# Appunti di Data Mining

Daniele Zago

28 febbraio 2021

# Indice

# Lezione 1

*Letture* Breiman (2001)

## 1.1 Introduzione

Con *data mining* si intende l'elaborazione di dati *complessi*, con lo scopo di estrarre informazione utile a fini previsivi e/o interpretativi. La complessità dei dati si può ricercare in

1. *Dimensione*: grandi moli di dati, raccolti in modo automatico, di dimensione tale da necessitare algoritmi specifici.
2. *Struttura*: non necessariamente si hanno tabelle $n \times p$, ma anche immagini, suono, video, ...
3. *Velocità*: i dati possono essere ottenuti sotto forma di *data stream*, ovvero in tempo reale, e ciò necessita l'aggiornamento continuo della stima dei modelli.

Queste caratteristiche fanno si che l'analisi dei dati sia caratterizzata da nuovi problemi computazionali e inferenziali, che richiedono metodologie specifiche per poter essere affrontati.

Inoltre, i dati vengono spesso raccolti senza un piano sperimentale o osservazionale, per cui le regole inferenziali del campionamento non sono applicabili. Si tratta comunque di un'opportunità da non buttare, perché si può comunque estrarre informazione utile con i giusti accorgimenti.

## 1.2 Elaborazione dell'informazione

A queste problematiche si sono affacciati soprattutto gli informatici, a partire dai quali si è sviluppato un filone di analisi dei dati, detto *Knowledge Discovery in Database* (*KDD*), che consiste in:

1. Costruzione di un database strategico (*Data Warehouse*), a partire dalla pulizia e aggregazione di più database operativi (*Data Mart*) con tecniche di OLTP (*OnLine Transaction Processing*), basate su SQL. Il DWH può essere utilizzato per costruire ulteriori *Data Mart*, finalizzati agli scopi di analisi.
2. Analisi descrittiva con tecniche di OLAP (*OnLine Analytical Processing*), cioè elaborazioni e interrogazioni grezze, per estrarre sintesi preliminari dei dati.
3. Analisi dei dati con tecniche di *Data Mining*, per la ricerca e l'estrazione di informazioni rilevanti.

Le caratteristiche tipiche dei dati, oltre alla complessità, sono anche:

- *Elevata dimensionalità*: difficoltà di rappresentazione, interpretazione, stima dei modelli, ...
- *No campionamento*: problemi nel generalizzare i risultati inferenziali.
- *Dati mancanti*: spesso presenti e da trattare in qualche modo.

Fondamentale l'aspetto computazionale nel data mining, talmente tanto che spesso predomina un approccio operativo a "scatola nera", senza conoscere nel dettaglio il funzionamento di algoritmi e modelli. In questo corso si cercherà di minimizzare la componente a scatola nera, cercando di studiare le procedure di stima e l'interpretabilità dei vari modelli.

# Lezione 2

*Letture* Senn (2003)
Dunson (2018)

In generale, lo studio dei dati può essere motivato da diverse finalità:

1. Cogliere l'*andamento globale* di un fenomeno: profilare l'intera popolazione di clienti, ...
2. Ricercare certe *configurazioni speciali*: situazioni al di fuori del comportamento standard, cose che non pensavamo fossero presenti, ...

Le applicazioni che consideriamo in questo corso saranno principalmente aziendali e biomediche, ovvero quelle in cui la maggior parte degli studenti va a lavorare.

## 2.1 Complicanze nell'analisi dei dati

Nell'analisi moderna, spesso l'obiettivo non è dichiarato prima della raccolta dei dati, per cui mescolando i dati e cercando a fondo si trova sempre qualcosa di "significativo".

*If you torture the data long enough, Nature will always confess.*

(R. H. Coase)

| **Approccio tradizionale** | **Approccio moderno** |
|---|---|
| 1. Scelgo il modello | 1. Raccolgo dati |
| 2. Raccolgo i dati | 2. Scelgo il modello |
| 3. Faccio inferenza | 3. Faccio inferenza |

I due approcci sono completamente diversi e, anche se abbiamo delle garanzie che il primo approccio sia coerente, non si hanno invece garanzie nel secondo caso.

Tra le problematiche di questo nuovo approccio ci sono: il *data snooping*, cioè la definizione degli obiettivi di ricerca dopo aver osservato i dati; il *data dredging*, ovvero la valutazione ripetuta di ipotesi fino a trovare qualcosa di significativo; l'uso di *variabili leaker*, quelle variabili che sono ottimi predittori ma non sono disponibili nel momento in cui si fa previsione (e.g. tipologia di trasferimento del calciatore nel caso voglia prevederne il prezzo).

È importante comprendere gli strumenti per porter scegliere adeguatamente lo strumento da utilizzare e per poter *interpretare correttamente* i risultati e valutare l'*attendibilità dell'output*.

## 2.2 Statistica nel *Data Mining*

Uno dei temi della statistica non è solo stimare una quantità, ma anche misurare *quanto bene abbiamo stimato* tale quantità. Questo si può fare soprattutto attraverso i modelli statistici probabilistici, per i quali si dispone di un grado di affidabilità delle stime ottenute:

**Esempio (Stima della media)**

Si utilizza come stimatore la media aritmetica

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i,$$

che ha accuratezza stimata pari a

$$\mathrm{se}(\overline{x}) = \sqrt{\frac{1}{n(n-1)}\sum_{i=1}^{n}(x_i - \overline{x})^2}.$$

È importante osservare che gli stessi dati forniscono sia il valore della stima, sia la sua accuratezza.

I modelli probabilistici possono tuttavia diventare eccessivamente complessi, specialmente quando sono disponibili grandi quantità di dati e strutture di *cloud computing*.

# Lezione 3

*Riferimenti* Dunson (2018)
Meng (2018)

Problematica del $p$-value, con la controversia del *Basic and Applied Social Psychology* che ha bandito l'uso dei $p$-value nei paper pubblicati.

## 3.1 Campionamento ed errori

Un tema fondamentale è legato al processo di selezione del campione di dati (Dunson, 2018): se non si dispone di un campione casuale, come è la norma nel caso del data mining, l'efficienza nello stimare le quantità di interesse viene falsata.

Sono importanti da distinguere la *distorsione per selezione*, ovvero il caso in cui il campione rappresenta una popolazione diversa dalla popolazione $P$ di interesse, e il problema degli *errori di misura*, in cui magari non si sa se gli individui rispondono correttamente o meno.

Avere a disposizione big data non risolve il problema: si riduce l'incertezza nella stima di una quantità per la popolazione $Q \neq P$, ma non è generalizzabile a $P$. Per questo, big data $\not\equiv$ qualità della stima, di conseguenza è importante tenere conto della distorsione anche nel data mining (Meng, 2018).

### 3.1.1 Distorsione per selezione

Dato un campione, si supponga di voler stimare la media $\overline{X}_N$ di $X$ per una certa popolazione $P$. Si usa, come misura di qualità della stima, la quantità

$$\overline{X}_n - \overline{X}_N,$$

cioè la differenza tra media campionaria e quella della popolazione.

Nel caso del campionamento casuale semplice, si ha la decomposizione

$$\overline{X}_n - \overline{X}_N = \underbrace{\sqrt{\frac{1-f}{f}}}_{\text{quantità di dati}} \times \underbrace{\sigma_X}_{\text{difficoltà problema}},$$

con $f = n/N$. Questo funziona molto bene nella pratica, perché campionando con un $n$ sufficientemente grande, si riesce a controllare l'errore.

Quando invece il campionamento non è casuale, definendo con $R_i$ la variabile che indica se l'$i$-esima unità è stata osservata nel campione, allora si modifica la decomposizione, aggiungendo una quantità legata alla *qualità dei dati*:

$$\overline{X}_n - \overline{X}_N = \underbrace{\sqrt{\frac{1-f}{f}}}_{\text{quantità di dati}} \times \underbrace{\sigma_X}_{\text{difficoltà problema}} \times \underbrace{\rho_{R,X}}_{\text{qualità dei dati}},$$

dove $\rho_{R,X}$ è la correlazione tra il processo di scelta delle unità e la variabile di interesse:

$$\rho_{R,X} = \text{corr}(R, X).$$

Ovviamente, più $\rho_{R,X}$ si avvicina a 0, più ci si avvicina al campionamento casuale e migliore è la qualità dei dati a disposizione.

A parità di correlazione media, si ha che l'errore cresce a un tasso asintotico $\sqrt{N}$:

$$Z_{n,N} = \frac{\overline{X}_n - \overline{X}_N}{\sqrt{\mathbb{V}_{\text{ccs}}[\overline{X}_n]}} = \frac{\sqrt{\frac{1-f}{f}}\sigma_X \rho_{R,X}}{\sqrt{\frac{1-f}{f}\frac{1}{N-1}}\sigma_X} = \sqrt{N-1}\rho_{R,X},$$

cioè quando la numerosità $N$ della popolazione cresce, la correlazione tra $R$ e $X$ in modulo deve essere minore a $1/\sqrt{N}$ per poter controllare l'errore di sottostima o sovrastima.

Questa è una misura della relazione tra la qualità dei dati (corr. tra $R$ e $X$) e la *dimensione della popolazione*: più è grande la popolazione, più è piccola la soglia ammissibile di correlazione.

**Esempio (Distorsione causata dalla correlazione)**
Pensando alle elezioni di Trump del 2016, la popolazione votante era $N \approx 230$ mio, per cui il valore massimo in modulo ammissibile per l'autoselezione è $\rho_{R,X} \approx 0.000066$.

Le stime del *CCES* suggeriscono una correlazione di $\rho_{R,X} \approx -0.005$ relativa all'autodichiarazione di votare per Trump, a causa dei non rispondenti tra i votanti.

Con una tale correlazione, anche prendendo come campione $l'1\%$ della popolazione votante, circa 2.3 mio, a causa della correlazione si ottiene una qualità corrispondente a un ccs di $n \approx 400$ unità.

Quando i big data sono soggetti a distorsione per selezione ed errori di misura, non devono essere usati per decisioni senza valutazione e scetticismo.

In generale, bisogna ricordare il "rasoio di Occam", ovvero che (nel nostro caso) non bisogna usare modelli complessi quando non è necessario. Bisogna partire dalle cose semplici e complicarle se necessario.

**Esempio (Scelta delle variabili)**
Se sono interessato a prevedere a chi vendere un prodotto per il fitness, sono più utili variabili socio-demografiche rispetto ad altre variabili.

Campioni casuali dal database di analisi possono essere molto utili per fare stime non troppo distorte, specie se il database è molto grande.

## 3.2 Modellazione

*Letture* Cox (1990), Cox (1997)

*All models are wrong, but some are useful*

(G.P. Box)

Un *modello* è una rappresentazione semplificata del fenomeno di interesse e funzionale a un obiettivo specifico: la funzionalità è fondamentale, perché significa che si possono mantenere le caratteristiche di interesse e rimuove quello che non serve.

In generale, siccome si semplifica la realtà, diversi modelli possono essere utili a rappresentare lo stesso fenomeno:

1. In alcuni casi si possono interpretare i modelli come un'approssimazione del *modello vero*, come nella fisica e nell'astronomia.
2. In altri ambiti il modello ha una funzione *operativa*, per cui il criterio di bontà è l'utilità del modello: quanto bene prevede, quanto permette di migliorare le decisioni, ...

Anche nei modelli *black-box* è presente una componente relativa al modello, anche se meno visibile: si tratta di assunzioni sul tipo di funzione che viene stimata, oltre alla robustezza rispetto alle violazioni di questi assunti.

# Lezione 4

*Riferimenti* Azzalini e Scarpa (2012, §2.1-2.2)

## 4.1 Regressione lineare e prime estensioni

Cominciando dalla regressione lineare, si definisce il modello

$$Y = X\beta + \varepsilon,$$

da cui si vogliono stimare i valori di $\beta$ che soddisfano

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - x_i^\mathsf{T}\beta\right)^2,$$

$$\hat{\beta} = (X^\mathsf{T}X)^{-1}X^\mathsf{T}Y \implies \hat{Y} = X\hat{\beta}.$$

In generale, la linearità del modello è nei parametri, per cui si possono usare trasformazioni non lineari delle variabili come esplicative, per migliorare la capacità previsiva.

### Risposte multiple

Nel caso in cui si dispone di più di una variabile risposta, si può comunque utilizzare un modello lineare convertendo $Y$ in una variabile matriciale:

$$Y_{n\times q} = X_{n\times p}B_{p\times q} + E_{n\times q},$$

con $B$ matrice di parametri e $\mathbb{V}[\tilde{E}_i] = \Sigma$, varianza di ciascuna delle $n$ righe di $E$. Applicando i minimi quadrati, si ottiene

$$\hat{Y} = X\hat{B}, \quad \hat{B} = (X^\mathsf{T}X)^{-1}X^\mathsf{T}Y,$$

con varianza

$$\hat{\Sigma} = \frac{1}{n-p}Y^\mathsf{T}(I-P)Y, \quad P = X(X^\mathsf{T}X)^{-1}X^\mathsf{T}.$$

## 4.2 Aspetti computazionali

La soluzione ai minimi quadrati si basa sull'inversione di $X^\mathsf{T}X$, che di solito si effettua tramite la fattorizzazione di Cholesky, con costo computazionale $p^3 + \frac{np^2}{2}$. Con questa soluzione, il tempo di risoluzione è dominato dal numero $p$ di colonne.

Alternativamente si può usare la decomposizione $QR$, se $n$ non è troppo grande: si decompone $X$ in

$$X = QR,$$

da cui

$$\begin{aligned}\hat{\beta} &= (X^\mathsf{T}X)^{-1}X^\mathsf{T}y \\ &= (R^\mathsf{T}Q^\mathsf{T}QR)^{-1}R^\mathsf{T}Q^\mathsf{T}y \\ &= R^{-1}(R^\mathsf{T})^{-1}R^\mathsf{T}Q^\mathsf{T}y \\ &= R^{-1}Q^\mathsf{T}y\end{aligned}$$

con costo computazionale $2np^2$.

Per scomporre la matrice, si utilizza l'algoritmo di Gram-Schmidt, algoritmo di ortonormalizzazione di una base di vettori. Si inizializza $z_0 = x_0 = 1$ e si effettua

1. Per $j = 1, 2, \ldots, p$ si ottengono i coefficienti di regressione

$$\hat{\gamma}_{lj} = \frac{z_l^\mathsf{T} x_j}{z_l^\mathsf{T} z_l}$$

con vettore dei residui

$$z_j = x_j - \sum_{k=1}^{j-1} \hat{\gamma}_{kj} z_k.$$

2. Si effettua la regressione di $y$ sui residui $z_p$ per ottenere le stime $\hat{\beta}_p$.

Con questo algoritmo si sfrutta la proprietà che il vettore dei residui $z_j$ è ortogonale alle altre variabili, per cui la regressione multipla ha lo stesso coefficiente delle regressioni semplici.

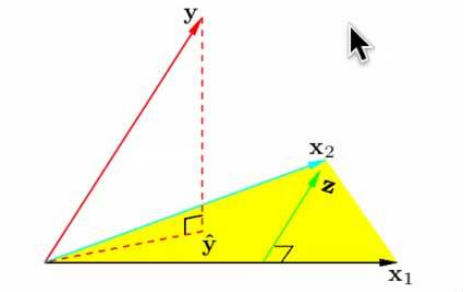


Figura 1: Regressione sul vettore dei residui.

## 4.3 Minimi quadrati ricorsivi

Quando i dati vengono forniti come un flusso continuo (*data stream*), oppure non possono essere caricati completamente in memoria, è necessario modificare la procedura di stima dei minimi quadrati.

Consideriamo le quantità di base

$$W_{p\times p} = X^\mathsf{T}X, \quad u_{p\times 1} = X^\mathsf{T}y,$$

chiamiamo $\tilde{x}_i$ la $i$-esima riga di $X$ messa in colonna:

$$\tilde{x}_{p\times 1} = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,p} \end{pmatrix}, \quad X_{n\times p} = \begin{pmatrix} \tilde{x}_1^\mathsf{T} \\ \vdots \\ \tilde{x}_n^\mathsf{T} \end{pmatrix}.$$

In particolare, osserviamo che

$$\tilde{x}_i\tilde{x}_i^\mathsf{T} = \begin{pmatrix} x_{i,1} \\ \vdots \\ x_{i,p} \end{pmatrix} (x_{i,1}, x_{i,2}, \dots, x_{i,p}) = \begin{pmatrix} x_{i,1}^2 & x_{i,1}x_{i,2} & \dots & x_{i,1}x_{i,p} \\ \vdots & \ddots & \dots & \vdots \\ x_{i,1}x_{i,p} & x_{i,2}x_{i,p} & \dots & x_{i,p}^2 \end{pmatrix}$$

per cui si può scrivere

$$W = \sum_{i=1}^{n} \tilde{x}_i\tilde{x}_i^\mathsf{T}, \quad u = \sum_{i=1}^{n} \tilde{x}_i y_i.$$

Questa rappresentazione è molto comoda, perché dipende solamente dalla singola osservazione $(x_i, y_i)$. Dunque, aggiungendo la $j$-esima osservazione, le quantità di interesse diventano

$$W_j = W_{j-1} + \tilde{x}_j\tilde{x}_j^\mathsf{T}, \quad u_j = u_{j-1} + \tilde{x}_j y_j,$$

da cui

$$\hat{\beta}_j = W_j^{-1}u_j.$$

Se fossimo in grado di invertire $W_j$ ricorsivamente, sarebbe possibile aggiornare le stime $\hat{\beta}_j$ per ogni nuova osservazione in ingresso.

**Inversione di $W_{j+1}$**

Supponiamo di disporre delle stime $\hat{\beta}_n$ e $V_n = W_n^{-1}$, allora

$$X_{n+1} = \begin{pmatrix} X_n \\ \tilde{x}_{n+1}^\mathsf{T} \end{pmatrix}, \quad W_{n+1} = X_{n+1}^\mathsf{T}X_{n+1} = X_n^\mathsf{T}X_n + \tilde{x}_{n+1}\tilde{x}_{n+1}^\mathsf{T}.$$

Utilizziamo la formula di Sherman-Morrison per invertire la matrice:

$$(A + bd^\mathsf{T})^{-1} = \underbrace{A^{-1}}_{\text{inversa}} - \underbrace{\frac{1}{1 + d^\mathsf{T}A^{-1}b}A^{-1}bd^\mathsf{T}A^{-1}}_{\text{aggiornamento}}.$$

$$\begin{aligned} W_{n+1}^{-1} &= \left(W_n + \tilde{x}_{n+1}\tilde{x}_{n+1}^\mathsf{T}\right)^{-1} \\ &= W_n^{-1} - h_{n+1}W_n^{-1}\tilde{x}_{n+1}\tilde{x}_{n+1}^\mathsf{T}W_n^{-1} \end{aligned}$$

dove $h_{n+1} = \frac{1}{1+\tilde{x}_{n+1}^\mathsf{T}W_n^{-1}\tilde{x}_{n+1}}$.

Da qui si ottiene

$$\begin{aligned}
\beta_{n+1} &= W_{n+1}^{-1} u_{n+1} \\
&= W_{n+1}^{-1}(X_n^{\mathsf{T}} y_n + \tilde{x}_{n+1} y_{n+1}) \\
&= (W_n^{-1} - h_{n+1} W_n^{-1} \tilde{x}_{n+1} \tilde{x}_{n+1}^{\mathsf{T}} W_n^{-1})(X_n^{\mathsf{T}} y_n + \tilde{x}_{n+1} y_{n+1}) \\
&= \hat{\beta}_n + h_{n+1} W_n^{-1} \tilde{x}_{n+1} \big( \frac{1}{h_{n+1}} y_{n+1} - \tilde{x}_{n+1}^{\mathsf{T}} W_n^{-1} X_n^{\mathsf{T}} y_n - \tilde{x}_{n+1}^{\mathsf{T}} \tilde{x}_{n+1} y_{n+1} \big) \\
&= \hat{\beta}_n + h_{n+1} W_n^{-1} \tilde{x}_{n+1} (y_{n+1} - \tilde{x}_{n+1}^{\mathsf{T}} \hat{\beta}_n) \\
&= \hat{\beta}_n + \underbrace{h_{n+1} W_n^{-1} \tilde{x}_{n+1}}_{\text{guadagno}} \underbrace{(y_{n+1} - \tilde{x}_{n+1}^{\mathsf{T}} \hat{\beta}_n)}_{\text{errore}} \\
&= \hat{\beta}_n + k_n e_{n+1}.
\end{aligned}$$

Dunque abbiamo una formula che inverte la matrice tramite un *filtro lineare*, che modifica la vecchia stima $\hat{\beta}_n$ in base all'errore di previsione $e_{n+1}$ pesato dal *guadagno* $k_n$ del filtro.

In questo senso si dice che lo stimatore "impara dagli errori", aggiustando di volta in volta la stima corrente.

Nella pratica, siccome di solito disponiamo di una grande quantità di dati, modificare leggermente la matrice $V_0$ non porta delle gravi conseguenze. Allora, si possono scegliere come valori iniziali

$$\hat{\beta}_0 = 0_p, \quad V_0 = I_p,$$

in modo da evitare l'inversione della matrice $p \times p$ iniziale. Questo si può fare, perché se disponiamo di una grande quantità di dati, la distorsione sarà minima.

Esercizi per casa

1. Verificare che la formula di Sherman-Morrison fornisce l'inversa di $(A + bd^{\mathsf{T}})$.
2. Verificare la formula dei minimi quadrati ricorsivi.
3. Scrivere la formula per l'aggiornamento di $\hat{\sigma}^2$, (Sugg. : $\|\hat{\varepsilon}\|^2 = y^{\mathsf{T}} y - y^{\mathsf{T}} X \hat{\beta}$).
4. Qual è la differenza fra l'intervallo di confidenza del valore della funzione e quello di previsione, entrambi relativi ad una futura osservazione?

# Lezione 5

*Riferimenti* Azzalini e Scarpa (2012, §3.1-3.3)
Hastie et al. (2013, §7.1-7.3)

## 5.1 Sovradattamento, distorsione e varianza

Si consideri un problema di $n = 30$ coppie $(x_i, y_i)$, generate da una legge di tipo

$$y = f(x) + \varepsilon.$$

L'interesse è la stima di una regola $\hat{y} = \hat{f}(x)$, che consenta di prevedere $y$ sulla base di $x$ anche per dei *nuovi dati* in arrivo e non osservati in precedenza. Il presupposto implicito è che il processo generatore dei dati sia lo stesso sia per dati vecchi che per dati nuovi (campionamento, ecc. discusso nella Lezione 3.1).

La cosa più semplice è interpolare i dati con un polinomio, il cui grado massimo possibile è pari a $n - 1$:

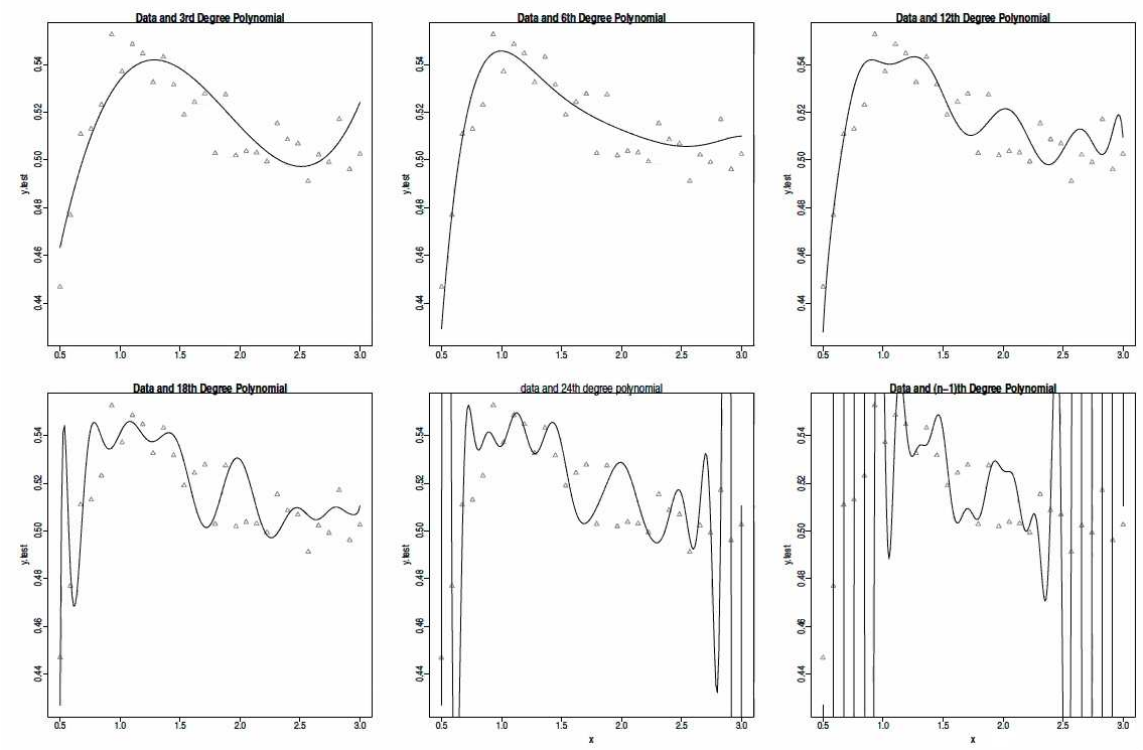


Figura 2: Regressione polinomiale con vari gradi: 3, 6, 12, 18, 24, 29.

Scelto un valore $x'$, vogliamo ottenere $\hat{y} = \hat{f}(x)$, per un generico stimatore. Se conoscessimo completamente il meccanismo generatore dei dati, compreso $f(x')$, potremmo valutare la qualità dello stimatore $\hat{y}$ con l'errore quadratico medio (EQM, MSE)

$$\text{EQM}(\hat{y}) = \mathbb{E}[\{\hat{y} - f(x')\}^2],$$

dove il valore atteso si sta calcolando sulla distribuzione di $\hat{y}$ (stima frequentista).

Calcolando l'EQM per ciascun grado del polinomio $p$, che rappresenta la complessità del modello, non si ottiene un andamento monotono (Figura 3).

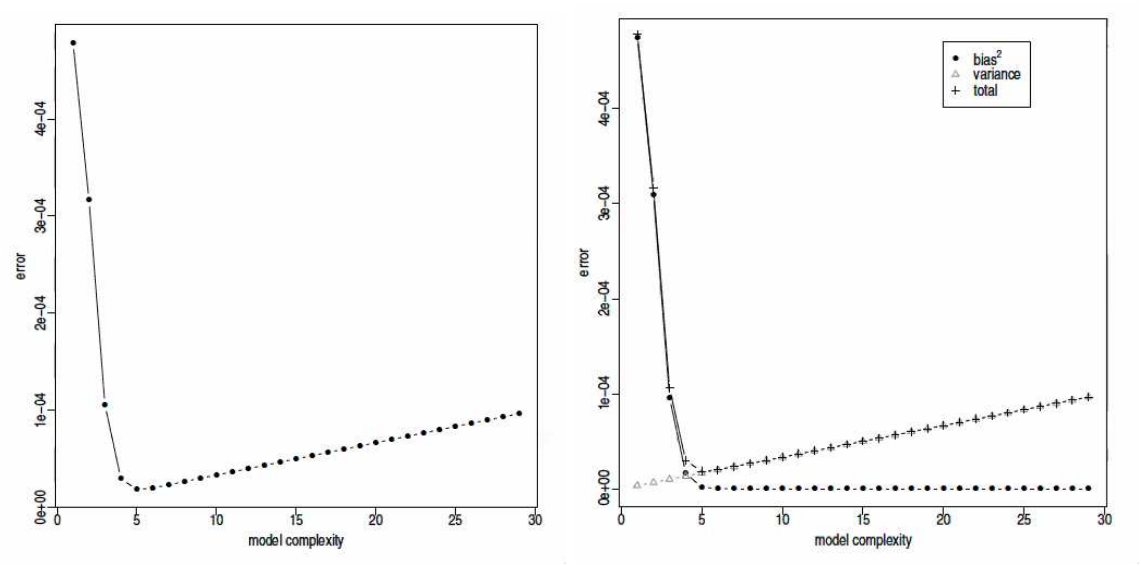


Figura 3: *Sinistra*: EQM al variare del grado del polinomio, rispetto alla vera $f(x)$. *Destra*: stessa curva, decomposta in somma di distorsione al quadrato e varianza.

Si ricorda che l'errore quadratico medio si può scomporre in

$$\begin{aligned}\mathbb{E}[\{\hat{\vartheta} - \vartheta\}^2] &= \mathbb{E}[\{\hat{\vartheta} - \mathbb{E}[\hat{\vartheta}]\}^2] + \{\mathbb{E}[\hat{\vartheta}] - \vartheta\}^2 \\ &= \mathbb{V}[\hat{\vartheta}] + \text{Bias}^2(\hat{\vartheta}).\end{aligned}$$

Vediamo che, a differenza degli stimatori in cui si assume il modello statistico corretto, nel nostro caso la distorsione non è nulla. In generale, questo vale ogni qual volta il processo generatore dei dati non fa parte della classe dei modelli utilizzati.

In generale, la variabilità del modello cresce al crescere della sua complessità, in quanto un maggior numero di parametri tenderà a catturare la variabilità locale dei dati. La distorsione, invece, diminuisce al crescere della complessità.

Esiste allora un grado $p$ migliore degli altri, in termini di EQM, che tuttavia non possiamo identificare con certezza se non conoscendo il vero processo $f$.

### Stima dell'EQM

L'idea è di utilizzare la stima del modello sui dati "di ieri" per confrontare le previsioni con i dati "di oggi", per avere un indice dell'errore che commettiamo. Nel fare questo processo, assumiamo che

1. I dati nuovi sono generati con lo stesso processo che ha generato i dati vecchi.
2. I dati nuovi sono indipendenti dai dati vecchi.

Aumentare il grado $p$ del polinomio migliora le prestazioni per i dati su cui si stima il modello, ma le previsioni a un certo punto peggiorano: si ha il fenomeno di *sovra-adattamento* (*overfitting*), ovvero il modello segue troppo le caratteristiche locali dei dati, che in realtà non vengono replicate quando ottengo nuove osservazioni.

**Importante**

Non bisogna mai valutare un modello usando la capacità previsiva sui dati su cui è stato stimato, ma sempre su un insieme di dati "nuovo". In questo modo, la scelta del modello migliore sarà guidata da un compromesso tra distorsione e varianza.

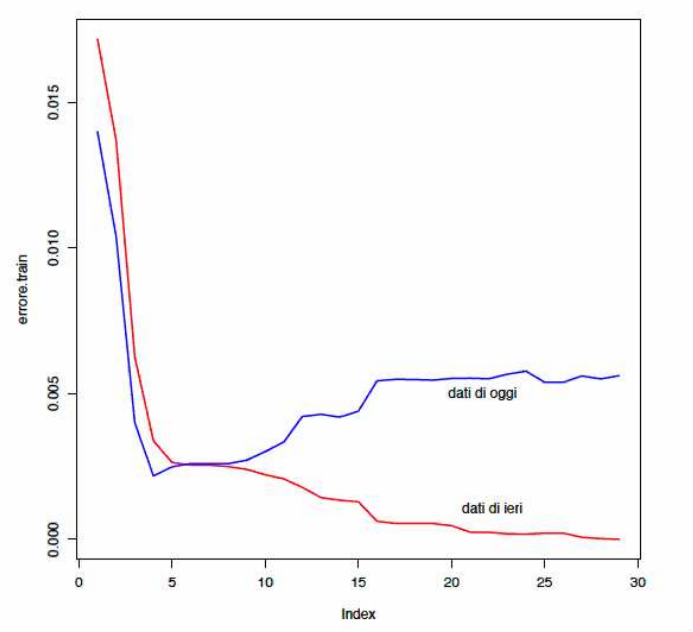


Figura 4: Varianza residua in funzione del grado del polinomio, per dati vecchi (*rosso*) e nuovi (*blu*).

# Lezione 6

*Riferimenti* Azzalini e Scarpa (2012, §3.5)
Hastie et al. (2013, §7.10)

## 6.1 Errore di stima e suddivisione dei dati

Abbiamo bisogno di un meccanismo per *generare nuovi dati* su cui stimare l'EQM. Nel caso in cui disponiamo di tante osservazioni, possiamo dividere casualmente i dati in tre insiemi, ciascuno con un compito diverso:

1. *Training set* (insieme di stima): dati su cui adattiamo i vari modelli.
2. *Validation set* (insieme di verifica): dati su cui si valuta la capacità previsiva di ciascun modello.
3. *Test set* (insieme di controprova): dati con cui si valuta la capacità previsiva del modello finale.

Le proporzioni dei dati negli insiemi di solito sono 50%, 25%, 25% nel caso in cui si usino tutte e tre le divisioni, altrimenti 75%, 25%, 0%.

