Università degli studi di Milano–Bicocca
Scuola di Economia e Statistica

Corso di Laurea in
Scienze Statistiche ed Economiche

Regressione nonparametrica tramite processi gaussiani

Relatore: Dott. Tommaso Rigon

Tesi di laurea di:
Erica Delvino
Matricola N. 851269

Anno Accademico 2021/2022

Indice

Capitolo 1

Regressione nonparametrica

In un problema di regressione diversi sono i possibili approcci da utilizzare per modellare la relazione tra la variabile risposta e le covariate.

Nella regressione parametrica viene specificata la forma funzionale che lega l'una alle altre, in modo tale che ci sia un coefficiente fissato per ciascun predittore. Sebbene uno dei grandi vantaggi di questi metodi sia la semplicità di interpretazione, talvolta fissare una funzione è un'ipotesi restrittiva che potrebbe portare ad avere previsioni di scarsa qualità.

L'esigenza di una maggiore flessibilità nel modello ha portato a ricercare strumenti che non si basano su forti assunzioni riguardo la forma della relazione tra le variabili, abbandonando l'idea di conoscere a priori la funzione sottostante i dati.

Molti metodi nonparametrici sono stati proposti originariamente tra la fine degli anni '40 e l'inizio degli anni '50. Da allora ci sono state numerose pubblicazioni che sviluppano proprietà, modifiche ed estensioni delle procedure di base, nonché nuove tecniche. A questo argomento è ad oggi ancora dedicata una notevole ricerca. Per citarne alcuni, metodi come *spline*, *random forest*, *reti neurali* e *wavelets* hanno natura di tipo nonparametrico.

1.1 Metodi di regressione nonparametrica

Dato un insieme di coppie di dati $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ tale che

$$y = f(x) + \epsilon, \tag{1.1}$$

l'obiettivo dei metodi di regressione nonparametrica è di stimare la funzione f che meglio approssima i dati senza ipotizzare che appartenga ad una specifica classe di funzioni parametriche, in quanto la struttura del modello non è specificata a priori, ma è determinata dai dati.

Proprio grazie alla loro flessibilità esistono tantissimi modelli di stima nonparametrica, di seguito ne verranno presentati alcuni.

1.1.1 Regressione locale

Fissando un generico punto x_0 appartenente alla classe dei numeri reali l'obiettivo iniziale è quello di stimare $f(x)$ della (1.1) nello specifico punto x_0, sotto alcune condizioni di regolarità. Se $f(x)$ è una funzione derivabile con derivata continua in x_0 allora è possibile applicare lo sviluppo in serie di Taylor, in cui $f(x)$ è localmente approssimata da una retta passante per $(x_0, f(x_0))$:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \text{resto},$$

dove $f(x_0)$ rappresenta l'intercetta β_0, $f'(x_0)$ la pendenza β_1 di una semplice retta di regressione lineare e il "resto" è una quantità di errore con un ordine di grandezza inferiore a $|x - x_0|$.

In questo contesto essendo le osservazioni pesate in base alla loro distanza dal punto x_0 si ricava il valore dei parametri risolvendo il problema attraverso il *criterio dei minimi quadrati pesati*:

$$\min_{\beta_0,\beta_1} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1(x_i - x_0))^2 w_i \tag{1.2}$$

con i pesi w_i dati da:

$$w_i = \frac{1}{h} w\left(\frac{x_i - x_0}{h}\right), \tag{1.3}$$

dove $w(\cdot)$ è detta *funzione kernel* ed è una funzione di densità simmetrica rispetto all'origine, mentre h è detto *parametro di lisciamento* o *finestra mobile*.
Poiché i pesi sono costruiti attraverso un'ottica "locale" attorno a x_0, più la distanza $|x_i - x_0|$ è bassa più il peso w_i sarà alto.

È possibile osservare che la (1.2) oltre che dipendere da w_i, dipende dal punto arbitrario x_0; per avere quindi una stima di f in un generico punto dello spazio x bisognerebbe ripetere più volte il processo di minimizzazione. Si può dimostrare

che è possibile ottenere la stima di f per un qualsiasi vettore x ed è una formula del tipo:

$$\hat{f}(x) = S_h y \tag{1.4}$$

dove S_h è detta *matrice di lisciamento*, infatti dipende dal parametro h.

Il risultato ottenuto rappresenta una stima non iterativa e lineare nel vettore risposta y e ciò sarà utile da un punto di vista computazionale per alcune considerazioni successive.

Per quanto riguarda la stima di $\hat{f}$ la scelta della specifica *funzione kernel* (1.3) non è molto impattante, al contrario lo è quella di h.

Se h è "piccolo" allora la curva $\hat{f}$ sarà maggiormente oscillatoria in quanto dipende prevalentemente dal comportamento locale dei dati. All'aumentare del parametro h la curva $\hat{f}$ diventerà sempre più smussata dato che il sistema dei pesi opera su una finestra più grande.

Per quanto riguarda la scelta ottimale del parametro h è necessario enunciare alcune proprietà di $\hat{f}$. Da Azzalini & Scarpa (2012):

- la distorsione è multiplo di h^2 ;
- la varianza è multipla di $\frac{1}{nh}$.

Idealmente si vorrebbe quindi scegliere $h \to 0$ per annullare la distorsione, ma ciò farebbe esplodere la varianza di $\hat{f}$ a ∞; si risconterebbe il problema opposto se si scegliesse $h \to \infty$ per annullare la varianza, in quanto il bias sarebbe portato ad ∞. Si ha quindi la necessità di trovare un compromesso.

A livello teorico la scelta ottimale h^* del *parametro di lisciamento* si ottiene per via asintotica. É interessante analizzare due caratteristiche di questo stimatore, da Wasserman (2006):

- $h^* = \mathcal{O}(n^{-1/5})$: h deve tendere a zero come $n^{-1/5}$, quindi molto lentamente;
- l'errore quadratico medio con h^* tende a zero con la velocità di $n^{-4/5}$.

Nei modelli parametrici invece l'errore quadratico medio tende a zero ad una velocità di $1/n$. Il tasso di decrescita più lento nei modelli nonparametrici è il prezzo da pagare per avere una maggiore flessibilità.

Operativamente la scelta di h dipende, oltre che dalla grandezza del campione, da diversi criteri di scelta che saranno discussi più avanti.

1.1.2 Processi Gaussiani

Un secondo metodo di regressione nonparametrica su cui si porrà attenzione sono i *processi gaussiani*.

A differenza della *regressione locale*, che è un metodo più classico di natura algoritmica, i *processi gaussiani* sono un metodo probabilistico che ha una interpretazione bayesiana; quindi è possibile trarre considerazioni su quest'ultimo da un punto di vista inferenziale, in quanto si assume una distribuzione sottostante le variabili casuali.

Questo metodo sarà approfondito nel secondo capitolo in modo più dettagliato.

1.2 Compromesso Distorsione Varianza

Tornando all'obiettivo iniziale di un generico problema di regressione, lo scopo è di stimare $f(x)$ dalla (1.1) per prevedere nuovi valori di y prodotti dallo stesso meccanismo generatore dei dati.

Se si volesse modellare una regressione polinomiale, ovvero un particolare tipo di regressione parametrica in cui la funzione che descrive i dati è costituita da un polinomio, l'obiettivo sarebbe quello di trovare il grado p più adatto del polinomio tra 1 e n (date n osservazioni).

Al crescere di p migliora l'adattamento della funzione ai dati. Il caso limite, detto *overfitting*, è quello in cui $p = n$ dove il polinomio interpola esattamente i punti. Il problema in questo caso è che si ottiene un perfetto adattamento sui dati osservati ma pessime previsioni su nuovi dati.

Continuando ad aggiungere gradi al polinomio aumenta la sua complessità, ottenendo via via una riduzione della distorsione ma un aumento della varianza della stima.

Se si conoscesse $f(x)$ dato un qualsiasi valore x' potremmo determinare $f(x')$ e calcolare alcuni indici di interesse sulla bontà dello stimatore $\hat{y}$ di $f(x)$, come

l'*errore quadratico medio*, definito come:

$$\begin{aligned} MSE &= \left[\mathbf{E}(\hat{y}) - f(x')\right]^2 + var\{\hat{y}\} \\ &= distorsione^2 + varianza \end{aligned}$$

Naturalmente l'obiettivo è quello di trovare il valore del polinomio p che minimizza l'*MSE*. All'aumentare della complessità del modello, quindi del grado p, si ha inizialmente un guadagno in termini di *errore quadratico medio* seguito poi da una perdita.

Se p è "basso" la distorsione, che è dovuta alla mancanza di conoscenza del meccanismo generatore dei dati, sarà alta e la varianza bassa; se "alto" viceversa. Distorsione e varianza sono quindi entità che entrano in conflitto, non è possibile minimizzarle entrambe. È necessario trovare il giusto compromesso tra le due parti.

Dato che nella realtà $f(x)$ è ignota, non è possible calcolare l'*MSE* e di conseguenza valutare il trade-off distorsione- varianza, anche se è possibile ottenere una stima di esso. L'obiettivo rimane sempre quello di cogliere il meccanismo strutturale intrinseco nei dati e le caratteristiche essenziali del campione osservato cercando di evitare l'*overfitting*.

Una strategia utile può essere quella di calcolare la *devianza*, ovvero la somma dei quadrati degli scarti dalla media e vedere dove è minima tra i dati di input e dati sconosciuti usati per fare previsione.

Metodi come quello della *regressione locale* si basano proprio su questo conflitto: se h è "grande" la distorsione è elevata e la varianza bassa, viceversa se h è "piccolo".

1.3 Selezione del modello

Per poter studiare alcuni metodi di selezione del miglior modello è necessario dividere in maniera casuale il dataset a disposizione in due parti differenti:

- *training set*: porzione di dati utilizzata nella fase di stima del modello

- *test set*: porzione di dati "sconosciuta" utilizzata esclusivamente per fare previsione sul modello stimato

1.3.1 Criteri basati sull'informazione

Tornando all'esempio della *regressione polinomiale*, una possibile strategia per selezionare il modello con il migliore grado p del polinomio potrebbe essere quella di confrontare la massimizzazione della log-verosimiglianza unita ad una quantità che penalizza l'utilizzo di un eccessivo numero di parametri.
Esiste una famiglia di criteri che segue tale logica, costituita dai *criteri basati sull'informazione* e sono del tipo:

$$\mathrm{IC} = -2\log\mathcal{L}(\hat{\theta}) + \mathtt{penalità}(p). \tag{1.5}$$

Il criterio più conosciuto è quello di Akaike ed è così descritto:

$$\mathrm{AIC} = -2\log\mathcal{L}(\hat{\theta}) + 2p. \tag{1.6}$$

A questa formulazione originaria sono seguite alcune modifiche, una delle più usate è il criterio di Akaike-Schwarz:

$$\mathrm{BIC} = -2\log\mathcal{L}(\hat{\theta}) + p\log n. \tag{1.7}$$

Il modello preferibile è quello che presenta il più basso valore tra tutti. Ovviamente a parità di risultato è bene scegliere il modello più semplice, così da non complicarlo inutilmente.

La differenza principale tra questi due criteri è che l'AIC, avendo una penalizzazione costante pari a 2, non sfavorisce a sufficienza i modelli complessi; il BIC invece utilizza una penalizzazione che cresce con n, quindi tiene conto della dimensione del campione.

In pratica, se il campione diventa più grande logicamente si hanno più informazioni e seguendo il criterio del BIC si è disposti ad abbandonare il modello più semplice solo se la complessità viene ripagata dal fatto di avere più informazioni, cosa che nell'AIC non viene considerata.

Ritornando invece alla regressione nonparametrica non è possibile in via naturale

utilizzare i criteri basati sull'informazione poichè non andando a stimare dei parametri viene meno il calcolo del termine di penalità, che dipende da p.

Se però la stima nonparametrica con cui si ha a che fare è un lisciatore lineare nella risposta del tipo (1.4), come il caso della *regressione locale*, è possibile determinare una sorta di numero di parametri, sebbene approssimati, detti *gradi di libertà equivalenti* che corrispondono alla traccia della *matrice di lisciamento*: $\mathrm{tr}(S_h)$. Procedendo in questa via è possibile approssimare le quantità corrispondenti ai criteri di informazione anche per stime di tipo nonparametrico, purchè sia una procedura lineare nella risposta.

1.3.2 Cross-Validation

Il metodo del *cross-validation* come strumento di scelta del modello richiede la divisione sopra citata in *training* e *test set*.

Per poter ottenere un risultato più stabile è bene che si alternino a rotazione le porzioni di *training set* e di *test set*. Naturalmente la stima del modello risulterà più accurata maggiore è la porzione di dati compresi nella porzione di allenamento.

Il caso limite è costituito dall'Algoritmo 1.1 *leave-one-out cross-validation* in cui, date n osservazioni, il campione di ampiezza *n-1* costituisce il *training set* e solo l'*i-esima* osservazione costituisce il *test set*. Questo procedimento viene ripetuto n volte, perciò a turno ogni osservazione sarà usata per verificare l'adeguatezza del modello.

