed apprezzare.

Un ringraziamento a tutti i compagni di corso ed alle persone conosciute in questi anni di vita padovana: nomi ed elenchi sarebbero troppo lunghi e rischierei ingiustamente di dimenticare qualcuno.

Un ultimo ringraziamento, più formale e diretto al lavoro svolto in questa tesi, a Paolo Scopelliti e Vincenzo Agosto per avermi permesso di sfruttare le risorse computazionali messe a disposizione dell'ateneo sulle quali eseguire alcune delle simulazioni computazionalmente più onerose svolte in questo lavoro.