In generale questo procedimento riduce la numerosità dei dati su cui stimiamo il modello, ma non è un problema quando $n$ è elevato. Diventa invece un problema quando $n$ è piccolo o $p > n$, caso in cui la tecnica non si può utilizzare.

**Osservazioni**

- Questo procedimento *non necessita* che sia completamente esplicitato un modello, ma è sufficiente avere una procedura di stima e una di previsione.
- I dati vanno ovviamente separati in modo *casuale* e va tenuto conto che quello che otteniamo è una semplice stima dell'EQM. Nell'esempio dei polinomi, la verità dice che il minimo è in $p = 5$, mentre con la suddivisione dei dati si ottiene un minimo in $p = 4$.
- Non sempre si può fare e non sempre è la scelta giusta: se devo prevedere il futuro in una serie storica, non posso campionare la parte iniziale, ma quella finale.
- L'insieme di verifica deve contenere una parte della dipendenza contenuta nei dati, se questi non sono indipendenti (e.g. serie storiche).

## 6.2 Convalida incrociata

Supponendo di effettuare una divisione 75% e 25%, potremmo essere sfortunati e avere insiemi di dati completamente diversi tra stima e verifica. Gli obiettivi sono allora:

1. Svincolarci da *quegli* specifici 75% e 25%.
2. Se $n$ non è grande, la stima sul solo 75% dei dati non è particolarmente buona.

Si può usare il procedimento di *convalida incrociata*, ovvero la suddivisione del dataset in porzioni, che vengono utilizzate a rotazione per la stima e la verifica. Ottenute le stime dell'EQM per ciascun modello, si può fare la media degli errori di ognuno e scegliere quello con valore più piccolo.

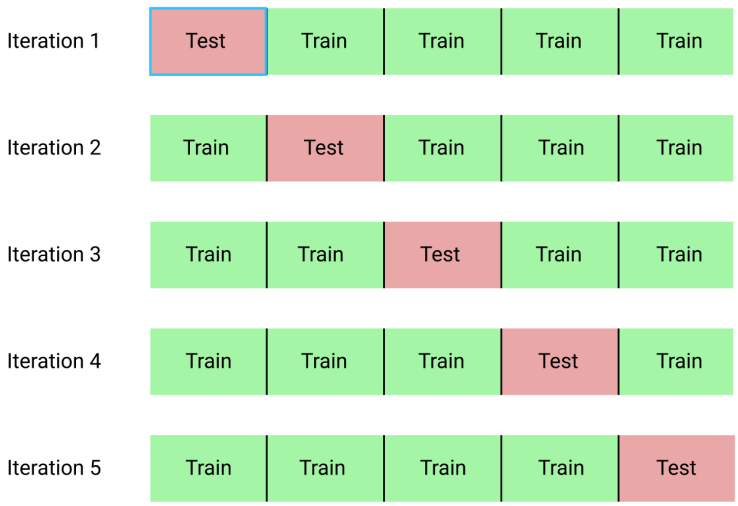


Figura 5: Esempio di convalida incrociata a 5 insiemi.

Ovviamente il costo computazionale viene moltiplicato per il numero di divisioni effettuate e il massimo numero di divisioni possibili è $n$, ovvero il caso in cui l'insieme di test sia composto da una sola osservazione (*leave-one-out cross-validation*).

**Algoritmo 1** Leave-one-out cross-validation

**for** $j = 1, \dots, p$ **do**
  **for** $i = 1, \dots, n$ **do**
    Calcolare $\hat{y}_{-i}$
    $e_i = y_i - \hat{y}_{-i}$
  **end for**
  $D^*(j) = \sum_{i=1}^{n} e_i^2$
**end for**

### Osservazione

Stiamo utilizzando i dati due volte, sia per la stima sia per la valutazione dell'errore, esattamente come nel caso di media e varianza campionarie. Ci sono risultati teorici che garantiscono proprietà ottimali per $n \longrightarrow \infty$.

### Leave-one-out cross-validation e stimatori lineari

Il ciclo al variare di $n$ è computazionalmente intensivo, ma nel caso degli *stimatori lineari* è possibile calcolare $e_i$ senza ri-stimare il modello da capo. Dato uno stimatore del tipo $\hat{y} = Py$, esiste un risultato che dice

$$y_i - \hat{y}_{-i} = \frac{y_i - \hat{y}_i}{1 - P_{ii}},$$

dove $\hat{y}_i$ è la stima usando tutte le osservazioni. Dunque, si ottiene

$$D^*(p) = \frac{1}{n} \sum_{i=1}^{n} \left( \frac{y_i - \hat{y}_i}{1 - P_{ii}} \right)^2.$$

Ricordiamo che $|P_{ii}|$ indica l'influenza che il punto $(x_i, y_i)$ ha sul modello.

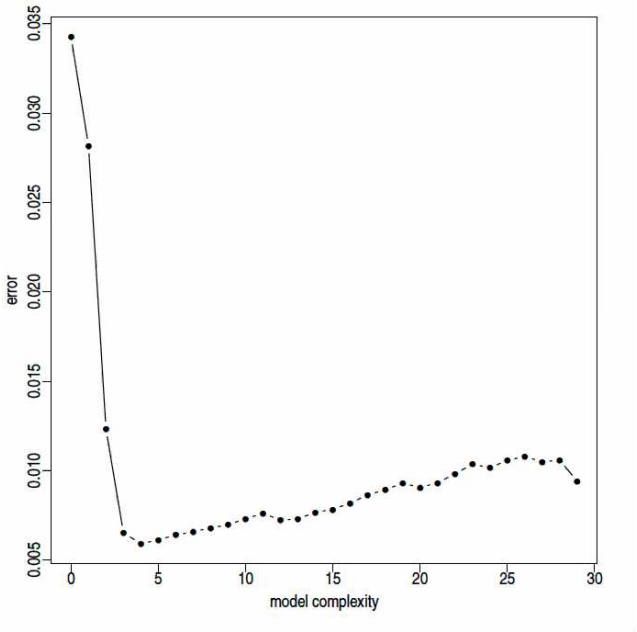


Figura 6: Convalida incrociata al variare del grado $p$ del polinomio.

# Lezione 7

*Riferimenti* Hastie et al. (2013, §7.3-7.4)

Vediamo ora delle procedure per la valutazione della qualità del modello, nei casi in cui non disponiamo di sufficienti osservazioni per separare i dati o usare la convalida incrociata.

Ricordiamo infatti, che nel caso si disponga di 60 osservazioni, è difficile giustificare una suddivisione del dataset in $45 - 15$, poiché si perderebbe gran parte dell'informazione.

## 7.1 Penalizzazioni e criteri di informazione

Supponiamo allora di disporre di un solo insieme di dati, su cui viene stimato il modello. Come indice di qualità del modello non si può usare la varianza dei residui, perché

1. È una stima troppo ottimista della qualità del modello.
2. La procedura di stima del modello in genere va proprio a minimizzare la varianza dei residui, che quindi è monotona rispetto alla complessità del modello.

Una strada è la *penalizzazione della devianza*, con l'aggiunta di un termine che penalizza l'aumento di complessità del modello. Data

$$D = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

utilizziamo equivalentemente la trasformazione monotona data dalla log verosimiglianza nel caso di dati normali

$$-2 \log L = n \log \sigma^2 + \frac{D}{\sigma^2} + \text{const.}$$

Esercizio : Verificare la formula della log-verosimiglianza.

L'idea è di costruire un *criterio di informazione* (IC) della forma

$$\text{IC}(p) = -2 \log L + \text{penalità}(p),$$

scegliendo opportunamente la funzione di penalità, in base a considerazioni pratiche.

Nel caso gaussiano, sappiamo che l'ipotesi che valuta la nullità dei coefficienti nel modello più grande ha statistica test

$$(2 \log L_{p_1} - 2 \log L_{p_2}) \overset{H_0}{\sim} \chi^2_{p_1 - p_2}.$$

Nel caso di un parametro aggiuntivo, si ha che

$$(2 \log L_{p+1} - 2 \log L_p) \overset{H_0}{\sim} \chi_1.$$

Questa distribuzione dice che il *costo* di aggiungere un parametro non necessario al modello è dato da "una distribuzione $\chi^2_1$". Il costo medio sotto questa distribuzione è pari a 1, per cui una penalizzazione deve aggiungere almeno 1 alla log-verosimiglianza.

Usiamo dunque $-2\log L$ come quantità di base visto che sappiamo che asintoticamente è un chi quadro, perché è possibile penalizzarla con una scala di misura nota.

**Criterio di informazione di Akaike (AIC)**

Nella costruzione del suo criterio di informazione, Akaike era partito da una logica diversa, ovvero dalla divergenza di Kullback-Leibler tra il vero modello $p_*(\cdot)$ e il modello stimato $p(\cdot;\vartheta)$.

Questa quantità misura la divergenza tra la distribuzione delle future osservazioni dei dati e quella del modello:

$$\begin{aligned} KL(p_*(\cdot), p(\cdot;\theta)) &= \mathbb{E}_{p_*}\left[\log \frac{p_*(Y)}{p(Y;\theta)}\right] \\ &= \mathbb{E}_{p_*}[\log p_*(Y)] - \mathbb{E}_{p_*}[\log p(Y;\vartheta)] \end{aligned}$$

Poiché la verità si assume fissa, si può agire solamente sul secondo termine, utilizzando la stima $\hat{\vartheta}_y$, come funzione dei dati, per prevedere il comportamento sui dati futuri $Y$. Si considera allora

$$\mathbb{E}_y\left\{\mathbb{E}_Y[\log p(Y;\hat{\vartheta}_y)]\right\},$$

che, con qualche assunzione e approssimazione analitica, fornisce il risultato

$$\mathrm{AIC}(p) = -2\log p(y;\hat{\vartheta}) + 2p.$$

| **Criterio** | **Penalizzazione** |
|---|---|
| AIC | $2p$ |
| $\mathrm{AIC}_c$ | $2p + \frac{2p(p+1)}{n-(p+1)}$ |
| BIC | $p\log n$ |
| HQ | $cp\log\log n$ |

Tabella 1: Comuni penalizzazioni della log-verosimiglianza.

Questi criteri si applicano anche a modelli *non annidati e non gaussiani*, scegliendo la forma di $\log L$ più appropriata.

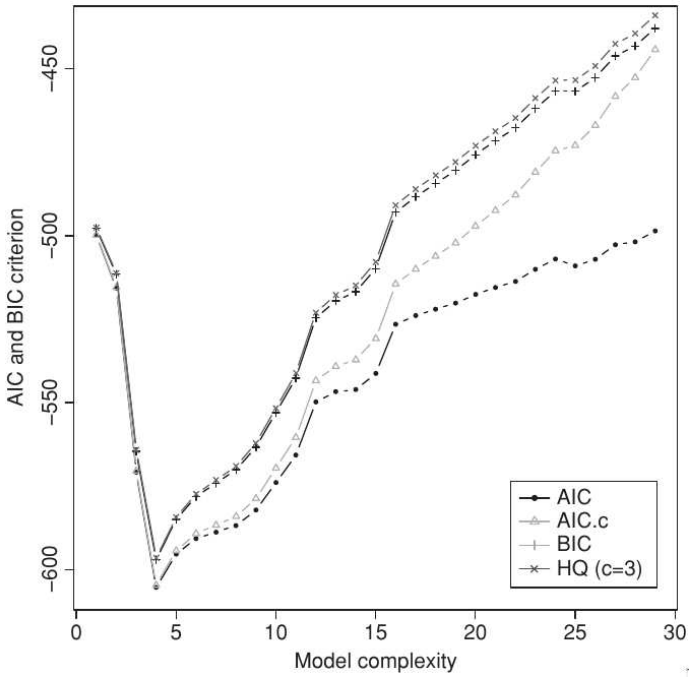


Figura 7: Criteri di informazione al variare della complessità del modello.

**Confronto tra metodi**

Non c'è una regola unica nella scelta della procedura di valutazione del modello, ma solo considerazioni di carattere empirico.

I criteri di informazione sono solo un'approssimazione delle procedure di splitting, che tuttavia utilizzano tutto l'insieme di dati nel calcolare la stima.

Nel caso dei criteri di informazione, devo essere in grado di scrivere la forma della log verosimiglianza del modello, cosa non immediata in alcuni casi.

Quando si stimano diversi tipi di modello, si possono usare approcci differenti per il confronto nelle diverse famiglie. Ad esempio, si può fare un confronto con AIC tra modelli gaussiani e un confronto con convalida incrociata per alberi di regressione.

## 7.2 Variabilità nella stima

Lo scenario ideale nella stima di un modello lineare è quando le variabili esplicative sono incorrelate, visto che la correlazione può mascherare l'effetto delle variabili.

Con un disegno di esperimento bilanciato, si possono stimare i coefficienti separatamente e ognuno di essi significa effettivamente il "cambiamento unitario quando tutte le altre restano fisse".

Quando il disegno non è bilanciato, la correlazione tra due variabili causa

- Aumento della varianza dei coefficienti.

- Interpretazioni rischiose delle derivate parziali, perché una correlazione tra $x_j$ e $x_k$ implica che quando $x_j$ cambia, anche $x_k$ tende a cambiare.

# Lezione 8

*Riferimenti* Hastie et al. (2013, §3.3)

Il compromesso distorsione-varianza è sempre presente quando si analizzano dati, in quanto l'insieme dei modelli considerati non comprende mai il vero processo generatore dei dati.

## 8.1 Selezione delle variabili

Per migliorare la performance del modello, è importante studiare in che modo possiamo selezionare automaticamente le variabili rilevanti. La selezione automatica delle variabili è utile per

1. Migliorare la *capacità previsiva* del modello, regolarizzando i coefficienti di regressione.
2. Migliorare l'*interpretabilità* del modello, identificando un sottoinsieme di variabili che contribuiscono maggiormente nella regressione.

### Best subset regression

L'approccio più ovvio è provare a selezionare tutti i possibili sottoinsiemi di variabili e vedere quello che si adatta meglio. Siccome le variabili in generale non sono ortogonali, è necessario considerare *tutti* i sottoinsiemi di variabili che sono pari a $2^p$.

Questo procedimento si può implementare se il numero di variabili $p$ è contenuto ($\leq 20$), altrimenti il numero di sottoinsiemi diventa troppo grande per poter essere esplorato.

In questo caso, il parametro che regola la complessità del modello è dato dal *numero* di variabili esplicative del modello, per cui

1. Si trova per ogni $s \in \{0, 1, \dots, p\}$ il sottoinsieme di dimensione $s$ che fornisce la più piccola somma dei quadrati dei residui, visto che il numero di variabili è lo stesso (penalizzazione AIC è la stessa).
2. Per scegliere $s$ si usa il compromesso tra distorsione e varianza (EQM, AIC, ...).

$$AIC = \sum_{i=1}^{n} \frac{(y_i - x_i\hat{\beta})^2}{\sigma^2} + 2p$$

$$C_p = \sum_{i=1}^{n} \frac{(y_i - x_i\hat{\beta})^2}{\sigma^2} - n + 2p$$

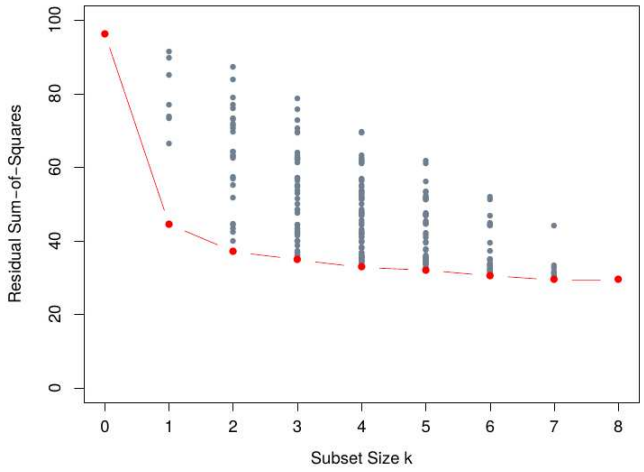


Figura 8: Esempio di best subset selection.

Esercizio: si potrebbero usare i test $T$ o $F$ per selezionare le variabili?

Si può utilizzare il test $F$ solo nel caso in cui il modello vero sia contenuto tra i modelli lineari sotto analisi, in pratica se si assume che il bias del modello sia trascurabile.

In ogni caso, non si può usare il test $R^2$, visto che $R^2(k)$ è una funzione monotona crescente al variare di $k$.

Anziché usare tutti i possibili modelli, si possono usare le selezioni automatiche stepwise, ovvero selezioni *greedy* che guardano solamente un sottoinsieme di questi modelli:

| **Stepwise** | **Procedura** |
|---|---|
| Forward | Si parte dall'intercetta e si aggiunge la variabile che migliora di più l'adattamento |
| Backward | Si parte dal modello completo e si eliminano le variabili meno rilevanti. |
| Hybrid | Si inserisce la variabile più utile e si rimuove quella meno rilevante. |

**Commenti**

1. Se parto dal modello vuoto e aggiungo variabili con una procedura forward, verosimilmente si selezionerà un modello più parsimonioso rispetto alla procedura backward.
2. Nel modello finale backward potrebbero esserci variabili non significative tra i predittori (perché?).

Tutte le procedure hanno uno o più *parametri di regolazione*, che ne modificano la complessità:

1. **Regressione polinomiale**: grado del polinomio.
2. **Best subset selection**: dimensione del sottoinsieme.
3. **Stepwise regression**: posizione lungo il percorso.

Usando la convalida incrociata si possono anche calcolare le deviazioni standard per ciascuno step della regressione stepwise.

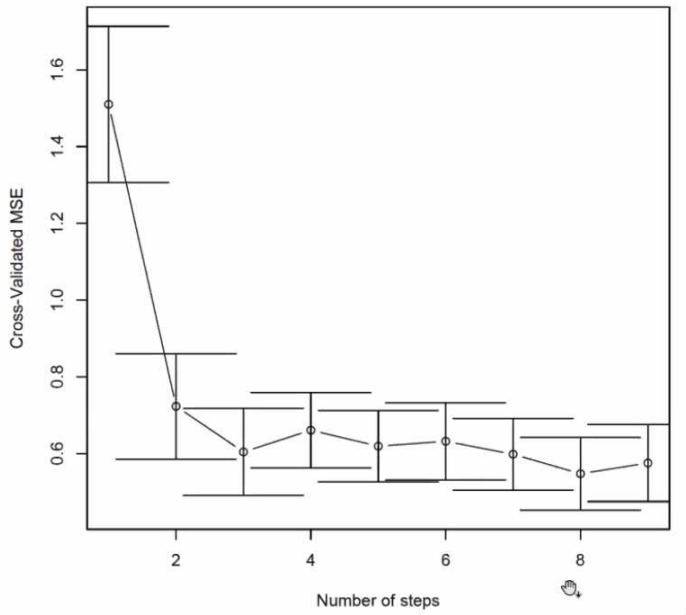


Figura 9: Media e deviazione standard degli errori di convalida incrociata al variare dello step.

**Pro e contro**

+ La selezione stepwise forward si può usare anche quando $p > n$, perché aggiungo una variabile per volta. La selezione backward non permette di farlo, in quanto si comincerebbe direttamente da $p$ variabili.
+ Spesso la stepwise forward porta a modelli parsimoniosi.
− Non si possono togliere variabili, una volta che questa è entrata nel modello. In quali casi sarebbe utile poter rimuovere variabili?
− Si può sovradattare ai dati più velocemente rispetto ad altri metodi di regolarizzazione.

## 8.2 Riduzione della dimensionalità

*Riferimenti* Hastie et al. (2013, §3.5)

L'idea, in comune con la selezione delle variabili, è la *riduzione della dimensionalità del problema*, inteso come dimensione della complessità del modello.

Se ruotiamo gli assi, potremmo essere in grado di trovare una direzione in cui le variabili variano molto di più rispetto ad altre.

Il metodo più conosciuto è quello delle *componenti principali*, ovvero si trova una sequenza di combinazioni lineari delle variabili esplicative, che sono incorrelate e ordinate per varianza

decrescente.

Se $\Sigma$ è la matrice di varianza covarianza di $x_1, x_2, \ldots, x_p$, si ha

$$\Sigma = \Gamma \Lambda \Gamma^{\mathsf{T}},$$

da cui si ottengono le componenti principali

$$Z = X\Gamma.$$

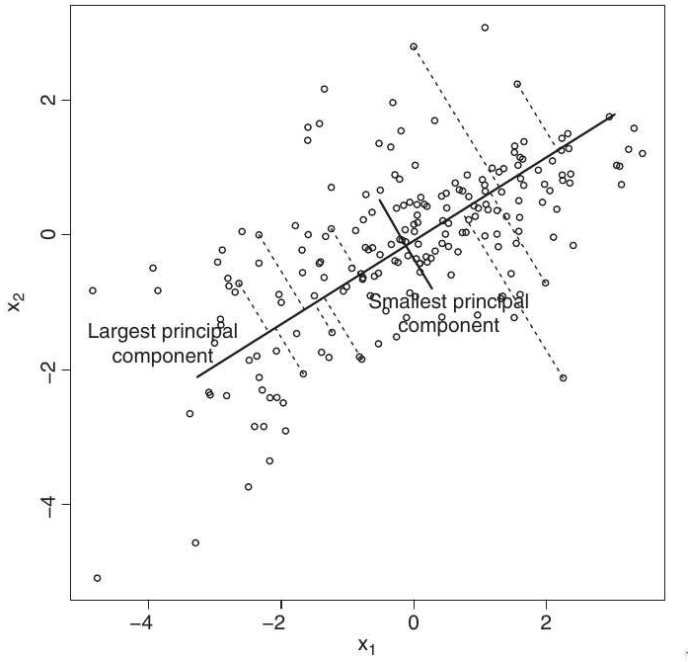


Figura 10: La prima componente principale è la retta di regressione rispetto alle distanze ortogonali.

Poiché le $z_j$ sono *ortogonali*, la regressione è la somma delle regressioni univariate

$$\hat{y} = \overline{y} + \sum_{i=1}^{J} \hat{\gamma}_j z_j,$$

per una scelta di $J \in \{1, \ldots, p\}$.

**Problemi**

Con questo tipo di regressione non sono in grado di interpretare i parametri, a meno che le componenti principali abbiano una specifica interpretazione.

Le componenti principali si ottengono solo guardando le variabili esplicative, non quelle più correlate con la variabile risposta.

Le componenti principali dipendono da tutte le covariate, per cui generalmente si ottengono tutti i coefficienti $\{\beta_j\}_{j=1,\dots,p}$ del modello.

## 8.3 Metodi di regolarizzazione

Il *subset selection* è un processo discreto, ovvero un coefficiente o è incluso o è escluso dal modello, per cui si ottiene una soluzione interpretabile, ma con alta variabilità.

I metodi di *shrinkage* sono un'alternativa continua alla subset selection che non soffre di alta variabilità: lo scopo è *contrarre* (*shrink*) i coefficienti verso lo zero, in modo da ridurre la variabilità dei coefficienti.

In questo modo, accettando una distorsione nelle stime dei coefficienti, si può mediamente ridurre l'errore quadratico medio e migliorare la capacità predittiva del modello.

## 8.4 Regressione ridge

*Riferimenti* van Wieringen (2020) (Discussione approfondita)
Hastie et al. (2013, §3.4)

La regressione ridge risolve il problema di minimo

$$\min_{\beta\in\mathbb{R}^p} (y - X\beta)^\mathsf{T}(y - X\beta)$$

$$\text{s.t. } \sum_{j=1}^{p} \beta_j^2 \leq s$$

o equivalentemente in forma di Lagrange

$$\min_{\beta\in\mathbb{R}^p} (y - X\beta)^\mathsf{T}(y - X\beta) - \lambda\beta^\mathsf{T}\beta,$$

che ha soluzione esplicita

$$\hat{\beta}_\lambda = (X^\mathsf{T}X + \lambda I)^{-1}X^\mathsf{T}y.$$

*Dim.*

$$\frac{\partial}{\partial\beta}\{(y - X\beta)^\mathsf{T}(y - X\beta) - \lambda\beta^\mathsf{T}\beta\} = 0 \iff$$

$$\frac{\partial}{\partial\beta}\left\{y^\mathsf{T}y - \beta^\mathsf{T}X^\mathsf{T}y - y^\mathsf{T}X\beta + \beta^\mathsf{T}X^\mathsf{T}X\beta - \lambda\beta^\mathsf{T}\beta\right\} = 0 \iff$$

$$-X^\mathsf{T}y - X^\mathsf{T}y + 2X^\mathsf{T}X\beta - 2\lambda\beta = 0 \iff$$

$$\hat{\beta}_\lambda = (X^\mathsf{T}X + \lambda I)^{-1}X^\mathsf{T}y$$

□

Questo è uno stimatore distorto (esercizio) ma, per alcuni valori di $\lambda > 0$, può presentare un EQM minore rispetto a quello dei minimi quadrati.

**Osservazioni**

1. $\lambda = 0 \implies \hat{\beta}_\lambda = \hat{\beta}_{\text{ols}}$ mentre $\lambda \to \infty \implies \hat{\beta}_\lambda \to 0$.
2. Tipicamente non si penalizza l'intercetta $\beta_0$, in quanto la penalizzazione di interesse è sui coefficienti delle esplicative. Una penalizzazione dell'intercetta farebbe dipendere la stima dall'origine di $Y$.
3. La penalizzazione si effettua sulle variabili standardizzate, perché la soluzione $\hat{\beta}_\lambda$ non è invariante per riscalamento.
4. Spesso è utile quando le variabili sono *collineari*, perché anche un piccolo $\lambda > 0$ fornisce una soluzione $\hat{\beta}_\lambda$ numericamente e statisticamente stabile.
5. Poiché la matrice $X^\mathsf{T}X + \lambda I$ è sempre invertibile, si può stimare un modello lineare anche se $p > n$.

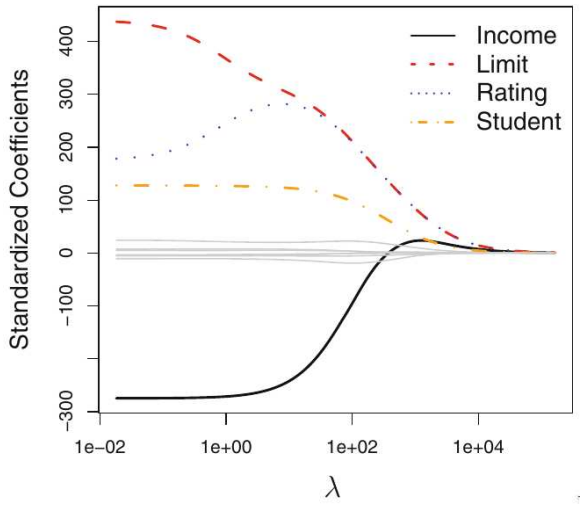


Figura 11: Profilo dei parametri di regressione ridge al variare del parametro di penalizzazione $\lambda$.

# Lezione 9

## 9.1 Interpretazione bayesiana della soluzione ridge

*Riferimenti* Hastie et al. (2013, §3.4.3)

La stima ridge può essere vista dal punto di vista bayesiano. Supponendo che

$$Y_i|\beta_j \sim \mathcal{N}(\beta_0 + x_i^\mathsf{T}\beta, \sigma^2)$$

$$\beta_j \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \tau^2) \quad j = 1, \dots, p$$

da cui, la densità a posteriori ha log-verosimiglianza proporzionale a

$$(y - X\beta)^\mathsf{T}(y - X\beta) + \lambda\beta^\mathsf{T}\beta,$$

dove $\lambda = \frac{\sigma^2}{\tau^2}$ e la soluzione ridge è la *moda a posteriori* (o valore atteso, essendo normale). Infatti, la distribuzione a posteriori è

$$\begin{aligned} p(\beta|x, y) &\propto L(\beta; y, x)p(\beta) \\ &= \exp\left\{\frac{1}{2\sigma^2}(y - X\beta)^\mathsf{T}(y - X\beta)\right\} \times \dots \end{aligned}$$

Aggiungendo e togliendo $X\hat{\beta} = X(X^\mathsf{T}X)^{-1}X^\mathsf{T}y$ in $(y - X\beta)$ si ottiene (esercizio)

$$\propto \exp\left\{-\frac{1}{2\sigma^2}\left[(y - X\beta)^\mathsf{T}(y - X\beta) + (\beta - \hat{\beta})^\mathsf{T}X^\mathsf{T}X(\beta - \hat{\beta})\right]\right\}$$

e il doppio prodotto è nullo, perché i residui sono ortogonali.

$$L(\beta; y, X, \sigma^2) \propto \exp\left\{-\frac{1}{2\sigma^2}(\beta - \hat{\beta})^\mathsf{T}X^\mathsf{T}X(\beta - \tilde{\beta})\right\}$$

e la densità a priori è

$$p(\beta) \propto \exp\left\{-\frac{\beta^\mathsf{T}\beta}{2\tau^2}\right\}.$$

Dunque, facendo i conti si ottiene

$$\beta|y \sim \mathcal{N}\left(\left(X^\mathsf{T}X + \frac{\sigma^2}{\tau^2}I\right)^{-1}X^\mathsf{T}y, \sigma^2\left(X^\mathsf{T}X + \frac{\sigma^2}{\tau^2}I\right)^{-1}\right).$$

In questo senso si potrebbe vedere la stima bayesiana come una regolarizzazione della stima frequentista. Tuttavia, in un caso otteniamo una distribuzione di probabilità e nell'altro otteniamo una stima puntuale.

## 9.2 Ridge e componenti principali

*Riferimenti* Hastie et al. (2013, §3.4)

C'è un collegamento tra regressione sulle componenti principali e regressione ridge: quest'ultima proietta le $y$ sulle componenti principali, *comprimendo* le componenti con bassa varianza rispetto a quelle con alta varianza.

Scrivendo $X = UDV^\mathsf{T}$ la decomposizione a valori singolari di $X$, la regressione sulle componenti principali usa i valori previsti

$$\hat{y}_{\text{cp}} = X(X^\mathsf{T}X)^{-1}X'y = U \operatorname{diag}(\underbrace{1, 1, \dots, 1}_{k}, 0, \dots, 0)U^\mathsf{T}y,$$

che è una selezione discreta del numero $k$ di componenti da mantenere. Nella regressione ridge, invece, i valori previsti sono

$$\hat{y}_{\text{ridge}} = X(X^\mathsf{T}X - \lambda I)^{-1}X^\mathsf{T}y = U \operatorname{diag}\left(\frac{d_j^2}{d_j^2 + \lambda}\right) U^\mathsf{T}y,$$

ovvero una versione "lisciata" della regressione sulle componenti principali.

*Dim.*

$$\begin{aligned}
\hat{y}_{\text{ridge}} = X\hat{\beta}_\lambda &= X^\mathsf{T}(X^\mathsf{T}X + \lambda I)^{-1}X^\mathsf{T}y \\
&= UDV^\mathsf{T}(VD^2V^\mathsf{T} + \lambda I)^{-1}VDU^\mathsf{T}y \\
&= UD(\underbrace{V^\mathsf{T}V}_{I} D^2 \underbrace{V^\mathsf{T}V}_{I} + \lambda \underbrace{V^\mathsf{T}V}_{I})^{-1}DU^\mathsf{T}y \\
&= UD(D^2 + \lambda I)^{-1}DU^\mathsf{T}y \\
&= U \operatorname{diag}\left(\frac{d_i^2}{d_i^2 + \lambda}\right) U^\mathsf{T}y
\end{aligned}$$

□

Nella regressione ridge si definisce *gradi di libertà effettivi* del modello la quantità

$$\begin{aligned}
\mathrm{df}(\lambda) &= \operatorname{tr}\left(X(X^\mathsf{T}X + \lambda I)^{-1}X^\mathsf{T}\right) \\
&= \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}
\end{aligned}$$

gradi di libertà effettivi sono una misura della complessità del modello e sono inversamente proporzionali a $\lambda$.

La formula ha una diretta analogia con i gradi di libertà $p$ del modello lineare, per il quale si può calcolare esplicitamente usando la proprietà di invarianza sotto permutazioni cicliche della traccia

$$\operatorname{tr}\left(X(X^\mathsf{T}X)^{-1}X^\mathsf{T}\right) = \operatorname{tr}\left(X^\mathsf{T}X(X^\mathsf{T}X)^{-1}\right) = \operatorname{tr}\left(I_p\right) = p.$$

## 9.3 Regressione lasso

*Riferimenti* Tibshirani (1996)
Hastie et al. (2013, §3.4.2)

Il lasso risolve il problema di minimo

$$\min_{\beta \in \mathbb{R}^p} (y - X\beta)^\mathsf{T}(y - X\beta)$$

$$\text{s.t. } \sum_{j=1}^{p} |\beta_j| \leq s$$

o equivalentemente

$$\min_{\beta \in \mathbb{R}^p} (y - X\beta)^\mathsf{T}(y - X\beta) + \lambda \|\beta\|_1.$$

La penalizzazione lasso non ha soluzione esplicita e non è lineare in $y$, per cui è necessario ricorrere ad algoritmi numerici di risoluzione.