Algoritmo 1.1 *Leave-one-out cross-validation*

- Ciclo per $i = 1, \ldots, n$
  - stimare il modello eliminando l'*i-esima* osservazione
  - ottenere la previsione di $\hat{y}_{-i}$ per y_i in corrispondenza del punto x_i
  - calcolare l'errore di previsione $e_i = y_i - \hat{y}_{-i}$
- Calcolare il quadrato della somma degli errori di previsione $R = \sum_{i=1}^{n} e_i^2$
- Scegliere il modello per cui R è minimo

Questa procedura al crescere di n diventerebbe molto onerosa da un punto di vista computazionale e poco efficiente. In alcuni casi fortunatamente è possibile

ottenere stime di un modello usando dati privati da una singola osservazione mediante operazioni basate sulle stime utilizzando il dataset completo. Da Wasserman (2006):

Teorema 1 (Lisciatori lineari)**.** *Data $\hat{f}(x)$ lisciatore lineare, il quadrato della somma degli errori di previsione è dato da:*

$$R = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \hat{f}(x)}{1 - S_{ii}} \right)^2 \tag{1.8}$$

dove S_{ii} è l'*i-esimo* elemento della *matrice di lisciamento* S.

Questo risultato è un grande vantaggio in quanto permette di calcolare in un'unica iterazione la somma degli errori di previsione, senza dover impostare un ciclo per ogni osservazione del dataset. Essendo un teorema applicabile ai *lisciatori lineari* è possibile utilizzarlo in maniera agevole nella *regressione locale* per la scelta del parametro h.

Capitolo 2

Processi gaussiani

Rispetto alla *regressione locale*, come già accennato, i *processi gaussiani* sono un metodo nonparametrico che ha il vantaggio di poter fare inferenza in modo semplice sulle singole funzioni una volta stimate.

Un *processo gaussiano* è definito come una *generalizzazione della distribuzione di probabilità gaussiana*. È interessante soffermarsi su due aspetti di tale definizione:

- È un *processo*, perché governa le proprietà delle funzioni, come un qualsiasi processo stocastico;
- È una *distribuzione di probabilità*, perché descrive fenomeni aleatori.

2.1 Introduzione alla statistica bayesiana

Nella statistica frequentista l'approccio che si segue è quello di trattare i parametri come delle quantità fisse e ottenere una stima di essi utilizzando dei campioni dalla popolazione, ma da campioni diversi nascono però delle stime differenti.

L'approccio bayesiano è un modo diverso di pensare la statistica. Esso analizza la variazione delle credenze sui parametri, senza considerarli fissi.

I parametri sono trattati come delle variabili casuali, nel caso dei *processi gaussiani* sono delle funzioni, che possono essere descritti da distribuzioni di probabilità.

Per i frequentisti la probabilità è legata alla frequenza relativa di un evento, per i bayesiani è legata alla certezza o all'incertezza di un evento.

Nella statistica bayesiana si attribuisce una *probabilità a priori* ai parametri di

interesse, stabilita in maniera personale in base al grado di credibilità del verificarsi dell'evento. Solo dopo questo passaggio, a posteriori, si osservano i dati e si valuta la funzione che meglio li rappresenta.

Essendo la *probabilità a posterori* basata su un'informazione a priori, oltre che essere basata sui dati osservati, non è una probabilità assoluta ma condizionata.

È importante notare che la specificazione a priori è fondamentale in quanto si fissano le caratteristiche delle funzioni usate per fare inferenza. Nei *processi gaussiani* le proprietà sulla quale ci si concentra sono due e riguardano la funzione di covarianza:

- **autocorrelazione** che riporta al discorso sull'*overfitting* del capitolo precedente: più le osservazioni sono correlate tra loro più la funzione risulterà smussata;

- **stazionarietà** ovvero che al variare di x la funzione f rimane simile.

2.2 Weight space view

2.2.1 Modello lineare da un punto di vista bayesiano

È possibile studiare il semplice modello di regressione lineare da un punto di vista bayesiano in cui, seguendo la notazione di Rasmussen & Williams (2006):

$$f(x) = x^T \beta, \tag{2.1}$$

dove x è un vettore colonna $(1 \times p)$ parte della *matrice del disegno* X di dimensioni $(p \times n)$.

I valori di y della 1.1 differiscono da $f(x)$ per un rumore ϵ, che viene assunto seguire una distribuzione Normale con media nulla e varianza pari a σ_n^2:

$$\epsilon \sim \mathcal{N}(0, \sigma_n^2). \tag{2.2}$$

Come nella statistica classica è possibile calcolare la *funzione di verosimiglianza*, ovvero la densità di probabilità delle osservazioni, che corrisponde alla probabilità

di y condizionata ad X:

$$p\left(y|X,\beta\right) = \prod_{i=1}^{N} p(y_i|x_i,\beta) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{\left(y_i - x_i^\intercal \beta\right)^2}{2\sigma_n^2}\right) = \\ = \left(\frac{1}{2\pi\sigma_n^2}\right)^{n/2} \exp\left(-\frac{1}{2\sigma_n^2}\sum_{i=1}^{N}\left(y_i - x^\intercal\beta\right)^2\right).$$

Seguendo la logica bayesiana è necessario definire una probabilità a priori sui parametri. Si assume $\beta \sim \mathcal{N}_p(0, \Sigma_p^2)$ a cui corrisponde la probabilità a priori $p(\beta)$, che fa riferimento alle credenze che si hanno sui parametri prima di guardare i dati.
Combinando la probabilità condizionata con la probabilità a priori si ottiene la probabilità a posteriori data dal Teorema di Bayes:

Teorema 2.1 (Teorema di Bayes)**.**

$$p(\beta|y,X) = \frac{p(y|X,\beta)p(\beta)}{p(y|X)} = \frac{verosimiglianza * priori}{verosimiglianza\ marginale}\,, \tag{2.3}$$

dove la *verosimiglianza marginale* è pari a $\int p(y|X,\beta)p(\beta)d\beta$.

Concentrandosi sul numeratore del (2.3), quindi sulle quantità che dipendono dai pesi β, si ottiene che:

$$\begin{aligned} p\left(\beta|X,y\right) &\propto \exp\left(-\frac{1}{2\sigma_n^2}\left(y - X^\intercal\beta\right)^\intercal\left(y - X^\intercal\beta\right)\right)\exp\left(-\frac{1}{2}\beta^\intercal\Sigma_p^{-1}\beta\right) \\ &\propto \exp\left(-\frac{1}{2}\left(\beta - \bar{\beta}\right)^\intercal A\left(\beta - \bar{\beta}\right)\right)\,. \end{aligned} \tag{2.4}$$

Pertanto, $(\beta|x,y) \sim \mathcal{N}_p(\bar{\beta}, A^{-1})$, dove $\bar{\beta}$ è il vettore delle medie, pari a:

$$\bar{\beta} = \frac{1}{\sigma_n^2}A^{-1}Xy \tag{2.5}$$

e A^{-1} è la matrice di varianza e covarianza pari a:

$$A^{-1} = \frac{1}{\sigma_n^2}XX^\intercal + \Sigma_p^{-1}\,. \tag{2.6}$$

Previsione della distribuzione a priori

Nella statistica bayesiana per prevedere nuovi dati di input si considera la distribuzione a posteriori del previsore lineare, ovvero (2.1). La previsione della funzione $f^* = x_*^T\beta$ per nuovo punto di *test set* x_* è data da:

$$(f^*|X, y) = (x_*^T\beta|X, y) \sim \mathcal{N}(x_*^T\bar{\beta}, x_*^T A^{-1} x_*) \tag{2.7}$$

ed è una gaussiana con media pari alla media di 2.5 pre-moltiplicata per x_* e varianza pari a una forma quadratica della matrice di varianza-covarianza della distribuzione a posteriori 2.6 nel punto nel punto x_*.

2.2.2 Feature space view

Essendo il modello (2.1) lineare rispetto agli input, la sua espressività risulta essere piuttosto limitata.

Un modo per renderlo più flessibile può essere quello di proiettare le osservazioni in uno spazio ad alta dimensionalità facendo uso di un insieme di funzioni, dette *funzioni base* e applicando il modello lineare direttamente a questo spazio.

Perseguendo questa via si perde la linearità negli input ma si conserva la linearità nei parametri.

Si introduce quindi la funzione $\phi(x)$ che proietta un vettore di input p-dimensionale, appartenente allo spazio delle osservazioni, in un particolare spazio N-dimensionale delle *features*:

$$\phi : x \to \phi(x) = (\phi_1(x), \dots \phi_N(x)) \ .$$

Il nuovo modello di regressione che si ottiene è pari a:

$$f(x) = \phi(x)^T\beta. \tag{2.8}$$

Al posto della *matrice del disegno* X si avrà la nuova matrice $\Phi(X)$ data da:

$$\Phi = \Phi(X) = \begin{bmatrix} \phi_1(x_1) & \dots & \phi_1(x_n) \\ \vdots & \ddots & \vdots \\ \phi_N(x_1) & \dots & \phi_N(x_n) \end{bmatrix}.$$

Ora la distribuzione della funzione di previsione f^* per il punto x_* diventa:

$$f^*|x_*, X, y \sim \mathcal{N}\left(\frac{1}{\sigma_n^2}\phi(x_*)^\intercal A^{-1}\Phi y,\ \phi(x_*)^\intercal A^{-1}\phi(x_*)\right) \tag{2.9}$$

con $A = \sigma_n^{-2}\Phi\Phi^\intercal + \Sigma_p^{-1}$.
È bene osservare che al crescere di N, quindi al crescere della dimensione dello spazio delle features, l'inversione della matrice A di dimensione $(N \times N)$ si rivelerebbe un calcolo oneroso da un punto di vista computazionale.

È possibile aggirare questo problema ricorrendo ad una formulazione alternativa definendo innanzitutto la *funzione kernel* o *funzione di covarianza* $k(\cdot,\cdot)$ tale per cui, dati due punti x e x′: $k(x, x') = \phi(x)^\intercal\Sigma_p\phi(x')$. Si può dimostrare che la (2.9) è equivalente alla seguente scrittura:

$$f^*|x_*, X, y \sim \mathcal{N}\left(\phi_*^\intercal\Sigma_p\Phi(K + \sigma_n^2 I)^{-1}y,\ \phi_*^\intercal\Sigma_p\phi_* - \phi_*^\intercal\Sigma_p\Phi(K + \sigma_n^2 I)^{-1}\Phi^\intercal\Sigma_p\phi_*\right), \tag{2.10}$$

dove ϕ_* corrisponde a $\phi(x_*)$ e K corrisponde a $\Phi^\intercal\Sigma_p\Phi$.

Il vantaggio di questa formulazione alternativa è dato dal fatto che la matrice da invertire nella (2.10) ha dimensione $(n \times n)$, perciò si ha un guadagno in termini computazionali quando $n<N$.

2.3 Function space view

Un modo alternativo che consente di ottenere gli stessi risultati della *feature space view* è possibile facendo inferenza direttamente nello *spazio delle funzioni*, usando i *processi gaussiani*.

Da Rasmussen & Williams (2006): Un processo gaussiano è una collezione di variabili casuali, tra cui un numero finito di esse ha una distribuzione congiunta gaussiana.

La specificazione di un *processo gaussiano* avviene tramite la definizione della *funzione media* e della *funzione di varianza*:

$$f(x) \sim GP\left(m(x), k(x, x')\right) \tag{2.11}$$

in cui:

- $m(x) = \mathbf{E}[f(x)]$ rappresenta la *funzione media*;
- $k(x, x') = \mathbf{E}[(f(x) - m(x))(f(x') - m(x'))]$ rappresenta la *funzione di covarianza* o *funzione kernel*.

