

Foundation of Machine Learning

Stefano Di Lena

2025

Indice

1	Statistical Learning	1
1.1	Trade-Off tra Accuratezza delle Predizioni e Interpretabilità del Modello	3
1.2	Regression Problems vs Classification Problems	5
1.3	Misurare la qualità di un Fit	5
1.4	Bias-Variance Trade-Off	7
1.5	Classification Setting	8
1.5.1	Bayes Classifier	9
1.5.2	K-Nearest Neighbors	9
2	Linear Regression	12
2.1	Simple Linear Regression	12
2.1.1	Stimare i coefficienti	12
2.1.2	Valutazione della Precisione del Modello	14
2.2	Multiple Linear Regression	15
2.2.1	Stimare i coefficienti	16
2.3	Qualitative Predictors	18
2.3.1	Predittori con solo Due Livelli	19
2.3.2	Predittori con Più di Due Livelli	20
2.4	Estensione del Modello Lineare	21
2.4.1	Rimuovere l'Additive Assumption	21
2.4.2	Rimuovere la Linear Assumption	23
3	Classification	24
3.1	Logistic Regression	25
3.1.1	Logistic Model	25
3.1.2	Stimare i coefficienti di regressione	26
3.1.3	Making Prediction	26
3.1.4	Multiple Logistic Regression	27
3.1.5	Multinomial Logistic Regression	28
3.2	Generative Models per la Classification	28
3.2.1	Linear Discriminant Analysis per $p = 1$	29
3.2.2	Linear Discriminant Analysis per $p > 1$	31
3.2.3	Naive Bayes	34
4	Resampling Methods	36
4.1	Cross-Validation	36
4.1.1	Validation Set	36
4.1.2	Leave-One-Out Cross-Validation	37
4.1.3	K-Fold Cross-Validation	38
4.1.4	Cross-Validation nei problemi di Classificazione	39
4.2	Bootstrap	40

5	Linear Model Selection and Regularization	41
5.1	Subset Selection Methods	41
5.1.1	Best Subset Selection	41
5.1.2	Stepwise Selection	42
5.1.3	Scegliere il Modello Ottimale	44
5.2	Shrinkage Methods	46
5.2.1	Ridge Regression	46
5.2.2	The Lasso	48
5.2.3	Selezionare i Parametri di Tuning	51
5.3	Dimension Reduction Methods	52
5.3.1	Principal Components Regression	52
5.3.2	Partial Least Squares	55
6	Tree-Based Methods	56
6.1	Regression Trees	56
6.2	Classification Trees	61
6.3	Vantaggi e Svantaggi degli Alberi	63
6.4	Bagging	64
6.5	Random Forest	66
6.6	Boosting	67
7	Support Vector Machines	69
7.1	Maximal Margin Classifier	69
7.2	Support Vector Classifiers	71
7.3	Support Vector Machines	73
7.4	SVMs con più di Due Classi	76
7.4.1	One-Versus-One (OVO) Classification	76
7.4.2	One-Versus-All (OVA) Classification	77
7.5	Relazioni con la Logistic Regression	77
8	Deep Learning	78
8.1	Single Layer Neural Networks	78
8.2	Multilayer Neural Networks	79
8.3	Convolutional Neural Networks	82
8.3.1	Convolution Layers	83
8.3.2	Pooling Layers	84
8.3.3	Architettura di una CNN	84
8.3.4	Risultati Utilizzando un Pretrained Classifier	85
8.4	Fittare una Neural Network	86
8.4.1	Backpropagation	87
8.4.2	Dropout Learning	87
8.5	Document Classification	88
8.6	Recurrent Neural Networks	89

9	Unsupervised Learning	91
9.1	Principal Component Analysis	91
9.1.1	Un'altra Interpretazione delle Componenti Principali	93
9.1.2	La Proporzioe della Varianza Spiegata	94
9.2	Clustering Methods	95
9.2.1	K-Means Clustering	95
9.2.2	Hierarchical Clustering	98