Il vantaggio del lasso è che alcuni coefficienti vengono stimati *esattamente a zero* se il vincolo $\lambda$ è sufficientemente grande (o $s$ sufficientemente piccolo). Il lasso produce una "selezione continua" delle variabili del modello, oltre a ridurre la variabilità delle stime come la ridge.

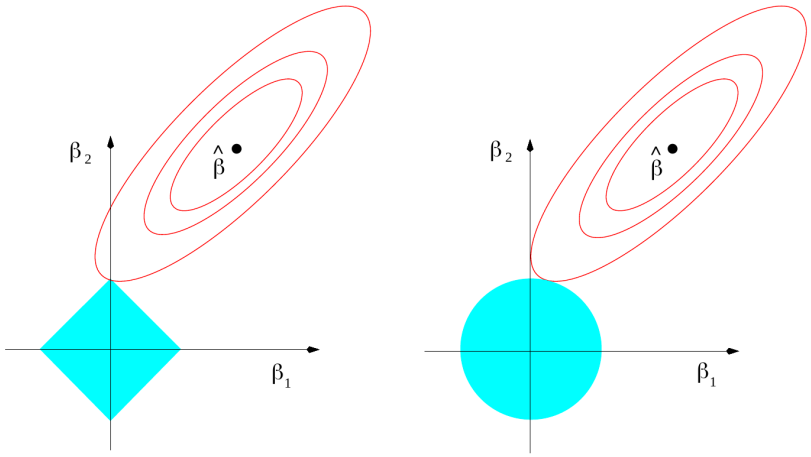


Figura 12: Funzione di verosimiglianza e regioni corrispondenti ai vincoli del lasso (sinistra) e ridge (destra).

Per qualunque paraboloide, a meno che abbia asse maggiore esattamente su una bisettrice, il lasso ammette un valore di $s$ con cui si trova una soluzione sparsa.

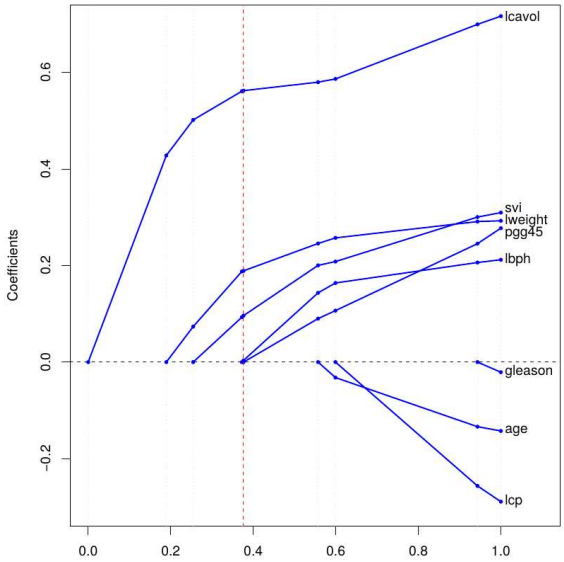


Figura 13: Profilo dei coefficienti lasso, con variabili standardizzate.

### 9.3.1 Confronto tra lasso e ridge

Se le variabili esplicative sono ortogonali, la ridge *moltiplica* il coefficiente ai minimi quadrati per $\kappa < 1$, mentre il lasso li *trasla* per una costante verso lo zero, ponendoli esattamente a zero quando diventano sufficientemente piccoli.

$$\begin{aligned} &\text{Ridge} \quad \hat{\beta}_j/(1+\lambda) \\ &\text{Lasso} \quad \operatorname{sgn}(\hat{\beta}_j)(|\hat{\beta}_j - \lambda|)_+ \end{aligned}$$

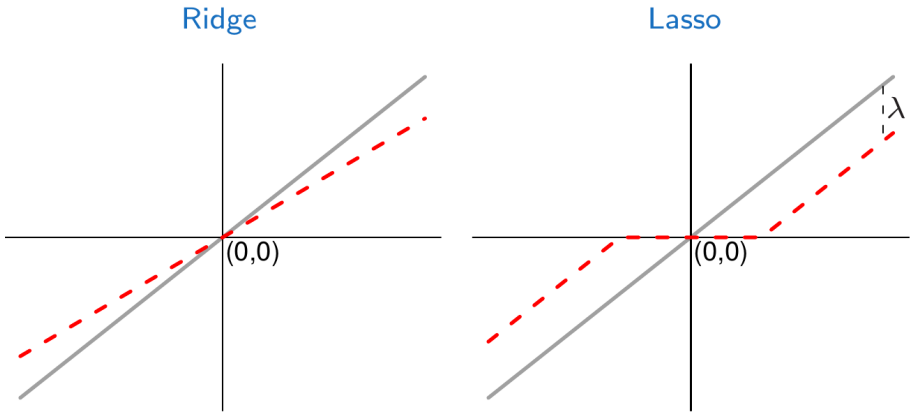


Figura 14: Effetto dello stimatore ridge e dello stimatore lasso rispetto ai minimi quadrati (bisettrice), nel caso di un singolo coefficiente.

# Lezione 10

## 10.1 Consistenza nella selezione del lasso

Avendo osservato che il lasso effettua contemporaneamente shrinkage e selezione delle variabili, è naturale chiedersi se i coefficienti posti a zero siano quelli che, nel vero processo generatore dei dati, sono nulli.

In pratica, ci si chiede se la selezione avvenga in modo "consistente", nel senso frequentista del termine. Alcuni autori hanno studiato le proprietà asintotiche di identificazione del modello del lasso, al crescere di $n$ e $p$.

### Impostazione del problema

Definiamo $S = \{j : \beta_j^0 \neq 0,\ j = 1, \dots, p\}$ l'insieme dei parametri *veri* non nulli nel modello e sia $\hat{S}_\lambda = \{j : \hat{\beta}_\lambda \neq 0,\ j = 1, \dots, p\}$ l'insieme delle variabili *scelte dal lasso* per un dato valore di $\lambda$. La domanda è se

$$\exists \lambda : S \subset \hat{S}_\lambda.$$

(libro di Bühlmann e van Geer). Un risultato asintotico mostra che, sotto condizioni abbastanza restrittive esiste un $\lambda$ che soddisfa la consistenza di selezione.

Le condizioni, imposte alla matrice $X$, prendono il nome di *stabilità nell'intorno* o *irrepresentabilità*, e si riferiscono alla correlazione (regressione) tra le variabili presenti nel modello non presenti:

$$\| \overbrace{(X_S^\mathsf{T} X_S)^{-1} X_S^\mathsf{T} X_{S^c}}^{\text{Regressione tra } S \text{ e } S^c} \|_\infty \leq 1 - \varepsilon \quad \text{per qualche } \varepsilon \in [0, 1].$$

In pratica, si richiede che le variabili collegate alla risposta non devono essere troppo correlate alle variabili non collegate alla risposta.

Sotto questa condizione, e assumendo che i coefficienti non nulli siano sufficientemente grandi, esiste un opportuno $\lambda \gg \sqrt{\log(p)/n}$ tale che

$$P(\hat{S} = S) \xrightarrow{n,p \to \infty} 1.$$

### Lasso adattivo

Nella realtà, la relazione di irrepresentabilità non è soddisfatta e il lasso stima delle cose che potremmo non volere.

Una possibile soluzione è il *lasso adattivo* (Zou e Hastie, 2005), che è una generalizzazione del lasso in cui si pesano i coefficienti nella penalizzazione:

$$\hat{\beta}(\lambda) = \underset{\beta}{\operatorname{argmin}} (y - X\beta)^\mathsf{T}(y - X\beta) + \lambda \sum_{j=1}^{p} \frac{|\beta_j|}{|\hat{\beta}_{\text{init},j}|},$$

dove $\hat{\beta}_{\text{init}}$ è uno stimatore "buono" per $\beta$, ad esempio la stima ai minimi quadrati.

Ci sono risultati teorici che mostrano come il lasso adattivo fornisca stime consistenti, mantenendo soprattutto la convessità della funzione obiettivo.

Il lasso adattivo fa parte dei metodi di regolarizzazione pesata, nei quali la componente di regolarizzazione $J(\beta)$ è della forma

$$\sum_{j=1}^{p} w_j |\beta_j|.$$

## 10.2 Regolarizzazione di Tikhonov

Guardando alla regolarizzazione come un problema vincolato con una norma $\|\cdot\|_q$, lasso e ridge sono casi particolari della *regolarizzazione di Tikhonov*, ovvero

$$\tilde{\beta} = \underset{\beta}{\operatorname{argmin}} (y - X\beta)^{\intercal}(y - X\beta) + \lambda \sum_{j=1}^{p} |\beta_j|^q, \quad q \geq 0.$$

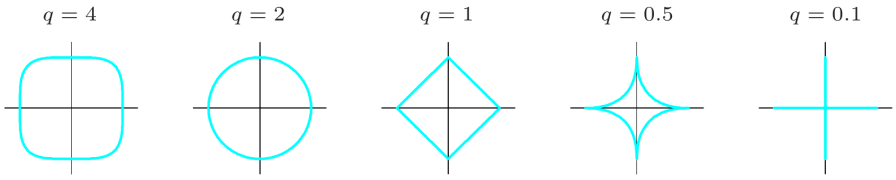


Figura 15: Regioni di vincolo per alcune scelte di $q$. Il lasso, $q = 1$, è il più piccolo valore di $q$ per cui la regione presenta punti non differenziabili agli angoli.

Nell'ottica della stima bayesiana, $|\beta_j|^q$ si può pensare come il logaritmo del kernel di una densità a priori per $\beta_j$, che ha verosimiglianza gaussiana.

# Lezione 11

## 11.1 Least Angle Regression (LAR)

*Riferimenti* Efron et al. (2004) (Trattazione approfondita)
Hastie et al. (2013, §3.4.4)

Studiamo come si calcolano i coefficienti della regressione lasso, introducendo un nuovo algoritmo per calcolare i minimi quadrati ordinari.

**Algoritmo 2** Algoritmo LAR per i minimi quadrati

**Input:** $x_1, \dots, x_p$ standardizzate

1: **Init:**

$r \leftarrow y$
$\hat{\beta}_1, \dots, \hat{\beta}_p \leftarrow 0$

2: Cerco il predittore $x_j$ maggiormente correlato con $r$.

3: Incremento il valore di $\beta_j$ nella direzione di $\text{corr}(r, x_j)$ finché un'altra $x_k$ ha tanta correlazione con i residui quanta ne presenta $x_j$.

4: Aggiorno stima e residui $r \leftarrow y - X_A \hat{\beta}_A$

5: Incremento il valore di $\beta_j$ e $\beta_k$ nella direzione di equicorrelazione, cioè lungo la bisettrice, fino a che trovo una terza variabile correlata con i residui quanto $x_j$ e $x_k$.

6: Si procede in questo modo fino a che tutte le variabili sono entrate nel modello, cioè quando $\text{corr}(x_j, r) = 0 \ \forall j$.

Questo algoritmo permette di ottenere la stima i minimi quadrati, ma con una piccola modifica si può ottenere anche l'intero percorso di stima lasso.

### Least Angle Regression and Selection (LARS)

1. Inizio della stima con l'algoritmo LAR.
2. Se un coefficiente $\hat{\beta}_j$ incrocia lo zero, l'algoritmo si ferma.
3. Si elimina la variabile $x_j$ corrispondete e si riprende l'algoritmo LAR.

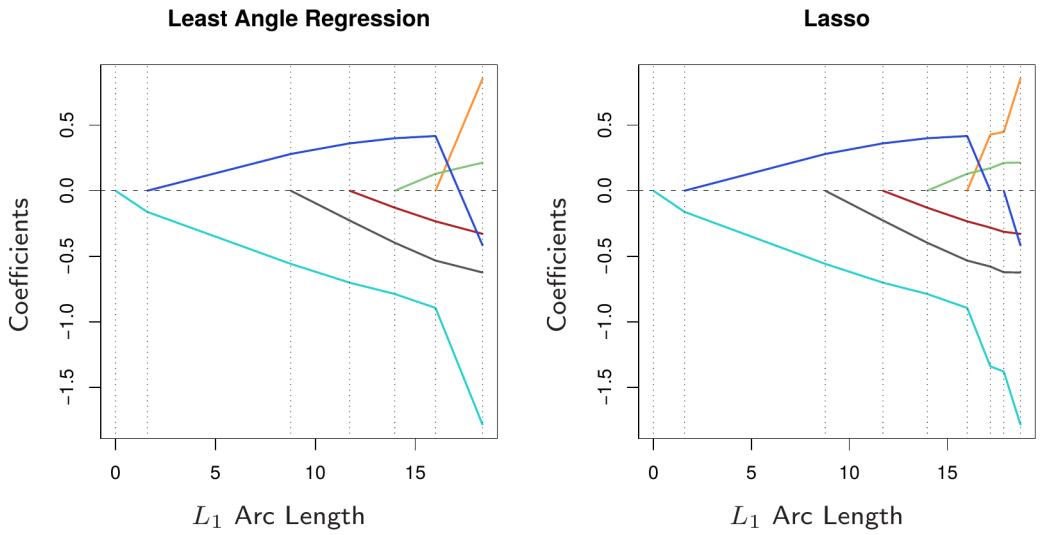


Figura 16: Profilo dei coefficienti di regressione per il LAR, a sinistra, e del lasso, a destra. I profili sono identici fino a che la variabile blu incontra lo zero.

**Osservazioni**

- Con questo algoritmo, si può calcolare l'intero percorso del lasso, per qualunque valore di $\lambda$, con lo stesso ordine di complessità dell'algoritmo ai minimi quadrati.
- Spesso la procedura farà più di $p$ passi, ma va bene uguale perché asintoticamente rimane dello stesso ordine di complessità.

## 11.2 Pathwise coordinate descent

*Riferimenti* Hastie et al., (2013, §3.8.6)

L'idea dell'algoritmo è di ottimizzare la funzione obiettivo una coordinata alla volta, invece di proseguire nella direzione più ripida.

**Idea** Se si considera il problema di ottimizzazione una singola coordinata alla volta, si dispone di una soluzione esplicita per il lasso univariato. Iterando una coordinata per volta e mantenendo fisse le altre, si converge alla soluzione globale del lasso.

Si consideri la funzione obiettivo

$$\text{argmin}_\beta f(\beta) = \text{argmin}_\beta \left\{ \underbrace{\frac{1}{2n}\sum_{i=1}^{n}(y_i - x_i\beta)^2}_{f_1(\beta)} + \underbrace{\lambda|\beta|}_{f_2(\beta)} \right\},$$

allora la soluzione del problema è $\hat{\beta}_\lambda = \text{sgn}(\hat{\beta})(|\hat{\beta}| - \lambda)_+$.

*Dim.*

Derivando la funzione rispetto a $\beta$, e utilizzando la definizione di subdifferenziale, supponendo che le variabili siano standardizzate si ottiene

$$\frac{\mathrm{d}f_1(\beta)}{\mathrm{d}\beta} = -\frac{1}{n}\sum_{i=1}^{n} x_i\,(y_i - x_i\beta) = -\frac{1}{n}\sum_{i=1}^{n} x_i y_i + \beta$$

$$\frac{\mathrm{d}f_2(\beta)}{\mathrm{d}\beta} = \begin{cases} -\lambda & \text{se} \quad \beta < 0 \\ [-\lambda, \lambda] & \text{se} \quad \beta = 0 \\ \lambda & \text{se} \quad \beta > 0 \end{cases}$$

Uguagliando a 0 la somma dei due termini, si ottiene

$$0 = \frac{\mathrm{d}f(\beta)}{\mathrm{d}\beta}$$

$$0 = \frac{\mathrm{d}f_1(\beta)}{\mathrm{d}\beta} + \frac{\mathrm{d}f_2(\beta)}{\mathrm{d}\beta}$$

$$0 = \begin{cases} -\frac{1}{n}\sum_{i=1}^{n} x_i y_i + \beta - \lambda & \text{se} \quad \beta < 0 \\ \left[-\frac{1}{n}\sum_{i=1}^{n} x_i y_i - \lambda, -\frac{1}{n}\sum_{i=1}^{n} x_i y_i + \lambda\right] & \text{se} \quad \beta = 0 \\ -\frac{1}{n}\sum_{i=1}^{n} x_i y_i + \beta + \lambda & \text{se} \quad \beta > 0 \end{cases}$$

$$\hat{\beta}_{\text{lasso}} = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} x_i y_i + \lambda & \text{se} \quad \frac{1}{n}\sum_{i=1}^{n} x_i y_i < -\lambda \\ 0 & \text{se} \quad -\lambda < \frac{1}{n}\sum_{i=1}^{n} x_i y_i < \lambda \\ \frac{1}{n}\sum_{i=1}^{n} x_i y_i - \lambda & \text{se} \quad \frac{1}{n}\sum_{i=1}^{n} x_i y_i > \lambda \end{cases}$$

Ponendo $\hat{\beta} = \frac{1}{n}\sum_{i=1}^{n} x_i y_i$ lo stimatore ai minimi quadrati per le variabili standardizzate, si ottiene

$$\hat{\beta}_{\text{lasso}} = \begin{cases} \hat{\beta} + \lambda & \text{se } \hat{\beta} < -\lambda \\ 0 & \text{se } -\lambda \le \hat{\beta} \le \lambda \\ \hat{\beta} - \lambda & \text{se } \hat{\beta} > \lambda \end{cases}$$

$$= \operatorname{sgn}(\hat{\beta}) \cdot (|\hat{\beta}| - \lambda)_+$$

□

**Estensione nel caso di $p$ esplicative**

Al variare di $k$, si calcolano i residui $r_i^k = y_i - \sum_{j \neq k} x_{ij}\hat{\beta}_j$ e si applica la funzione *soft-threshold* sulle coppie di nuovi dati $(x_{ij}, r_i^k)$. In particolare, si riscrive la funzione come

$$R(\tilde{\beta}, \beta_j) = \frac{1}{2}\sum_{i=1}^{n}\Big(y_i - \overbrace{\sum_{j \neq j} x_{ik}\tilde{\beta}_k}^{\tilde{y}^i} - x_{ij}\beta_j\Big)^2 + \lambda\sum_{k \neq j}|\tilde{\beta}_k(\lambda)| + \lambda|\beta_j|,$$

che si può pensare come un problema di lasso univariato rispetto ai residui $y_i - \sum_{k \neq j} x_{ik}\tilde{\beta}_k$. La soluzione esplicita porta a

$$\tilde{\beta}_j(\lambda) = S\left(\sum_{i=1}^{n} x_{ij}(y_i - \tilde{y}^i); \lambda\right),$$

dove $S(t, \lambda) = \text{sgn}(t)(|t| - \lambda)_+$ è l'operatore di *soft-thresholding*. Per calcolare il valore di $\lambda$ ottimale, è necessario scegliere una griglia di valori di $\lambda$ e utilizzare le stime relative a $\lambda_{t-1}$ come valori iniziali per stimare $\tilde{\beta}(\lambda_t)$.

Nonostante si debba esplorare una griglia di valori di $\lambda$, in generale si ottiene un algoritmo più efficiente del LARS, specialmente per valori elevati di $p$.

## 11.3 Scelta del parametro

Generalmente, i parametri di regolazione del modello si scelgono attraverso la convalida incrociata, tramite la quale otteniamo media ed errore standard dell'errore di stima.

Di solito si prende, come stima conservativa del parametro di complessità, il134Gvalore più basso a distanza di un errore standard dalla stima migliore.

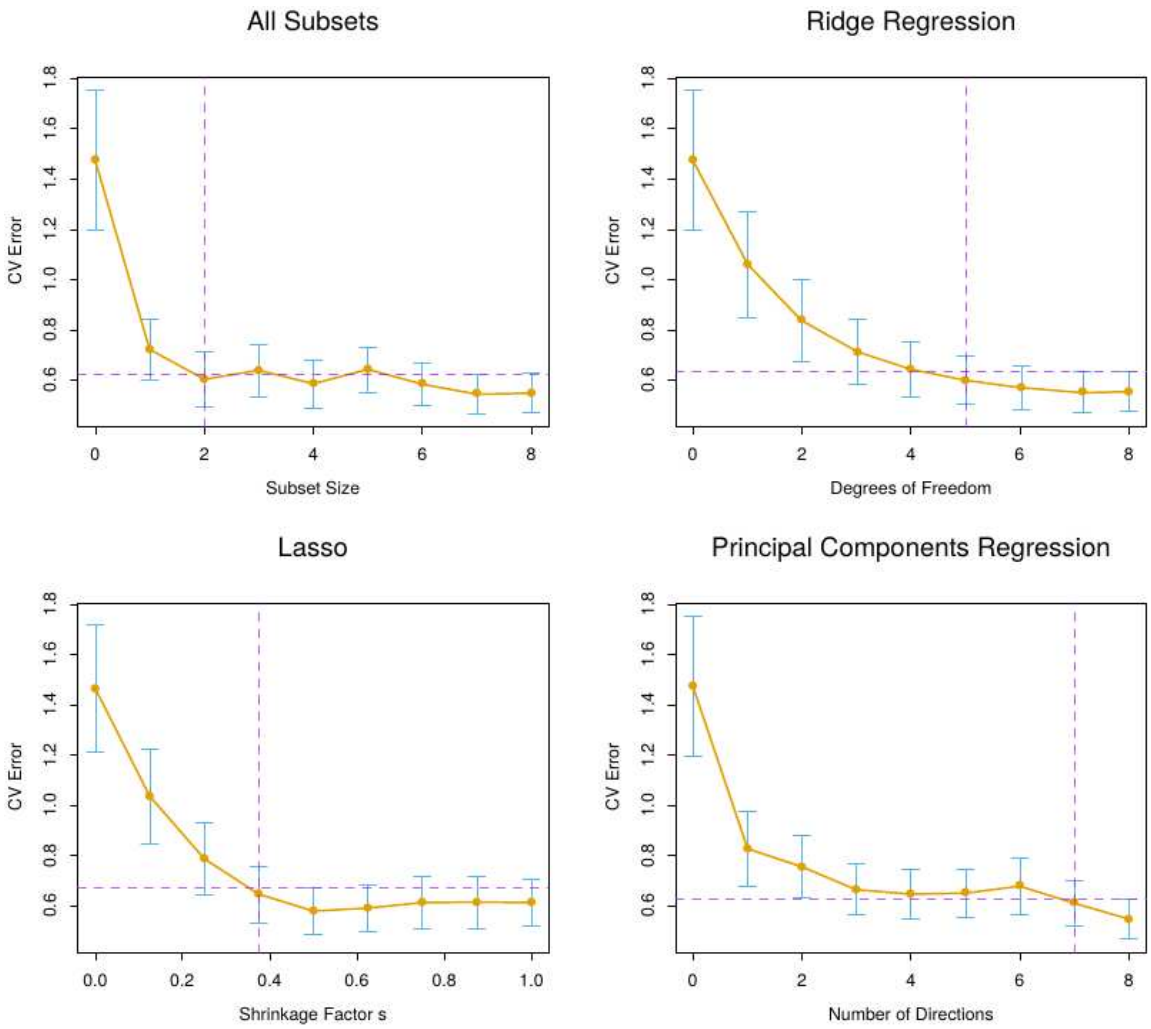


Figura 17: Scelta del parametro di complessità ottimo con la *one-standard-deviation rule*.

# Lezione 12

## 12.1 Regressione non parametrica

*Riferimenti* Hastie et al., (2013, §6.1)

La regressione non parametrica è particolarmente utile quando disponiamo di una grande quantità di osservazioni, tale da permetterci di rilassare le assunzioni lineari e specificare invece un modello flessibile.

Si consideri un vettore di esplicative $x \in \mathbb{R}^p$, scegliamo $f$ in modo che minimizzi l'errore quadratico medio

$$\mathbb{E}\left[\big(y - f(x)\big)^2\right],$$

la cui soluzione di minimo è la *funzione di regressione*

$$f(x) = \mathbb{E}\left[Y|X = x_0\right].$$

Dal punto di vista matematico, senza specificare alcuna assunzione parametrica, abbiamo comunque un minimizzatore dell'errore quadratico medio. Purtroppo, questa regola non aiuta nella pratica, perché non osserviamo $(x_i, y_i)$ per tutti i possibili valori dello spazio $\mathcal{X}$, ma solo per un sottoinsieme finito.

### *k*-nearest-neighbours

Quello che si può invece fare è utilizzare un'approssimazione della funzione di regressione, attraverso la media delle $y_i$ le cui $x_i$ sono in un *intorno* $N(x_0)$ di $x_0$. Lo stimatore per la funzione è dunque

$$\hat{f}(x) = \text{Ave}\big(y_i|x_i \in N(x)\big).$$

Come intorno, si intendono tutte le osservazioni vicine a $x_i$, in riferimento a una data misura di distanza $d(x_i, z) = \|x_i - z\|$.

**Assunzioni** L'assunzione implicita del modello non parametrico è che in valori vicini di $x_i$, la variabile risposta $y$ si comporti in modo abbastanza simile.

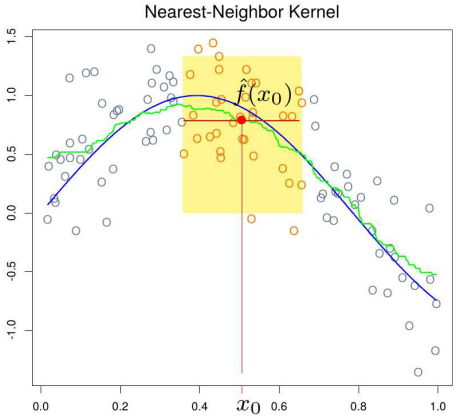


Figura 18: Stima (*verde*) della funzione tramite $k$-nearest-neighbours.

## 12.2 Local linear regression

La regressione lineare locale generalizza il metodo dei $k$-nearest-neighbours, pesando maggiormente il contributo delle $x$ vicine a $x_0$ attraverso una *funzione di peso*. Considerando un modello

$$y = f(x) + \varepsilon,$$

con $\mathbb{E}\left[\varepsilon\right] = 0$, e assumendo che $f(x)$ sia una funzione derivabile, si può scrivere lo sviluppo in serie di Taylor di $f$ in un intorno di $x_0$:

$$f(x) = \underbrace{f(x_0)}_{\beta_0} + \underbrace{f'(x_0)}_{\beta_1}(x - x_0) + o(\|x - x_0\|^2).$$

Questa formulazione suggerisce che, se la funzione è sufficientemente regolare, allora si può approssimare *localmente* con una retta, incorporando il termine $o(\|x - x_0\|^2)$ in $\varepsilon$. Per imporre la condizione di località, i minimi quadrati vengono modificati da una funzione che pesa maggiormente le osservazioni vicine a $x_0$:

$$\min_{\alpha,\beta} \sum_{i=1}^{n} \{y_i - (\beta_0 + \beta_1(x_i - x_0))\}^2 \, w(x_i - x_0; h),$$

per una certa funzione di peso (*nucleo*) $w(\cdot; h)$, dove $h$ è un parametro che determina il grado di lisciamento della funzione risultante.

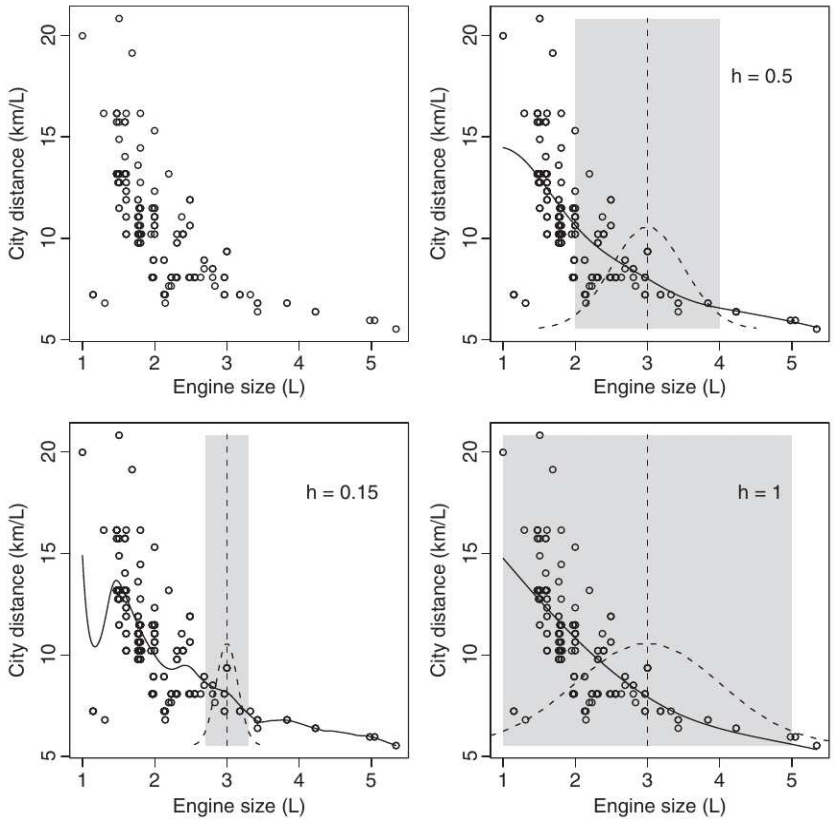


Figura 19: Stima della funzione tramite kernel gaussiano, per diverse scelte del parametro di lisciamento $h$.

La stima si ottiene dai minimi quadrati generalizzati, tramite

$$\hat{\beta} = (X^\mathsf{T} W X)^{-1} X^\mathsf{T} W y,$$

dove $X_{n\times 2} = \big(1, (x_i - x_0)\big)$ e $W = \operatorname{diag}\left(w(x_i - x_0; h)\right)$. Nel caso in cui si cerchi la stima per un valore $x$ generico, si può dimostrare che

$$\begin{aligned}\hat{f}(x) &= \frac{1}{n}\sum_{i=1}^{n} \frac{\{a_2(x;h) - a_1(x;h)(x_i - x)\}\, w(x_i - x; h)}{a_2(x;h)a_0(x;h) - a_1(x;h)^2} y_i, \\ &= s_h^\mathsf{T} y\end{aligned}$$

dove $a_r(x;h) = \frac{1}{n}\sum_{i=1}^{n}(x_i - x)^r w(x_i - x; h)$. Calcolando $\hat{f}(x)$ in $m$ punti equispaziati, si può utilizzare la singola operazione matriciale

$$\underbrace{\hat{f}(x)}_{m\times 1} = \underbrace{S_h}_{m\times n}\underbrace{y}_{n\times 1}$$

**Osservazioni**

- Si tratta di una stima esplicita, non iterativa e lineare nelle $y_i$, per cui si può utilizzare la formula semplice per la leave-one-out cross-validation.
- Se $n$ è molto grande, si può ridurre la dimensione della matrice tramite un raggruppamento in classi della variabile $x$, ottenendo una matrice di dimensione $m \times n'$.
- Dal momento che abbiamo una coppia di parametri $(\beta_0, \beta_1)$ per ciascuno dei (teoricamente) infiniti valori di $x$ che vogliamo stimare, si tratta di una regressione *non parametrica*.
- Nel momento in cui rendo $h$ variabile, in modo da stimarlo attraverso le osservazioni di $Y$, lo stimatore non è più lineare.

# Lezione 13

## 13.1 Scelta del kernel

La scelta della funzione nucleo $w(\cdot)$ è invece poco rilevante, in quanto è sufficiente utilizzare una funzione che sparisca agli estremi.

$$w(t;h) = \frac{1}{h} w_0\left(\frac{t}{h}\right).$$

Tabella 2: Possibili scelte di $w_0(\cdot)$ per la regressione locale

| | |
|---|---|
| Normale | $\mathcal{N}(0,1)$ |
| Biquadratica | $\frac{15}{16}(1-t^2)^2 \mathbb{1}_{(-1,1)}(t)$ |
| Tricubica | $\frac{70}{81}(1-\lvert t\rvert^3)^3 \mathbb{1}_{(-1,1)}(t)$ |
| Epanechnikov | $\frac{3}{4}(1-t^2)\mathbb{1}_{(-1,1)}(t)$ |

Spesso avere supporto limitato riduce l'onere computazionale, così come il calcolo delle potenze invece della funzione esponenziale. Il nucleo di Epanechnikov è il kernel con le proprietà più interessanti, a livello teorico.

## 13.2 Scelta del parametro di lisciamento

L'elemento cruciale di questo metodo di regressione è il parametro $h$, che regola il lisciamento della curva risultante. Si può dimostrare che, per $n \longrightarrow \infty$, valgono i seguenti risultati:

$$\mathbb{E}\left[\hat{f}(x)\right] \approx f(x) + \frac{h^2 \sigma_w^2}{2} f''(x)$$

$$\mathbb{V}\left[\hat{f}(x)\right] \approx \frac{\sigma^2}{nh} \frac{\alpha(w)}{g(x)},$$

dove $g(x)$ è la densità da cui sono campionate le $x$, $\sigma_w^2 = \int z^2 w(z)\, dz$, $\alpha(w) = \int w(z)^2\, dz$.