Riguardo la *funzione kernel*, come affermato precedentemente nel primo capitolo, non esiste un'unica *funzione kernel*. Di seguito nella tabella 2.1 vengono riportati alcuni esempi:

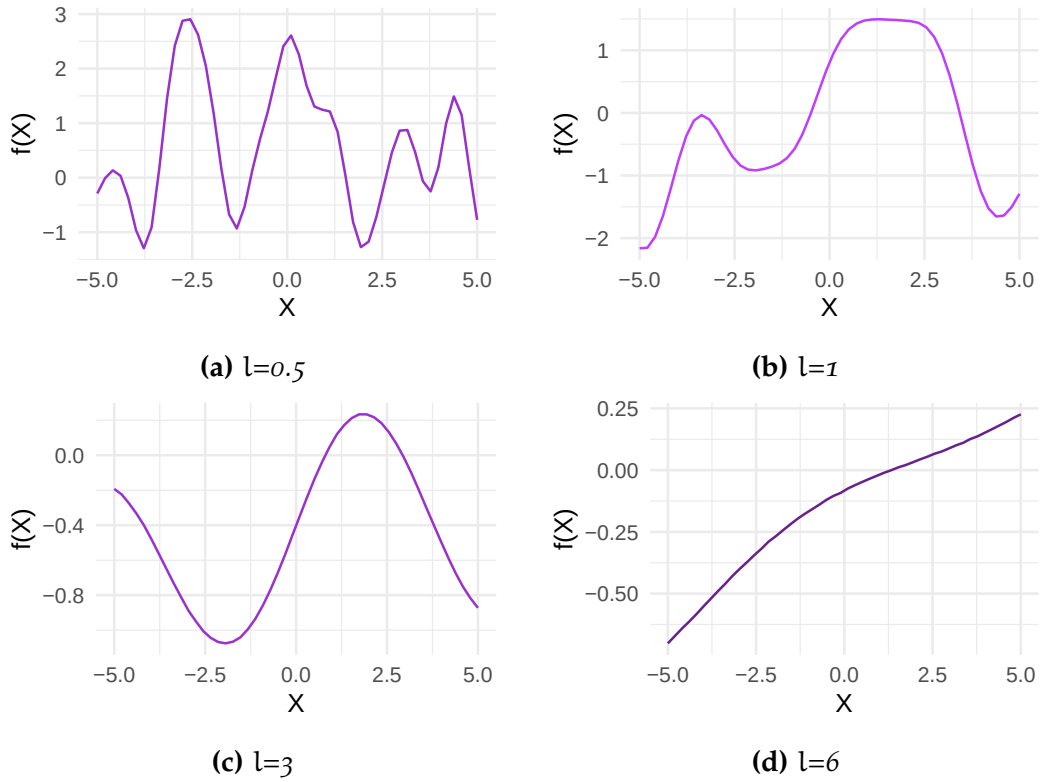
Tabella 2.1

Funzione kernel	funzione
squared exponential	$\exp\left(-\frac{1}{2}\frac{\lvert x-x'\rvert^2}{l^2}\right)$
exponential	$\exp\left(-\frac{1}{2}\frac{\lvert x-x'\rvert}{l^2}\right)$
rational quadratic	$\left(1+\frac{\lvert x-x'\rvert^2}{2\alpha l^2}\right)^{-\alpha}$

È interessante notare la relazione tra covarianza ed input negli esempi di *funzioni kernel* della tabella 2.1. Se $|x - x'| \approx 0$ allora i due punti sono vicini e $cov(\cdot, \cdot) \approx 1$; all'aumentare della distanza tra i punti x e x' la *funzione di covarianza* decresce.

Un'importante proprietà di cui godono i *processi gaussiani* è la **proprietà di marginalizzazione** la quale afferma che considerare un insieme più grande di variabili non cambia la distribuzione dell'insieme più piccolo. Facendo quindi un esempio, se $(y_1, y_2) \sim \mathcal{N}(\mu, \Sigma)$, $y_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$ dove Σ_{11} è la sottomatrice di Σ.

Nei *processi gaussiani* si ha a che fare con oggetti infinito dimensionali e si campiona una parte di essi da una distribuzione gaussiana multivariata di parametri pari alla media indotta dalla *funzione media* e covarianza indotta dalla *funzione kernel*.



(a) l=*0.5* **(b)** l=*1*

(c) l=*3* **(d)** l=*6*

Figura 2.1: Alcune realizzazioni della funzione a priori per differenti valori del parametro di scala.

La specificazione della *funzione di covarianza* implica una distribuzione sulle funzioni. Infatti per generare la *distribuzione a priori* f_*:

- si sceglie un determinato set di dati $X_* \sim \mathcal{N}_p(0, \Sigma_p)$
- si ottiene la *matrice di covarianze* $K(X_*, X_*)$
- si genera un vettore casuale con distribuzione gaussiana $f_* \sim \mathcal{N}_p(0, K(X_*, X_*))$

La figura Figura 2.1 mostra la generazione di quattro differenti funzioni a priori al variare del parametro di lisciamento l utilizzando la funzione di covarianza *squared exponential*.

2.3.1 Previsione della distribuzione a posteriori

Avendo un insieme di dati a disposizione, è opportuno combinare la specificazione della *distribuzione a priori* f_* con essi. È bene osservare che, avendo un modello del tipo 1.1, la covarianza tra i dati sarà data da:

$$cov(y) = K(X; X) + \sigma_n^2 I, \tag{2.12}$$

avendo osservazioni con un rumore additivo ϵ.

La distribuzione congiunta dei valori osservati nel *training set* e dei valori di *test set* della la *distribuzione a priori* è data da:

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}_p \left(0, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix} \right) \tag{2.13}$$

Condizionando la *distribuzione a priori* sulle osservazioni si ottiene la specificazione della *distribuzione a posteriori* con le sue funzioni di media e covarianza associate:

$$f_*|X, y, X_* \sim \mathcal{N}(\bar{f}_*, cov(f_*)) \tag{2.14}$$

dove:

$$\bar{f}_* = K(X_*,X)[K(X,X) + \sigma_n^2 I]^{-1} y = S_y y \tag{2.15}$$

$$cov(f_*) = K(X_*,X_*) - K(X_*,X)[K(X,X) + \sigma_n^2 I]^{-1} K(X,X_*) \tag{2.16}$$

Concentrandosi sulla (2.15) è possibile notare che la *funzione media* è combinazione lineare della risposta y. I *processi gaussiani* quindi, così come la *regressione locale*, sono dei *lisciatori lineari*; infatti è possibile notare l'equivalenza con la formulazione (1.4).

Analizzando invece la (2.16) si può osservare che la *funzione kernel* non dipende dalla risposta, bensì solo dai valori di input ed è data dalla differenza tra due termini: il primo costituisce la covarianza a priori a cui viene sottratto il secondo termine che riguarda le informazioni che le osservazioni danno sulla funzione.

2.3.2 Equivalenza delle formulazioni da weight-space a function space view

È possibile dimostrare che la scrittura della *funzione media* per i *processi gaussiani* della (2.5) è analoga a (2.15), così come la *covarianza* della (2.6) è analoga a (2.16).

Per dimostrare queste due equivalenze è necessario avvalersi della seguente identità di matrici:

Teorema 2.2 (Matrix Inversion Lemma)**.**

$$(Z + UDV^\intercal)^{-1} = Z^{-1} - Z^{-1}U(D^{-1} + V^\intercal Z^{-1} U)^{-1} V^\intercal Z^{-1} \tag{2.17}$$

Da questo Teorema è possibile scomporre la matrice A^{-1} della (2.6) in:

$$\begin{aligned} A^{-1} &= \left(\frac{1}{\sigma_n^2} XX^\intercal + \Sigma_p^{-1}\right)^{-1} \\ &= \Sigma_p - \Sigma_p X^\intercal \left(\sigma_n^2 I + X\Sigma_p X^\intercal\right)^{-1} X\Sigma_p \ , \end{aligned} \tag{2.18}$$

con: $Z = \Sigma_p^{-1}$, $U = X^\intercal$, $V^\intercal = X$ e $D = 1/(\sigma_n^2 I)$

Equivalenza in media

Dimostrazione. Equivalenza delle formulazioni in media.

$$\begin{aligned} &\frac{1}{\sigma_n^2} x_*^\intercal A^{-1} X^\intercal y = \\ &= \frac{1}{\sigma_n^2} x_*^\intercal \left[\Sigma_p - \Sigma_p X^\intercal (\sigma_n^2 I + X\Sigma_p X^\intercal)^{-1} X\Sigma_p\right] X^\intercal y \\ &= \frac{1}{\sigma_n^2} x_*^\intercal \Sigma_p X^\intercal \left[I - (\sigma_n^2 I + X\Sigma_p X^\intercal)^{-1} X\Sigma_p X^\intercal\right] y \\ &= \frac{1}{\sigma_n^2} x_*^\intercal \Sigma_p X^\intercal \left[I - \left(I + \frac{X\Sigma_p X^\intercal}{\sigma_n^2}\right)^{-1} \frac{X\Sigma_p X^\intercal}{\sigma_n^2}\right] y \end{aligned}$$

Sostituendo $Q = \frac{X\Sigma_p X^\intercal}{\sigma_n^2}$ all'interno dell'uguaglianza:

$$= \frac{1}{\sigma_n^2} x_*^\intercal \Sigma_p X^\intercal \left[I - (I + Q)^{-1} Q\right] y$$

Concentrandosi sul termine tra parentesi quadre è possibile applicare nuovamente il *matrix inversion lemma* (2.17) in cui $Z = U = V^\intercal = I$ e $D = Q$:

$$\begin{aligned} I - (I + Q)^{-1} Q = I - (Q^{-1} + I)^{-1} = \\ = (I + Q)^{-1} = \\ = \left(I + \frac{X\Sigma_p X^\intercal}{\sigma_n^2}\right)^{-1} \end{aligned}$$

Tornando all'equazione iniziale:

$$\begin{aligned}
&\frac{1}{\sigma_n^2} x_*^\intercal \Sigma_p X^\intercal \left[I - (I + Q)^{-1} Q \right] y = \\
&= \frac{1}{\sigma_n^2} x_*^\intercal \Sigma_p X^\intercal \left[I + \frac{X \Sigma_p X^\intercal}{\sigma_n^2} \right]^{-1} y = \\
&= x_*^\intercal \Sigma_p X^\intercal \left(\sigma_n^2 I + X \Sigma_p X^\intercal \right)^{-1} y = \\
&= k_*^\intercal (\sigma_n^2 I + K)^{-1} y
\end{aligned}$$

□

Equivalenza in varianza

Dimostrazione. Equivalenza delle formulazioni in varianza.

$$\begin{aligned}
&x_*^\intercal A^{-1} x_* = \\
&= x_*^\intercal \left[\Sigma_p - \Sigma_p X^\intercal \left(\sigma_n^2 I + X \Sigma_p X^\intercal \right)^{-1} X \Sigma_p \right] x_* = \\
&= x_*^\intercal \Sigma_p x_* - x_*^\intercal \Sigma_p X^\intercal (\sigma_n^2 I + X \Sigma_p X^\intercal)^{-1} X \Sigma_p x_* = \\
&= k(x_*, x_*) - k_*^\intercal (K + \sigma_n^2 I)^{-1} k_*
\end{aligned}$$

□

2.3.3 Verosimiglianza marginale

Imponendo una distribuzione sulle funzioni, è possibile calcolare la *funzione di verosimiglianza marginale* e utilizzarla come possibile criterio di scelta del miglior modello, oltre a quelli precedentemente esplorati nel primo capitolo.

La *funzione di verosimgilianza marginale* è l'integrale della funzione di verosimiglianza moltiplicata per la probabilità a priori.

$$p(y|X) = \int p(y|f, X) p(f|X) \, df.$$

Poiché $y \sim \mathcal{N}(0, K + \sigma_n^2 I)$ derivando la funzione di *log-verosimiglianza marginale*, seguendo la notazione di Rasmussen & Williams (2006), si ottiene:

$$\log p(y|X) = -\frac{1}{2} y^\intercal (K + \sigma_n^2 I)^{-1} y - \frac{1}{2} \log |K + \sigma_n^2 I| - \frac{n}{2} \log(2\pi), \tag{2.19}$$

dove $|K + \sigma_n^2 I|$ rappresenta il determinante di tale matrice.

2.4 Scelta degli Iperparametri

Tipicamente le *funzioni di covarianza* che si utilizzano hanno alcuni parametri liberi. Considerando per esempio la funzione di covarianza *squared exponential*, che ha equazione del tipo:

$$k(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(x_p - x_q)^2\right) + \sigma_n^2 \delta_{pq}, \tag{2.20}$$

dove:

- σ_f^2 è la varianza a priori;
- l è il parametro di lisciamento o parametro di scala;
- σ_n^2 identifica la varianza residuale, che si ricava dai dati;
- δ_{pq} è il *delta di Kronecker* ed è pari a 1 se $p = q$, 0 altrimenti

ha un vettore di tre parametri liberi, detti *iperparametri*: $(\sigma_f^2, l, \sigma_n^2)$.

Si possono utilizzare differenti metodi per determinare la combinazione migliore di tali parametri, per esempio: *cross validation* e *criteri basati sull'informazione*, già analizzati nel primo capitolo, oppure è possibile scegliere la combinazione di parametri che massimizza la *log-verosimiglianza marginale*.

Riguardo la scelta del *parametro di scala* l, se è "troppo piccolo" la curva f avrà un basso rumore ma una varianza elevata, perciò risulterà molto oscillatoria; viceversa se l fosse "troppo grande".

2.5 Applicazione

Questa semplice applicazione ha lo scopo di illustrare visivamente l'utilizzo dei *procesi gaussiani* come metodo di regressione nonparametrica.
Il dataset preso in considerazione è *jaws.txt* https://data-flair.training/blogs/r-nonlinear-regression/, composto da due variabili:

- *Age*: covariata che rappresenta l'età di un cervo;
- *Bone*: variabile risposta che rappresenta la dimensione dell'osso della mandibola di un cervo.

L'obiettivo è quello di stimare tramite *processi gaussiani* la relazione tra la le due variabili mediante differenti metodi di scelta del modello.

Nella tabella 2.2 viene riportata una sintesi dei risultati ottenuti.