Elenco delle tabelle

2.1	Tabella di regressione per il data set Advertising	14
2.2	RSE, R^2 ed F-statistic per la regressione lineare del numero di unità vedute in pubblicità TV per il data set Advertising	15
2.3	Coefficienti di regressione multipli stimati quando i budget pubblicitari di TV, radio e newspaper vengono usati per predire le vendite del prodotto usando il data set Advertising	16
2.4	Matrice di correlazione tra TV, radio, newspaper e sales per il data set Advertising	17
2.5	Altri RSE, R^2 ed F-statistic per la regressione del numero di unità vedute in pubblicità TV per il data set Advertising	17
2.6	Stima dei coefficienti associati con la regressione di balance in own nel data set Credit	19
2.7	Stima dei coefficienti associati con la regressione di balance in region nel data set Credit	20
2.8	Coefficienti stimati associati al modello dell'equazione (2.22).	22
2.9	Stima dei coefficienti associati alla regressione di mpg in horsepower ed horsepower ² per il data set Auto	23
3.1	Coefficienti stimati che prevedono la probabilità di default usando il balance per il data set Default	26
3.2	Coefficienti stimati che prevedono la probabilità di default usando lo student status per il data set Default	27
3.3	Coefficienti stimati con la regressione logistica per il data set Default	27
3.4	Una confusion matrix per comparare le predizione LDA ai true default statuses per 10000 osservazioni di training nel data set Default (gli elementi diagonali della matrice rappresentano gli individui con default status predictato correttamente, mentre quelli off-diagonals rappresentano gli elementi misclassificati).	32
5.1	I primi quattro modelli selezionati con il best subset e la forward stepwise per il data set Credit	43
8.1	Test error rate per le neural networks con due forme di regolarizzazione e per la multinomial logistic regression e la linear discriminant analysis sul data set MNIST	81
9.1	I loading vectors ϕ_1 e ϕ_2 per il dataset USArrest	93

9.2	Sommario dei quattro tipi più comunemente usati di <i>linkage</i> nel cluster gerarchico.	102
-----	---	-----

Elenco delle figure

1.1	Il dataset Advertising	1
1.2	il data set Income	2
1.3	Trade-Off tra interpretabilità e flessibilità.	4
1.4	Clustering data set.	4
1.5	Sinistra: Dati simulati da f e tre stime per essa. Destra: trainig MSE vs test MSE.	5
1.6	Sinistra: Dati simulati da f e tre stime per essa. Destra: trainig MSE vs test MSE.	6
1.7	Sinistra: Dati simulati da f e tre stime per essa. Destra: trainig MSE vs test MSE.	7
1.8	Bias, Varianza, MSE per tre diversi data set.	8
1.9	Data set consistente in 100 osservazioni appartenenti a due distinti gruppi (blu, arancione). La regione tratteggiata in viola indica il decision boundary del classificatore Bayes.	9
1.10	KNN, usando $K=3$	10
1.11	KNN, usando $K=10$	10
1.12	Decision boundary a confronto per diversi valori di K	11
1.13	Confronto tra il test error rate ed il training error rate per il KNN.	11
2.1	Least squares fit per la regressione di sales in TV per l' Advertising data set.	13
2.2	Least square fit per il data set toy con $p = 2$ predittori.	16
2.3	Scatterplots del data set Credit contenente informazioni per un numero di potenziali customers.	18
2.4	Per il data set Advertising , linear regression fit di sales usando TV e radio come predittori.	21
2.5	Relazione tra balance ed income per il data set Credit	22
2.6	Relazione tra mpg ed horsepower per un certo numero di macchine del data set Auto	23
3.1	Default data set.	24
3.2	Classification usando il data set Default	25
3.3	Confounding nel data set Default	28
3.4	Due funzioni di densità <i>normali</i> e divisione di 20 osservazioni nelle due classi.	30
3.5	Esempi di distribuzione Gaussiana multivariata con $p = 2$	31
3.6	Tre classe Gaussiane sono mostrate con un class-specific mean vector ed una matrice covarianza comune.	32
3.7	Per il data set Default sono mostrati alcune Error Rate in funzione del Threshold.	33
3.8	Curva ROC per il classificatore LDA sul data set Default . La linea tratteggiata rappresenta il "no information" classifier.	34