La distorsione è allora $O(h^2)$ e la varianza è $O((nh)^{-1})$, per cui abbiamo ancora un *conflitto distorsione-varianza*. Certamente è necessario che $n \longrightarrow \infty \implies h \longrightarrow 0$, ma la relazione non specifica la scelta di $h$.

**Nota** Non si può minimizzare l'EQM direttamente, in quanto $\mathbb{E}\left[\hat{f}(x)\right] - f(x)$ dipende da $f''(x)$.

Per ovviare a questo problema, si può usare allora la suddivisione dei dati in stima e verifica, oppure la leave-one-out cross-validation, dal momento che è uno stimatore lineare. La stima asintoticamente ottima per $h$, che purtroppo coinvolge termini non stimabili, è (esercizio)

$$h_{\text{opt}} = \left(\frac{\sigma^2 \alpha(w)}{\sigma_w^4 f''(x)^2 g(x) n}\right)^{1/5}.$$

*Dim.*
Asintoticamente,

$$\begin{aligned}\mathrm{EQM}_h\big(\hat{f}(x)\big) &= \mathbb{V}\left[\hat{f}(x)\right] + \left\{\mathbb{E}\left[\hat{f}(x)\right] - f(x)\right\}^2 \\ &\approx \frac{\sigma^2}{nh}\frac{\alpha(w)}{g(x)} + \left\{\frac{h^2}{2}\sigma_w^2 f''(x)\right\}^2 \\ &= \frac{\sigma^2}{nh}\frac{\alpha(w)}{g(x)} + \frac{h^4}{4}\sigma_w^4 f''(x)^2,\end{aligned}$$

per cui il minimo si ottiene derivando rispetto ad $h$:

$$\frac{\partial \mathrm{EQM}_h\big(\hat{f}(x)\big)}{\partial h} = -\frac{\sigma^2}{nh^2}\frac{\alpha(w)}{g(x)} + \frac{\cancel{4}h^3}{\cancel{4}}\sigma_w^4 f''(x)^2,$$

che è pari a 0 se

$$h^5 n\sigma_w^4 f''(x)^2 g(x) = \sigma^2\alpha(w) \iff h_{\mathrm{opt}} = \left(\frac{\sigma^2\alpha(w)}{f''(x)^2\sigma_w^4 g(x) n}\right)^{1/5}$$

□

In ogni caso, il meglio che si può ottenere per l'EQM con questa stima è

$$\mathbb{E}\left[\big(f(x) - \hat{f}(x)\big)^2\right] = O(n^{-4/5}),$$

che si può confrontare con il modello lineare, il quale ha un errore quadratico medio dell'ordine $O(n^{-1})$. Questo significa che, per $n \to \infty$, se il modello è corretto, la regressione locale ha un errore quadratico medio asintoticamente maggiore. Quindi, se so che il modello corretto è lineare, sbaglio di meno con il modello lineare rispetto a quello non parametrico.

**Bande di variabilità**

Utilizzare gli intervalli di confidenza del tipo

$$\left(\hat{\vartheta} - z_{1-\frac{\alpha}{2}}\mathrm{se}(\hat{\vartheta}), \hat{\vartheta} + z_{1-\frac{\alpha}{2}}\mathrm{se}(\hat{\vartheta})\right)$$

è appropriato solamente nel caso in cui lo stimatore sia esattamente o asintoticamente non distorto, cioè $\mathbb{E}\left[\hat{\vartheta}\right] = \vartheta$. Siccome nel termine di distorsione è presente $f''(x)$, siamo in grado di stimare la variabilità ma non il bias, quindi non è possibile nemmeno costruire intervalli di confidenza approssimati.

Invece usare correzioni complicate per tenere conto della distorsione, possiamo calcolare questi intervalli ugualmente per ottenere delle *bande di variabilità*, che ignorano la distorsione ma danno un'indicazione del grado di precisione con cui si stima la curva.

(a) Non sono intervalli di confidenza.

(b) Valgono puntualmente ma non globalmente per la funzione.

## 13.3 Parametro di lisciamento variabile e LOESS

### Ampiezza di banda variabile

Dal momento che la variabilità della stima dipende inversamente dalla densità $g(x)$, come abbiamo visto dalla formula

$$\mathbb{V}\left[\hat{f}(x)\right] \approx \frac{\sigma^2}{nh} \cdot \frac{\alpha(w)}{g(x)},$$

in molti casi è vantaggioso permettere che $h$ sia variabile in base alla concentrazione locale delle $x$.

La regressione *loess* viene effettuata prefissando una percentuale costante di punti rilevanti da includere nel kernel: questo significa che la finestra del kernel ($h$) viene ampliata o contratta, in base alla concentrazione locale dei valori di $x$.

I kernel con supporto limitato sono più adatti alla regressione loess, visto che distinguono chiaramente tra punti utilizzati e non utilizzati. Come default, R usa il kernel biquadratico per la stima della loess.

Dal momento che si rischia di ottenere una stima poco robusta rispetto agli outlier e influenzata da osservazioni lontane, la procedura di stima non si basa sui minimi quadrati, bensì su un procedimento di stima robusta.

# Lezione 14

## 14.1 Regressione lineare locale in 2D

Consideriamo ora due variabili esplicative, e un modello

$$y = f(x_1, x_2) + \varepsilon,$$

l'estensione della regressione lineare locale è, assumendo che il kernel bidimensionale sia un prodotto di kernel univariati,

$$\min_{\alpha,\beta,\gamma} \sum_{i=1}^{n} \{y_i - \beta_0 - \beta_1(x_{i1} - x_{01}) - \beta_2(x_{2i} - x_{02})\}\, w(x - x_1; h_1) w(x - x_2; h_2).$$

In forma matriciale, scrivendo una matrice di regressione

$$X = [1,\ (x_{i1} - x_{01}),\ (x_{i2} - x_{02})]$$

e $W$ matrice diagonale di pesi, allora la stima è di nuovo data da

$$\hat{f}(x) = (X^\mathsf{T} W X)^{-1} X^\mathsf{T} W y.$$

Se vogliamo stimare la superficie in $k$ punti del piano, possiamo farlo con un unico insieme di operazioni matriciali e l'operatore è ancora una volta lineare.

Come nel caso, precedente, non è possibile estrapolare la funzione al di fuori della chiusura convessa del supporto osservato (perché?).

**Motivo** Probabilmente perché estrapolare dipende fortemente dai valori osservati agli estremi della chiusura convessa. Inoltre, l'estrapolazione è costante man mano che ci si allontana.

## 14.2 Maledizione della dimensionalità

Al crescere di $p$, i punti osservati diventano sempre più rarefatti, anche se $n$ è grande: per $x_0$ fissato, un intorno di $x_0$ contiene un numero piccolissimo di osservazioni al crescere di $p$. In particolare

- Gli intorni locali sono vuoti, oppure
- Gli intorni con vicini più prossimi non sono locali
- Tutti i punti sono vicini alla frontiera

Per *qualunque* metodo non parametrico, vale questa limitazione.

Esercizio: perché questo non è un problema nel modello parametrico lineare?

#### Formulazione probabilistica

Sia $x = (x_1, x_2, \ldots, x_p)$ e un insieme di stima $\{y_i, x_i\} \in \mathbb{R}^{p+1}$, vogliamo stimare una funzione in $\mathbb{R}^p$ più vicino possibile ai dati. Se $f(x)$ è (quasi) arbitrariamente complessa, è necessario un campione

*denso* per poterla specificare bene.

Consideriamo $n$ punti da $X \sim \text{Unif}_p(0,1)$, allora la distanza mediana dall'origine al punto più vicino è (esercizio)

$$r(p,n) = \left(1 - 0.5^{\frac{1}{n}}\right)^{\frac{1}{p}}.$$

*Dim.*
Sia $m$ la distanza mediana al più vicino degli $n$ punti, allora poiché la distribuzione è uniforme, si ha che $P(X_i \geq m) = 1 - m$. Quindi,

$$P(X_i \geq m \text{ per ogni } i) = (1-m)^n \stackrel{?}{=} 0.5 \iff m = 1 - 0.5^{\frac{1}{n}}.$$

In due dimensioni, la probabilità di osservare punti a distanza $m$ è data dal rapporto dell'area infinitesimale rispetto all'area totale:

$$\frac{\pi m^2}{\pi} = m^2,$$

dunque

$$P(X_i \geq m \text{ per ogni } i) = (1-m^2)^n \stackrel{?}{=} 0.5 \iff m = \left(1 - 0.5^{\frac{1}{n}}\right)^{\frac{1}{p}}.$$

Analogamente si può estendere a $p$ dimensioni, visto che il volume è

$$V(p) = \frac{\pi^{\frac{p}{2}}}{\frac{p}{2}\Gamma\left(\frac{p}{2}+1\right)} R^p$$

e i termini si cancellano facendo il rapporto.

□

Provando con un po' di numeri, la distanza mediana del punto più vicino dall'origine, in un cubo di lato 1, è

$$r(10, 500) \approx 0.52$$

$$r(10, 1000) \approx 0.48$$

$$r(11, 1000) \approx 0.52$$

$$r(20, 1000) \approx 0.69$$

$$r(100, 1000) \approx 0.93$$

In pratica, la maggior parte dei punti è molto più vicina al bordo dello spazio campionario che ad ogni altro punto: la previsione è molto più difficile vicino ai bordi (estrapolazione contro interpolazione).

**Formulazione geometrica**

Considero la sfera $d$-dimensionale di raggio r, il suo volume è

$$V(d) = \frac{\pi^{\frac{d}{2}}}{\frac{d}{2}\Gamma\left(\frac{d}{2}+1\right)} R^d.$$

Dall'approssimazione di Stirling, per $n$ intero, si ha che

$$\Gamma(n+1) \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n,$$

per cui il denominatore cresce molto più velocemente del numeratore e $\lim_{d\to\infty} V(d) = 0$. Dunque, in alte dimensioni, il volume si concentra agli angoli dello spazio.

Le distanze tra i punti sono all'incirca dist $\approx (\text{vol})^{\frac{1}{p}}$: considero $X \sim \text{Unif}_p(0,1)$ e un sotto-cubo dall'origine che comprende una frazione $d$ dei dati. Quale distanza bisogna coprire su ciascun asse? Con 10 variabili, per avere l'1% delle osservazioni, bisogna coprire il $0.01^{\frac{1}{10}} \approx 63\%$ del campo di variazione di ciascuna variabile esplicativa.

Le funzioni in elevate dimensioni hanno la potenzialità di essere molto più complicate di quelle in piccole dimensioni, e queste complicazioni sono difficili da identificare.

**Nota** questo problema è intrinseco di qualunque modello non parametrico, non solo della regressione locale.

Per poter sconfiggere la "maledizione", è necessario incorporare *assunzioni corrette* sulla $f(x)$, che vadano oltre ai dati. Nel momento in cui si sceglie un metodo (CART, rete neurale, ...), si impone una serie di assunzioni e ciascun modello si differenzia per:

1. Natura delle assunzioni imposte.
2. Forza delle assunzioni imposte.
3. Robustezza rispetto a violazioni degli assunti.

# Lezione 15

## 15.1 Funzioni splines

Una *spline* è una funzione matematica, che ha una naturale applicazione nell'approssimazione di funzioni arbitrariamente complesse.

Si consideri un problema di interpolazione, in cui ci sono $(\xi_k, y_k)_{k=1,\dots,K}$ coppie di *nodi* e osservazioni da interpolare con una funzione. La funzione scelta può essere costituita da una serie di segmenti (funzione di regressione), oppure una serie di funzioni più lisce. Nel caso dell'interpolazione tramite splines, si cerca una funzione polinomiale a tratti di grado $d$ liscia, ottenibile tramite l'imposizione della continuità fino alla derivata $(d-1)$-esima.

### Polinomi cubici

I polinomi cubici sono universalmente usati, in quanto l'occhio umano non riesce a distinguere discontinuità nelle derivate superiori a 2.
Si impone una struttura polinomiale a tratti nei $(K-1)$ intervalli $(\xi_i, \xi_{i+1})$, che soddisfi

$$\begin{aligned} &f(\xi_k) = y_k && k = 1, \dots, K \\ &f^{(j)}(\xi_k^+) = f^{(j)}(\xi_k^-) && k = 2, \dots, K-1 \quad j = 0, \dots, d-1. \end{aligned}$$

Nel caso del polinomio cubico, ho

- $4(K-1)$ parametri da stimare
- $K$ vincoli per interpolare
- $3(K-2)$ vincoli sulle derivate

In totale, ci sono $4(K-1) - \{K + 3(K-2)\} = 2$ parametri ancora liberi, per cui la soluzione non è univocamente identificata. Ci sono molte possibilità per il vincolo, ad esempio le *splines cubiche naturali* utilizzano

$$f''(\xi_1) = f''(\xi_K) = 0,$$

il che significa che i polinomi agli estremi sono lineari.

## 15.2 Splines di regressione

Esistono due modi diversi per utilizzare queste splines nella regressione, per cui bisogna stare attenti alle differenze.

Nelle splines di *regressione*, si scelgono $K \ll n$ nodi, in modo da suddividere l'asse delle $x$ in $K+1$ intervalli, su cui effettuare un'interpolazione via splines. Dal momento che non c'è più legame tra le coppie $\xi_i$ e $y_i$, non si possono utilizzare i $K$ vincoli $f(\xi_i) = y_i$. In particolare, si hanno

- $4(K+1)$ parametri da stimare.
- $3K$ vincoli sulle derivate.

Dunque, in totale si hanno $4(K+1) - 3K = K + 4$ gradi di libertà nelle funzioni da stimare, per le quali è possibile dimostrare che valga la rappresentazione

$$f(x) = \sum_{j=1}^{K+4} \beta_j h_j(x),$$

dove la *base di funzioni* è costituita da

$$h_j(x) = x^{j-1} \qquad j = 1, \ldots, 4$$

$$h_j(x) = \left(x - \xi_j\right)_+^3 = \max\left\{0, (x - \xi_j)^3\right\} \qquad j = 1, \ldots, K$$

Nel caso di due nodi $\xi_1, \xi_2$, la base è data da

$$\begin{aligned} h_1(x) &= 1 \\ h_2(x) &= x \\ h_3(x) &= x^2 \\ h_4(x) &= x^3 \\ h_5(x) &= (x - \xi_1)_+^3 \\ h_6(x) &= (x - \xi_2)_+^3 \end{aligned}$$

Interpretando le splines come base di funzioni, è possibile fare i minimi quadrati senza vincoli in maniera molto semplice, trasformando opportunamente la variabile $x$ nella matrice $X$ di regressione.

**Nota** Visto che ci sono $K+4$ parametri da stimare, in che modo si differenzia dalla solita regressione lineare? La differenza è l'elevato *grado di flessibilità* di queste curve, quasi quanto i modelli non parametrici.

**Nota 2** Poiché il grado del polinomio a tratti è fissato e pari a 3, la misura di complessità delle splines di regressione è il numero di nodi, se questi sono equispaziati. Visto che abbiamo una struttura di modello parametrico, il numero di nodi si può scegliere sia tramite stima-verifica, sia tramite criteri di informazione.

## 15.3 Splines di lisciamento

Consideriamo i minimi quadrati penalizzati della forma

$$D(f, \lambda) = \sum_{i=1}^{n} \{y_i - f(x_i)\}^2 + \lambda \int_{-\infty}^{\infty} f''(t)^2 \, dt.$$

La penalizzazione sulla derivata seconda misura quanto la funzione è regolare, da un estremo $\lambda \to 0$ dato dalla media condizionata, ad una funzione lineare con $f''(x) \equiv 0$ per $\lambda \to \infty$. Il comportamento di $\lambda$ è simile al parametro di lisciamento $h$ della regressione lineare locale.

È possibile dimostrare che la soluzione di $\min_f D(f, \lambda)$ sono le splines naturali cubiche, i cui nodi sono i punti $x_i$, che si possono scrivere nella forma

$$f(x) = \sum_{j=1}^{n_0+4} N_j(x)\vartheta_j = N\vartheta,$$

dove $n_0$ è il numero di $x_i$ distinti e $N_j$ sono le basi delle splines naturali. In particolare, sostituendo la soluzione in $D(N\vartheta, \lambda)$, si ha

$$D(f, \lambda) = (y - N\vartheta)^\mathsf{T}(y - N\vartheta) + \lambda\vartheta^\mathsf{T}\Omega\vartheta,$$

dove $\Omega$ è la matrice con generico elemento $\int N_j''(t)N_k''(t)\,dt$. La forma è equivalente al problema ridge con vincolo ellittico, che ha soluzione

$$\hat{\vartheta} = (N^\mathsf{T}N + \lambda\Omega)^{-1}N^\mathsf{T}y.$$

La soluzione è quindi

$$\hat{y} = N(N^\mathsf{T}N + \lambda\Omega)^{-1}N^\mathsf{T}y = S_\lambda y,$$

per cui si tratta di un lisciatore lineare se $\lambda$ è fissato.

Splines di regressione e lisciamento sono diversi, in quanto

Tabella 3: caption

| **Regressione** | **Lisciamento** |
|---|---|
| $k$ nodi (variabili) | $n_0$ nodi (fissi) |
| Minimi quadrati (fissi) | Ridge ($\lambda$ variabile) |

## 15.4 Splines multivariate

**Lisciamento: Thin-plate splines**

L'estensione delle splines al caso multivariato non è immediata, ad esempio si può scegliere di imporre la penalizzazione di $D(f, \lambda)$ pari a

$$\lambda \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left\{ \left(\frac{\partial^2 f(x)}{\partial^2 x}\right)^2 + \left(\frac{\partial^2 f(x)}{\partial x \partial y}\right)^2 + \left(\frac{\partial^2 f(x)}{\partial^2 y}\right)^2 \right\} dx\,dy$$

**Regressione: Splines prodotto tensoriale**

Generalizzazione delle splines di regressione è l'utilizzo del prodotto tensoriale: se $h_{1k}(x_1)$ e $h_{2k}$ sono le funzioni splines di base delle due variabili $x_1$ e $x_2$, la base prodotto tensoriale è

$$g_{jk}(x) = h_{1j}(x_1)h_{2k}(x_2), \quad j = 1, \ldots, K_1 + 4, \quad k = 1, \ldots, K_2 + 4.$$

Dunque, la funzione bidimensionale scegliendo polinomi di terzo grado è

$$g(x) = \sum_{j=1}^{K_1+4} \sum_{k=1}^{K_2+4} \vartheta_{jk} g_{jk}(x) = \sum_{j=1}^{K_1+4} \sum_{k=1}^{K_2+4} \vartheta_{jk} h_j(x_1) h_k(x_2),$$

da cui si possono ricavare le stime di $\vartheta_{jk}$ tramite minimi quadrati.

Ulteriori generalizzazioni a dimensioni più elevate sono possibili, ma si incorre nel solito problema di maledizione della dimensionalità.

# Lezione 16

## 16.1 Gradi di libertà equivalenti

Tutti questi metodi possono essere degli stimatori lineari, nel senso che i valori previsti sono

$$\hat{y} = S_\lambda y$$

per una certa matrice $S_\lambda$ che non dipende da $y$, con $\lambda$ parametro di lisciamento. Si osserva che, se la stima di $\lambda$ si fa tramite $y$, lo stimatore non è più lineare: per questo motivo, si usano dati nuovi per stimare il valore di $\lambda$.

Dalla teoria dei modelli lineari, i valori previsti sono

$$\hat{y} = Py,$$

dove $P = X(X^\mathsf{T}X)^{-1}X^\mathsf{T}$ è una matrice di proiezione, cioè simmetrica, idempotente e di rango $p = \operatorname{tr}(P)$. Ricordiamo inoltre che

$$\mathbb{E}\left[\|\hat{\varepsilon}\|^2\right] = \sigma^2(n-p).$$

Da qui, con l'aggiunta di ipotesi di normalità,

$$\|\hat{\varepsilon}\|^2 \sim \sigma^2\chi^2_{n-p}, \quad \|\hat{y}\|^2 = y^\mathsf{T}y \sim \sigma^2\chi^2_p(\delta),$$

dove $\delta$ è un parametro di non centralità. Grazie a questi risultati, si può scomporre la variabilità totale in

$$\|y\|^2 = \|\hat{\varepsilon}\|^2 + \|\hat{y}\|^2,$$

e il termine $\|\hat{y}\|^2$ si può scomporre in componenti individuali, per ciascuna variabile esplicativa, con corrispondente scomposizione dei gradi di libertà.

Nel caso di stima non parametrica, vogliamo dei risultati simili partendo da

$$\hat{y} = S_h y, \quad S_h \text{ matrice } n \times n.$$

con residui corrispondenti $\hat{\varepsilon} = (I - S_h)y$. Anche assumendo normalità di $\varepsilon$, la distribuzione di $\|\hat{\varepsilon}\|^2$, $\|\hat{y}\|^2$, ecc... non risultano più della forma $\chi^2$. Si può tuttavia argomentare che, per $\|\hat{\varepsilon}\|^2$, sussiste un andamento non troppo lontano da quello di un $\chi^2$. Scriviamo allora le distribuzioni approssimate

$$\|\hat{\varepsilon}\|^2 \overset{\text{app.}}{\sim} \sigma^2\chi^2_?, \quad \|\hat{y}\|^2 \overset{\text{app.}}{\sim} \sigma^2\chi^2_{??}(\delta),$$

per delle opportune scelte di gradi di libertà. Cerchiamo allora una forma di "gradi di libertà" approssimati: per una variabile $\chi^2$, i gradi di libertà sono il valore atteso della distribuzione. Nel nostro caso,

$$Q = \sum_{i=1}^n \hat{\varepsilon}_i^2 = y^\mathsf{T}(I_n - S_h)^\mathsf{T}(I_n - S)y,$$

per cui

$$\begin{aligned}\mathbb{E}\left[Q\right] &= \mathbb{E}\left[Y^\mathsf{T}(I_n - S)^\mathsf{T}(I_n - S)Y\right] \\ &= \mu^\mathsf{T}(I_n - S)(I_n - S)\mu + \sigma^2 \operatorname{tr}\left((I_n - S)^\mathsf{T}(I_n - S)\right)\end{aligned}$$

usando la formula per il valore atteso di una forma quadratica.

$$\mathbb{E}\left[X^\mathsf{T}AX\right] = \mu^\mathsf{T}A\mu + \operatorname{tr}(AV)$$

dove $\mu = \mathbb{E}\left[X\right]$ e $V = \mathbb{V}\left[X\right]$.

*Dim.*
Siano $A$ una matrice simmetrica definita positiva, $X \in \mathcal{X} \subseteq \mathbb{R}^p$ una variabile casuale, allora

$$\begin{aligned}\mathbb{E}\left[X^\mathsf{T}AX\right] &= \mathbb{E}\left[\sum_{i=1}^{p}\sum_{j=1}^{p} a_{ij}X_iX_j\right] \\ &= \sum_{i=1}^{p}\sum_{j=1}^{p} a_{ij}\,\mathbb{E}\left[X_iX_j\right] && \text{(linearità)} \\ &= \sum_{i=1}^{p}\sum_{j=1}^{p} a_{ij}\big(\sigma_{ij} + \mu_i\mu_j\big) && \text{(formula covarianza)} \\ &= \sum_{i=1}^{p}\sum_{j=1}^{p} a_{ij}\sigma_{ji} + \sum_{i=1}^{p}\sum_{j=1}^{p} a_{ij}\mu_i\mu_j && (\Sigma \text{ matrice simmetrica}) \\ &= \sum_{i=1}^{p}\left[A\Sigma\right]_{ii} + \mu^\mathsf{T}A\mu \\ &= \operatorname{tr}(A\Sigma) + \mu^\mathsf{T}A\mu.\end{aligned}$$

□

Dalla formula di $\mathbb{E}\left[Q\right]$, possiamo provare a simplificarla con delle approssimazioni ragionevoli:

$$\begin{aligned}&(I_n - S)\mu \approx 0 && \text{(come modello lineare)} \\ &(I_n - S)^\mathsf{T}(I_n - S) \approx I_n - S && \text{(come fosse proiezione)}\end{aligned}$$

per cui si ottiene

$$\begin{aligned}\mathbb{E}\left[Q\right] &= \mu^\mathsf{T}\underbrace{(I_n - S)(I_n - S)}_{\approx I_n - S}\mu + \sigma^2 \operatorname{tr}\Big(\underbrace{(I_n - S)^\mathsf{T}(I_n - S)}_{\approx I_n - S}\Big) \\ &= \underbrace{\mu^\mathsf{T}(I_n - S)\mu}_{\approx 0} + \sigma^2 \operatorname{tr}\left(I_n - S\right) \\ &= \sigma^2\big(n - \operatorname{tr}(S)\big).\end{aligned}$$

Chiamiamo allora $\mathrm{df}(\lambda) = \operatorname{tr}(S_\lambda)$ *gradi di libertà equivalenti per il lisciatore* e $n - \operatorname{tr}(S)$ i *gradi di libertà equivalenti per il termine di errore*. Ci sono altre approssimazioni per i gradi di libertà

equivalenti, ad esempio

$$\operatorname{tr}(SS^\mathsf{T}), \quad \operatorname{tr}(2S - SS^\mathsf{T}),$$

tuttavia questa è la più semplice da ricavare e interpretare.

## 16.2 Modelli additivi generalizzati (GAM)

*Riferimenti* Hastie et al. (2013, §9.1)
Azzalini e Scarpa (2012, §4.5)

Se il numero $p$ di variabili è grande, i dati si disperdono in $\mathbb{R}^p$ e siamo colpiti dalla "maledizione della dimensionalità" se utilizziamo un modello di regressione non parametrico.

Domanda *Perché non succede nel caso del modello lineare?* Perché si impone una forte struttura a priori sul modello, per cui ci si limita ad una singola forma funzionale (iperpiani). Quindi, nel caso parametrico la stima del modello non si basa solo sui punti nell'intorno $N_0(x)$, ma su tutti i punti dello spazio.

Assumendo un modello

$$y = f(x) + \varepsilon,$$

si può definire un *modello additivo* restringendo $f(x)$ ad una rappresentazione come somma di funzioni per singole variabili, coppie, triple, ...

$$f(x) = \alpha + \sum_{i=1}^{p} f_j(x_j) + \sum_{k<j} f_{kj}(x_k, x_j) + \dots$$

Troncando lo sviluppo a termini di primo o secondo grado, si impone una limitazione nella forma di $f(x)$, tuttavia permettendo ampi margini di flessibilità. Nel caso dei termini di primo grado, ci si ferma a

$$y = f(x) + \varepsilon = \alpha + \sum_{j=1}^{p} f_j(x_j) + \varepsilon.$$

La stima di queste funzioni si può fare attraverso un algoritmo di *backfitting*, che si appoggia ad un lisciatore univariato $S$ scelto tra loess, spline, ecc.

### 16.2.1 Algoritmo di backfitting

A priori, si standardizzano le $y$ per evitare problemi di identificabilità del modello. Fissata una direzione $x_j$, si può scrivere

$$f_j(x_j) = y - \big(\alpha + \sum_{k \neq j} f_k(x_k) + \varepsilon\big),$$

per cui si possono considerare questi residui parziali come realizzazioni di $f_j(\cdot)$, che ora è univariata. Si possono allora lisciare i residui parziali nella variabile $x_j$ tramite il lisciatore univariato $S$ scelto.

**Algoritmo 3** Backfitting per un modello additivo (AM)

1: **Init:**
   $\hat{\alpha} = \sum_{i=1}^{n} y_i/n$
   $\hat{f}_j = 0$ per ogni $j = 1, \dots, p$
2: **while** *tol* $> \varepsilon$ **do** $j = 1, \dots, p$
3: $\quad \hat{f}_j = S\big(y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik})\big)$
4: $\quad \hat{f}_j = \hat{f}_j - n^{-1} \sum_{i=1}^{n} \hat{f}_j(x_{ij})$ ▷ $\hat{f}$ ha media nulla, ma a causa di rounding errors conviene farlo
5: **end while**

La logica del modello lineare rimane, nel senso che in ciascuna esplicativa si ottiene l'effetto di $x_j$ al netto di quello che viene stimato nelle altre variabili. Nel caso della regressione lineare i parametri erano numeri, mentre qui sono funzioni.

**Osservazione**

- L'algoritmo di *backfitting* è una variante del metodo di Gauss-Seidel.
- Si può mostrare che l'algoritmo *nel caso semplice* converge, mentre nel caso più complesso (GAM) non c'è dimostrazione formale della convergenza, per cui ci si fiderà dell'output.

Sarebbe interessante poter costruire un test per verificare che una determinata $x_j$ sia significativa nel modello: in analogia con il modello lineare, si può costruire una struttura di ANOVA con $\hat{y}_0$ vettore dei valori adattati nel modello ridotto

$$F = \frac{\|\hat{y} - \hat{y}_0\|^2}{\hat{\sigma}^2} \frac{n-p}{nq},$$

per cui possiamo utilizzare i gradi di libertà equivalenti dati da

$$n - \operatorname{tr}(S), \quad \operatorname{tr}(S) - \operatorname{tr}(S_0),$$

dove $\hat{y} = Sy$ e $\hat{y}_0 = S_0 y$, dunque la distribuzione di riferimento è

$$F_{\operatorname{tr}(S)-\operatorname{tr}(S_0),\ n-\operatorname{tr}(S)}.$$

**Nota** Questa approssimazione è qualitativa, non asintotica come nel caso del teorema del limite centrale.

L'effetto nei grafici è attorno allo zero, visto che si toglie la media, al netto dell'effetto delle altre variabili.

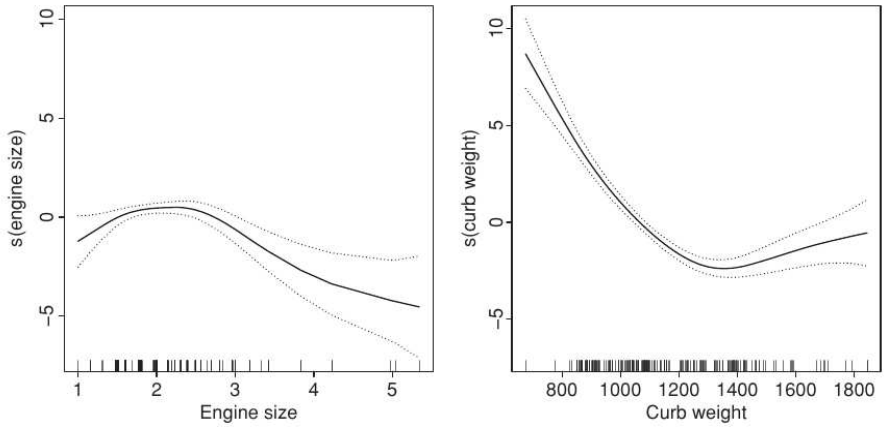


Figura 20: Effetto marginale di ciascuna variabile sulla risposta, al netto delle restanti covariate e della media globale.

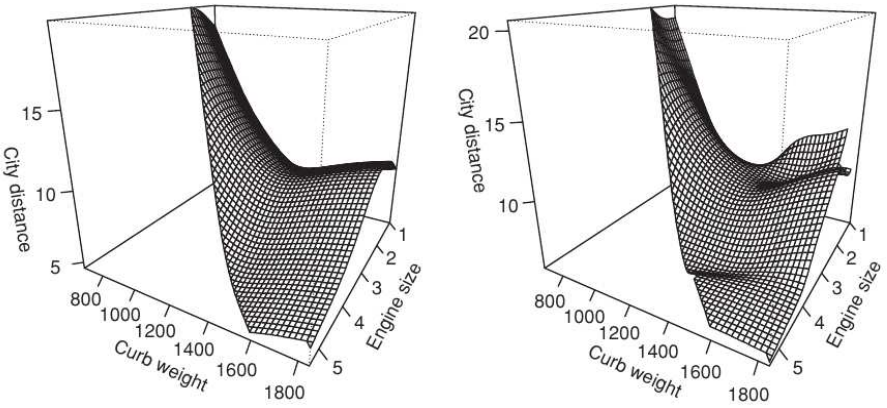


Figura 21: *Sinistra*: modello additivo, non si ammettono interazioni tra le variabili. *Destra*: modello con interazione, la discrepanza rispetto al modello additivo è minima.

Dalla figura, si osserva la caratteristica principale dei modelli additivi: l'additività impone che la forma dell'effetto di $x_j$ sia la stessa al variare del valore delle altre variabili. In particolare, non sono ammessi picchi locali, perché sarebbe necessaria una modifica locale della curva.