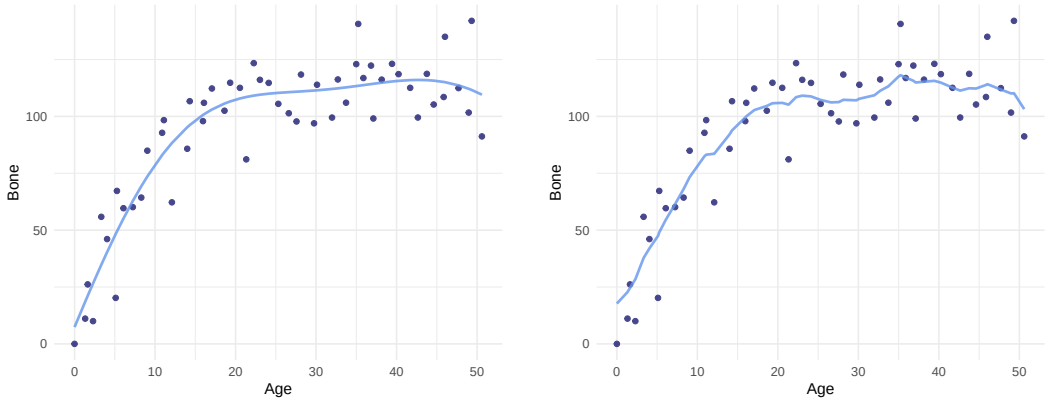
Tabella 2.2

Criterio	Funzione kernel	σ_f^2	l	σ_n^2
Cross validation	Squared Exponential	700	21	10
Cross validation	Exponential	500	4	170
Log-verosimiglianza	Squared Exponential	$4.7e^{11}$	14.3	$6.4e^{11}$
Log-verosimiglianza	Exponential	444	7.2	26.1
AIC	Squared Exponential	2000	2	10
BIC	Squared Exponential	2000	3	10
AIC	Exponential	2000	1	10
BIC	Exponential	1000	1	10

Il primo metodo utilizzato come criterio di scelta del modello è stato *cross validation*, applicato alla funzione di covarianza *squared exponential* e successivamente alla funzione *exponential*.

È stata inizialmente creata una griglia con varie possibili combinazioni dei tre parametri per la funzione di covarianza e in seguito è stato applicato l'algoritmo *leave-one-out cross-validation*.

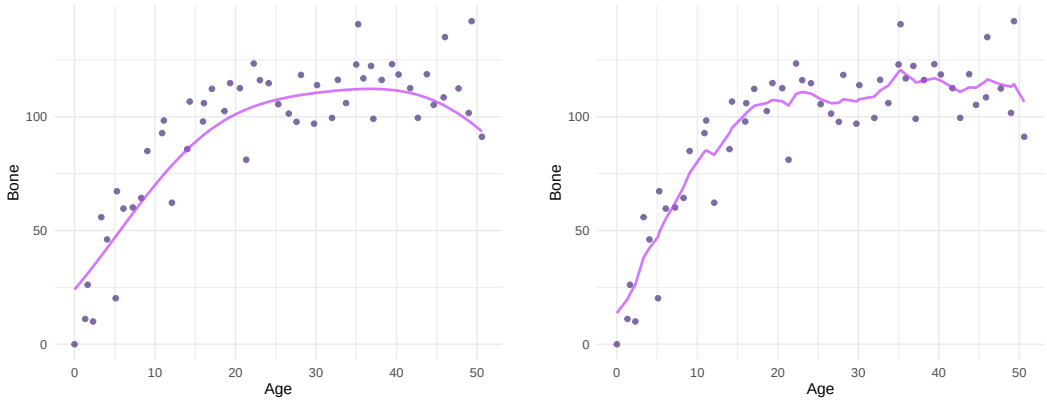
In entrambi i casi l'algoritmo sembra performare bene nella previsione della risposta.



(a) Squared exponential kernel **(b)** Exponential kernel

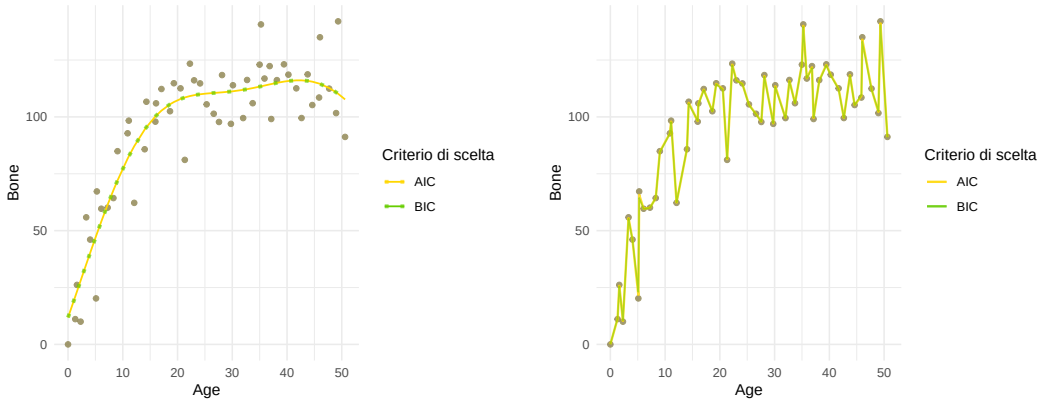
Figura 2.2: Cross validation

Nel secondo esempio il criterio di scelta utilizzato è stato quello della massimizzazione della *log-verosimiglianza marginale* applicato alla funzione *squared exponential* ed *exponential*. Come si può vedere dalla Figura 2.3 e dal valore dei parametri stimati dalla Tabella 4.2 la previsione ha prodotto un risultato non ottimale per la funzione di covarianza *squared exponential*; probabilmente la funzione è arrivata a convergenza su un punto di ottimo locale.



(a) Squared exponential kernel **(b)** Exponential kernel

Figura 2.3: Massimizzazione della log-verosimiglianza marginale



(a) Squared exponential kernel **(b)** Exponential kernel

Figura 2.4: Criteri basati sull'informazione

Nell'ultimo esempio vengono confrontati due diversi *criteri basati sull'informazione*, *AIC* e *BIC*, applicati alle funzione di covarianza *squared exponential* ed *exponential*, considerando come gradi di libertà fittizi la traccia della *matrice di lisciamento* S_h.

Si può notare dalla Figura 2.4 che i parametri selezionati dai due differenti criteri danno luogo allo stesso risultato per la funzione *squared exponential*, mentre

per la funzione *exponential* i risultati sono simili ma i parametri selezionati corrispondono ad una curva che si adatta perfettamente ai dati e riporta al problema di *overfitting* discusso nel capitolo precedente.

In conclusione è possibile osservare che la funzione di covarianza *exponential* risulta essere segmentata rispetto alla funzione *squared exponential* che si presenta con un tratto continuo, a prescindere dal criterio di scelta del modello adottato.

Riguardo invece i diversi metodi di scelta del modello il criterio di *cross-validation* sembra essere quello che permette di avere un risultato più stabile.

Capitolo 3

Estensione al caso multivariato

Quanto visto nei precedenti capitoli fa riferimento al caso univariato, in cui c'è corrispondenza biunivoca tra la risposta e la variabile indipendente.

Se si volesse estendere al caso multivariato i metodi studiati finora è necessario introdurre nuove strutture che permettano di applicare i *processi gaussiani* in più dimensioni.

3.1 La maledizione della dimensionalità

In senso algoritmico è relativamente semplice estendere i *lisciatori lineari* univariati in più dimensioni. Incrementando però il numero di variabili le prestazioni diventano sempre meno efficienti, infatti all'aumentare del numero p dello spazio delle covariate aumenta esponenzialmente la dispersione dei punti nello spazio.

Facendo un semplice esempio, da Azzalini & Scarpa (2012), date un numero di osservazioni con $n = 500$ in un intervallo $(0, 1)$ disposte casualmente su una retta x, se si utilizzassero questi n punti per stimare la funzione $f(x)$ si otterrebbe una stima affidabile, grazie alla piccola distanza media che li separa.

Se lo stesso numero di punti fosse disperso causalmente su un piano (x_1, x_2) essi sarebbero più distanti tra loro essendo distribuiti non più su una retta ma sul quadrato di dimensioni $(0, 1)^2$.

Procedendo in questa maniera, all'aumentare della dimensione dello spazio la distanza tra i punti aumenta e la stima di f diventa sempre meno attendibile.

Per compensare l'aumento dello scostamento tra i dati si avrebbe bisogno di un numero di punti dell'ordine di grandezza di n^p, condizione molto improbabile se le covariate sono parecchie. Questa situazione di impossibilità di stimare la funzione f in modo accurato è chiamata *maledizione della dimensionalità*.

È necessario quindi introdurre dei vincoli che diano una sorta di struttura al modello così che, seppur più rigido, possa produrre delle stime ragionevoli.

3.2 Modelli Additivi

Un possibile modo per aggirare la *maledizione della dimensionalità* quando p è "alto" riguarda l'utilizzo dei *modelli additivi*. Questi ultimi introducono una struttura nel modellare la funzione f, cercando di non irrigidire troppo il modello e di non farlo perdere di flessibilità.

I *modelli additivi* presuppongono che la media della risposta possa essere scomposta in una somma di funzioni univariate sui predittori. Da Arnold et al. (2019):

$$\mathbf{E}(y_i|x_i) = \alpha + \sum_{j=1}^{p} f_j(x_{i,j}), \tag{3.1}$$

dove $x_{i,j}$ è la *j-esima* variabile per l'*i-esima* unità.

Quindi la previsione del vettore f è data da:

$$\hat{f}(x) = \hat{f}(x_1, \ldots, x_p) = \alpha + \sum_{j=1}^{p} \hat{f}_j(x_j). \tag{3.2}$$

Dalla scrittura dei *modelli additivi* si può notare che essi sono una naturale estensione del modello lineare.

La struttura esterna è quella di una normale regressione lineare in cui non ci sono interazioni tra le variabili, proprio per questo il modello è di tipo "additivo": l'effetto della variazione di una qualsiasi covariata è indipendente dai valori delle altre, perciò i punti nella *j-esima* dimensione sono gli unici dati necessari per stimare f_j.

La struttura interna invece cattura la non linearità in una dimensione, infatti le funzioni $f_1, \ldots, f_p$ sono funzioni di regressione nonprarametrica in una variabile.

Il modello così descritto gode di un problema di non identificabilità in quanto, per esempio, se si aggiungesse una costante fissa ad f_1 e si sottraesse la stessa quantità da f_2 la relazione con y rimarrebbe invariata: due differenti valori dei parametri darebbero luogo alla stessa funzione.

Dunque per risolvere questo problema ogni funzione f_j viene centrata sullo zero nei dati di training, cioè:

$$\sum_{i=1}^{n} \hat{f}_j(x_{i,j}) = 0 \ .$$

Per stimare le funzioni f_j dai dati di training si utilizza l'algoritmo di *backfitting*, descritto dall'Algoritmo 3.1, ovvero una procedura iterativa basata su un metodo di stima nonparametrica di funzioni a una variabile. Da Azzalini & Scarpa (2012):

Algoritmo 3.1 *Backfitting*

1. Inizializzazione: $\hat{\alpha} = \sum_{i=1}^{n} y_i/n$, $\qquad f_j = 0 \ \ \forall j$
2. Stima: ciclo fino a stabilizzazione per $j = 1, 2, \ldots, p, \ldots \ \ldots, 1, 2, \ldots, p$
   - $\hat{f}_{i,j} = S\left[y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_{i,j}\right]$ dove S è uno stimatore nonparametrico
   - $\hat{f}_{i,j} = \hat{f}_{i,j} - \frac{1}{n}\sum_i \hat{f}_{i,j}$ così da centrare i valori sullo zero.

Il metodo scelto per la stima nonparametrica non è determinante, l'importante è che lo stimatore S sia un *lisciatore lineare* sia del tipo 1.4; per esempio è possibile scegliere come possibile metodo i *processi gaussiani* o la *regressione locale*.

3.3 Modelli Additivi Generalizzati

Fino ad ora il focus sulla relazione tra input ed output è stato su problemi di regressione, in cui la risposta assume valori continui.

Apportando opportune trasformazioni è possibile anche estendere alcune considerazioni oggetto di studio al caso in cui la risposta sia di tipo discreto, concentrandosi dunque su un problema di classificazione.

In particolare, quando la risposta è di tipo dicotomico si può applicare una naturale generalizzazione del *modello additivo*.

3.3.1 Regressione Logistica

Quando lo studio della relazione tra le variabili ha natura dicotomica nella risposta applicare il modello di regressione lineare semplice produrrebbe scarse previsioni.

Una semplice estensione di tale modello è il *modello di regressione logistica* in cui la variabile di risposta y per un dato soggetto è una variabile casuale di Bernoulli, che assume valore 1 se l'evento si è verificato con successo, 0 altrimenti e la cui probabilità di successo, chiamata $\pi(x)$, dipende dalle covariate.