L'idea del modello additivo si può generalizzare facilmente al caso dei modelli lineari generalizzati, considerando una formula additiva per il predittore lineare

$$g\big(\mathbb{E}\left[Y|X\right]\big) = \alpha + \sum_{j=1}^{p} f_j(x_j),$$

per la stima del quale si usa una combinazione di backfitting e minimi quadrati pesati iterati.

# Lezione 17

*Riferimenti* Hastie et al. (2013, §11.2)
Azzalini e Scarpa (2012, §4.6)

In generale, il modello additivo non è in grado di catturare curvature non parallele agli assi cartesiani, in quanto sono permessi solo effetti principali nelle singole variabili. Si può allora sviluppare un modello additivo basato su una rotazione degli assi, invece che degli assi stessi, per modellare la funzione.

## 17.1 Projection Pursuit Regression

Sia $x = x_1, x_2, \ldots, x_p$ il vettore delle esplicative e $a \in \mathbb{R}^p$ un vettore unitario. La *regressione projection pursuit* (*PPR*) vuole stimare un modello di tipo

$$f(x) = \sum_{m=1}^{M} f_m(a_m^\mathsf{T} x),$$

dove le funzioni $f_m : \mathbb{R}^p \to \mathbb{R}$ si dicono *funzioni dorsali* (*ridge functions*) e i vettori di rotazione $\{a_m\}_{m=1,\ldots,M}$ non sono necessariamente ortogonali tra di loro. Fissata una rotazione $a$, la corrispondente funzione dorsale si stima sempre attraverso lisciatori univariati, per cui la flessibilità del modello è molto elevata.

Visto che le rotazioni non sono ortogonali, il loro numero $M$ non è fisso e può essere scelto arbitrariamente: in particolare, questo significa che la PPR può approssimare qualsiasi funzione in $\mathbb{R}^p$ arbitrariamente bene e per questo si definisce *approssimatore universale*.

Domanda: *Un modello lineare è un approssimatore universale?* No, perché anche se il modello lineare permette di inserire trasformazioni di variabili, non permette di stimare funzioni diverse da quelle specificate dai parametri.
Il modello PPR permette di catturare qualunque tipo di interazione, perché sono contenute nel fatto che $f_m(a_m^\mathsf{T} x)$ è funzione della combinazione lineare delle $x_j$.

La caratteristica di poter approssimare qualunque funzione significa che il modello coglierà la variabilità locale delle osservazioni, dunque sovradattamento. Giocando sulla scelta di $M$ nei soliti modi (CV, AIC, ...), in quanto parametro di complessità permette di limitare il sovradattamento.

Con questo modello si perde interpretabilità del risultato, in quanto diventa molto complicato dare un'interpretazione sensata dell'effetto delle singole covariate. Tuttavia, è ancora possibile farlo nel caso in cui le combinazioni $a_m^\mathsf{T} x$ abbiano una loro interpretazione come *fattori latenti*.

Ad ogni modo, è improbabile che $a_m^\mathsf{T} x$ sia interpretabile, dal momento che ci si adatta a un campione osservato e per di più suddiviso in stima e verifica.

### 17.1.1 Stima del modello

L'algoritmo di stima consta di uno step di backfitting e uno di ottimizzazione:

1. Si consideri un termine solo $M = 1$.
2. Fissate $f_m$, si possono trovare i parametri $a$ attraverso ottimizzazione numerica (Newton-Raphson).
3. Ottenuto $a$, si effettua la rotazione $z = a^\mathsf{T} x$ e si liscia la funzione $f(z)$ tramite loess, spline, ...
4. Iterazione di 2) e 3) fino a convergenza dell'algoritmo.
5. Se $M > 1$, si può adottare una strategia stepwise forward per selezionare il numero di termini $M$.
6. Una volta identificate le $M$ direzioni, le stime delle funzioni $f_m$ sono riaggiustate tramite backfitting:
   (a) Fisso le direzioni $a$.
   (b) Modifico le funzioni $f_m$ tramite backfitting.

## 17.2 Splines di regressione multivariate adattive (MARS)

*Riferimenti* Hastie et al., (2013, §9.4)
Azzalini e Scarpa (2012, §4.4.5)

La regressione MARS funziona bene in dimensioni elevate, ed è una modifica del metodo CART per la regressione. In particolare, utilizza splines prodotto tensoriale lineari a coppie con un solo nodo nel punto $t$, cioè funzioni di tipo

$$\mathbb{C} = \{(x_j - t)_+, (t - x_j)_+\}_{\substack{t \in \{x_1,\dots,x_p\} \\ j=1,\dots,p}}.$$

Anche se le funzioni in $\mathbb{C}$ dipendono da una singola $x_j$, conviene considerarle come funzioni in $\mathbb{R}^p$.

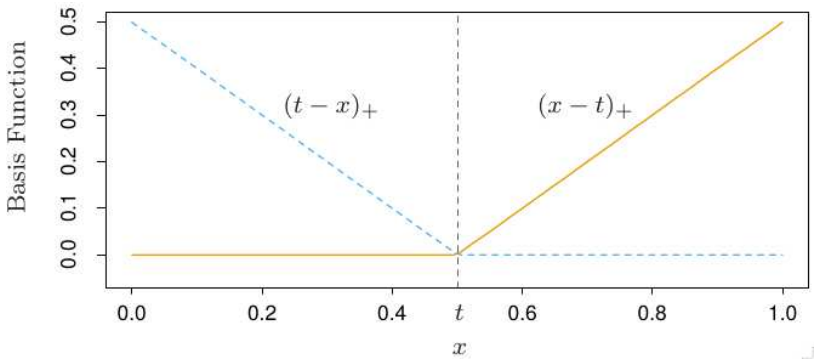


Figura 22: Esempio di funzioni di base $(x - t)_+$ (*arancione*) e $(t - x)_+$ (*blu*).

Si cerca dunque un modello adattivo della forma

$$f(x) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(x),$$

dove le funzioni di base $h_m(x)$ sono funzioni in $\mathbb{C}$ o prodotti di due o più funzioni in $\mathbb{C}$. Fissate le funzioni $h_m(\cdot)$, i corrispondenti coefficienti $\beta$ sono stimati semplicemente con i minimi quadrati.

Siccome una spline prodotto tensoriale porta a scontrarsi con la maledizione della dimensionalità, è necessario scegliere quali funzioni di base far entrare nel modello.

L'algoritmo di stima procede come la regressione stepwise forward, suddividendo le funzioni $h_m$ in due insiemi: $\mathcal{M}$, di funzioni incluse, e $\mathcal{C}$ di funzioni candidate (Figura 23).

- Si inizializzano $M = 0$, $h_0(x) = 1$, ovvero si inizia con la sola intercetta.
- Ad ogni step successivo $M + 1$, si sceglie di introdurre nel modello la coppia di funzioni

$$\hat{\beta}_{M+1} h_\ell(x) \cdot (x_j - t)_+ + \hat{\beta}_{M+2} h_\ell(x) \cdot (t - x_j)_+, \quad h_\ell \in \mathcal{M},$$

  tale che si massimizzi il miglioramento dell'adattamento ai dati. I coefficienti $\hat{\beta}_{M+1}$ e $\hat{\beta}_{M+2}$ sono stimati con i minimi quadrati, come gli altri coefficienti del modello.

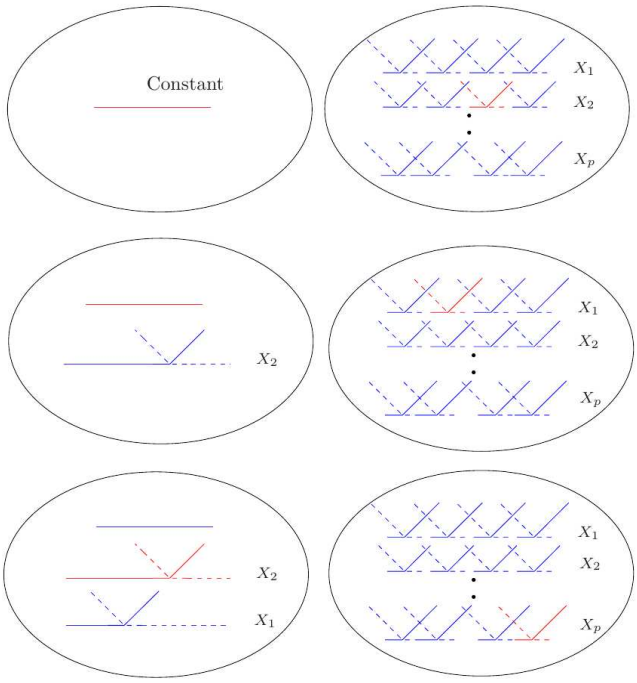


Figura 23: Insiemi di funzioni incluse $\mathcal{M}$ (*sinistra*) e candidate $\mathcal{C}$ (*destra*).

Una volta costruito il modello completo, volutamente sovradattato, si procede all'indietro "potandolo" dalle funzioni di base meno utili, attraverso un altro criterio (GCV, definito più avanti). In questo senso, il MARS si può vedere come un albero di regressione con funzioni lineari a tratti in ciascun nodo, invece che costanti a tratti.

La potatura del modello si fa in funzione del compromesso distorsione-varianza, ad esempio valutando la potatura sulla base di

- suddivisione in stima e verifica, se i dati sono tanti;
- convalida incrociata (computazionalmente onerosa);

- *convalida incrociata generalizzata* ($GCV$): un'approssimazione del valore della leave-one-out cross-validation, ottenuta usando una formula simile alla *loocv* nel caso di uno stimatore lineare:

$$GCV = \frac{\text{Errori residui}}{\left(1 - \frac{\text{df}}{n}\right)^2} = \frac{\sum_{i=1}^{n} \left(y_i - \hat{f}_\lambda(x_i)\right)^2}{\left(1 - \frac{\text{df}(\lambda)}{n}\right)^2},$$

dove la quantità $\text{df}(\lambda)$ è la misura della complessità del modello (e.g. gdl equivalenti, ...). La quantità $\text{df}(\lambda)/n$ è la complessità "media" di ciascuna osservazione nel modello. In questo caso,

$$\text{df} = \text{n}^\circ \text{ basi} + c \cdot \text{ n}^\circ \text{ nodi}, \quad c = 2 \text{ o } 3.$$

Perché si scelgono le funzioni lineari a tratti?

- Buona capacità locale, perché i prodotti tensoriali di funzioni soft-threshold sono picchi locali. Questo significa che la funzione di regressione viene costruita in modo parsimonioso, con componenti locali solo dove sono necessarie.

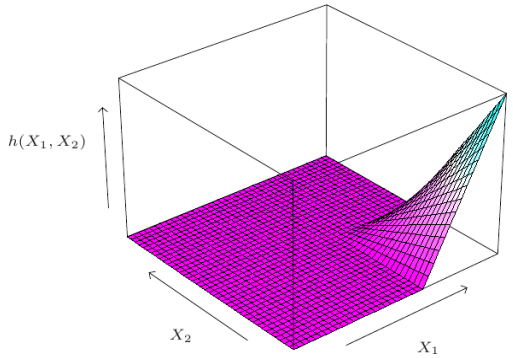


Figura 24: Esempio di funzione prodotto tensoriale $(x_1 - t_1)_+ \cdot (t_2 - x_2)_+$.

- Computazionalmente efficiente, il costo è $n \cdot O(n) \implies O(n^2)$.
- Strategia di selezione gerarchica, ovvero le basi vengono aggiunte in sequenza e i termini di interazione più alti vengono aggiunti solo se già presenti quelli inferiori.

I MARS possono cogliere gli effetti principali come i modelli additivi, se si tengono nel modello i "nodi genitori". Le loro funzioni di previsione hanno qualche punto angoloso, data la natura delle funzioni di base, ma sono pressoché lisce.

Attenzione che i MARS sono difficili da interpretare, anche se più interpretabili di una PPR.

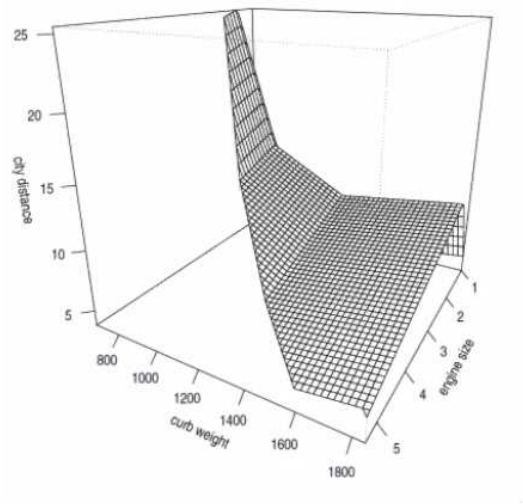


Figura 25: Modello MARS senza estrapolazione al di fuori delle covariate osservate.

# Lezione 18

## 18.1 Alberi di regressione (CART)

*Riferimenti* Breiman et al. (1984)
Azzalini e Scarpa (2012, §4.8.1)
Hastie et al. (2013, §9.2)

Il modo più semplice per approssimare una funzione è utilizzare una funzione costante a tratti, poiché l'approssimazione può sempre essere migliorata usando suddivisioni più fini degli intervalli.

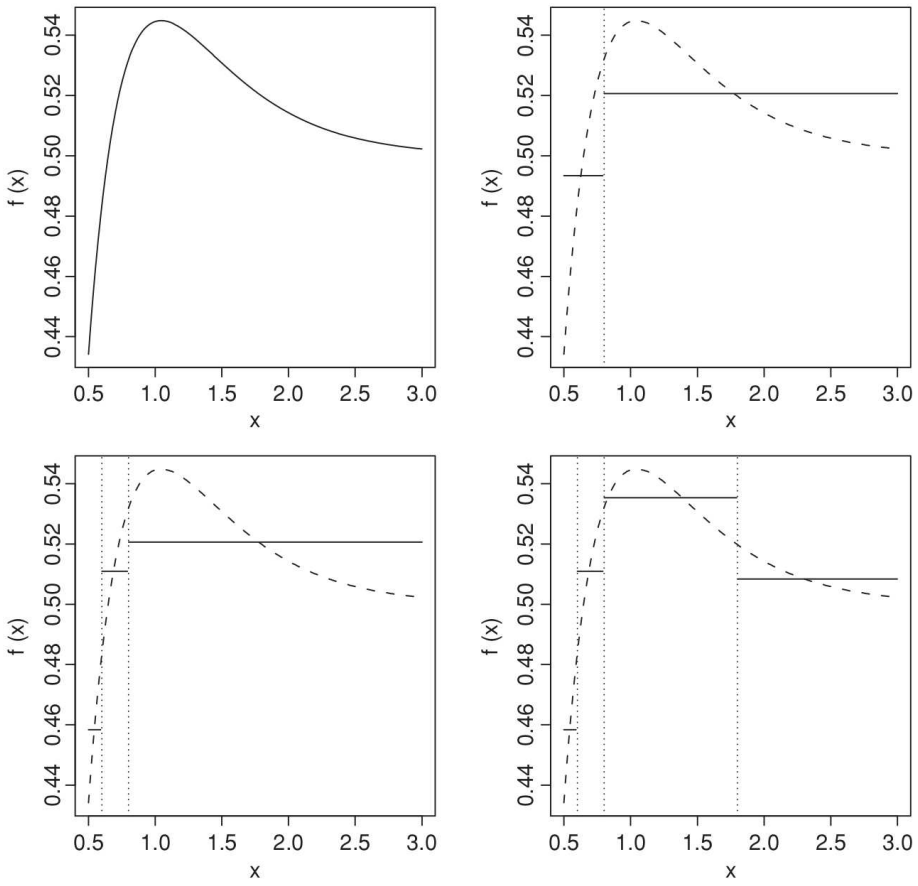


Figura 26: treeRegression

In generale, per trovare una rappresentazione di questo tipo, è necessario definire:

1. *Dove* effettuare i tagli: conviene concentrarsi laddove la funzione è più ripida.
2. *Quale* valore assegnare ad ogni intervallo: si usa la media delle $y$ nell'intervallo.
3. *Quante* suddivisioni effettuare: saranno guidate dal compromesso distorsione-varianza.

Questa logica si applica anche al caso di una funzione di $p$ variabili, tuttavia per mantenere la semplicità computazionale del modello si impongono *tagli paralleli* agli assi coordinati. Si aggiunge allora un quarto aspetto da definire: su quale variabile operare il successivo taglio ad ogni passo.

Dato un insieme di valori $(x_i, y_i) \in \mathbb{R}^{p+1}$, si cerca un'approssimazione della forma

$$f(x) = \sum_{j=1}^{J} c_j \mathbb{1}_{R_j}(x),$$

dove $R_j$ è un rettangolo $p$-dimensionale e $c_1, \ldots, c_J$ sono costanti da stimare tramite minimizzazione della devianza

$$\begin{aligned} D &= \sum_{i=1}^{n} \Big(y_i - \hat{f}(x_i)\Big)^2 \\ &= \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{c}_j)^2 \\ &= \sum_{j=1}^{J} D_j \end{aligned}$$

Il procedimento di regressione è *miopica* (*greedy*), ovvero si procede per suddivisioni successive di singole covariate, per costruire un albero binario con $J$ nodi terminali (*foglie*). Con questo procedimento si arriva ad avere $n$ foglie, che corrisponde ad una regressione con un polinomio di grado $n-1$.

Una volta terminata la crescita, si procede con una fase di *potatura* dell'albero per minimizzare il sovradattamento, utilizzando un criterio diverso dalla devianza.

### 18.1.1 Algoritmo di stima

Si parte con $J = 1$, $R_J = \mathbb{R}^p$, $D = \sum_{i=1}^{n} \big(y_i - M(y)\big)^2$. Si procede ripetendo seguenti passi:

1. Fissato un rettangolo $R_j$, il valore $c_j$ corrispondente è $\hat{c}_j = M(y_i : x_i \in R_j)$.
2. Suddividendo $R_j$ in due parti, $R'_j$ e $R''_j$, l'addendo $D_j$ della devianza viene sostituito da

   $$D_j^* = \sum_{i \in R'_j} (y_i - \hat{c}'_j)^2 + \sum_{i \in R''_j} (y_i - \hat{c}''_j)^2,$$

   per cui si ha un "guadagno" in termini di abbassamento della devianza pari a $D_j - D_j^*$.
3. La suddivisione di $R_j$ viene scelta in modo da massimizzare il guadagno della devianza: per ciascuna variabile, si considera ciascuna modalità osservata e si valuta ognuno degli $n$ split possibili.
4. L'algoritmo si ferma quando si raggiunge un numero minimo specificato di osservazioni presenti nelle foglie.

La potatura si effettua utilizzando una penalizzazione della devianza, basata sul numero $J$ di foglie dell'albero

$$C_\alpha(J) = \sum_{j=1}^{J} D_j + \alpha J,$$

dove $\alpha$ è un parametro positivo di penalizzazione. Il ruolo di parametro di complessità in questo caso è giocato dal numero di foglie dell'albero.

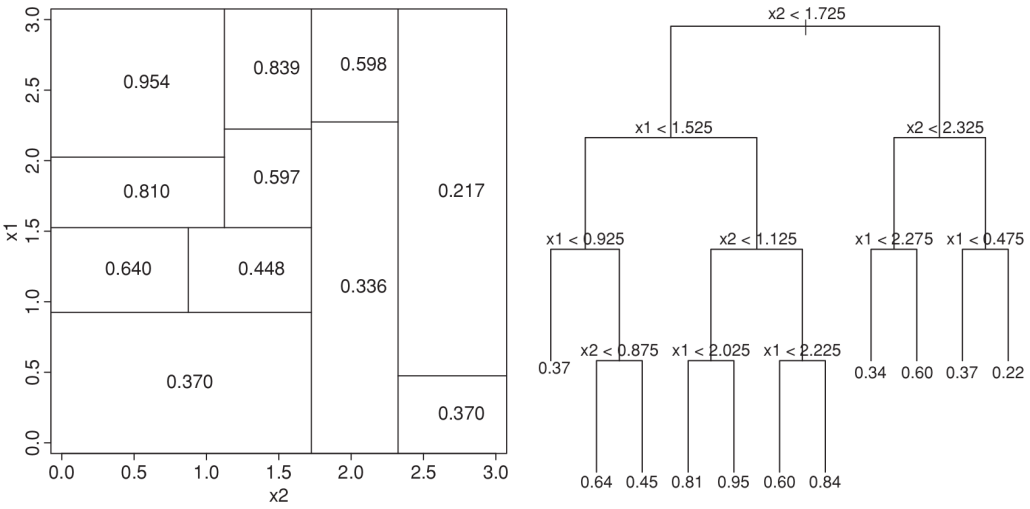


Figura 27: Partizione dello spazio $\mathcal{X}$ e rappresentazione ad albero stimate per una funzione $f(x_1, x_2)$.

Per ottenere la previsione di un'osservazione, si può scendere lungo l'albero e ottenere la foglia corrispondente alle $x_j$ osservate.

Dal momento che i tagli dello spazio campionario sono effettuati con iperpiani paralleli agli assi, ci sono molte partizioni di $\mathcal{X}$ che non possono essere rappresentate in forma di albero binario:

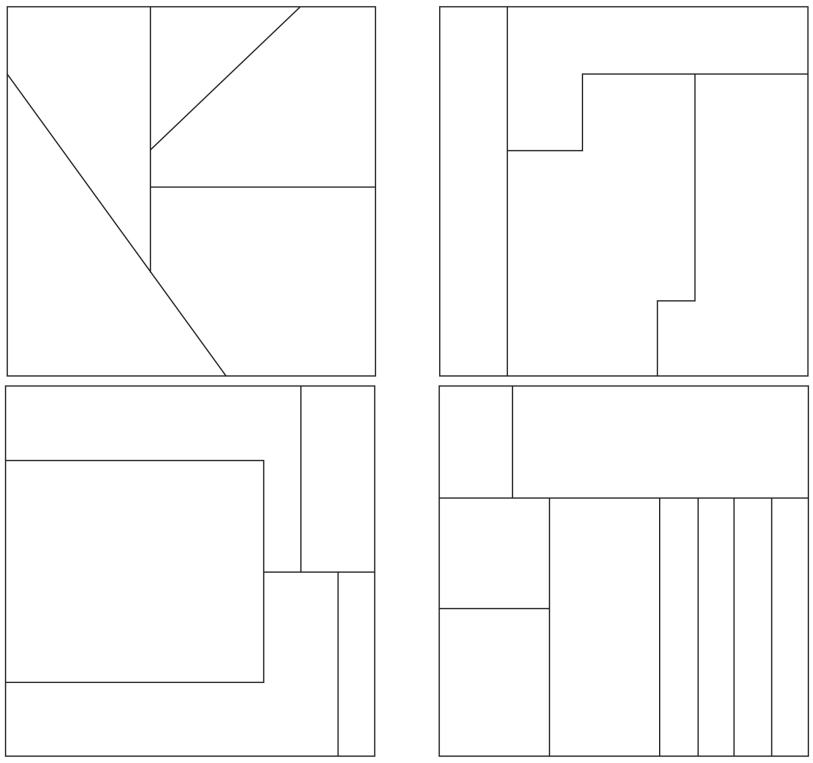

Figura 28: Delle quattro partizioni, solo quella in basso a destra corrisponde ad un albero binario

**Risultato fondamentale**

Il CART è il miglior procedimento di stima degli alberi di regressione, perché

1. chiaramente, all'aumentare della penalizzazione $\alpha$ l'albero scelto si riduce;
2. si può mostrare (Breiman et al., 1984) che, aumentando $\alpha$ l'albero ottimo viene trovato con operazioni di potatura sull'albero corrente. Per ogni $\alpha$, l'albero ottimo con $j$ foglie è l'albero ottimo con $j+1$ foglie potato di una foglia.

L'operazione di potatura è dunque annidata: si riduce progressivamente l'albero, selezionando la foglia la cui eliminazione comporta il minor incremento di $\sum_j D_j$.

Per scegliere $\alpha$, si può procedere con una logica simile all'AIC per ottenere $\alpha = 2\hat{\sigma}^2$, dove $\hat{\sigma}^2$ è la miglior stima della varianza residua. Questo procedimento non è molto affidabile e tende in generale a sovradattare il modello ai dati.

Si preferisce allora suddividere in due parti l'insieme di dati: una per far crescere l'albero e una per effettuare la potatura; in questo caso non è necessario usare $C_\alpha(J)$, ma è sufficiente guardare l'errore di previsione

Infine, se i dati sono pochi, si può usare la $k$-fold cross-validation.

**Osservazioni**

- Suddividendo i dati per stima e potatura, si ottiene un valore di $\alpha$ associato a ciascun albero in modo indiretto.
- Gli alberi di regressione non sono funzioni lisce, ma possono fare delle ottime previsioni.

**Pro**

1. Semplicità di comunicazione per persone "non quantitative", specialmente in campo medico, ma solo se l'albero è piccolo.
2. Rappresentazione analitica semplice e spazio occupato in memoria basso.
3. Rapidità di calcolo, perché si fanno medie di sottogruppi. Si può parallelizzare, ma di solito non si fa.
4. Dati mancanti trattabili naturalmente.
5. Selezione automatica delle variabili rilevanti.

**Contro**

1. Molto variabile rispetto a perturbazioni del campione (ottimo vantaggio nel *bagging*).
2. Non aggiornabile sulla base di nuovi dati.
3. Funzioni ripide sono difficili da approssimare, perché si usano tagli paralleli.
4. Non ci sono procedure di inferenza formali (verifica di ipotesi, ...).
5. Difficile attribuire importanza alle variabili rimaste nell'albero

Come misura qualitativa dell'importanza di ciascuna variabile, una possibilità è usare la somma del miglioramento della devianza su tutti i nodi in cui compare $x_j$. Tuttavia, in questo modo le variabili continue sono avvantaggiate rispetto alle qualitative.

# Lezione 19

## 19.1 Reti neurali

*Riferimenti* Hastie et al. (2013, §11.3)
Azzalini e Scarpa (2012, §4.9)

Date le esplicative $x_1, \dots, x_p$ e le variabili risposta $y_1, \dots, y_K$, la rete neurale è un modello che vuole emulare quello che si riteneva essere il meccanismo di un neurone. La specificità di questo metodo è l'inserimento di uno *strato latente*, che agisce da congiunzione tra le variabili di input e le variabili risposta.

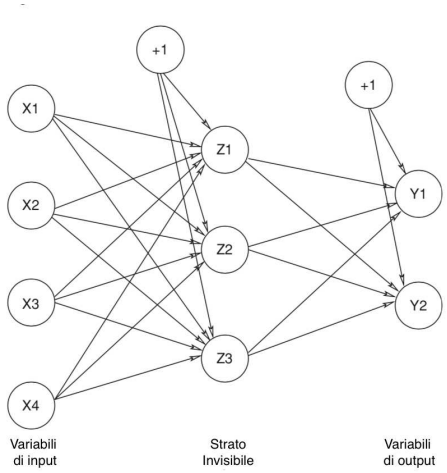


Figura 29: neuralNetwork

La struttura analitica tipica di una rete neurale è

$$z_j = f_0\Big(\sum_{i\to j} w_{ij} x_i\Big)$$

$$y_k = f_1\Big(\sum_{j\to k} w_{jk} z_j\Big) = f_1\Big(\sum_{j\to k} w_{jk} f_0\Big(\sum_{i\to j}^{n} w_{ij} x_i\Big)\Big),$$

dove $w_{rs}$ sono parametri da determinare e $f_1, f_0$ sono funzioni specificate, di cui almeno una non lineare: se fossero tutte lineari, il modello sarebbe equivalente a un modello lineare, in quanto

$$f_0, f_1 \text{ lineari} \implies f_1 \circ f_0 \text{ lineare.}$$

Generalmente, in questo caso per $f_0$ si sceglie la funzione logistica, chiamata in questo contesto *sigmoide*, mentre per $f_1$ si sceglie l'identità.

$$f_0(u) = \frac{e^u}{1+e^u}, \quad f_1(u) = u.$$

Se $f_1(u) = f_0(u) = u$, si avrebbe un semplice modello lineare, dove

$$y_k = \sum_{j \to k} \sum_{i \to j} w_{jk} w_{ij} x_i.$$

La struttura del modello è molto simile a quella dei modelli additivi, in particolare se $f_0$ fosse stimata tramite un lisciatore non parametrico, si otterrebbe una regressione projection pursuit.

Data la complessità del modello, questo metodo è molto flessibile e permette di adattarsi a un elevato numero di funzioni, essendo come un modello PPR un *approssimatore universale*. Tuttavia, la stessa complessità pone dei problemi computazionali e statistici non banali.

### 19.1.1 Algoritmo di backpropagation

1. Si sceglie la devianza come funzione obiettivo da minimizzare:

$$D_0 = \sum_{i=1}^{N} \|y_i - \hat{f}(x_i)\|^2 = \sum_{i=1}^{N} \sum_{k=1}^{K} (k y_i - k \hat{f}(x_i))^2,$$

   dove

$$\hat{f}(x) = f_1\Big( \sum_{j \to k} \beta_{jk} f_0\Big( \sum_{h \to j} \alpha_{hj} x_h \Big) \Big) = f_1\Big( \beta^{\mathsf{T}} f_0\big( \alpha^{\mathsf{T}} x_h \big) \Big).$$

2. La minimizzazione avviene attraverso algoritmi come il *backpropagation*, che tuttavia di solito individua un minimo relativo. Questo algoritmo lavora sul gradiente, calcolabile tramite differenziazione a catena con una doppia passata (andata e ritorno), tenendo traccia delle quantità solo localmente per ogni unità.

L'inizializzazione dei valori si fa con pesi vicini a zero, in modo da cominciare con un modello quasi lineare e muoversi gradualmente verso la non linearità.

| **Pro** | **Contro** |
|---|---|
| Semplice, parallelizzabile e con stime aggiornabili. Ciò implica la possibilità di stimare il modello su insiemi molto grandi e aggiornare i parametri nel tempo. | Molto lento da stimare, richiede una capacità computazionale notevole |

Per le reti neurali, il sovra-adattamento è enfatizzato dalla presenza di un numero molto elevato di parametri. In generale, si utilizza una procedura di regolarizzazione, dove le soluzioni comunemente usate sono:

- Fermata anticipata prima del sovradattamento (ad hoc ma efficace).
- Penalizzazione della devianza:

$$D = D_0 + \lambda J(\alpha, \beta),$$

dove il termine di penalizzazione $J(\alpha, \beta)$ si può scegliere tra

$$J(\alpha, \beta) = \int \sum_{h,k} \frac{\partial^2 y_k}{\partial x_h^2}\, dx \approx \frac{1}{n} \sum_{i=1}^{n} \sum_{h,k}^{n} \frac{\partial^2 y_{ki}}{\partial x_{hi} \partial x_{hi}}$$

$$J(\alpha, \beta) = \|\alpha\|^2 + \|b\|^2 \qquad \text{(“weight decay”)}$$

Ripley suggerisce di scegliere $\lambda \in (10^{-4}, 10^{-2})$.

Le penalizzazioni hanno senso se le variabili hanno la stessa unità di misura, in quanto non si ha alcuna interpretazione per gli strati latenti. È quindi opportuno normalizzare le variabili, in modo da riportarle in un intervallo tra 0 e 1, oppure standardizzarle.

$$\text{Normalizzazione :} \qquad \tilde{x} = \frac{x}{\max x - \min x}$$

$$\text{Standardizzazione :} \qquad \hat{x} = \frac{x - \bar{x}}{\mathrm{sd}(x)}$$

Nello strato latente è preferibile avere più unità, piuttosto che troppo poche: con poche unità latenti il modello non ha sufficiente flessibilità, mentre con tante unità latenti alcune possono essere poste a 0. Per problemi complessi, si possono scegliere tra 5 e 100 unità latenti, a seconda del numero di esplicative e di osservazioni.

Le connessioni si possono anche scegliere in base alla conoscenza del problema che si sta analizzando: nell'identificazione di immagini si usano le *Convolutional Neural Networks*, modelli dove le variabili vengono selettivamente collegate solo ad alcuni nodi latenti.

Si può anche scegliere un diverso numero di strati latenti, a seconda della struttura gerarchica del problema in analisi.

Qualsiasi rete multistrato si può riscrivere come una rete a strato singolo con un numero sufficiente di nodi, quindi in teoria potrebbero essere inutili. Tuttavia, reti con più strati (*deep learning*) permettono di inserire vincoli solo su alcuni strati e di avere più connessioni con meno nodi.

Tabella 4: “Traduzione” di vari termini nei domini di applicazione.

| **Reti neurali** | **Statistica** | **Neural nets** |
|---|---|---|
| Rete | Modello | Network |
| Pesi | Parametri | Weights |
| Apprendimento | Adattamento | Learning |
| Generalizzazione | Risultati su insieme di verifica | Generalization |
| Apprendimento supervisionato | Regressione/classificazione | Supervised learning |
| Apprendimento non supervisionato | Stima della densità/clustering | Unsupervised learning |
| Danni ottimi al cervello | Selezione del modello | Optimal brain damage |

# Lezione 20

*Riferimenti* Azzalini e Scarpa (2012, §5)

## 20.1 Classificazione

Nei problemi di classificazione, si vuole individuare una regola per classificare delle unità in una categoria, in un insieme di $K$ possibili. Nell'*analisi supervisionata*, il numero di categorie è fissato e si dispone di unità statistiche, delle quali si conosce la classe di appartenenza.

Particolarmente importante è il caso in cui $K = 2$, i cui metodi in generale permettono di essere estesi a problemi multiclasse.

In tutti gli esempi, scegliendo di allocare un individuo a *uno soltanto* dei $K$ gruppi si hanno $K(K-1)$ errori possibili, in genere non di uguale rilevanza: dare credito a un cliente insolvente dà luogo a una perdita completa dei soldi prestati; viceversa, una risposta negativa ad un cliente solvente provoca un mancato guadagno degli interessi.

In generale, nella pratica, la base informativa disponibile è costituita da unità *note* e non da un campione della popolazione di interesse. Da qui, tutti i problemi di cui si è parlato nella Lezione 3.1.

### 20.1.1 Classificazione con regressione lineare

Nel caso con $K = 2$ classi, si può utilizzare il modello lineare per ottenere

$$\hat{y}_i = x_i^\mathsf{T} \hat{\beta},$$

con la quale classificare nel gruppo 1 se $\hat{y} > 1/2$ e nel gruppo 0 se $\hat{y} < 1/2$. In questo caso le regioni di classificazione sono separate dall'iperpiano $x^\mathsf{T}\hat{\beta} = 1/2$, che può diventare non lineare in $x$ se si includono trasformazioni dell'esplicativa.

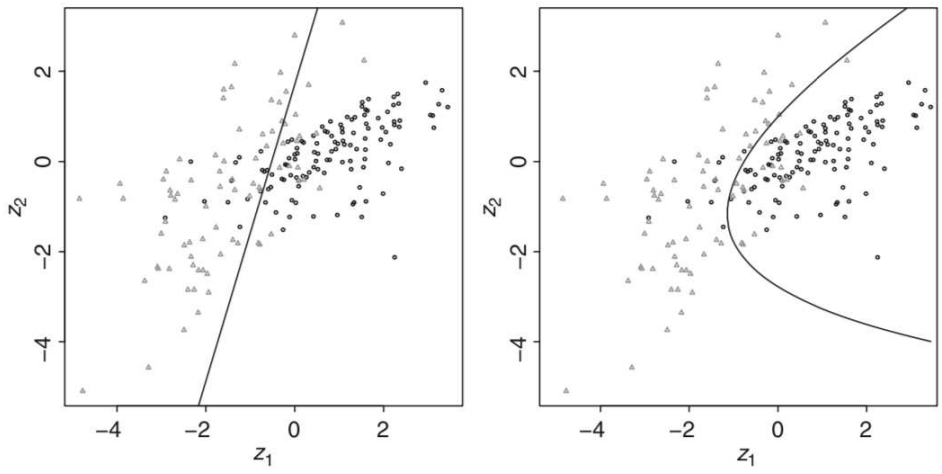


Figura 30: Classificazione mediante regressione lineare, con una trasformazione quadratica della variabile esplicativa.

Per variabili multiclasse si ricodifica la risposta $y$, usando variabili indicatrici, per ottenere una *matrice* di variabili risposta indicatrici

$$Z = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \implies Z = XB + \varepsilon.$$

Si possono usare quindi i minimi quadrati con variabile risposta multidimensionale

$$\hat{Z} = X(X^\mathsf{T}X)^{-1}X^\mathsf{T}Z$$

e si classifica il punto $x$ nel gruppo che massimizza $\hat{z}_k$. Le due procedure di stima $\hat{y} > \frac{1}{2}$ e $\hat{z} = \text{argmax}_j\, z_j$ sono diverse, ma per $K = 2$ si dimostra (esercizio) che sono equivalenti.

Si mostra inoltre che, per due classi bilanciate, la separazione data dalla regressione lineare e dall'analisi discriminante lineare sono esattamente le stesse (esercizio più avanti).

**Problemi**

- Si possono ottenere stime fuori dall'intervallo $[0, 1]$ (meno rilevante).
- Possono accadere fenomeni di *mascheratura* (più rilevante):

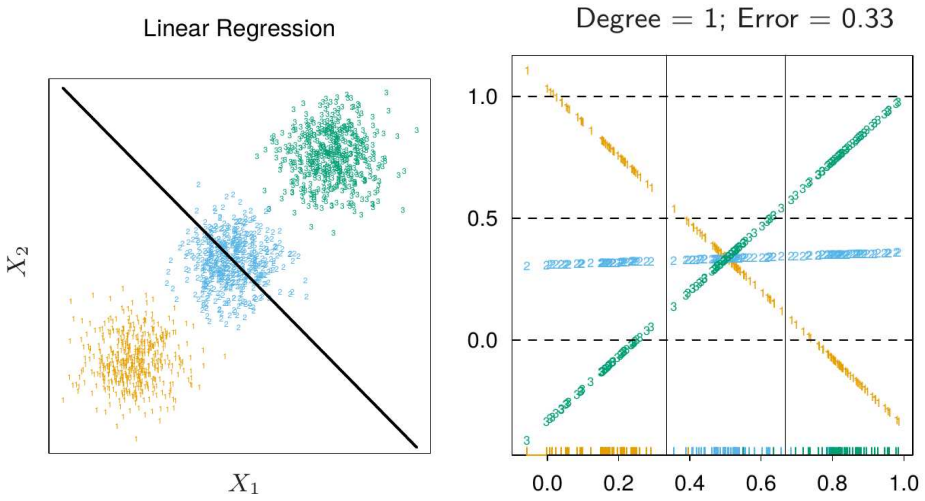


Figura 31: Mascheratura nella classificazione con più variabili tramite regressione lineare: i punti non vengono mai classificati nell'insieme *blu*.

### 20.1.2 Regressione logistica multidimensionale

Con $K$ categorie, si usa il *logit multivariato* (*multilogit*)

$$g(\pi_j) = \log \frac{\pi_j}{\pi_K} = \eta_k(x) = X\beta_k,$$

da cui

$$\pi_j(x) = \frac{e^{\eta_j(x)}}{1 + \sum_{k=1}^{K-1} e^{\eta_k(x)}}$$

$$\pi_K(x) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\eta_j(x)}}.$$

In questo caso, di fatto si adattano $K - 1$ modelli di regressione logistica.

## 20.2 Strumenti di valutazione

*Riferimenti* Azzalini e Scarpa (2012, §5.2)

### Errata classificazione

Per valutare quale modello è migliore, ovviamente non si può usare la somma dei quadrati dei residui. Si usa allora la *matrice di confusione*, ad esempio per $K = 2$

| | **Osservati** | | |
|---|---|---|---|
| **Predetti** | **-** | **+** | **Totale** |
| **-** | $n_{11}$ | $n_{12}$ | $n_{1\cdot}$ |
| **+** | $n_{21}$ | $n_{22}$ | $n_{2\cdot}$ |
| **Totale** | $n_{\cdot 1}$ | $n_{\cdot 2}$ | n |

| | **Osservati** | |
|---|---|---|
| **Predetti** | **-** | **+** |
| **-** | $1 - \alpha$ | $\beta$ |
| **+** | $\alpha$ | $1 - \beta$ |
| **Totale** | 1 | 1 |

Figura 32: Matrice di errata classificazione (*sinistra*) e la sua versione probabilistica (*destra*).

Quantità riassuntive del modello sono:

- Tasso errata classificazione: $\frac{n_{12} + n_{21}}{n}$
- Accuratezza: $\frac{n_{11} + n_{22}}{n}$
- Sensibilità: $1 - \beta = \frac{n_{22}}{n_{12} + n_{22}}$
- Specificità: $1 - \alpha = \frac{n_{11}}{n_{11} + n_{21}}$
- False discovery rate: $\frac{n_{21}}{n_{21} + n_{22}} = \frac{\alpha}{1 + \alpha - \beta}$
- Precisione: $\frac{n_{22}}{n_{21} + n_{22}} = \frac{1 - \beta}{1 + \alpha - \beta}$
- $F_1$ score: $\frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precisione}}} = \frac{2n_{22}}{2n_{22} + n_{21} + n_{12}}$

**Nota** Queste quantità si riferiscono ad una *fissata* soglia di separazione tra le due classi.

### Curva ROC

Facendo invece variare la soglia di separazione, in base ai valori di sensibilità e $1-$specificità $(\hat{\alpha}, 1-\hat{\beta})$ si ottiene la Curva ROC (*Receiving Operating Curve*), cioè un grafico della qualità previsiva del

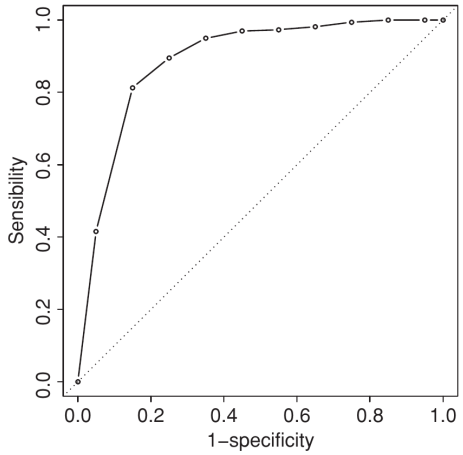


Figura 33: curvaRoc

modello nel suo complesso. Ci sono varie soluzioni ottimali possibili, in base a quanto sono rilevanti specificità e sensibilità. L'area sotto la curva (AUC) è una misura della qualità globale del modello, ma per la classificazione è necessario scegliere il modello corrispondente ad una particolare coppia $(\alpha_0, 1-\beta_0)$ sulla curva. Per questo motivo, è utile usare la curva ROC per una valutazione teorica globale del modello, ma per la particolare classificazione interessa solo il punto relativo al singolo modello.

### Curva lift

La curva *lift* dà un indicatore di quanto migliora un certo metodo rispetto a quello che opera casualmente con probabilità pari alla frazione osservata di positivi. Il *lift*, o fattore di miglioramento, è definito per una certa soglia come

$$\text{lift} = \frac{n_{22}/n_{2\cdot}}{n_{2\cdot}/n}$$

La curva lift è sempre decrescente, se si usano gli stessi dati di stima, e misura quanto migliora la previsione rispetto a quella casuale per una proporzione scelta di unità dal campione. La curva lift sull'insieme di stima tende a 1 se la frazione di unità tende a 1.

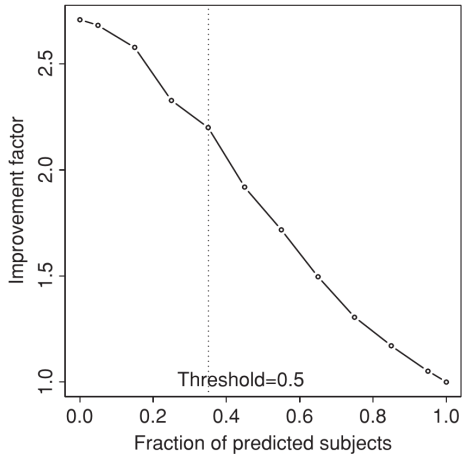


Figura 34: liftCurve

# Lezione 21

## 21.1 Classificazione non parametrica

*Riferimenti* Azzalini e Scarpa (2012, §5.6)

Come per la regressione, si possono considerare metodi di classificazione anche non parametrici. Si può applicare la logica della regressione logistica nella regressione non parametrica, cioè utilizzare una *funzione di legame* per legare variabile risposta e covariate:

$$\operatorname{logit}\Big(\mathbb{E}\left[Y|X\right]\Big) = f(x).$$

L'estensione al caso di $K$ categorie si può fare tramite la funzione *multilogit*.

### 21.1.1 GAM

I modelli GAM possono essere adattati immediatamente, per costruire il modello

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

che si stima utilizzando una generalizzazione dell'algoritmo di backfitting, detto punteggio locale (*local scoring*).

**Algoritmo 4** *Local scoring* (backfitting + IRLS)

**Init:**
  $\hat{f}_j \leftarrow 0,\ j = 1, \dots, p$
  $\hat{\alpha} \leftarrow \log(\bar{y}/(1-\bar{y}))$
  $\eta_i \leftarrow \hat{\alpha} + \sum_{j=1}^{p} \hat{f}_j(x_{ij})$
  $\hat{p}_i \leftarrow \frac{1}{1+\exp(-\hat{\eta}_i)}$
**while** $tol > \varepsilon$ **do**
  $z_i \leftarrow \hat{\eta}_i + \frac{y_i - \hat{p}_i}{\hat{p}_i(1-\hat{p}_i)}$ ▷ Passo IRLS
  $w_i \leftarrow \hat{p}_i(1-\hat{p}_i)$
  $\hat{f}_j \leftarrow S\big(z_i - \sum_{k \neq j} \hat{f}_k(x_{ij}); w_i\big)$ ▷ Passo backfitting
  $\hat{\alpha} \leftarrow \dots$
**end while**

Si ricorda che il passo IRLS viene effettuato quando l'algoritmo di Newton-Raphson utilizza la matrice di informazione attesa $i(\hat{\beta}^{(k)})$ al posto di quella osservata nell'aggiornamento del parametro

$$\hat{\beta}^{(k+1)} = \hat{\beta}^{(k)} + \big(j(\hat{\beta}^{(k)})\big)^{-1} \nabla \ell\big(\hat{\beta}^{(k)}\big).$$

$$\hat{\beta}^{(k+1)} \approx \hat{\beta}^{(k)} + \big(i(\hat{\beta}^{(k)})\big)^{-1} \nabla \ell\big(\hat{\beta}^{(k)}\big).$$

$$i(\hat{\beta}^{(k)})\hat{\beta}^{(k+1)} \approx \underbrace{i(\hat{\beta}^{(k)})\hat{\beta}^{(k)}}_{X^\intercal \hat{W} X \hat{\beta}^{(k)}} + \underbrace{\nabla \ell\big(\hat{\beta}^{(k)}\big)}_{X^\intercal W u}.$$

$$\hat{\beta}^{(k+1)} = (X^\intercal \hat{W} X)^{-1} X^\intercal \hat{W} (X\hat{\beta}^{(k)} + u).$$

Analogamente, si può fare il local scoring per un GLM con qualsiasi distribuzione della famiglia di dispersione esponenziale. Anche nel caso dei GAM, si può usare la logica della regressione lineare e utilizzare una regressione ai minimi quadrati per classificare.

### 21.1.2 MARS

Da una parte, si possono usare i minimi quadrati per classificare, ricodificando la risposta in $K$ variabili dicotomiche.

Alternativamente, la libreria `PolyMARS` permette di stimare il modello MARS massimizzando la log-verosimiglianza della multinomiale. Dal momento che la massimizzazione della log-verosimiglianza non è esplicita, bisognerebbe usare gli IRLS ad ogni funzione di base che si inserisce. Il problema diventerebbe esponenzialmente complesso, per cui si preferisce invece massimizzare un'approssimazione quadratica della log-verosimiglianza.

Il modello viene adattato ai dati usando l'approssimazione quadratica della verosimiglianza, per poi essere potato tramite convalida incrociata generalizzata.

### 21.1.3 Rete neurale

La rete neurale usa funzioni

$$z_j = f_0\Big(\sum_{i\to j} w_{ij}x_i\Big), \quad y_k = f_1\Big(\sum_{j\to k} w_{jk}z_j\Big),$$

dove $f_0, f_1$ sono note e $w_{rs}$ sono pesi da stimare. In problemi di classificazione, le $y_i$ sono variabili indicatrici, per cui si utilizzano sia per $f_0$ sia per $f_1$ la funzione logistica:

$$f_0(u) = f_1(u) = \frac{e^u}{1+e^u}.$$

Altre funzioni $f : \mathbb{R} \to (0,1)$ sono più computazionalmente vantaggiose per la stima dei coefficienti (conferenza *deep learning*): nel caso multi classe si può usare la funzione *softmax*, preferita alla multilogit, data da

$$f_{1k} = \frac{e^{\eta_i}}{\sum_{j=1}^k e^{\eta_j}}$$

### 21.1.4 CART

Gli alberi di regressione stimano la probabilità di appartenere a ciascuna classe con una funzione a gradini:

$$p(x) = \sum_{j=1}^J P_j \mathbb{1}_{R_j}(x),$$

dove la probabilità di ciascuna classe è stimata con

$$\hat{P}_j = \frac{1}{n_j}\sum_{i\in R_j} \mathbb{1}_{R_j}(y_i).$$

Avendo a che fare con una multinomiale, la funzione da minimizzare non è più la devianza, ma si utilizza l'analogo della devianza nei GLM, cioè $-2\log L(\vartheta)$:

$$\begin{aligned} D &= -2\sum_{i=1}^{n}\{y_i \log p_i + (1-y_i)\log(1-p_i)\} \\ &= -2\sum_{j=1}^{J} n_j\{\hat{P}_j \log \hat{P}_j + (1-\hat{P}_j)\log(1-\hat{P}_j)\} \\ &= 2\sum_{j=1}^{J} n_j Q(\hat{P}_j)\,, \end{aligned}$$

Vale tutto quello che valeva per gli alberi di regressione, a livello di interpretazione del modello e per quanto riguarda l'algoritmo di stima. Si usa la verosimiglianza della binomiale/multinomiale invece della devianza, e ad ogni passo dell'algoritmo si sceglie lo split che massimizza la diminuzione della devianza ecc..., fino alla stima dell'albero completo. La potatura viene effettuata con un criterio diverso, esattamente come nel caso della regressione.

Con $K$ modalità, è necessario fare un'altra modifica nella funzione di log-verosimiglianza: partendo dall'ultima espressione,

$$D = 2\sum_{j=1}^{J} n_j Q(\hat{P}_j),$$

si definisce $Q(\cdot)$ il *grado di impurità* delle foglie. Nel caso binomiale, il grado di impurità è misurato dall'*entropia*

$$\begin{aligned} Q(P_j) &= -\sum_{j=0,1} P_{jk}\log P_{jk} \\ &= -\big[P_{j1}\log P_{j1} + (1-P_{j1})\log(1-P_{j1})\big]. \end{aligned}$$

L'entropia/impurità misura quanto la variabile varia rispetto alla moda, guardare il legame tra entropia e varianza: scomposizione dell'entropia, che è simile a quella della varianza. L'idea alla base della minimizzazione dell'entropia è di avere le osservazioni meno mutabili possibili in ogni rettangolo.

Alternativamente, si può quantificare l'impurità con l'*indice di Gini*, che (esercizio) si può dimostrare essere un'approssimazione di Taylor dell'entropia:

$$Q(P_j) = \sum_{k=0,1} P_{jk}(1-P_{jk}).$$

*Dim.*
Per $|x|<1$, si ha che

$$\log(1+x) = x + o(x),$$

dunque

$$\log p \approx p-1 \implies H(p) = -\sum_k p_k \log p_k \approx \sum_k p_k(1-p_k).$$

□

Il fatto che si possa riscrivere la devianza in questo modo implica che

1. Si può usare una variante dell'entropia.
2. Si può generalizzare a più classi usando la versione multinomiale di entropia e Gini

$$H = -\sum_{k=1}^{K} P_{jk} \log P_{jk};$$

$$G = \sum_{k=1}^{K} P_{jk}(1 - P_{jk}).$$

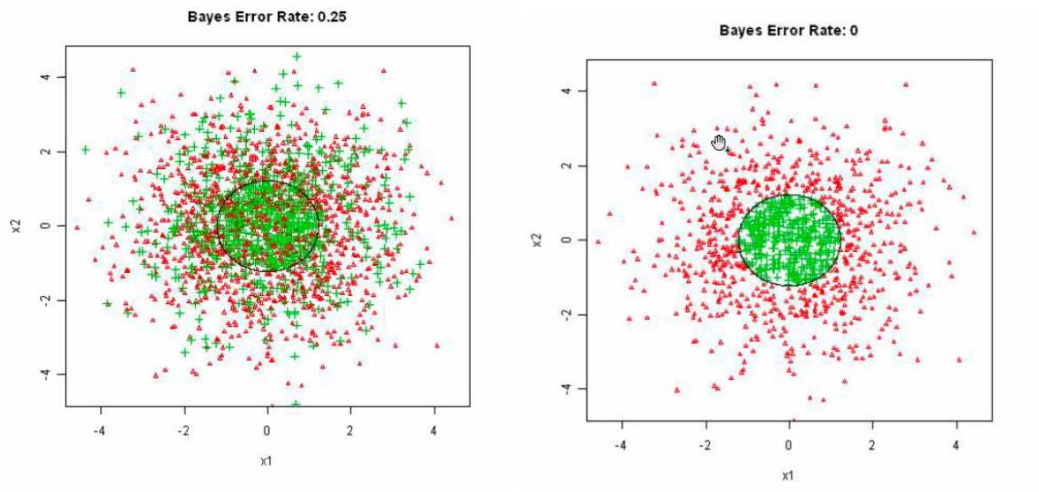


Figura 35: bayesDecision

# Lezione 22

## 22.1 Frontiera decisionale di Bayes

*Riferimenti* James et al. (2013, §2.2.3)

Analogamente alla funzione di regressione, soluzione del problema

$$\hat{f}(x) = \underset{f}{\operatorname{argmin}}\, \mathbb{E}\left[\big(y - f(x)\big)^2\right],$$

si può definire il *classificatore di Bayes* come la soluzione del problema di minimizzazione del tasso di errata classificazione

$$\hat{f}(x) = \underset{f}{\operatorname{argmin}}\, \mathbb{E}\left[\mathbb{1}_{y \neq f(x)}\right].$$

Si può mostrare che il classificatore che minimizza la funzione obiettivo è data dalla classe che massimizza la probabilità condizionata di appartenere alla classe:

$$\hat{f}(x) = \underset{j}{\operatorname{argmax}}\, P(Y = j | X = x).$$

La curva che separa le zone in cui la frequenza osservata è maggiore in un gruppo rispetto all'altro si dice *frontiera decisionale di Bayes*, ed è la classificazione ottima per l'insieme di dati.

Supponendo di classificare con un albero di regressione, si ottiene un output leggermente diverso dalla frontiera decisionale ottima, con un tasso di errore nel caso a destra del 4.87%.

Specialmente in dimensione elevata, quando le regioni decisionali ottime sono sferiche, l'albero di classificazione produce una regola $\hat{C}(X)$ molto inaccurata, con tasso di errore che può essere anche intorno al 30%.

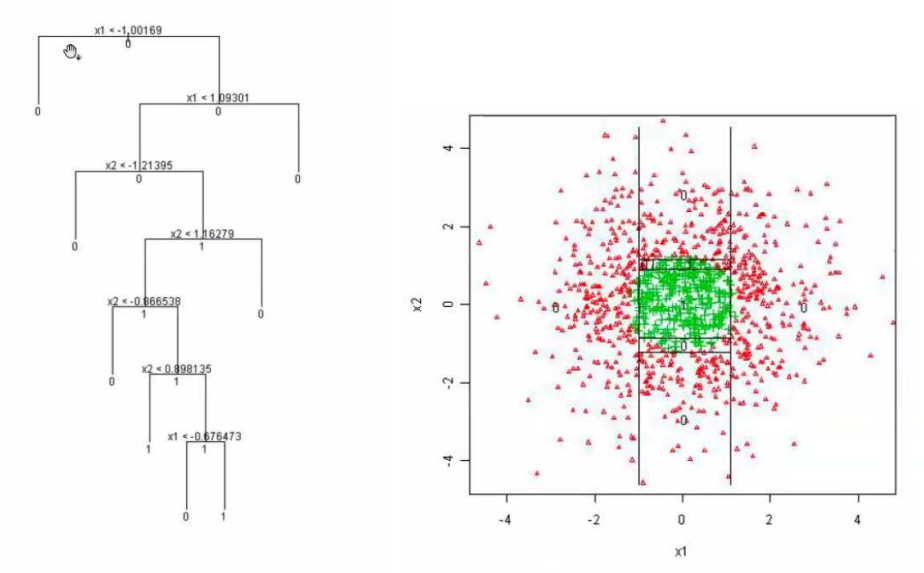


Figura 36: bayesDecisionTree2

Per migliorare questa situazione, si possono usare metodi di combinazione per ottenere modelli migliori, a partire dai singoli alberi di classificazione. In generale, gli alberi producono modelli di classificazione inappropriati, con molto rumore o comunque poco predittivi.

**Tecniche di combinazione di modelli**

- *Bagging*: Si adattano molti alberi non potati a versioni bootstrap del dataset di stima, si classifica per "voto di maggioranza" dei modelli.
- *Boosting*: Si adattano alberi poco profondi a versioni *pesate* del dataset di stima.
- *Random Forests*: Versione più sofisticata del *bagging*, che opera un bootstrap anche sulle colonne del dataset.

In generale, Boosting > Random Forests > Bagging > Albero singolo.

## 22.2 Combinazione di modelli

*Riferimenti* Azzalini e Scarpa (2012, §5.9)

L'idea è di combinare gli output di diversi modelli attraverso una funzione di aggregazione, per ottenere una previsione in ciascun punto sulla base delle esplicative $t = (x_1, x_2, \ldots, x_n)$.

Siano $f_1(x), \ldots, f_M(x)$ diversi classificatori ("*esperti*"). Un semplice *voto di gruppo* produce la previsione

$$\hat{f}(x) = \frac{1}{M} \sum_{m=1}^{M} f_m(x),$$

che ha senso sia in regressione sia in classificazione. In un problema di classificazione, per ogni classe $k$ si ha una previsione $f_m^k(x,t)$, per cui

$$\hat{f}^k(x) = \frac{1}{M}\sum_{m=1}^{M} f_m^k(x).$$

La combinazione di modelli ha senso soprattutto quando i modelli sono sostanzialmente diversi l'uno dall'altro, cioè quando ciascuno mette in evidenza diverse caratteristiche dei dati.

### 22.2.1 Stacking

Il metodo di *stacking* utilizza una media pesata come funzione di aggregazione

$$\hat{f}(x) = \sum_{m=1}^{M} w_m f_m(x),$$

pesando ciascun modello $f_m$ per $w_m$. Per stimare i $w_m$ si potrebbe utilizzare una regressione lineare di $y$, usando $f_1(x), \dots, f_M(x)$ come variabili esplicative. Tuttavia, questo approccio non tiene conto delle diverse complessità dei modelli $f_m$.

Una strategia più utile è utilizzare l'analogo della LOOCV, per trovare la soluzione

$$\underset{w_1,\dots,w_M}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{m=1}^{M} w_m f_m^{-i}(x_i)\right)^2,$$

in cui ogni modello viene stimato senza la $i$-esima osservazione. Si effettua allora la regressione di $y_i$ su $f_1^{-i}(x), f_2^{-i}(x), \dots, f_m^{-i}(x)$ con dei vincoli $w_m \geq 0$, altrimenti due modelli con pesi opposti si potrebbero annullare, ed essere perciò inutili.

### 22.2.2 Bagging

Si può fare finta che le osservazioni costituiscano una popolazione, da cui ottenere $B$ nuovi campioni tramite estrazioni con reinserimento. Su ciascuno di questi campioni bootstrap, si può adattare lo stesso modello ed ottenerne uno aggregato prendendone la media.

Sia $\mathcal{C}(\mathcal{S}, x)$ un metodo di classificazione; si definisce il classificatore *bagging* tramite la votazione di $B$ copie di $\mathcal{C}$, adattate su campioni bootstrap $\mathcal{S}^b$ di $\mathcal{S}$:

$$\mathcal{C}_{\text{bag}}(x) = \underset{k}{\operatorname{argmax}} \frac{1}{B}\sum_{b=1}^{B}\mathcal{C}(S^b, x).$$

La procedura di bagging ha l'effetto di mantenere costante la distorsione del modello, riducendo di la varianza grazie all'operazione di media.

Le procedure di classificazione instabili (poco distorte, molto variabili) sono quindi dei buoni candidati per il bagging, grazie al quale di solito si ottiene un miglioramento della previsione. Di contro, il bootstrap fa perdere ogni interpretabilità del modello.

Gli alberi sono un ottimo tipo di modello per il bagging, perché sono particolarmente sensibili allo specifico campione estratto.

Si riduce la varianza, ma si aumenta leggermente la distorsione, visto che gli alberi adattati sono generalmente meno profondi degli alberi completi.

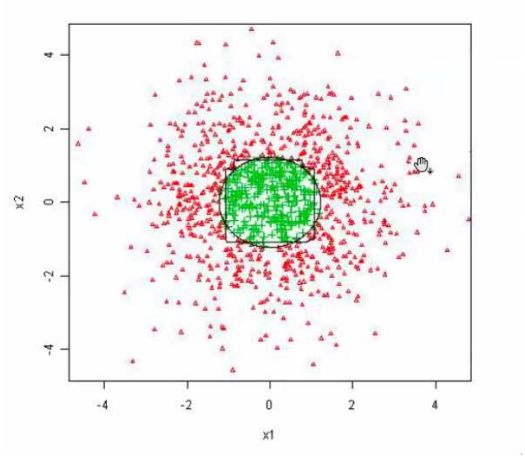


Figura 37: bayesBagging

Come si può vedere dalla Figura 37, il classificatore bagging produce frontiere decisionali più lisce del modello di partenza.

Il bagging non soffre del problema del sovradattamento, nemmeno per alberi molto profondi: la chiave è assumere che gli alberi *non siano correlati*, a causa della grande variabilità degli split:

1. Piccola distorsione (buon adattamento) e alta varianza.
2. Il bagging "media" le previsioni, per cui se i modelli sono più o meno indipendenti e con accuratezza superiore al 50%, la media rimane simile e la varianza viene ridotta.

Il bagging affronta il problema dell'alta varianza, che causa il sovradattamento del modello ai dati.

#### Previsione *out-of-bag*

L'utilizzo dei campioni casuali permette di usare la tecnica *out-of-bag* per ottenere la stima dell'errore di previsione: per ciascun campione bootstrap, i dati esclusi dal campionamento vengono usati come insieme di verifica, evitando così la necessità della convalida incrociata.

### 22.2.3 Bumping

Anziché prendere la media, per non perdere interpretabilità si può scegliere il modello bootstrap che prevede meglio nell'insieme di stima. Questa procedura si dice *bumping*, o "ricerca del modello stocastica". Effettuare il bumping con procedure come alberi e reti neurali può portare a un buon adattamento.

### 22.2.4 Boosting

Si usa un processo di ricampionamento sequenziale, come nel bagging, ma con probabilità non uguali per ogni osservazione: alla stima del $b$-esimo modello, vengono assegnati pesi maggiori alle osservazioni classificate male dai $b-1$ precedenti classificatori, per far lavorare maggiormente il modello su ciò che non è stato ben classificato.

In realtà, di solito si usano le stesse unità, invece di un campione casuale, pesate con il loro peso aggiornato.

Il sistema di pesi riduce la correlazione tra i classificatori, focalizzandosi sulle regioni "mancate" da quelli precedenti.

**Algoritmo 5** Algoritmo AdaBoost

1: **Init:**

$$w_i = 1/n, \quad i = 1, \dots, n$$

2: **for** $b = 1, \dots, B$ **do**

3: Si stima il modello $\mathcal{C}_b(x)$, pesando le osservazioni per $w_i$.

4: Si calcolano

$$e_b \leftarrow \frac{\sum_{i=1}^n w_i \mathbb{1}_{y_i \neq C_b(x_i)}}{\sum_{i=1}^n w_i}$$

$$\alpha_b \leftarrow \log \frac{1-e_b}{e_b}$$

5: Si assegnano i nuovi pesi

$$w_i \leftarrow w_i \exp\left\{\alpha_b \mathbb{1}_{y_i \neq \mathcal{C}_b(x_i)}\right\}$$

6: **end for**

7: Il classificatore finale è

$$\mathcal{C}(x) = \begin{cases} 1 & \text{se } \frac{\sum_{b=1}^B \alpha_b \mathcal{C}_b(x)}{\sum_{b=1}^B \alpha_b} > \frac{1}{2} \\ 0 & \text{altrimenti} \end{cases}$$

Di solito non è una buona idea usare alberi molto profondi, perché se l'errore di classificazione è nullo, l'algoritmo si ferma immediatamente. Conviene invece utilizzare uno "stump", cioè alberi con uno o due split, che classifichino appena meglio del 50%. In questo caso, quello che si ottiene dal boosting è un modello a sé stante e non una semplice combinazione di alberi.

Se si utilizza un singolo split non si ammette alcuna interazione tra variabili, perché vengono usate una alla volta: per le interazioni, bisogna utilizzare almeno quattro split (profondità due).

### 22.2.5 Random Forests

Sempre ricampionamento e media tra modelli, ma questa volta anche sulle colonne: ad ogni split del $b$-esimo albero, si sceglie un sottoinsieme $F$ delle colonne da utilizzare come predittori.