Considerato un set di covariate $(x_1, \ldots, x_p)$ e indicando con $\eta(x)$, detto *predittore lineare*, la combinazione di covariate lineare nei parametri:

$$\eta(x) = x^\intercal \beta = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p \ , \tag{3.3}$$

la *regressione logistica* appartiene alla famiglia dei *GLM*, ovvero i *modelli lineari generalizzati*, in cui la relazione tra le variabili esplicative e la risposta è legata tramite un'opportuna funzione $g(\cdot)$, detta *funzione legame*, ed è del tipo:

$$g\left[\mathbf{E}\{y|x_1, \ldots, x_n\}\right] = \eta(x) \ . \tag{3.4}$$

Nel caso in questione la *funzione legame* corrisponde alla funzione *logit*:

$$g(\pi) = logit(\pi) = \log \frac{\pi}{1-\pi} \tag{3.5}$$

Dall' inverso della funzione *logit* si ottiene il *modello di regressione logistica*:

$$\pi(x) = \frac{\exp(\eta(x))}{1 + \exp(\eta(x))} \ , \tag{3.6}$$

in cui la probabilità di successo $\pi(x)$ dipende dal vettore x tramite il *predittore lineare* (3.3).

Attraverso questa trasformazione si è riusciti a mettere in relazione le variabili ottenendo un output la cui risposta appartiene all'intervallo unitario $(0, 1)$.

3.3.2 GAM

I *modelli additivi* possono essere estesi alla classe dei *GAM*, ovvero i *modelli additivi generalizzati*, in cui la relazione tra la variabile risposta e le esplicative è regolata mediante la *funzione legame* $g(\cdot)$ e sono del tipo:

$$g(\mathbf{E}\left[y_i|x_i\right]) = \alpha + \sum_j f_j(x_{i,j}) \ . \tag{3.7}$$

I *GAM* possono essere usati per problemi di classificazione scegliendo un'adeguata distribuzione della variabile risposta e della *funzione legame*.

Nel caso di risposta dicotomica, $g(\cdot)$ è normalmente assunta come la funzione *logit* (3.5) e il modello che si ottiene è del tipo:

$$\text{logit}(\pi) = \alpha + \sum_{j=1}^{p} f_j(x_j) \ , \tag{3.8}$$

dove π rappresenta la probabilità di successo.

Per stimare le funzioni ad una variabile per il *modello additivo generalizzato*, da Hastie & Tibshirani (1987), si utilizza una combinazione dell'algoritmo di *backfitting* con il criterio dei *minimi quadrati pesati iterativi*, applicati al caso dei *GLM*. Questa procedura è chiamata *local scoring* ed è descritta dall'Algoritmo 3.2:

Algoritmo 3.2 *Local scoring per il modello additivo logistico*

1. Inizializzazione: $\hat{\alpha} = \text{logit}(y) \qquad \hat{f}_j = 0 \ \ \forall j$
2. Stima al passo *(m)*:
   - $\hat{\eta}^{(m)}(x_i) = \hat{\beta_0} + \sum_{j=1}^{p} \hat{f}_j^{(m)}(x_{i,j})$
   - $\hat{p}_i = \frac{1}{1+\exp(-\hat{\eta}^{(m)}(x_i))}$
   - $z_i = \hat{\eta}^{(m)}(x_i) + \frac{y_i - \hat{p}_i}{\hat{p}_i(1-\hat{p}_i)}$
   - $w_i = \hat{p}_i(1-\hat{p}_i)$
3. Si ottiene la nuova stima di $\hat{f}_j^{(m+1)}$ e di $\hat{\alpha}$ applicando l'algoritmo di *backfitting* alla pseudo-risposta z con covariate X e osservazioni pesate per W, fino a convergenza.

È possibile combinare l'utilizzo dei *GAM* con la stima delle funzioni ad una variabile tramite *processi gaussiani pesati*, essendo questi ultimi dei *lisciatori lineari*. È necessario però svolgere dei passaggi per trovare la *matrice di lisciamento* pesata alla matrice dei pesi W.

È da sottolineare che in questo caso si considerano i *processi gaussiani* come dei generici *lisciatori lineari* di tipo frequentista, andando invece a perdere la parte inferenziale che li caratterizza.

3.4 Processi gaussiani pesati

Di seguito viene presentata la dimostrazione per la stima della *funzione media* per i *processi gaussiani pesati*. La seguente è stata svolta passando attraverso una serie di step:

- Inizialmente si deriva il coefficiente di previsione per la *ridge regression* ed in seguito lo si deriva per la stessa pesata da una matrice dei pesi W.
- Paragonando poi l'interpretazione tra la media nella *ridge regression* e la media a posteriori nei *processi gaussiani pesati*, dato che entrambi presentano un termine di penalità nella funzione di log-verosimglianza, viene ricavata la funzione a posteri della media per i *processi gaussiani pesati* nella *weight-space view*.
- Come ultimo step, applicando il *matrix inversion lemma*, si presenta la stima del coefficiente dal punto di vista della *function space view*, così da poter applicare i *processi gaussiani pesati* più agevolmente da un punto di vista computazionale all'algoritmo di *local scoring*.

3.4.1 Ridge regression

La *ridge regression* è un metodo di regolarizzazione utilizzato per risolvere il problema della *maledizione della dimensionalità* quando sono presenti molte covariate.

Seguendo questo approccio tutte le variabili vengono mantenute, ma i coefficienti β di regressione vengono sottoposti ad un vincolo che li "restringe" verso la media, senza penalizzare l'intercetta.

Il problema di ottimizzazione che risolve la *ridge regression* è, da Arnold et al. (2019):

$$\min_{\beta} ||y - X\beta||^2 + \lambda||\beta||^2 \tag{3.9}$$

La funzione da minimizzare sopra citata è data dalla somma di due quantità: la prima è la somma dei residui al quadrato del modello lineare standard e la seconda è il termine di penalità λ che varia tra l'intervallo $(0, 1)$ sui coefficienti β.

Il coefficiente β nella *ridge regression* è pari a $\hat{\beta}_{ridge} = (X^T X + \lambda I_p)^{-1} X^T y$.

Dimostrazione.

$$
\begin{aligned}
&S_{ridge}(\beta) = (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta \\
&\hat{\beta}_{ridge} = \mathtt{argmin}(S_{ridge}(\beta)) \\
&\frac{\partial}{\partial \beta}(S_{ridge}(\beta)) = -(2y^T X)^T + 2X^T X\beta + 2\lambda\beta \\
&\frac{\partial}{\partial \beta}(S_{ridge}(\beta)) = 0 \Leftrightarrow \hat{\beta}_{ridge} = (X^T X + \lambda I_p)^{-1} X^T y
\end{aligned}
$$

□

Nella *regressione rigida* il coefficiente λ governa il *compromesso distorsione-varianza* già discusso nel primo capitolo: rispetto a uno stimatore standard ai minimi quadrati, avendo ridotto i coefficienti con il termine λ, più esso aumenta più le soluzioni avranno una varianza più bassa e una distorsione più accentuata verso lo zero.

Interpretazione bayesiana della ridge regression

La *ridge regression* può essere interpretata come uno stimatore bayesiano per la distribuzione a priori dello stimatore β:

$$\beta_j \sim \mathcal{N}(0, \sigma^2 \lambda^{-1}) \tag{3.10}$$

e le funzioni di verosimiglianza implicano l'assegnazione di variabili casuali indipendenti e identicamente distribuite ad una Normale:

$$y_i | \beta \sim \mathcal{N}(x_i^T \beta, \sigma_n^2) \ . \tag{3.11}$$

Dalle distribuzioni (3.10) e (3.11) è possibile applicare il *Teorema di Bayes* (2.3) per trovare la funzione di densità $f(\beta|y)$. Da Arnold et al. (2019):

$$\begin{aligned}
f(\beta|y) &\sim \prod_{j=1}^{p} \exp\left\{\frac{-\lambda}{2\sigma^2}\beta_j^2\right\} \prod_{i=1}^{n} \exp\left\{\frac{-1}{2\sigma^2}(y_i - x_i^\intercal\beta)^2\right\} \\
&= \exp\left\{\frac{-1}{2\sigma^2}(|y - X\beta|^2 + \lambda|\beta|^2)\right\} \\
&= \exp\left\{\frac{-1}{2\sigma^2}(y^\intercal y - \beta^\intercal X^\intercal X y - y^\intercal X\beta + \beta^\intercal X^\intercal X\beta + \lambda\beta^\intercal\beta)\right\} \\
&= \exp\left\{\frac{-1}{2\sigma^2}(y^\intercal y - \beta^\intercal(X^\intercal X\lambda I_p)\hat{\beta}_\lambda - \hat{\beta}_\lambda^\intercal(X^\intercal X + \lambda I_p)\beta + \beta^\intercal(X^\intercal X + \lambda I_p)\beta)\right\} \\
&= \exp\left\{\frac{-1}{2\sigma^2}(y^\intercal y - y^\intercal X(X^\intercal X + \lambda I_p)^{-1}X^\intercal y + \left[\beta - \hat{\beta}_\lambda\right]^\intercal (X^\intercal X + \lambda I_p)\left[\beta - \hat{\beta}_\lambda\right])\right\} \\
&\Rightarrow f(\beta|y) \sim \exp\left\{\frac{-1}{2\sigma^2}\left[\beta - \hat{\beta}_\lambda\right]^\intercal (X^\intercal X + \lambda I_p)\left[\beta - \hat{\beta}_\lambda\right]\right\},
\end{aligned} \tag{3.12}$$

dove il termine $\hat{\beta}_\lambda$ indica lo stimatore *ridge* che introduce una penalità λ; perciò lo stimatore β ha distribuzione: $\beta \sim \mathcal{N}(\hat{\beta}_\lambda, \sigma^2(X^\intercal X + \lambda I_p)^{-1})$.

Ridge regression pesata

Se si pesassero le osservazioni per una matrice W dei pesi, lo stimatore β per la *ridge regression* diventa:

$$\hat{\beta}_{\text{ridge,w}} = (X^\intercal W^{-1} X + \lambda I_p)^{-1} X^\intercal W^{-1} y \tag{3.13}$$

con W^{-1} matrice inversa dei pesi tale che: $I = W^{-1}W$.

Dimostrazione.

$$\begin{aligned}
&S_{\text{ridge,w}}(\beta) = (y - X\beta)^\intercal W^{-1}(y - X\beta) + \lambda\beta^\intercal\beta \\
&\hat{\beta}_{\text{ridge,w}} = \texttt{argmin}(S_{\text{ridge,w}}(\beta)) \\
&\frac{\partial}{\partial\beta}(S_{\text{ridge,w}}(\beta)) = -2(y^\intercal W^{-1}X)^\intercal + 2X^\intercal W^{-1}X\beta + 2\lambda\beta \\
&\frac{\partial}{\partial\beta}(S_{\text{ridge,w}}(\beta)) = 0 \Leftrightarrow \hat{\beta}_{\text{ridge,w}} = (X^\intercal W^{-1}X + \lambda I_p)^{-1}X^\intercal W^{-1}y.
\end{aligned}$$

□

Dalla scrittura dei coefficiente (3.13) si può notare un'analogia con il modello di regressione lineare con eteroschedasticità, dove la matrice W dei pesi va a rimuovere l'ipotesi che tutte le osservazioni abbiano la stessa varianza.

3.4.2 Processi gaussiani pesati - Weight space view

In via analoga ai passi affrontati per ricavare lo stimatore β nel caso della *ridge regression* si ricava il coefficiente β per i *processi gaussiani* nella *weight-space view* in quanto l'adattamento con i pesi in questo spazio risulta essere facilmente ricavabile.

Coefficiente β per i processi gaussiani - weight space

Per i *processi gaussiani*, in analogia alla scrittura (2.5), la stima del coefficiente β è data da: $\hat{\beta}_{GP} = \left(\frac{1}{2\sigma^2}X^TX + \Sigma^{-1}\right)^{-1} \frac{1}{2\sigma^2}X^Ty$.

Dimostrazione.

$$
\begin{aligned}
&S_{GP}(\beta) = \sigma_n^{-2}(y - X\beta)^T(y - X\beta) + \beta^T\Sigma^{-1}\beta \\
&\hat{\beta}_{GP} = \mathrm{argmin}(S_{GP}(\beta)) \\
&\frac{\partial}{\partial\beta}(S_{GP}(\beta)) = -2\sigma_n^{-2}X^Ty + 2\sigma_n^{-2}X^TX\beta + 2\Sigma^{-1}\beta \\
&\frac{\partial}{\partial\beta}(S_{GP}(\beta)) = 0 \Leftrightarrow \hat{\beta}_{GP} = \left(\frac{1}{2\sigma^2}X^TX + \Sigma^{-1}\right)^{-1} \frac{1}{2\sigma^2}X^Ty \, .
\end{aligned}
$$

□

Coefficiente β per i processi gaussiani pesati- weight space

Pesando le osservazioni per la matrice dei pesi W si ottiene che la stima della media nei *processi gaussiani pesati* è pari a:

$$
\hat{\beta}_{GP,w} = \left(\frac{1}{2\sigma^2}X^TW^{-1}X + \Sigma^{-1}\right)^{-1} \frac{1}{2\sigma^2}X^TW^{-1}y \, . \tag{3.14}
$$

Dimostrazione.

$$
\begin{aligned}
&S_{GP,w}(\beta) = \sigma_n^{-2}(y - X\beta)^T W^{-1}(y - X\beta) + \beta^T \Sigma^{-1}\beta \\
&\hat{\beta}_{GP,w} = \operatorname{argmin}(S_{GP,w}(\beta)) \\
&\frac{\partial}{\partial\beta}(S_{GP,w}(\beta)) = -2\sigma_n^{-2}X^T W^{-1}y + 2\sigma_n^{-2}X^T W^{-1}X\beta + 2\Sigma^{-1}\beta \\
&\frac{\partial}{\partial\beta}(S_{GP,w}(\beta)) = 0 \Leftrightarrow \hat{\beta}_{GP,w} = \left(\frac{1}{2\sigma^2}X^T W^{-1}X + \Sigma^{-1}\right)^{-1} \frac{1}{2\sigma^2}X^T W^{-1}y \ .
\end{aligned}
$$

□

3.4.3 Processi gaussiani pesati - Function space view

La formulazione (3.14) già permette di poter applicare l'algoritmo di *local scoring*, ma per ragioni computazionali già discusse al capitolo precedente è più conveniente applicare quest'ultimo allo *spazio delle funzioni*.

È possibile dimostrare che la *funzione media* nei *processi gaussiani pesati* nello *spazio delle funzioni* è pari a:

$$k_*^T(\sigma_n^2 I + W^{-1}K)^{-1}y \tag{3.15}$$

e la *funzione kernel* nello *spazio delle funzioni* è pari a:

$$k(x_*, x_*) - k_*^T(K + \sigma_n^2 W)^{-1}k_* \ . \tag{3.16}$$

Partendo dalla (3.14) la previsione per la media della risposta è data da:

$$
\begin{aligned}
&\hat{\beta}_{GP,w} = \left(\frac{1}{2\sigma^2}X^T W^{-1}X + \Sigma^{-1}\right)^{-1} \frac{1}{2\sigma^2}X^T W^{-1}y = \\
&= \frac{1}{\sigma_n^2}x_*^T A_w^{-1}X^T y \ ,
\end{aligned}
$$

$$\text{con } A_w = \tfrac{1}{\sigma_n^2}X^T W^{-1}X + \Sigma_p^{-1} \ .$$

Applicando il *Matrix inversion lemma* (2.17) alla matrice A_w^{-1} si ottiene:

$$
\begin{aligned}
A_w^{-1} &= \left(\frac{1}{\sigma_n^2}X^T W^{-1}X + \Sigma_p^{-1}\right)^{-1} = \\
&= \Sigma_p - \Sigma_p X^T(\sigma_n^2 W + X\Sigma_p X^T)^{-1}X\Sigma_p
\end{aligned}
\tag{3.17}
$$

con $Z = \Sigma_p^{-1}$, $U = X^\intercal$, $D = \frac{1}{\sigma_n^2} W^{-1}$ e $V^\intercal = X$.

Media dei processi gaussiani pesati nella function-space view

Dimostrazione. Equivalenza delle formulazioni in media.

$$\begin{aligned}
&\frac{1}{\sigma_n^2} x_*^\intercal A_w^{-1} X^\intercal y = \\
&= \frac{1}{\sigma_n^2} x_*^\intercal \left[\Sigma_p - \Sigma_p X^\intercal (\sigma_n^2 W + X \Sigma_p X^\intercal)^{-1} X \Sigma_p \right] X^\intercal y = \\
&= \frac{1}{\sigma_n^2} x_*^\intercal \Sigma_p X^\intercal \left[I - (\sigma_n^2 W + X \Sigma_p X^\intercal)^{-1} X \Sigma_p X^\intercal \right] y = \\
&= \frac{1}{\sigma_n^2} x_*^\intercal \Sigma_p X^\intercal \left[I - \left(W + \frac{X \Sigma_p X^\intercal}{\sigma_n^2} \right)^{-1} \frac{X \Sigma_p X^\intercal}{\sigma_n^2} \right] y =
\end{aligned}$$

Sostituendo $Q = \frac{X \Sigma_p X^\intercal}{\sigma_n^2}$ nell'espressione si ha:

$$\frac{1}{\sigma_n^2} x_*^\intercal \Sigma_p X^\intercal \left[I - (W + Q)^{-1} Q \right] y$$

Applicando il *Matrix inversion lemma* (2.17) al termine tra parentesi si ottiene:

$$\begin{aligned}
I - (W + Q)^{-1} Q &= (I + W^{-1} Q)^{-1} = \\
&= \left(I + W^{-1} X \frac{X \Sigma_p X^\intercal}{\sigma_n^2} \right)^{-1}
\end{aligned}$$

dove $Z = U = I$, $D^{-1} = W$ e $V^\intercal = Q$.
Sostituendo questo risultato all'espressione si ottiene:

$$\begin{aligned}
&\frac{1}{\sigma_n^2} x_*^\intercal \Sigma_p X^\intercal \left[I + W^{-1} \frac{X \Sigma_p X^\intercal}{\sigma_n^2} \right]^{-1} y = \\
&= x_*^\intercal \Sigma_p X^\intercal (\sigma_n^2 I + W^{-1} X \Sigma_p X^\intercal)^{-1} y = \\
&= k_*^\intercal (\sigma_n^2 I + W^{-1} K)^{-1} y
\end{aligned}$$

□

Varianza dei processi gaussiani pesati nela function-space view

Dimostrazione. Equivalenza delle formulazioni in varianza.

$$
\begin{aligned}
& x_*^\intercal A_w^{-1} x_* = \\
& = x_*^\intercal \left[\Sigma_p - \Sigma_p X^\intercal (\sigma_n^2 W + X \Sigma_p X^\intercal)^{-1} X \Sigma_p \right] x_* = \\
& = x_*^\intercal \Sigma_p x_* - x_*^\intercal \Sigma_p X^\intercal (\sigma_n^2 W + X \Sigma_p X^\intercal)^{-1} X \Sigma_p x_* = \\
& = k(x_*, x_*) - k_*^\intercal (K + \sigma_n^2 W)^{-1} k_*
\end{aligned}
$$

□

Capitolo 4

Applicazione: nycflights13

L'obiettivo della seguente applicazione è di prevedere se un volo proveniente da New York City arrivi o meno in ritardo, considerati 30 minuti come soglia per stabilire se un aereo sia in ritardo oppure no.

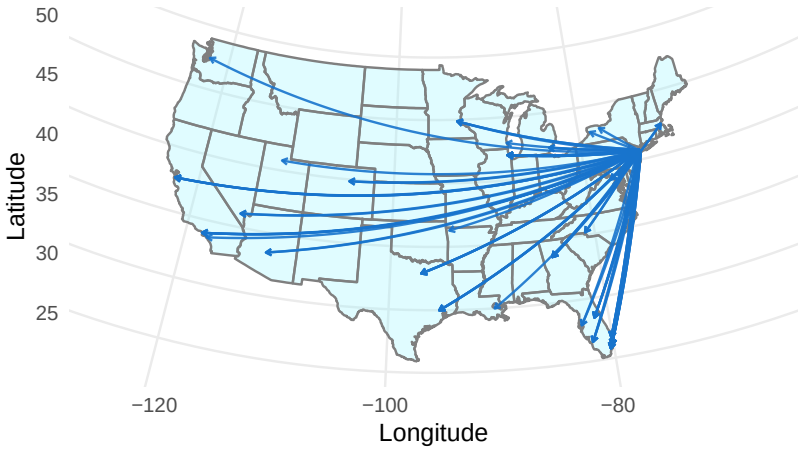


Figura 4.1: Rappresentazione grafica dei primi 80 voli partiti da NYC nel 2013.

L'applicazione è svolta seguendo l'esempio di Arnold et al. (2019) in cui viene applicato il *modello additivo generalizzato* con risposta binomiale in combinazione alla funzione di lisciamento a una variabile di tipo *spline*.

In questo esempio invece i *processi gaussiani* sono utilizzati per stimare le funzioni univariate applicati ai *modelli additivi generalizzati* attraverso l'algoritmo di *local scoring* per ottenere una risposta di tipo dicotomico.

4.1 Descrizione dei dati

I dati a disposizione sono stati ricavati dal pacchetto *nycflights13* presente in *R*. Quest'ultimo contiene tutti i voli in partenza dagli aeroporti di New York (cioè JFK, LGA o EWR) nel 2013 e metadati su compagnie aeree, aeroporti, meteo e aerei.

Il pacchetto contiene al suo interno 5 dataset in formato tibble, dalla descrizione di Wickham & RStudio (2021):

- *airlines*: dataset che contiene i metadati sui nomi delle compagnie aeree;
- *flights*: dataset che contiene le informazioni in tempo reale su tutti i voli in partenza da NYC nel 2013;
- *planes*: dataset che contiene i metadati relativi a tutti i numeri di coda degli aerei presenti nel registro aeronautico della FAA;
- *weather*: dataset che contiene dati meteorologici orari per gli aeroporti di LGA, JFK e EWR;
- *airports*: dataset che contiene i nomi delle compagnie aeree dai loro codici.

Partendo dai dataset *flights* e *weather* sono state svolte delle operazioni preliminari per creare un dataset ad hoc contente tutte le covariate di interesse per l'analisi.

Le variabili prese in considerazione per la creazione del dataset ***flights-weather*** sono state:

- *year, month, day*: variabili provenienti sia dal dataset *flights* sia dal dataset *weather* che rappresentano la data della partenza;
- *origin*: variabile proveniente sia dal dataset *flights* sia dal dataset *weather* che descrive l'aeroporto di origine;
- *dep-delay, arr-delay*: variabili provenienti dal dataset *flights* che descrivono i ritardi alla partenza e all'arrivo in minuti. I tempi negativi rappresentano partenze/arrivi anticipati;
- *dest*: variabile proveniente dal dataset *flights* che rappresenta l'aeroporto di destinazione;

- *distance*: variabile proveniente dal dataset *flights* che indica la distanza tra i due aeroporti in miglia;

- *visib*: variabile proveniente dal dataset *weather* che rappresenta la visibilità in miglia.

Operazioni preliminari con tidyverse

Per creare il dataset ***flights-weather*** è stato necessario usare la libreria *tidyverse* contenuta nel pacchetto *R* per poter manipolare i dati originali.

Inizialmente sono stati uniti i due tibble *flights* e *weather* mediante le variabili in comune *origin*, *year*, *month*, *day* e *hour*.

Successivamente è stata creata la variabile dicotomica ***delay***, data dalla somma di *dep-delay* e *arr-delay*.

È possibile notare visivamente dalla Figura 4.2, come era prevedibile, una correlazione positiva tra *dep-delay* e *arr-delay*; dunque sintetizzando la variabile *delay* con la somma del ritardo di andata e di arrivo non si perde molta informazione.

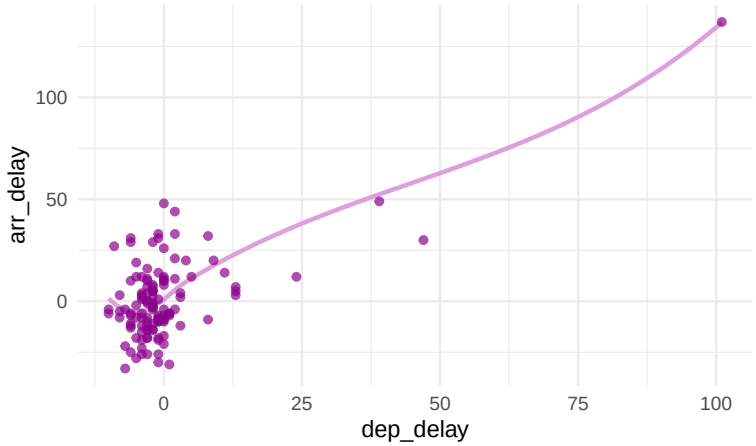


Figura 4.2: Scatterplot di *arr-delay* e *dep-delay*

È stata poi creata la variabile ***weekday*** che rappresenta il giorno della settimana in cui l'aereo è partito, ricavato mediante la funzione *weekdays*; questa variabile è stata poi ricodificata con degli interi che rappresentano il giorno della settimana.

Infine è stata creata la variabile ***volume*** che rappresenta la frequenza assoluta della variabile *dest*.

Sono state poi selezionate solo le variabili di interesse per il problema, sono state rimosse le righe contenenti dei valori mancanti ed è stato creato un sotto-dataset campionando casualmente 2500 osservazioni dal dataset *flights-weather*.

4.1.1 Variabili

Nella seguente tabella 4.1 sono elencate le variabili utilizzate nell'applicazione con una breve descrizione.

Tabella 4.1

Variabile	Descrizione
distance	Distanza espressa in miglia tra l'aeroporto di partenza e di arrivo
hour	Ora di partenza dell'aereo
visib	Visibilità in miglia
volume	Frequenza assoluta del numero di voli in un determinato aeroporto di arrivo
weekday	Giorno della settimana in cui è partito l'aereo
delay	Variabile risposta dicotomica: 0 se l'aereo non era in ritardo, 1 altrimenti

4.2 Applicazione del modello additivo generalizzato

Per questa applicazione è stata creata la funzione *beta.w* in *R* che rappresenta la previsione della *funzione media* per i *processi gaussiani pesati* corrispondente alla (3.15).