La costruzione di ogni albero è molto più veloce, perché si utilizzano poche variabili ad ogni nodo. Come nel bagging, ciascun albero viene fatto crescere fino alla sua massima profondità e non viene potato: è l'operazione di media che riduce la varianza e controlla il sovradattamento.

I parametri di regolazione del modello sono $F$ e il numero $B$ di alberi che costituiscono la foresta. Si può dimostrare che non ci sono problemi di sovradattamento al crescere di $B$, perché l'errore converge ad una soglia inferiore. Scegliendo un numero alto per $B$, si può ottenere un errore di previsione non molto distante dal suo limite inferiore e concentrarsi sull'ottimizzazione di $F$.

In molti esami capita che gli studenti utilizzino diverse strade, scegliendo $F = \log p$ oppure $F = \frac{p}{3}$, per cui bisogna studiare come funzionano e giustificarli nella relazione finale $\implies$ leggere perché.

L'utilizzo del bagging, cioè la costruzione di ogni albero su un sottoinsieme dei dati, permette di

1. Migliorare le previsioni.
2. Scegliere $F$ con le previsioni *out-of-bag*.
3. Ottenere misure di importanza delle variabili esplicative

**Importanza delle variabili**

Si fanno crescere $B$ alberi, si effettua la previsione sui dati *out-of-bag* e si calcola l'errore di previsione. Si effettua la previsione sullo stesso insieme *out-of-bag* con gli stessi valori permutati casualmente. Se la variabile è importante, l'errore di previsione sarà molto alto, per cui la differenza si può usare come indicatore di quanto sia importante.

Alla fine si fa una media standardizzata e si ottiene l'importanza della variabile $j$-esima al netto delle altre variabili, motivo per cui non si toglie semplicemente la variabile ma si usa una sua permutazione (se si togliesse la variabile, altre correlate verrebbero usate come surrogato).

C'è anche un altro modo su R, molto più semplice, guardare da soli come si fa. L'indicatore non mostra se l'effetto è calante o crescente, solo l'importanza. Le foreste casuali enfatizzano le interazioni, per cui non è sempre una buona domanda sapere se l'effetto è marginalmente positivo o negativo.

# Lezione 23

*Riferimenti* Azzalini e Scarpa (2012, §5.5)

## 23.1 Analisi discriminante

L'impostazione "statistica" del problema di classificazione assume che ci sia una variabile categoriale $G$, che rappresenta il gruppo a cui un soggetto appartiene. In concomitanza, si osserva la variabile esplicativa $p$-dimensionale $X$, che si vuole modellare condizionatamente al gruppo.

Ogni gruppo ha una distribuzione condizionata per le variabili $X$, cioè $X|G = k$, per cui la densità marginale è una mistura

$$p(x) = \sum_{k=1}^{K} \pi_k p_k(x).$$

Usando il teorema di Bayes, si può ricavare la probabilità a posteriori che un nuovo soggetto appartenga al gruppo $k$:

$$P(G = k|X = x) = \frac{p_k(x)\pi_k}{p(x)}.$$

Per confrontare la probabilità tra due classi si può usare il rapporto logaritmico, che semplifica i conti:

$$\Delta(k, m) = \log \frac{P(G = k|X = x)}{P(G = m|X = x)} = \underbrace{\log \frac{p_k(x)}{p_m(x)}}_{\text{a posteriori}} + \underbrace{\log \frac{\pi_k}{\pi_m}}_{\text{a priori}}.$$

Rimaneggiando la funzione, si può scrivere

$$\begin{aligned}\Delta(k, m) &= \log p_k(x) + \log \pi_k - \log p_m(x) - \log \pi_m \\ &= \delta_k(x) - \delta_m(x),\end{aligned}$$

per cui il confronto viene effettuato tra le *funzioni discriminanti*

$$\delta_j(x) = \log p_j(x) + \log \pi_j.$$

Quel valore $j$ che massimizza $\delta_j(x^*)$ per la nuova osservazione $x^*$ indica il gruppo in cui viene classificato il nuovo soggetto.

### Stima delle probabilità

Come probabilità a priori, si può stimare $\pi_k$ con le proporzioni del gruppo $k$ osservate nel campione:

$$\hat{\pi}_k = \frac{n_k}{n}.$$

Il problema principale è stimare $p(x)$, che si può fare in diversi modi, parametrici o non.

### 23.1.1 Analisi discriminante lineare

Nell'analisi discriminante *lineare*, si assume che $p_k \sim \mathcal{N}_p(\mu_k, \Sigma)$, con uguale matrice di varianza e covarianza per ogni gruppo, da cui si ottiene la funzione discriminante

$$\delta_k(x) = x^\intercal \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^\intercal \Sigma^{-1} \mu_k + \log \mu_k,$$

che è lineare in $x$, da cui il nome *analisi discriminante lineare* (LDA).

*Dim.*

$$\begin{aligned}
\delta_k(x) &= -\frac{p}{2} \log 2\pi - \log |\Sigma| - \frac{1}{2}(x - \mu_k)^\intercal \Sigma^{-1}(x - \mu) + \log \mu_k \\
&= \underbrace{-\frac{p}{2} \log 2\pi - \log |\Sigma| - \frac{1}{2} x^\intercal \Sigma^{-1} x}_{\text{costante in } k} + x^\intercal \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^\intercal \Sigma^{-1} \mu_k + \log \pi_k \\
&= x^\intercal \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^\intercal \Sigma^{-1} \mu_k + \log \pi_k.
\end{aligned}$$

□

Il problema rimane la stima dei parametri, che con il metodo dei momenti è

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i : g_i = k} x_i,$$

$$\hat{\Sigma} = \frac{1}{n - K} \sum_{k=1}^{K} \sum_{i : g_i = k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^\intercal.$$

Data la natura delle stime, si può fare un aggiornamento ricorsivo se i dati sono in streaming. Inoltre, si può utilizzare un algoritmo parallelo per velocizzare la stima del modello.

**Nota** Questa procedura si può giustificare anche senza la normalità, solamente usando considerazioni del secondo ordine.

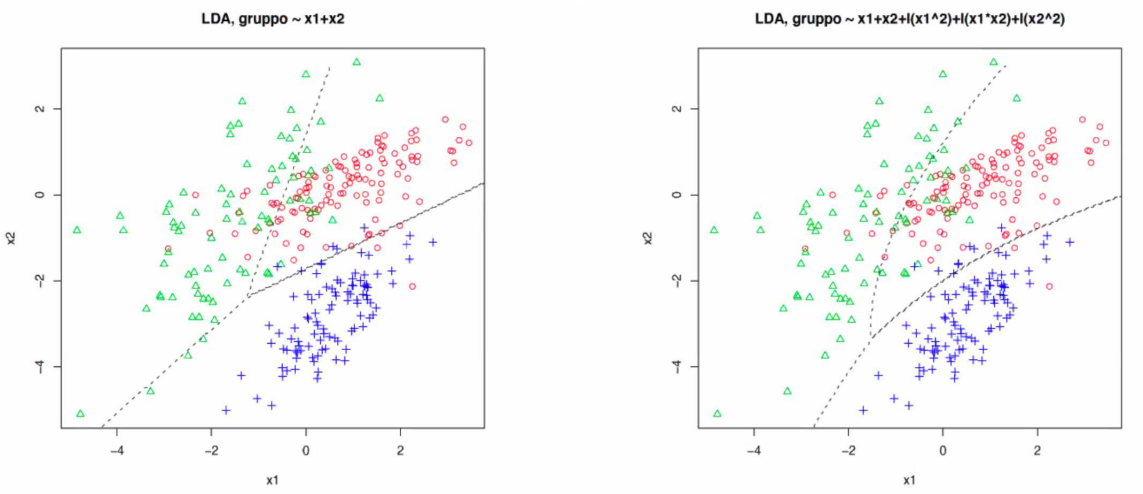


Figura 38: LDA

### 23.1.2 Analisi discriminante quadratica

Assumendo che le varianze siano diverse all'interno delle classi, cioè $p_k \sim \mathcal{N}_p(\mu_k, \Sigma_k)$, per l'analisi discriminante *quadratica* si ottiene la funzione discriminante

$$\delta_k(x) = \log \pi_k - \frac{1}{2}(x-\mu_k)^\mathsf{T}\Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\Sigma_k|.$$

La stima di $\Sigma_k$ è data dalla varianza all'interno del gruppo

$$\hat{\Sigma}_k = \frac{1}{n_k - 1}\sum_{i:g_i=k}(x_i - \hat{\mu}_k)^\mathsf{T}(x_i - \hat{\mu}_k).$$

**Pro**

- Appropriatezza del metodo.
- Include informazioni a priori.
- Semplicità di calcolo.
- Qualità e stabilità dei risultati.
- Robustezza rispetto alle ipotesi

**Contro**

- Ipotesi restrittive nella QDA, più legata all'ipotesi gaussiana.
- Numero di parametri (QDA).
- **!** Niente selezione o importanza delle variabili.
- Non robustezza delle stime.

**Nota** Se si hanno solo due classi bilanciate, si può mostrare che LDA e modello lineare sono esattamente equivalenti (esercizio). Si può dunque utilizzare la classificazione con la regressione lineare per selezionare le variabili più rilevanti per l'analisi discriminante.

## 23.2 Support-Vector Machines

L'idea alla base delle support-vector machines è di ricercare l'iperpiano $\{x : f(x) = \beta_0 + x^\mathsf{T}\beta = 0\}$, che separi nel modo migliore i gruppi. Si assumono due classi per $y_i$, codificate per comodità $+1$ e $-1$, ma è generalizzabile a più gruppi.

### 23.2.1 Iperpiani separanti ottimi

*Riferimenti* Hastie et al. (2013, §4.5.1-§4.5.2)

Si vuole trovare il miglior iperpiano separatore delle classi, ovvero quello che *massimizza la distanza dal punto più vicino.*

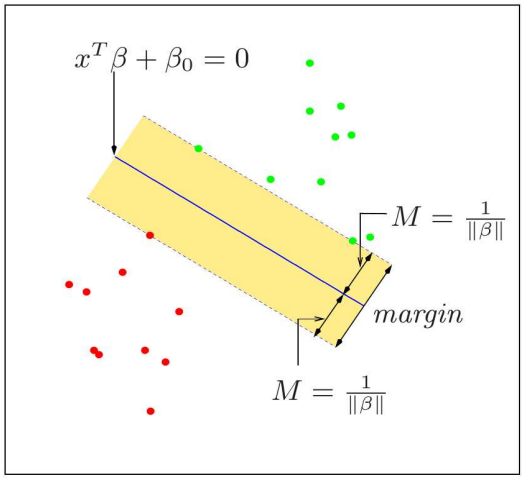


Figura 39: Support-vector classifiers, nel caso in cui i gruppi siano perfettamente separabili.

L'iperpiano individuato avrà la stessa distanza $m$ dal più vicino elemento di ciascuna delle due classi. Alla retta parallela sono associate due rette parallele tratteggiate, che passano per il punto più prossimo a ciascuna classe.

**Nota** Non sono necessarie tutte le osservazioni per identificare la separazione, ma solo quattro, che danno il confronto tra due rette migliori. Anche con milioni di punti, gli unici che interessano sono quelli vicini alla frontiera.

Per un qualunque iperpiano $A$, valgono le seguenti proprietà:

- Per ogni $x' \in A$, $\beta^\mathsf{T} x' = -\beta_0$
- Se $x', x'' \in A$, allora $\beta^\mathsf{T}(x' - x'') = -\beta_0 + \beta_0 = 0$.
- Segue che il vettore normale all'iperpiano è $\beta^* = \beta/\|\beta\|$, in quanto $x', x'' \in A \implies x' - x'' \in A$.
- La distanza con segno di un punto $x$ da $A$ è data da

$$\begin{aligned}\beta^\mathsf{T}(x - x_0) &= \frac{1}{\|\beta\|}(\beta^\mathsf{T} x + \beta_0) \\ &= \frac{1}{\|\beta\|} f(x).\end{aligned}$$

L'obiettivo è trovare l'iperpiano con $\|\beta\| = 1$ (identificabilità) che massimizzi la distanza tra i punti nei due gruppi. Per un dato iperpiano, un'unità risulta classificata correttamente o meno se

$$\begin{cases} y_i(\beta_0 + x_i^\mathsf{T}\beta) > 0 & \text{stesso segno} \\ y_i(\beta_0 + x_i^\mathsf{T}\beta) < 0 & \text{segni diversi} \end{cases}$$

Il problema di ottimizzazione si può formulare allora come

$$\max_{\beta_0,\beta} M$$

$$\text{s.t. } \|\beta\| = 1, y_i(\beta_0 + x_i^\mathsf{T}\beta) \geq m \quad i = 1, \ldots, n.$$

Il vincolo $\|\beta\| = 1$ è difficile da implementare, per cui si preferisce modificare i vincoli come

$$\frac{1}{\|\beta\|} y_i(\beta_0 + x_i^\mathsf{T}\beta) \geq m \implies y_i(\beta_0 + x_i^\mathsf{T}\beta) \geq m\|\beta\|,$$

che implica solo una modifica nella definizione di $\beta_0$.

Siccome la moltiplicazione non altera i vincoli, si può porre la condizione $\|\beta\| = 1/M$, e riscrivere la funzione obiettivo nella forma equivalente

$$\min_{\beta_0,\beta} \frac{1}{2}\|\beta\|^2$$

$$\text{s.t. } y_i(\beta_0 + x_i^\mathsf{T}\beta) \geq 1, \quad i = 1, \ldots, n.$$

Il problema in questa forma è di minimo per una funzione quadratica con vincoli lineari sullo spazio, dunque semplice.

La stima finale produce il classificatore

$$\mathcal{C}(x) = \begin{cases} +1 & \text{se } \hat{\beta}_0 + x^\mathsf{T}\hat{\beta} > 0 \\ -1 & \text{se } \hat{\beta}_0 + x^\mathsf{T}\hat{\beta} < 0 \\ \text{a caso} & \text{se } \hat{\beta}_0 + x^\mathsf{T}\hat{\beta} = 0 \end{cases}$$

La frontiera è l'insieme di valori per cui $f(X) = 0$, e i coefficienti che risolvono il problema sono definiti in termini di un (piccolo) numero di *punti di supporto*

$$\hat{\beta} = \sum_{j \in \mathcal{S}} \alpha_j x_j,$$

per opportuni pesi $\alpha_j$.

### 23.2.2 Punti non separabili

*Riferimenti* Hastie et al. (2013, §12.1-12.3)

Se i punti non sono separabili, si può generalizzare la procedura appena descritta inserendo delle

variabili di *slack* $\xi = (\xi_1, \xi_2, \ldots, \xi_n)$, che esprimono quanto i vari punti vengono classificati oltre il margine di classe. Se il punto $x_i$ è correttamente classificato, $\xi_i = 0$.

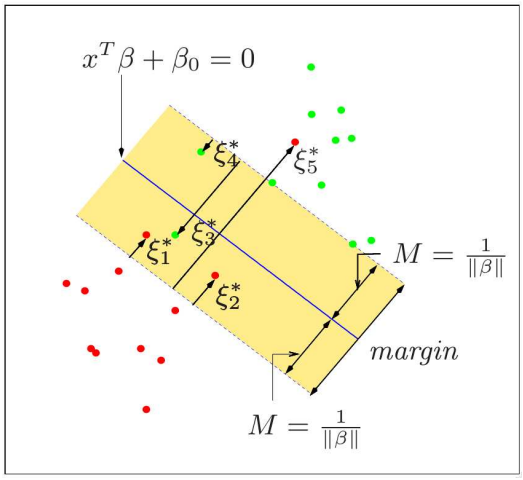


Figura 40: Support-vector machines, nel caso in cui i due gruppi non siano linearmente separabili.

La generalizzazione si ottiene permettendo ai punti di superare i margini, usando quindi *soft margins*, accettando un numero massimo $B$ di punti mal classificati

$$\sum_{i=1}^{n} \xi_i \leq B.$$

Utilizzando le variabili di *slack* $\xi_i$, si possono modificare i vincoli scegliendo

$$y_i(\beta_0 + x_i^{\mathsf{T}}\beta) \geq M - \xi_i$$

oppure

$$y_i(\beta_0 + x_i^{\mathsf{T}}\beta) \geq M(1 - \xi_i)$$

In questo caso si sceglie la seconda opzione, visto che la prima porterebbe a un problema di ottimizzazione non convessa, mentre la seconda lo manterrebbe convesso. Rielaborando il problema, si arriva a scriverlo come

$$\min_{\beta_0,\beta} \frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \begin{cases} y_i(\beta_0 + x_i^{\mathsf{T}}\beta) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \quad \forall i$$

In questo caso, $\gamma$ gioca il ruolo di parametro di regolazione e rappresenta il costo di violazione delle barriere.

In questo caso, la soluzione del problema di ottimo è del tipo

$$\hat{\beta} = \sum_{i=1}^{n} a_i x_i y_i \implies \hat{f}(x) = \hat{\beta}_0 + \sum_{i=1}^{n} a_i y_i x x_i$$

dove solo alcuni degli $a_i$ sono non nulli. Le osservazioni che vengono usate sono dette *support vectors* e in questo caso sono sia punti sulla frontiera sia dalla parte "sbagliata" della frontiera.

### 23.2.3 Support Vector Machines e Kernel

Per estendere il modello a regioni di separazione non lineari, si possono includere delle trasformazioni di $x$, come nel caso delle splines:

$$h(x) = \big(h_1(x), h_2(x), \ldots, h_q(x)\big)^\mathsf{T},$$

dove $q$ può essere maggiore/minore/uguale di $p$. In questo caso, l'iperpiano di separazione diventa

$$f(x) = \beta_0 + h(x)^\mathsf{T}\beta = 0$$

e la stima ha espressione del tipo

$$\begin{aligned}\hat{f}(x) &= \hat{\beta}_0 + \sum_{i=1}^{n} a_i y_i h(x)^\mathsf{T} h(x_i) \\ &= \hat{\beta}_0 + \sum_{i=1}^{n} a_i y_i \langle h(x), h(x_i)\rangle.\end{aligned}$$

Si possono specificare allora delle *funzioni nucleo* (*kernel functions*) definite solo in base al prodotto interno tra i punti

$$K(x, x') = \langle h(x), h(x')\rangle,$$

che calcola i prodotti interni nello spazio a dimensione maggiore delle variabili trasformate, senza però effettuare la trasformazione.

Ad esempio, si possono usare basi polinomiali

$$K(x, x_i) = \left(1 + \langle x, x_i\rangle\right)^d.$$

Al crescere di $p$ e $d$ lo spazio diventa enorme ma non lo si visita mai completamente, solamente nei punti di supporto

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{i\in\mathcal{S}} a_i y_i K(x, x_i).$$

Tabella 5: Principali funzioni nucleo comunemente adottate.

| **Nucleo** | $\boldsymbol{K(x, x')}$ |
|---|---|
| polinomiale grado $d$ | $\left(1 + \langle x, x'\rangle\right)^d$ |
| radiale | $\exp(-\gamma\|x - x'\|^2)$ |
| sigmoidale | $\tanh(\kappa_1\langle x, x'\rangle + \kappa_2)$ |

Con nucleo polinomiale di grado $d = 2$, se $p = 2$ si ha $x = (x_1, x_2)$ e la funzione nucleo si può scrivere

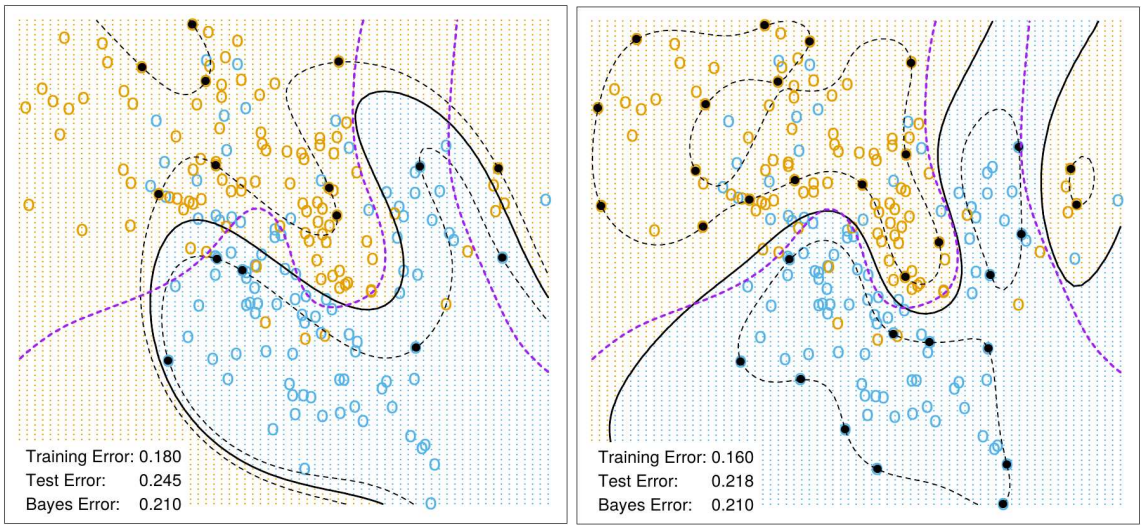


Figura 41: Support-vector machines con nucleo polinomiale di grado 4, a sinistra, e con nucleo radiale, a destra.

come

$$\begin{aligned} K(x, x_i) &= (1 + \langle x, x' \rangle)^2 \\ &= (1 + x_1 x_1' + x_2 x_2')^2 \\ &= 1 + (x_1 x_1')^2 + (x_2 x_2')^2 + 2x_1 x_1' + 2x_2 x_2' + 2x_1 x_1' x_2 x_2', \end{aligned}$$

che si può vedere come prodotto interno con 6 funzioni

$$h(x) = \big(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2\big).$$

I parametri di regolazione $(\gamma, d, \kappa_1, \kappa_2)$ controllano quanto "frastagliate" possono essere le regioni di classificazione, e vanno scelti in modo da controllare varianza e distorsione.

### 23.2.4 Nucleo radiale

*Riferimenti* Hastie et al. (2013, §6.7)

Il *nucleo radiale* è il più utilizzato e si definisce come

$$K(x, x_i) = e^{-d\|x - x_i\|^2}.$$

La base implicita $h(x)$ è di dimensione infinita e il parametro $\gamma$ misura il peso assegnato al kernel.

Le support-vector machines funzionano anche se $p \gg n$, perché anche se ci sono più iperpiani di separazione, si sceglie quello con margine di separazione maggiore.

Gli aspetti computazionali sono proibitivi? Pensarci per casa

| | Metodo | Errore nell'insieme di verifica (SE) | |
|---|---|---|---|
| | | Senza variabili "rumore" | Sei variabili "rumore" |
| 1 | SV Classifier | 0.450 (0.003) | 0.472 (0.003) |
| 2 | SVM/poly 2 | 0.078 (0.003) | 0.152 (0.004) |
| 3 | SVM/poly 5 | 0.180 (0.004) | 0.370 (0.004) |
| 4 | SVM/poly 10 | 0.230 (0.003) | 0.434 (0.002) |
| 5 | GAM | 0.084 (0.003) | 0.090 (0.003) |
| 6 | MARS | 0.156 (0.004) | 0.173 (0.005) |
| | *Bayes* | *0.029* | *0.029* |

Figura 42: SVMCurse

Problema delle sfere annidate e maledizione della dimensionalità: il test mostra che il SVM funziona bene, ma peggiora di molto quando si aggiungono variabili di rumore. Nei GAM e MARS invece no, perché per un'ipersfera basta una funzione quadratica.

# Lezione 24

## 24.1 Text Mining

*Riferimenti* Ceron et al. (2013)

Il text mining ha l'obiettivo di estrarre informazione da testi (documenti, tweet/post, ...), i quali sono caratterizzati dall'essere dati *destrutturati*. Il primo passo per l'analisi è, necessariamente, strutturare i testi in una classica matrice di dati, per poi effettuare l'analisi.
I passi da seguire nell'analisi sono i seguenti

1. Estrazione dell'insieme di dati (*corpus*).
2. *Preprocessing* dei dati.
3. Classificazione manuale (*tagging*) di un piccolo sottoinsieme di dati.
4. Modellazione e analisi.

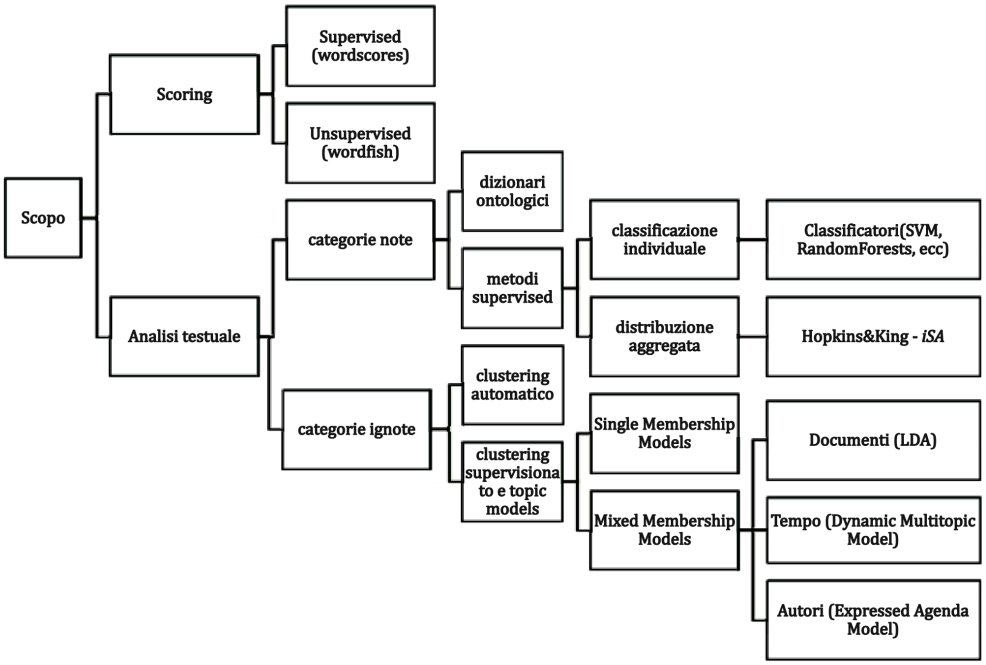


Figura 43: Esigenze e soluzioni nell'analisi dei testi.

### 24.1.1 Preprocessing

È la parte più complicata e dispendiosa di tempo, procede per più punti:

- **Analisi lessicale**:
  1. Eliminazione delle *stop words*, ovvero parole troppo/troppo poco frequenti, non specifiche e che non forniscono informazioni.

2. Eliminazione dei modi di dire.
3. Eliminazione di caratteri specifici (emoji, altri alfabeti, ...), caratteri speciali ($, %, ?, !, @, ...)

- **Stemming**: processo automatico, tramite il quale si riducono le parole alla loro radice fondamentale (*tema*). Le parole vengono convertite in *stilemi* (*stems*), che possono essere anche combinazioni di parole con significato specifico (unigrammi, bigrammi, trigrammi, ...)

| Post | $D_i$ | Stem s1 nucleare | Stem s2 paura | Stem s3 radiazioni | Stem s4 inquinamento | Stem s5 scorie | Stem s6 economico | ... |
|---|---|---|---|---|---|---|---|---|
| testo #1 | a favore | 1 | 0 | 0 | 0 | 0 | 1 | ... |
| testo #2 | NA | 1 | 0 | 0 | 0 | 1 | 0 | ... |
| testo #3 | contro | 1 | 1 | 1 | 1 | 1 | 0 | ... |
| testo #4 | contro | 1 | 1 | 1 | 1 | 1 | 0 | ... |
| testo #5 | a favore | 1 | 0 | 1 | 1 | 1 | 0 | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| testo #10000 | a favore | 1 | 0 | 1 | 0 | 0 | 1 | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

Figura 44: Riduzione dei dati ad una *matrice di stemming*.

### 24.1.2 Tagging

Il tagging consiste nella classificazione di un piccolo sottoinsieme di dati, a partire dal quale si effettua l'analisi dei restanti. Il tagging può essere

1. **Automatico** tramite *dizionari ontologici*: serve un dizionario raffinato, perché ogni dominio e argomento diverso necessitano di dizionari diversi.
2. **Manuale**: si codifica molto bene qualsiasi argomento, ma ha problematiche relative al costo, tempo e sensibilità dei temi da classificare. Al netto del costo è vantaggiosa.

### 24.1.3 Tipologia di analisi

La tipologia di analisi può essere orientata a:

- **Scoring**: allineamento dei testi secondo una dimensione latente, generalmente tramite PCA, analisi fattoriale, t-SNE, MultiDimensional Scaling, ...
- **Analisi del testo**: lo scopo è classificare i testi, in modo supervisionato e non. A seconda dell'interesse, si può avere
  1. Classificazione **individuale**: SVM, RandomForest, ...
  2. Classificazione **aggregata**: *Integrated Sentiment Analysis* (iSA), metodologia che produce informazioni aggregate all'intera popolazione.

## 24.2 Sentiment Analysis

*Riferimenti* Ceron et al. (2016)

**Pro**

+ Dati geolocalizzati.
+ Analisi retrospettiva, per cui opinioni già espresse.
+ Analisi in tempo reale, monitoraggio continuo.
+ Velocità di analisi elevata.
+ Raccolta di opinioni non sollecitate.
+ Analisi censuaria non campionaria, si riduce il problema di qualità dell'informazione

**Contro**

– Popolazione non rappresentativa della popolazione demografica.
– Analisi retrospettiva, quindi non c'è controllo.
– Linguaggio in evoluzione, per cui quello che andava bene ieri non per forza va bene oggi.
– Big data $\implies$ problemi computazionali.

Volendo classificare un corpus in $D_1, \dots, D_j$ categorie, $j$ non troppo grande, si suddivide il dataset in training e test, dove

1. Training set: testi letti e classificati manualmente da persone, per cui si conoscono .
2. Test set: testi non letti, di cui non conosciamo la "$y$" e di cui vogliamo la distribuzione aggregata, non quella individuale.

Se si usasse un approccio standard, si dovrebbe marginalizzare $P(\boldsymbol{D}|\boldsymbol{S})$, calcolata usando SMV/RandomForest, rispetto agli stilemi $\boldsymbol{S}$

$$P(\boldsymbol{D})_{k\times 1} = P(\boldsymbol{D}|\boldsymbol{S})_{k\times 2^m} \cdot P(\boldsymbol{S})_{2^m\times 1}.$$

Il problema è che con un tasso di errata classificazione del $2-3\%$, si ottiene un errore anche del $20\%$.

### 24.2.1 Integrated Sentiment Analysis

Per risolvere il problema di propagazione dell'errore, si può allora l'analogo del modello lineare per stimare $P(\boldsymbol{D})$ a partire da

$$P(\boldsymbol{S})_{2^m\times 1} = P(\boldsymbol{S}|\boldsymbol{D})_{2^m\times k} \cdot P(\boldsymbol{D})_{k\times 1}.$$

Si assume che la matrice $P(\boldsymbol{S}|\boldsymbol{D})$ stimata dal dataset di training sia la stima della stessa matrice nella popolazione. Il calcolo di $P(\boldsymbol{D})$ è come la stima dei parametri di un modello lineare, tramite

$$P(\boldsymbol{D}) = P(\boldsymbol{S}|\boldsymbol{D})^- P(\boldsymbol{S}),$$

dove $X^- = (X^\intercal X)^{-1} X^\intercal$. Il vantaggio di questo approccio è che si ottiene una classificazione aggregata in modo molto più efficiente, eliminando la propagazione individuale dell'errore con la regressione.

# Introduzione al Deep Learning

Deep learning nasce in contesto informatico e si è sviluppata a partire dal 2010 in poi, specialmente con lo sviluppo di provider cloud. I campi applicativi sono quelli *percettivi*, ovvero il riconoscimento di immagini, contenuti testuali, sistemi di raccomandazione, ...

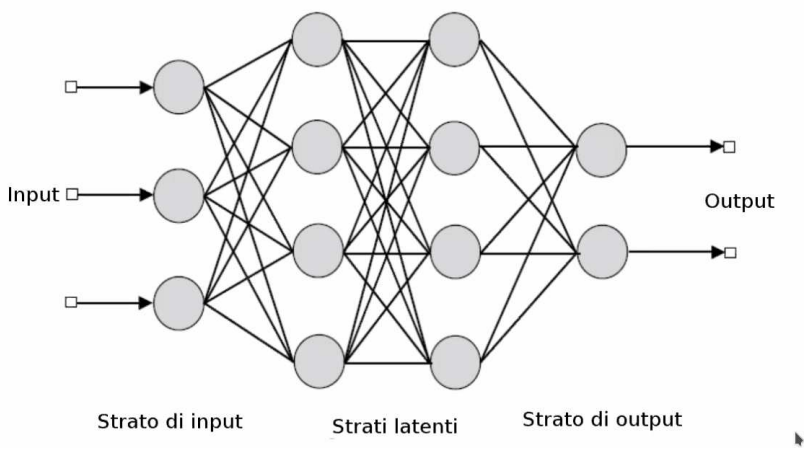


Figura 45: *Feed-forward neural network* (FFNN).