È stato poi implementato l'algoritmo di *local scoring* per poter applicare l'algoritmo di *backfitting* pesato.

Questa procedura è stata ripetuta su una griglia di combinazioni diverse di parametri così da poter trovare, attraverso l'algoritmo *leave-one-out cross-validation*, gli iperparametri "migliori" per il modello.

Il risultato ottenuto è dato dalla combinazione dei seguenti parametri:

- $\sigma_f^2 = 600$

- $l = 2$
- $\sigma_n^2 = 10$

Avendo ottenuto a questo punto una stima delle f_j si è ricavata la variabile risposta dicotomica data dalla (3.2).

Successivamente è stato confrontato il metodo sopra descritto con il *modello additivo generlizzato* a risposta binaria applicato alla funzione a una variabile stimata tramite *regressione locale*, mediante la funzione *gam* contenuta nel pacchetto *mgvc* di *R*.

4.3 Risultati

Nella tabella 4.2 sono presentate alcune righe del dataframe ***flights-weather*** contente le variabili utilizzate per stimare il modello, la variabile risposta *delay* e le stime ottenute dall'applicazione dei due modelli descritti nella sezione precedente, rispettivamente *delay-gp* e *delay-loess*.

Nella Figura 4.3 sono invece presenti rispettivamente le *matrici di confusione* relative alla stima del modello additivo tramite *processi gaussiani* e tramite *regressione locale* con il relativo valore di *accuratezza*.

Nella *matrice di confusione* è possibile osservare una rappresentazione sintetica dell'accuratezza della classificazione; ogni colonna della matrice rappresenta i valori predetti, mentre ogni riga rappresenta i valori reali. In questo esempio la matrice di confusione è stata calcolata esclusivamente per i dati di training.

L'*accuratezza* è un indicatore della percentuale di osservazioni classificate correttamente dal modello. Si può ricavare calcolando la traccia della *matrice di confusione* e dividendo questo risultato per il totale delle osservazioni.

4.4 Conclusioni

Analizzando i risultati è possibile osservare come gli errori più frequenti in entrambi i modelli siano dovuti all'errata previsione dei voli in ritardo.

Tabella 4.2

distance	hour	visib	volume	weekday	delay	delay-gp	delay-loess
301	22	6.00	4657	6	1	0	1
762	13	1.75.	17148	2	1	0	0
187	22	10.00	15434	7	0	0	0
1400	13	10.00	7166	3	0	0	0
1521	9	10.00	2422	7	0	0	0
2475	7	10.00	16087	7	0	0	0
488	10	10.00	9347	1	0	0	0
463	14	10.00	3511	6	0	0	0
997	15	10.00	7429	4	1	1	0
187	6	10.00	15434	5	0	0	0
963	10	5.00	1784	3	0	0	0
2227	18	1.50	5971	5	1	0	1
1096	18	0.50	11681	1	0	0	1
1065	18	10.00	11985	3	0	0	0
143	16	7.00	436	2	0	1	0
719	15	3.00	17219	5	1	0	0
733	16	10.00	17219	7	0	0	0
746	15	10.00	17148	3	0	0	0
1096	15	10.00	11681	6	0	0	0
1569	16	10.00	682	3	0	0	0
589	16	10.00	3924	1	1	0	0
544	17	10.00	14001	4	1	0	0
2475	8	3.00	16087	5	0	0	0
2475	20	10.00	16087	6	0	0	0
509	7	10.00	9347	4	0	0	0
1089	16	10.00	11681	6	0	0	0
228	12	10.00	5684	1	0	0	0
1391	16	10.00	8703	1	1	1	0
1389	15	10.00	8703	3	1	0	0

Refrence \ Prediction	0	1
0	1914	481
1	37	68

accuracy= 0.7928

(a) Processi gaussiani

Refrence \ Prediction	0	1
0	1926	520
1	25	29

accuracy= 0.782

(b) Regressione locale

Figura 4.3: Matrici di confusione con relativa accuratezza

Questo problema può essere una conseguenza del fatto che analizzando la proporzione tra voli in orario e in ritardo ci si trova in una situazione di risposta con classi sbilanciate, infatti il 78,4% di voli sono atterrati in orario e solamente il 21,96% sono atterrati in ritardo.

Essendo la maggior parte dei voli classificati in orario il modello con fatica riesce a distinguere quelli che sono stati realmente in ritardo.

Ponendo l'attenzione invece sui due modelli stimati, entrambi presentano risultati simili, perciò l'utilizzo dei *processi gaussiani* come *lisciatori lineari* applicati al *modello additivo generalizzato* può essere considerato una valida alternativa al classico utilizzo della *regressione locale* o della stima *spline*.