La struttura di una deep neural network è costituita da nodi $\boldsymbol{a}^{(l)}$ corrispondenti allo strato $l$-esimo e $W^{(l)}$ matrice dei parametri che collegano gli strati $l$ e $l+1$. Lo strato $l$ è collegato con lo strato precedente attraverso

$$\boldsymbol{z}^{(l)} = W^{(l-1)}\boldsymbol{a}^{(l-1)}$$

$$\boldsymbol{a}^{(l)} = g^{(l)}\big(\boldsymbol{z}^{(l)}\big),$$

dove $g^{(l)}(\cdot)$ è detta *funzione di attivazione*.

Le funzioni di attivazione dipendono fortemente dal tipo di problema che si cerca di risolvere:

**Regressione**

- Un solo nodo di output
- $g^{(L)}\big(\boldsymbol{z}^{(l)}\big) = z^{(l)}$

**Classificazione**

- $k$ nodi di output
- $g^{(L)}\big(\boldsymbol{z}^{(l)}\big) = \dfrac{e^{\boldsymbol{z}^{(l)}}}{\sum_{j=1}^{K} e^{\boldsymbol{z}^{(l)}}}$

Sia $\boldsymbol{W} = \big[W^{(1)}W^{(2)}\dots W^{(L-1)}\big]$, la stima dei parametri $\hat{\boldsymbol{W}}$ della rete neurale è data da

$$\hat{\boldsymbol{W}} = \underset{\hat{\boldsymbol{W}}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} L\big(y_i, f(x_i, \boldsymbol{W})\big).$$

In problemi di classificazione si minimizza la *cross-entropia*, cioè il negativo della log-verosimiglianza multinomiale:

$$H(\hat{\boldsymbol{W}}) = -\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik}\log f_k(x_i; \hat{\boldsymbol{W}}).$$

### Backpropagation

L'algoritmo largamente più utilizzato per stimare le reti neurali è il *backpropagation*, che permette di stimare i parametri con basso costo computazionale.

1. *Forward*: con il passo in avanti si ottiene $\hat{f}(x_i, \boldsymbol{W})$ tenendo fisso $\boldsymbol{W}$.
2. *Backward*: con il passo all'indietro vengono aggiornati i parametri in due fasi
   (a) *Propagazione*: calcolo del gradiente.
   (b) *Aggiornamento*: dato il gradiente, effettuare l'aggiornamento.

Nella fase di propagazione, si ricavano le derivate parziali rispetto a $\boldsymbol{z}$, definite $\delta_i^{(2)}(\boldsymbol{z}), \ldots, \delta_i^{(L)}(\boldsymbol{z})$. In realtà è necessario solo calcolare $\delta_i^{(L)}(\boldsymbol{z})$, riferita allo strato di output, perché poi viene *propagato all'indietro* ricorsivamente con la formula delle funzioni composte. Una volta calcolate le derivate rispetto $\boldsymbol{z}$, si possono calcolare quelle rispetto ai parametri $\delta_i^{(1)}(W), \ldots, \delta_i^{(L)}(W)$.

L'aggiornamento viene effettuato tramite la discesa del gradiente pesata per un *learning rate* $\eta \in (0, 1]$ :

$$W_{t+1}^{(l)} = W_t^{(l)} - \eta \cdot \nabla L\big(W_t^{(l)}\big), \quad l = 1, \ldots, L-1,$$

dove

$$\nabla L\big(W^{(l)}\big) = \frac{1}{n} \sum_{i=1}^{n} \delta_i^{(l)}(W)$$

### Mini-batch (stochastic) gradient descent

In contesti di analisi di grandi moli di dati, l'aggiornamento dei parametri è molto lento se si utilizzano tutte le $n$ osservazioni. Si usano invece sottocampioni di numerosità fissata $m \ll n$, effettuando uno step per ogni sottoinsieme di dati. In generale, si usano poche centinaia di osservazioni e si hanno due vantaggi:

1. Velocizzazione dell'algoritmo di stima.
2. Gradiente "più rumoroso", per cui è possibile esplorare lo spazio parametrico in modo più completo, uscendo dai minimi locali.

Le funzioni di attivazione sono la logistica e la tangente iperbolica

$$\begin{aligned} \text{logit}^{-1}(z) &= \frac{1}{1 + e^{-z}} \\ \tanh(z) &= 2 \cdot \text{logit}^{-1}(2z) - 1 \\ &= \frac{e^z - e^{-z}}{e^z + e^{-z}} \end{aligned}$$

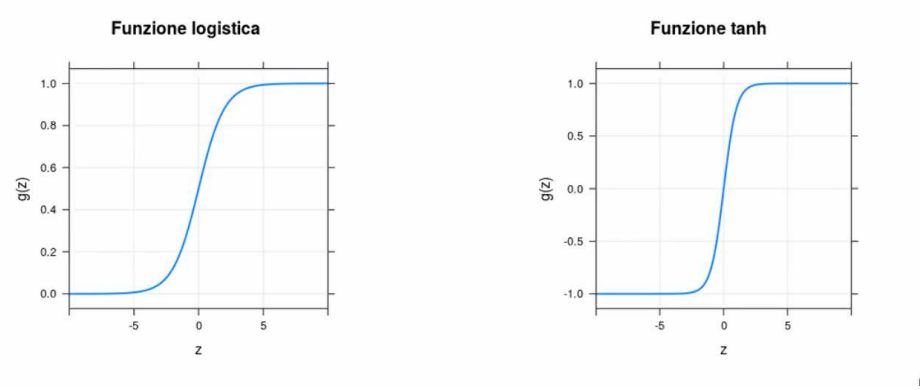


Figura 46: activFunctions

Queste funzioni hanno alcuni problemi: variazioni molto grandi di $z$ portano a variazioni molto minime della funzione. Inoltre, c'è un problema di "scomparsa del gradiente", cioè spostandosi dallo 0 si ha un gradiente quasi nullo e l'aggiornamento dei parametri è impossibile.

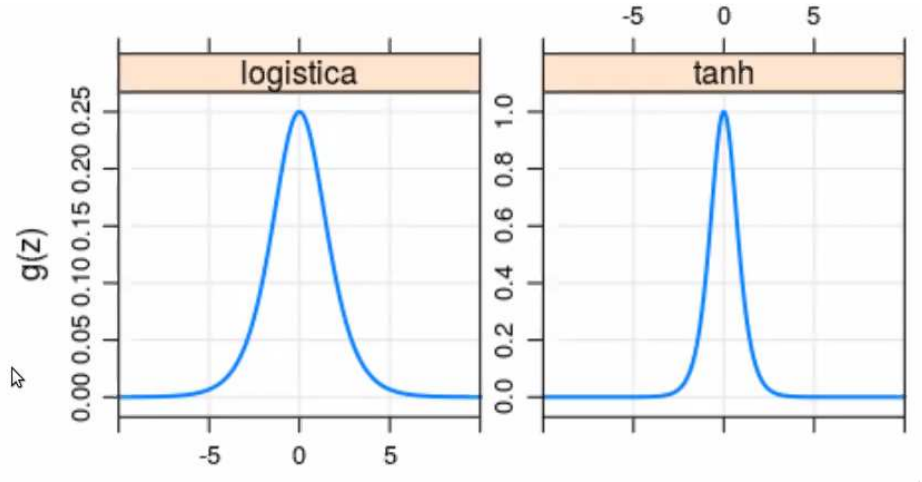


Figura 47: activFunctionDeriv

Si usa allora la funzione ReLU (*REctified Linear Unit*), che non è limitata e ha un gradiente nullo per il 50% dei nodi.

$$g(z) = \max(0, z) \implies g'(z) = \begin{cases} 0 & \text{se } z \leq 0 \\ 1 & \text{se } z > 0 \end{cases}$$

**Compromesso varianza-distorsione**

La scelta della configurazione migliore degli iperparametri viene fatta minimizzando il compromesso varianza-distorsione. Le tecniche principali per minimizzare il MSE sono

1. *Complessità*: stabilita dal numero di parametri e dal numero di stati latenti.
2. *Early stopping*: bloccare l'algoritmo dopo un certo numero di iterazioni.
3. *Altri metodi*: penalizzazione, dropout, ...

$$\hat{\boldsymbol{W}} = \underset{\hat{\boldsymbol{W}}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L\big(y_i, f(x_i, \boldsymbol{W})\big) + \lambda J(\boldsymbol{W}),$$

dove si può usare come penalizzazione il *weight decay*, che corrisponde alla penalizzazione $L_2$:

$$J(\boldsymbol{W}) = \frac{1}{2}\|\boldsymbol{W}\|_2^2 = \frac{1}{2} \sum_{l=1}^{L-1} \sum_{i=1}^{p_l} \sum_{j=1}^{p_{l+1}} \big(w_{ij}^{(l)}\big)^2$$

Il metodo dropout consiste nel porre a zero una porzione di nodi scelta casualmente.

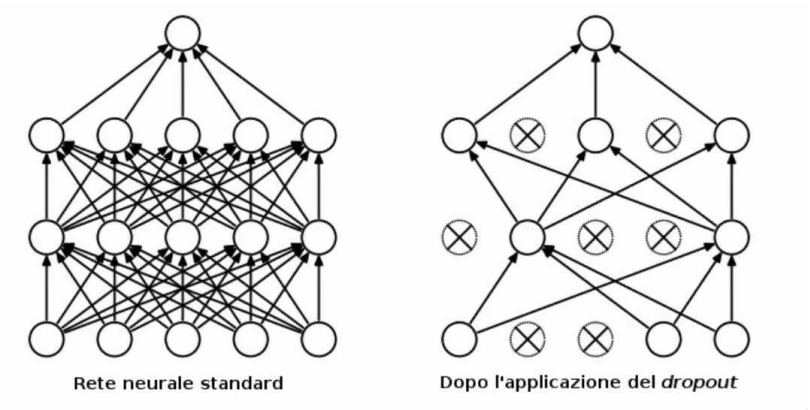


Figura 48: dropout

Il dropout ha la caratteristica di essere

- Approssimazione del risultato ottenibile come combinazione di classificatori.
- Simile alla logica delle random forest.
- Miglioramento dell'accuratezza previsiva e si evita sovradattamento.
- La probabilità $p$ di conservare un nodo è un parametro di regolarizzazione.
- Il dropout è particolarmente utile quando sono disponibili numerose osservazioni.

# Introduzione al calcolo parallelo e C++

| Problemi | Soluzioni |
|---|---|
| 1. Limiti della *CPU*: potenza di calcolo | Calcolo parallelo, C/C++ |
| 2. Limiti della *RAM*: capacità di memoria insufficiente | Aumentare la *RAM* |
| 3. Limiti *I/O*: lettura e scrittura di file su disco | Software efficiente |
| 4. Limiti di *trasferimento*: la copia dei dati richiede troppo tempo | Inviare dati per posta |

Il calcolo parallelo si basa sul suddividere le operazioni tra le unità di calcolo disponibili, sia per *CPU* che per *GPU*. I risultati ottenuti nei diversi blocchi vengono poi aggregati per ottenere il risultato finale.

Idea vecchia, ma ora i processori multicore sono praticamente ovunque, per cui è diventato spesso rilevante. La *GPU* è una recente aggiunta al panorama del calcolo parallelo, in particolare `tensorflow` si appoggia alla *GPU* per la stima in parallelo dei modelli.

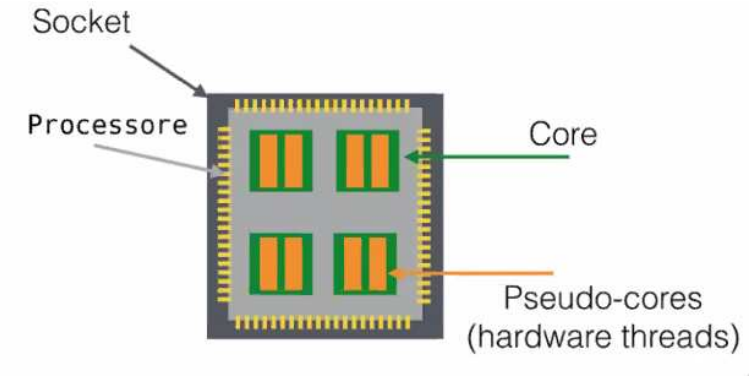


Figura 49: Processore a 8 core logici: 4 fisici con 2 thread per core.

Il processore è costituito da un socket, composto da più *core* fisici che possono lavorare in parallelo. In particolare, ogni core è talmente potente che le operazioni semplici non consumano tutte le risorse del core, per cui si possono fare più operazioni (*thread*) in contemporanea.

Il numero massimo di core disponibili è un limite massimo alla parallelizzazione, ma non necessariamente lo si può raggiungere: alcuni algoritmi sono sequenziali (backfitting, minimi quadrati ricorsivi, ...) e non si possono parallelizzare.

In generale, solo una frazione $f$ del codice totale può essere eseguita *simultaneamente*, mentre il resto è irriducibile.

- "Embarassingly parallel": $f \approx 1$, ad esempio se voglio sommare le colonne di una matrice.
- Casi realistici: $f < 1$

**Legge di Ahmdahl**

C'è un'equazione semplice che mette in relazione il guadagno nella parallelizzazione del codice: data $f$, proporzione parallelizzabile di codice e $k$, numero di processi paralleli, se si definisce il guadagno

$$\text{guadagno} = \frac{\text{tempo base}}{\text{tempo parallelo}},$$

allora si ha

$$\text{guadagno} \leq \frac{1}{f/k + (1-f)}.$$

Per $k \to \infty$, ad esempio se affittiamo un server con 100 mio. di processori,

$$\text{guadagno} \leq \frac{1}{1-f}.$$

Se $f = 95\%$, cioè solo il 5% del codice è non parallelizzabile, si può migliorare al massimo di 20 volte la velocità.

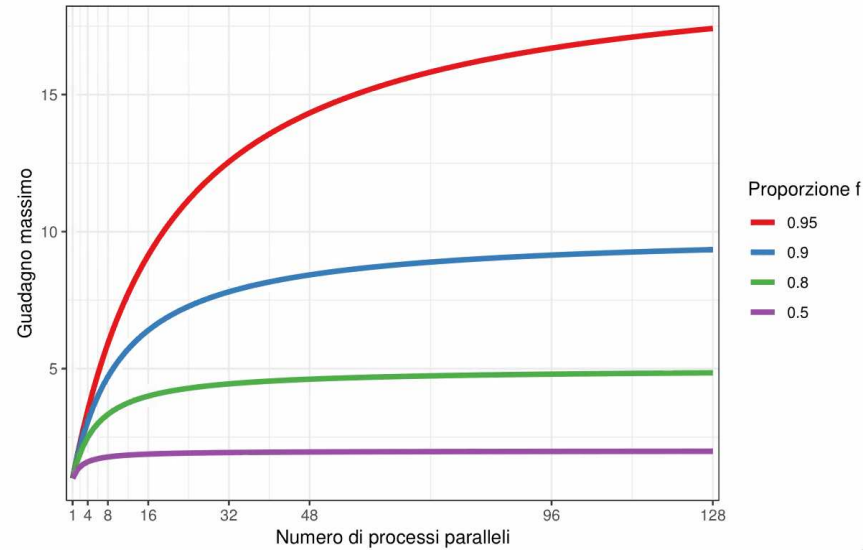


Figura 50: Esempio di guadagno massimo per diverse proporzioni $f$ di codice parallelizzabile.

Dalle curve, anche aumentando il numero di processi paralleli ($32 \to 96$ per $f = 0.8$), il guadagno non è così grande. Inoltre, nei casi pratici si va molto peggio dei casi teorici ideali, per vari motivi:

1. *Communication overhead*: trasferimento di dati tra processi, in input o output. I core devono comunicare, ad esempio quando ricevono dati o quando si vuole aggregare i risultati.
2. *Load balance*: assegnazione efficiente dei lavori ai diversi processori, non sempre quella "democratica" è la più efficiente.

Il calcolo parallelo funziona bene quando c'è poco communication overhead e quando le operazioni sono molto semplici da eseguire. Le *GPU* hanno tantissimi core e fanno operazioni molto semplici (e.g. calcolo matriciale).

## 2.1 Approcci alla parallelizzazione

Ci sono tre principali approcci alla parallelizzazione:

1. A livello di *dati* (*data parallelism*).
2. A livello di *algoritmo* (*model parallelism*).
3. A livello di *operazioni* (*operation parallelism*).

Dal punto di vista pratico, le librerie come `tensorflow` fanno operazioni in parallelo a livello di dati (1). Quello più diffuso è la parallelizzazione (3) a livello di operazioni di base, che chiamiamo parallelizzazione implicita. L'approccio (2) è utilizzato per algoritmi che *nascono* per essere parallelizzati, come ad esempio cicli `for` indipendenti.

### Parallelizzazione a livello di dati

Consideriamo di voler stimare un modello lineare con $p = 1$: $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$. Lo scopo è calcolare

$$\hat{\beta}_1 = \frac{\mathrm{Cov}(x, y)}{\mathbb{V}[x]}, \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}.$$

Dividiamo i dati in $k$ chunk di dimensione $n_k = n/k$ per $k + 1$ processori (l'ultimo aggrega):

$$x = \left(x^{(1)}, x^{(2)}, \ldots, x^{(k)}\right)^{\mathsf{T}}, \quad y = \left(y^{(1)}, y^{(2)}, \ldots, y^{(k)}\right)^{\mathsf{T}}.$$

Ad ogni core assegnamo uno di questi chunk, che li elabora e li trasmette i risultati all'ultimo processore.

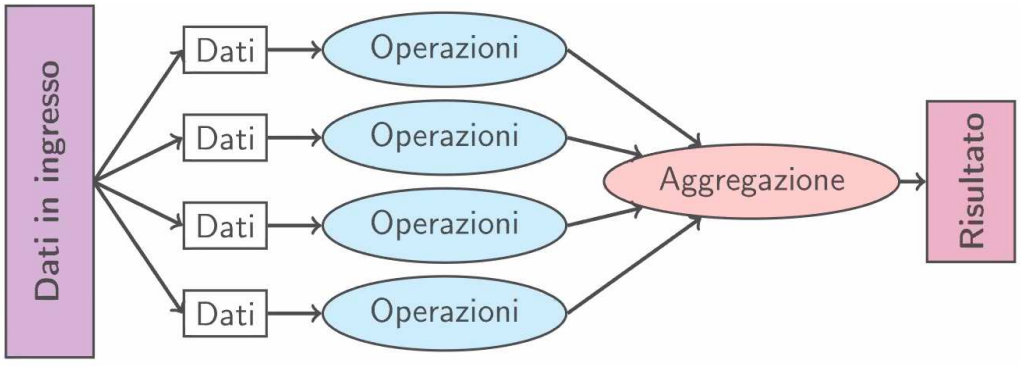


Figura 51: Approccio "map-reduce" per il calcolo parallelo.

A livello di dati, la covarianza è

$$\mathrm{Cov}(x, y) = \frac{1}{n} \sum_{i=1}^{n} x_i y_i - \bar{x} \cdot \bar{y},$$

per cui calcoliamo le medie parziali come

$$\bar{x}^{(j)} = \frac{1}{n_j} \sum_{i=1}^{n_j} x_i^{(j)}, \quad \bar{y}^{(j)} = \frac{1}{n_j} \sum_{i=1}^{n_j} y_i^{(j)}$$

e lo stesso con i prodotti

$$m_{xy}^{(j)} = \frac{1}{n_j} \sum_{i=1}^{n_j} x_i^{(j)} y_i^{(j)},$$

che dopo si aggrega con

$$\mathrm{Cov}(x, y) = \frac{1}{k} \sum_{j=1}^{k} m_{xy}^{(j)}$$

È anche necessario calcolare $\mathbb{V}\left[X\right]$ in parallelo, ad esempio si può usare

$$\mathbb{V}[X] = \mathbb{E}[\mathbb{V}[X|Z]] + \mathbb{V}[\mathbb{E}[X|Z]]$$

$$\mathsf{var}[x] = \frac{1}{n}\left(n_j \sum_{j=1}^{k} \mathsf{var}(x^{(j)}) + k n_j \left[\mathsf{var}\left([\bar{x}^{(1)}, \ldots, \bar{x}^{(k)}]\right)\right]\right)$$

Figura 52: result

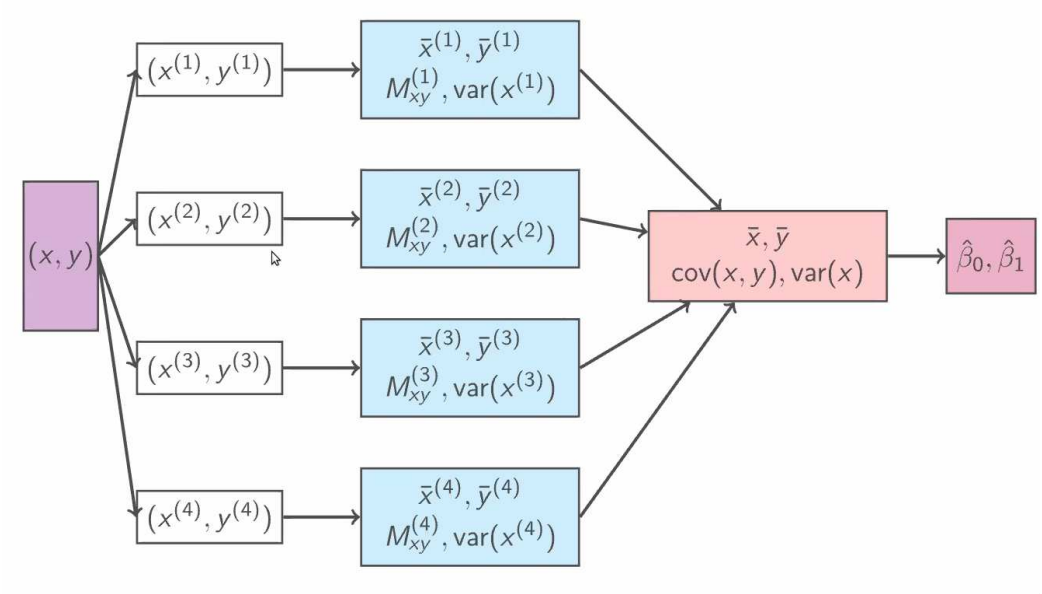


Figura 53: Implementazione in map-reduce per il modello lineare.

**Commenti**

- Utile per calcolare statistiche descrittive (media, varianza, max, min, ...)
- La divisione in chunk può essere effettuata in molti algoritmi, e.g. reti neurali usano stime locali per aggiornare quelle globali.
- A volte il meglio che possiamo fare è avere una stima diversa da quella globale, e ci si dovrà accontentare.

### Parallelizzazione a livello di algoritmo

Codice parallelizzabile, per cui ogni processore usa tutti i dati e calcola una parte diversa dell'algoritmo; successivamente, si aggregano i risultati e si ottiene la stima.

Un esempio naturale di parallelizzazione algoritmica è la convalida incrociata, assegnando un fold ad ogni processore:

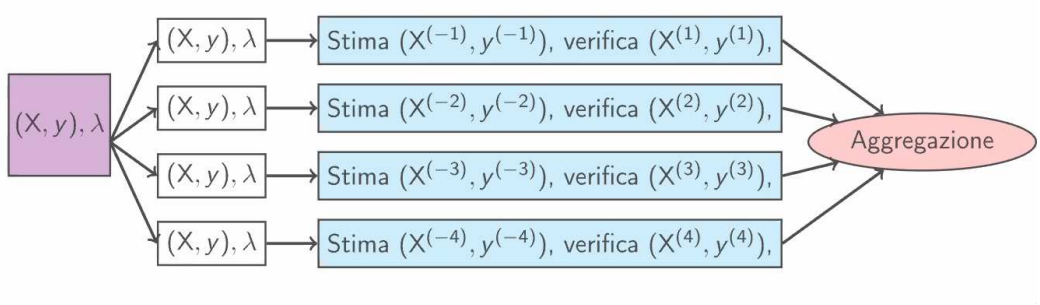


Figura 54: Convalida incrociata 4-fold in parallelo.

Problematico il fatto che ogni processore deve avere il dataset a disposizione, per cui non va bene se i dati sono massivi. Questo approccio è invece molto utile per operazioni non onerose in single core, e.g. ottimizzazione degli iperparametri per un modello non tanto pesante.

### Parallelizzazione a livello di operazioni

Le operazioni algebriche (prodotto matriciale, inversione, decomposizioni, ...) hanno un ruolo cruciale nell'analisi dei dati

$$\hat{\beta}_\lambda = (X^\mathsf{T} X + \lambda I)^{-1} X^\mathsf{T} y,$$

per cui si potrebbe provare a parallelizzare qualcuna di queste operazioni. R, Python e Matlab si appoggiano a librerie iper-ottimizzate già presenti di default nei sistemi operativi, come BLAS per le operazioni fondamentali e LAPACK per sistemi e decomposizioni. Ci sono anche alternative multithread, come

- OpenBLAS
- Intel MKL
- cuBLAS (usa la *GPU*)


Figura 55: recap

## 2.2 Take home

Uno tendenzialmente applicherebbe il più possibile tutti gli approcci contemporaneamente: sbagliatissimo, perché solo alcuni possono essere applicabili (o efficienti) per uno specifico problema.

Usare diversi approcci in contemporanea su più core di quanti sono disponibili potrebbe portare a risultati disastrosi: ad esempio, quando più sessioni parallele cercano di andare a loro volta in parallelo, il PC non riesce più a gestire bene il tutto.

Quale parallelizzazione conviene utilizzare? Come regola, si può utilizzare

1. Quello più efficiente: computazionalmente la più efficiente è parallelizzare i dati (haha deep learning go brrrr), ma purtroppo non si può fare sempre.
2. Quello più comodo: operazioni sono già parallelizzate in R e Python aggiornati.

Si possono comunque combinare gli approcci, se scelgo due core per i dati e quattro per l'algoritmo $\implies 2 \cdot 4$ core usati.

# Pregi e difetti dei vari modelli

### Modello lineare

**Pro**

- Familiarità ed efficacia del metodo.
- Semplicità di calcolo e formule di aggiornamento ricorsivo.
- Interpretabilità.

**Contro**

- Forzatura del modello per la classificazione, ma se interessa solo prevedere non è un problema.

### Analisi discriminante

**Pro**

- Appropriatezza del metodo (è stato costruito proprio per questo).
- Inclusione di informazione a priori.
- Semplicità di calcolo.
- Qualità e stabilità dei risultati rispetto all'arrivo di nuovi dati.
- Robustezza rispetto alle assunzioni, anche quando sono violate.

**Contro**

- Ipotesi restrittive nella QDA.
- Non ci sono metodi di selezione delle variabili o di calcolo dell'importanza.
- Elevato numero di parametri da stimare (! QDA).
- Non robustezza delle stime, ma si possono usare stimatori robusti.

### CART

**Pro**

- Piccoli alberi sono facili da interpretare.
- Applicabili a grandi dataset.
- Esplicative quantitative e qualitative.
- Ignorano facilmente variabili ridondanti.
- Dati mancanti modellati in modo naturale

**Contro**

- Grandi alberi sono difficili da interpretare.
- Se non vengono potati, sono portati al sovra-adattamento.
- Capacità predittiva spesso povera, soprattutto con frontiere decisionali diagonali.

### GLM/logit

**Pro**

- Bassa varianza.
- Funziona bene con frontiere decisionali diagonali.
- Facilmente interpretabili.

**Contro**

- Elevata distorsione.

### GAM

**Pro**

- Non soffrono della maledizione della dimensionalità.
- Sono più flessibili del modello lineare.
- Basso costo computazionale.

**Contro**

- Sono di difficile interpretazione.
- Non tengono conto delle interazioni tra variabili.

### MARS

**Pro**

- Più flessibili dei modelli lineari e tengono conto delle interazioni.
- Variabili esplicative qualitative e quantitative.
- Selezione automatica delle variabili.
- L'effetto di dati anomali è contenuto.
- Basso costo computazionale.
- Di solito hanno un buon compromesso varianza-distorsione.

**Contro**

- Non abbiamo una graduatoria della significatività delle variabili.
- Sono di difficile interpretazione.

### Projection Pursuit

**Pro**

- Tiene conto delle interazioni tra variabili, a differenza dei GAM.
- Approssimatore universale.
- Solitamente usa un basso numero di parametri, in quanto le rotazioni sono comprese in media tra 2 e 10.

**Contro**

- Perdiamo l'interpretabilità a meno che non riusciamo a dare un significato ad ogni rotazione.
- Se non si limita il numero di rotazioni si cade in sovra-adattamento.

### Bagging

**Pro**

- Il bootstrap riduce la varianza, rispetto ad alberi normali.
- Stesse caratteristiche degli alberi (NA's, qualitative, ...)
- L'out-of-bag si può utilizzare come insieme di verifica, senza ricorrere a stima-verifica o cv.
- Può sfruttare il calcolo in parallelo per la crescita degli alberi.

**Contro**

- Perdiamo l'interpretabilità degli alberi.
- Funziona male con classi rare.
- Non riduce la varianza se le variabili esplicative sono correlate.
- Non abbiamo una graduatoria della significatività delle variabili e nemmeno dell'importanza, a meno che non si proceda come nelle random forest.

### Boosting

**Pro**

- Più interpretabile del bagging e della random forest, poiché se si usano gli stumps è un modello additivo.
- Gestisce facilmente variabili qualitative.
- Migliora ad ogni iterazione dove nelle precedenti è stato carente: produce delle previsioni accurate.
- Tendenzialmente funziona bene anche con classi rare.

**Contro**

- Difficile da interpretare comunque.
- Si può sovradattare ai dati se c'è molto rumore o se i classificatori di base fanno schifo/sono poco distorti.
- Non c'è una graduatoria dell'importanza delle variabili.

### RandomForest

**Pro**

- Decorrela gli alberi più del bagging: con molte esplicative, queste sono di solito correlate.
- Non si sovradatta ai dati per costruzione.
- Riduce la varianza, rispetto ad alberi normali.
- Graduatoria dell'importanza delle variabili (ma non effetto).

**Contro**

- Perdiamo l'interpretabilità classica degli alberi.
- Non abbiamo una graduatoria della significatività delle variabili.

# Riferimenti bibliografici

Azzalini, A. e Scarpa, B. (2012). *Data Analysis and Data Mining: An Introduction*. Oxford University Press.

Breiman, L. (2001). «Random Forests». In: *Machine learning* 45.1, pp. 5–32.

Breiman, L. et al. (1984). *Classification and Regression Trees*. 1 edizione. Boca Raton: Chapman and Hall/CRC.

Ceron, A., Curini, L. e Iacus, S. M. (2013). *Social Media e Sentiment Analysis: L'evoluzione dei fenomeni sociali attraverso la Rete*. Springer.

Ceron, A., Curini, L. e Iacus, S. M. (2016). «iSA: A fast, scalable and accurate algorithm for sentiment analysis of social media content». In: *Information sciences* 367-368, pp. 105–124.

Cox, D. R. (1990). «Role of Models in Statistical Analysis». In: *Statistical science* 5.2, pp. 169–174.

Cox, D. R. (1997). «The Current Position of Statistics: A Personal View». In: *International statistical review* 65.3, pp. 261–276.

Dunson, D. B. (2018). «Statistics in the big data era: Failures of the machine». In: *Statistics & probability letters*. The Role of Statistics in the Era of Big Data 136, pp. 4–9.

Efron, B. et al. (2004). «Least angle regression». In: *The annals of statistics* 32.2, pp. 407–499.

Hastie, T., Tibshirani, R. e Friedman, J. (2013). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2 edizione. New York, NY: Springer Nature.

James, G. et al. (2013). *An Introduction to Statistical Learning: With Applications in R*. 2013 edizione. New York: Springer Verlag.

Meng, X.-L. (2018). «Statistical paradises and paradoxes in big data (I): Law of large populations, big data paradox, and the 2016 US presidential election». In: *The annals of applied statistics* 12.2, pp. 685–726.

Senn, S. (2003). *Dicing with Death: Chance, Risk And Health*. 1 edition. New York: Cambridge University Press.

Tibshirani, R. (1996). «Regression Shrinkage and Selection via the Lasso». In: *Journal of the royal statistical society. series b (methodological)* 58.1, pp. 267–288.

van Wieringen, W. N. (2020). «Lecture notes on ridge regression». In: *Arxiv:1509.09169 [stat]*.

Zou, H. e Hastie, T. (2005). «Regularization and variable selection via the elastic net». In: *Journal of the royal statistical society: series b (statistical methodology)* 67.2, pp. 301–320.