Appendice A

Codice R

```
####################
#### CAPITOLO 2 ####
####################

library(tidyverse)
library(gridExtra)
library(fields)
library(MASS)

#### GENERAZIONE FUNZIONE A PRIORI
set.seed(123)

prior <- function(grid, l) {
  mu <- rep(0, length(grid))
  D <- as.matrix(dist(grid))
  K <- exp(-1  * (D^2) /(2*(l^2))) + 1e-6 * diag(length(grid))
  A <- t(chol(K))
  Z <- rnorm(length(grid))
  X <- mu + A %*% Z
  dati <- tibble(X=X, grid=grid)
  list(priori=dati)
}

grid <- seq(from = -5, to = 5, length = 50)
```

```
f1 <- prior(grid, 0.5)
f2 <- prior(grid, 1)
f3 <- prior(grid, 3)
f4 <- prior(grid, 6)

(gg1 <- ggplot(data=f1$priori, aes(x=grid, y=X))+
    geom_line(color="darkorchid")+
    labs(x="X",y="f(X)")+ theme_minimal())

### APPLICAZIONE jaws.txt
deer <- deer[order(deer[,1], decreasing=F),]
X <- as.matrix(deer$age)
y <- as.matrix(deer$bone, ncol = 1)

#griglia di parametri
l1.grid <- c(100, 300, 500, 700, 1000, 1200, 2000)
l2.grid <- c(seq(1,10, by=1), seq(11,30, by=2), seq(30,200, by=10))
sigma2_n.grid <- c(10,50, 80,100, 120, 150, 170, 200)
Hp <- expand.grid(l1.grid, l2.grid, sigma2_n.grid)
colnames(Hp) <- c("l1", "l2", "sigma2_n")

#Funzione a posteriori con SQUARED EXPONENTIAL KERNEL
posterior_sq.exp<- function(X, y, l1, l2, sigma2_n) {
  n <- length(X)
  K <- l1 * exp(- rdist(X, X)^2 / (2 * l2^2))
  S <- K %*% solve(K + sigma2_n * as.matrix(diag(n)))
  #S matrice di smoothing, quando X = X* --> unica funzione Kernel
  L <- t(chol(K + sigma2_n * as.matrix(diag(n))))
  alpha <- solve(t(L), solve(L, y))
  fstar_mean <- t(K) %*% alpha
  v <- solve(L, K)
  V <- K - crossprod(v)
  m_lk <- (-1/2)*t(y) %*% alpha-sum(diag(L)) - (n/2)*log(2*pi)

  list(mean = fstar_mean, V = V, loglik = m_lk, S=S)
}
```

```
#Funzione a posteriori con EXPONENTIAL KERNEL
posterior_exp <- function(X, y, l1, l2, sigma2_n) {
  n <- length(X)
  K <- l1 * exp(- abs(rdist(X, X)) / (2 * l2^2))
  S <- K %*% solve(K + sigma2_n * as.matrix(diag(n)))
  L <- t(chol(K + sigma2_n * as.matrix(diag(n))))
  alpha <- solve(t(L), solve(L, y))
  fstar_mean <- t(K) %*% alpha
  v <- solve(L, K)
  V <- K - crossprod(v)
  m_lk <- (-1/2)*t(y) %*% alpha-sum(diag(L))-(n/2)*log(2*pi)
  list(mean = fstar_mean, V = V, loglik = m_lk, S=S)

}

# CRITERI DI SCELTA DEL MODELLO
#metodo cross validation applicato a SQUARED EXPONENTIAL KERNEL
cv_sq.exp <- function (X, y, grid) {
  R <- rep(NA, nrow(grid))
  r <- rep(NA, length(y))

  for(i in 1:nrow(grid)) {
    l1 <- grid[i,1]
    l2 <- grid[i, 2]
    sigma2_n <- grid[i,3]
    f <- posterior_sq.exp(X, y, l1, l2, sigma2_n)

    for(j in 1:length(y)) {
      r[j] <- ((y[j,1] - f$mean[j]) / (1 -f$S[j,j]))^2
    }
    R[i] <- mean(r)
  }
  list(R=R, r=r)
}
```

```
c.v_sq.exp <- cv_sq.exp(X, y, Hp)
which.min(c.v_sq.exp$R)
Hp[which.min(c.v_sq.exp$R),] #valori dei parametri
post_c.v_sq.exp <- posterior_sq.exp(X,y,Hp[which.min(c.v_sq.exp$R),1],
        Hp[which.min(c.v_sq.exp$R),2], Hp[which.min(c.v_sq.exp$R),3])

#metodo cross validation applicato a EXPONENTIAL KERNEL
cv_exp <- function (X, y, grid) {
  R <- rep(NA, nrow(grid))
  r <- rep(NA, length(y))

  for(i in 1:nrow(grid)) {
    l1 <- grid[i,1]
    l2 <- grid[i, 2]
    sigma2_n <- grid[i,3]
    f <- posterior_exp(X, y, l1, l2, sigma2_n)
    r <- rep(NA, length(y))

    for(j in 1:length(y)) {
      r[j] <- ((y[j,1] - f$mean[j]) / (1 -f$S[j,j]))^2
    }
    R[i] <- mean(r)
  }
  list(R=R, r=r)
}

c.v_exp <- cv_exp(X, y, Hp)
Hp[which.min(c.v_exp$R),]
post_c.v_exp <- posterior_exp(X,y,Hp[which.min(c.v_exp$R),1],
      Hp[which.min(c.v_exp$R),2], Hp[which.min(c.v_exp$R),3])

cross_vld <- tibble(X=X,y=y,cv_sq.exp=post_c.v_sq.exp$mean,
                    cv_exp=post_c.v_exp$mean)

(ggplot(cross_vld, aes(X,y))+
    geom_point(color="midnightblue", alpha=.8)+
```

```
    geom_line(aes(X,cv_sq.exp), color="cornflowerblue",
              size=.9, alpha=.8)+
    labs(x="Age", y="Bone")+
    theme_minimal())

#Log-likelihood applicato a SQUARED EXPONENTIAL KERNEL
max.marg.lk_sq.exp <- function(a) {
  n <- length(X)
  K <- a[1] * exp(- rdist(X, X)^2 / (2 * a[2]^2))
  L <- t(chol(K + a[3] * as.matrix(diag(n))))
  alpha <- solve(t(L), solve(L, y))
  fstar_mean <- t(K) %*% alpha
  v <- solve(L, K)
  V <- K - crossprod(v)
  m_lk <- 1/2*t(y) %*% alpha-sum(diag(L))-(n/2)*log(2*pi)
}

max.ml_sq.exp <- nlminb(c(1000,15,170),max.marg.lk_sq.exp, lower = 1e-5)
c(max.ml_sq.exp$par[1], max.ml_sq.exp$par[2], max.ml_sq.exp$par[3])
post.ml_sq.exp <-  posterior_sq.exp(X, y, max.ml_sq.exp$par[1],
              max.ml_sq.exp$par[2] , max.ml_sq.exp$par[3])

# Log-likelihood applicato a EXPONENTIAL KERNEL
max.marg.lk_exp <- function(a) {
  n <- length(X)
  K <- a[1] * exp(- abs(rdist(X, X)) / (2 * a[2]^2))
  L <- t(chol(K + a[3] * as.matrix(diag(n))))
  alpha <- solve(t(L), solve(L, y))
  fstar_mean <- t(K) %*% alpha
  v <- solve(L, K)
  V <- K - crossprod(v)
  m_lk <- 1/2*t(y) %*% alpha-sum(diag(L))-(n/2)*log(2*pi)
}

max.ml_exp <- optim(c(1000, 15, 170), max.marg.lk_exp, lower = 1e-5,
                    method = "L-BFGS-B",  control=list(fnscale=-1))
```

```
c(max.ml_exp$par[1],max.ml_exp$par[2],max.ml_exp$par[3])
post.ml_exp <-  posterior_exp(X, y, max.ml_exp$par[1],
                        max.ml_exp$par[2], max.ml_exp$par[3])

mlk <- tibble(X=X,y=y,mlk_sq.exp=post.ml_sq.exp$mean,
            mlk_exp=post.ml_exp$mean)

(ggplot(mlk, aes(X,y))+
    geom_point(color="mediumpurple4", alpha=.8)+
    geom_line(aes(X,mlk_sq.exp),color="darkorchid1",size=.9,alpha=.7)+
    labs(x="Age", y="Bone")+
    theme_minimal())

#AIC e BIC applicati a SQUARED EXPONENTIAL KERNEL
crit.inf_sq.exp <- function (X, y, grid) {
  AIC <- rep(NA, nrow((grid)))
  BIC <- rep(NA, nrow((grid)))
  ll <- rep(NA, nrow((X)))
  sum.ll <- rep(NA, nrow((grid)))

  for (j in 1: nrow(grid)) {
    l1 <- grid[j,1]
    l2 <- grid[j, 2]
    sigma2_n <- grid[j,3]
    f <- posterior_sq.exp(X, y, l1, l2, sigma2_n)

    for(k in 1:length(ll)) {
      ll[k] <- log(dnorm(y[k], mean=f$mean[k], sd=sqrt(sigma2_n)))
    }
    sum.ll[j] <- sum(ll)
    gdl <- sum(diag(f$S))
    AIC[j] <- -2*sum.ll[j] + 2*gdl
    BIC[j] <- -2*sum.ll[j] + gdl*log(nrow(X))
    }
  list(AIC=AIC, BIC=BIC, loglik=sum.ll)
}
```

```
#AIC e BIC applicati a EXPONENTIAL KERNEL
crit.inf_exp <- function (X, y, grid) {
  AIC <- rep(NA, nrow((grid)))
  BIC <- rep(NA, nrow((grid)))
  ll <- rep(NA, nrow((X)))
  sum.ll <- rep(NA, nrow((grid)))

  for (j in 1: nrow(grid)) {
    l1 <- grid[j,1]
    l2 <- grid[j, 2]
    sigma2_n <- grid[j,3]
    f <- posterior_exp(X, y, l1, l2, sigma2_n)

    for(k in 1:length(ll)) {
      ll[k] <- log(dnorm(y[k], mean=f$mean[k], sd=sqrt(f$V[k,k])))
    }
    sum.ll[j] <- sum(ll)
    gdl <- sum(diag(f$S))
    AIC[j] <- -2*sum.ll[j] + 2*gdl
    BIC[j] <- -2*sum.ll[j] + gdl*log(nrow(X))
    }
  list(AIC=AIC, BIC=BIC, loglik=sum.ll)
}

f_sq.exp <- crit.inf_sq.exp(X, y, Hp)

which.min(f_sq.exp$AIC)
Hp[which.min(f_sq.exp$AIC),] #valori dei parametri per AIC
Hp[which.min(f_sq.exp$BIC),] #valori dei parametri per BIC
post.aic_sq.exp <- posterior_sq.exp(X, y, Hp[which.min(f_sq.exp$AIC),1],
      Hp[which.min(f_sq.exp$AIC),2], Hp[which.min(f_sq.exp$AIC),3])
post.bic_sq.exp <- posterior_sq.exp(X, y, Hp[which.min(f_sq.exp$BIC),1],
      Hp[which.min(f_sq.exp$BIC),2], Hp[which.min(f_sq.exp$BIC),3])

aic_bic <- tibble(X=X,y=y,aic=post.aic_sq.exp$mean,
```

```
                  bic=post.bic_sq.exp$mean)
colors <- c("AIC"="gold", "BIC"="chartreuse3")

(ggplot(aic_bic, aes(X,y))+
    geom_point(color="lightgoldenrod4", alpha=.8)+
    geom_line(aes(X,aic,color="AIC"), size=.6)+
    geom_line(aes(X,bic,color="BIC"),size=1.3,alpha=.7,linetype="dotted")+
    labs(x="Age",
         y="Bone",
         color="Criterio di scelta")+
    scale_color_manual(values = colors)+
    theme_minimal())

which.min(f_exp$AIC)
Hp[which.min(f_exp$AIC),]
Hp[which.min(f_exp$BIC),]
post.aic_exp <- posterior_exp(X, y, Hp[which.min(f_exp$AIC),1],
          Hp[which.min(f_exp$AIC),2], Hp[which.min(f_exp$AIC),3])
post.bic_exp <- posterior_exp(X, y, Hp[which.min(f_exp$BIC),1],
          Hp[which.min(f_exp$BIC),2], Hp[which.min(f_exp$BIC),3])


####################
#### CAPITOLO 4 ####
####################

library(nycflights13)
library(tidyverse)
library(fields)
library(gam)
library(dplyr)
library(mgcv)

# Grafico primi 80 voli partiti da NYC nel 2013
flights_nyc <- flights %>%
  inner_join(select(airports, origin = faa,
```

```
        origin_lat = lat, origin_lon = lon), by = "origin") %>%
  inner_join(select(airports, dest = faa,
        dest_lat = lat, dest_lon = lon), by = "dest")

flights_nyc %>%
  slice(1:80) %>%
  ggplot(aes(
    x = origin_lon, xend = dest_lon,
    y = origin_lat, yend = dest_lat
  )) +
  borders("state", fill="cadetblue1", alpha=.3) +
  geom_segment(arrow = arrow(length = unit(0.1, "cm")),
          color="dodgerblue3", alpha=.9) +
  coord_map("albers",at0=45.5,lat1=29.5) +
  #scale_fill_viridis_c("")+
  labs(y = "Latitude", x = "Longitude")+
  theme_minimal()

# Scatterplot arr_delay e dep_delay
flights %>%
  slice(1:120) %>%
  ggplot(aes(x=dep_delay, y=arr_delay)) +
  geom_smooth(colour="plum", alpha=1, se=F, method="loess")+
  geom_point(colour="darkmagenta", alpha=.7)+
  theme_minimal()

### APPLICAZIONE

#Creazione del dataframe FLIGHTS-WEATHER
#Join dei due tibble tramite le variabili in comune
flights_weather <-
  flights %>%
  inner_join(weather, by = c(
    "origin" = "origin",
    "year" = "year",
    "month" = "month",
```

```
    "day" = "day",
    "hour" = "hour"
  ))

#Variabile DELAY
flights_weather <- mutate(flights_weather,
          delay = ifelse(dep_delay+arr_delay>=30,1,0),
          data=as.Date(paste(year, month, day,sep="-"),"%Y-%m-%d"))

#Variabile WEEKDAY
flights_weather <- as.data.frame(flights_weather)
flights_weather$weekday <- weekdays(flights_weather$data)
flights_weather <- as_tibble(flights_weather)
flights_weather <- mutate(flights_weather,
    weekday=ifelse(weekday=="Lunedì",1,
      ifelse(weekday=="Martedì",2,
        ifelse(weekday=="Mercoledì",3,
          ifelse(weekday=="Giovedì",4,
            ifelse(weekday=="Venerdì",5,
              ifelse(weekday=="Sabato",6,7)))))))

#Variabile VOLUME
volume <- table(flights_weather$dest)
prova <- tibble(dest=names(volume),
                volume=as.numeric(volume))
flights_weather <-
  flights_weather %>%
  inner_join(prova, by = c(
    "dest"="dest"
  ))

#Seleziono le variabili su cui applicare il modello
flights_weather <- select(flights_weather, distance, hour,
                          visib, volume, weekday, delay)%>%drop_na()

flights_weather <- as.data.frame(flights_weather)
```

```
#Coefficiente di regressione per GP pesati
beta.w<-function(X, y, w.i, l1, l2, sigma2_n) {
  n <- length(X)
  K <- l1 * exp(- rdist(X, X)^2 / (2 * l2^2))
  W <- as.matrix(diag(w.i))
  inv.W <- as.matrix(diag(length(w.i)))
  for(w in 1:nrow(inv.W)){
    inv.W[w,w] <- inv.W[w,w]/w.i[w]
  }
  S <- K %*% solve(sigma2_n*as.matrix(diag(n))+inv.W%*%K)
  media <- K %*% solve(sigma2_n*as.matrix(diag(n))+inv.W%*%K)%*%y

  list(media=media, S=S)
}

#Cross-validation per GP pesati
cv <- function (X, y, w.i, grid) {
  R <- rep(NA, nrow(grid))
  r <- rep(NA, length(y))

  for(i in 1:nrow(grid)) {
    l1 <- grid[i,1]
    l2 <- grid[i, 2]
    sigma2_n <- grid[i,3]
    f <- beta.w(X, y, w.i, l1, l2, sigma2_n)

    for(j in 1:length(y)) {
      r[j] <- ((y[j,1] - f$media[j]) / (1 -f$S[j,j]))^2
    }
    R[i] <- mean(r)
  }
  list(R=R, r=r)
}

# Algoritmo BACKFIT pesato
```

```
backfit <- function (X, y, w.i, grid,beta.0){
  #inizializzazioine
  alpha <- beta.0
  f.j <- matrix(0, ncol = ncol(X), nrow = nrow(X))
  resid <- matrix(0, ncol = ncol(X), nrow = nrow(X))

  for(i in 1:4) {
    #ciclo per 4 volte --> fino a stabilizzazione
    for(j in 1:ncol(f.j)) {
      resid[,j] <- y - beta.0 - apply(f.j[,-j], 1, sum)
      c.v_sq.exp <- cv(X[,j], as.matrix(resid[,j]), w.i, grid)
      post_c.v_sq.exp <- beta.w(X[,j], as.matrix(resid[,j]), w.i,
                           grid[which.min(c.v_sq.exp$R),1],
                           grid[which.min(c.v_sq.exp$R),2],
                           grid[which.min(c.v_sq.exp$R),3])
      f.j[,j] <- post_c.v_sq.exp$media
    }
  }
  l1 <- grid[which.min(c.v_sq.exp$R),1]
  l2 <- grid[which.min(c.v_sq.exp$R),2]
  sigma2_n <- grid[which.min(c.v_sq.exp$R),3]
  list(f.j = f.j, alpha=alpha, l1=l1, l2=l2, sigma2_n=sigma2_n)
}

#Algoritmo di LOCAL SCORING
local_scoring <- function(X, y, grid){
  #inizializzazione
  beta.0 <- log(mean(y)/(1-mean(y)))
  f.j <- matrix(0, ncol = ncol(X), nrow = nrow(X))

  for(q in 1:4) {
    #fino a stabilizzazione
    eta.i <- beta.0 +  apply(f.j, 1, sum)
    p.i <- 1 / (1+exp(-eta.i))
    z.i <- eta.i + (y-p.i)/(p.i*(1-p.i))
    w.i <- p.i*(1-p.i)
```

```
    back <- backfit(X, as.matrix(z.i), w.i, grid,beta.0)
  }
  f.j <- back$f.j
  beta.0 <- back$alpha
  l1 <- back$l1
  l2 <- back$l2
  sigma2_n <- back$sigma2_n

  list(f.j=f.j, beta.0=beta.0, l1=l1, l2=l2, sigma2_n=sigma2_n)
}

flights_weather_small <- flights_weather[sample(
  nrow(flights_weather),2500),]
y <- as.matrix(flights_weather_small[,"delay"],ncol=1)
X<- as.matrix(flights_weather_small[,-6],ncol=5)

l <- local_scoring(X, y, Hp)

y.pred <- l$beta.0+l$f.j[,1]+l$f.j[,2]+l$f.j[,3]+l$f.j[,4]+l$f.j[,5]
nycflights_weather <- cbind(X[,c(1,2,3,4,5)], y,
                            l$f.j[,c(1,2,3,4,5)], y.pred)
colnames(nycflights_weather)[c(6,7,8,9,10,11,12)]<-
     c("delay","f1","f2","f3","f4","f5","response_gp")

#Confusion matrix applicato a risposta dicotomica
nycflights_weather[,"response_gp"] <- ifelse(
  nycflights_weather[,"response_gp"]>=0.5, 1, 0)
confusionMatrix(as.factor(nycflights_weather[,"delay"]),
                as.factor(nycflights_weather[,"response_gp"]))

#Confronto con GAM applicati alla funzione loess
y1 <- mgcv::gam(delay ~ lo(distance)+ lo(hour)+
      lo(visib)+ lo(volume)+lo(weekday),
       data=as.data.frame(nycflights_weather),family=binomial())

summary(y1)
```

```
response_loess <- ifelse(y1$fitted.values>=0.5, 1, 0)
nycflights_weather <- cbind(nycflights_weather, response_loess)
nycflights_weather[292:219,c(1:6,12,13)] #tabella 4.2

confusionMatrix(as.factor(nycflights_weather[,"delay"]), as.factor(
  nycflights_weather[,"response_loess"]))
```

Bibliografia

Arnold, T., Kane, M. & Lewis, B. W. (2019). *A computational approach to statistical learning*. Chapman and Hall/CRC.

Azzalini, A. & Scarpa, B. (2009). *Analisi dei dati e data mining*. Springer Science - Business Media.

Azzalini, A. & Scarpa, B. (2012). *Data analysis and data mining: An introduction*. Oxford University Press.

Bowman, A. W. & Adelchi, A. (1997). *Applied smoothing techniques for data analysis: the kernel approach with S-Plus illustrations*. clarendon press.

Gibbons, J. D. & Fielden, J. D. G. (1993). *Nonparametric statistics: An introduction*. SAGE publications.

Hastie, T. & Tibshirani, R. (1987). Generalized additive models: some applications. *Journal of the American Statistical Association 82.398* .

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

Wasserman, L. (2006). *All of nonparametric statistics*. Springer Science - Business Media.

Wickham, H. & RStudio (2021). *Flights that Departed NYC in 2013*.