3.9	Esempio <code>toy</code>	35
4.1	Esempio schematico del validation set approach. Un set di n osservazioni viene diviso casualmente in training set (in blu, contenente le osservazioni 7, 22, 13, ed altre...) ed un validation set (in beige, contenente l'osservazione 91, ed altre...).	36
4.2	Validation set approach sul data set <code>Auto</code>	37
4.3	Esempio schematico del LOOCV. Un set di n data points viene diviso nel training set (in blu, contenente tutte le osservazioni tranne 1) ed il validation set (in beige, contenente la restante osservazione).	38
4.4	LOOCV sul data set <code>Auto</code>	38
4.5	Esempio schematico del 5-fold.	39
4.6	Stima e valore vero del test MSE.	39
4.7	Esempio schematico del bootstrap approach su un piccolo sample, contenente $n=3$ osservazioni.	40
5.1	RSS ed R^2 in funzione del numero di predittori per il data set <code>Credit</code>	42
5.2	C_p , BIC ed adjusted R^2 per il migliore modello di ogni dimensione prodotto dalla best subset selection sul data set <code>Credit</code>	44
5.3	BIC, validation set error, cross-validation error per il data set <code>Credit</code>	46
5.4	Stime dei coefficienti utilizzando la ridge regression per il data set <code>Credit</code>	47
5.5	Squared bias (nero), varianza (verde) e test MSE (rosa) per la ridge regression. La linea orizzontale tratteggiata indica il minimo MSE possibile, la X in rosa indica il modello di ridge regression che ha MSE minore.	48
5.6	Stime dei coefficienti utilizzando il lasso per il data set <code>Credit</code>	49
5.7	Contorni degli errori e delle funzioni vincolari per il lasso (a sinistra) e la ridge regression (a destra). Le aree blu sono le regioni di vincolo, considerando $p = 2$, $(\beta_1 + \beta_2 \leq s$ e $\beta_1^2 + \beta_2^2 \leq s)$, mentre le ellissi rosse sono i contorni del RSS.	49
5.8	Squared bias (nero), varianza (verde) e test MSE (rosa) per il lasso. La linea orizzontale tratteggiata indica il minimo MSE possibile, la X in rosa indica il modello di lasso che ha MSE minore.	50
5.9	Squared bias (nero), varianza (verde) e test MSE (rosa) per il lasso. La linea orizzontale tratteggiata indica il minimo MSE possibile, la X in rosa indica il modello di lasso che ha MSE minore.	50
5.10	Scelta di λ utilizzando la LOOCV sulla ridge regression per il data set <code>Credit</code>	51
5.11	10-folds cross-validation applicata sul lasso utilizzando i dati simulati in Figura 5.9.	51
5.12	La <i>population size</i> (<code>pop</code>) e l' <i>ad spending</i> (<code>ad</code>) per 100 differenti città è rappresentato con i pallini rosa. La linea verde continua indica la prima componente principale e la linea blu tratteggiata la seconda componente principale.	52

5.13	Un subset dell'advertising data. Il punto in blu è $(\overline{pop}, \overline{ad})$ ovvero il pop medio e l'ad medio rispetto a tutti i valori.	53
5.14	Plots dei first principal component scores z_{i1} in relazione a pop e ad. Notiamo una forte relazione.	53
5.15	Plots dei first principal component scores z_{i2} in relazione a pop e ad. Notiamo una relazione debole.	54
5.16	PCR applicata ai dati simulati della Figura 5.8 (sinistra) e 5.9 (destra).	54
5.17	PCR applicata ad data set Credit	55
6.1	Un regression tree per predire il log salary di un baseball player per il data set Hitters	56
6.2	La regione di partizione dell'albero per il data set Hitters con il regression tree illustrato in Figura 5.1.	57
6.3	In alto a sinistra: una partizione bi-dimensionale di un feature space che non può uscire dal recursive binary splitting. In alto a destra: l'output di un recursive binary splitting su un esempio bi-dimensionale. In basso a sinistra: un albero che corrisponde alla partizione del pannello in alto a destra. In basso a destra: Un <i>perspective plot</i> del <i>prediction surface</i> corrispondente a quell'albero.	58
6.4	Albero di regressione <i>unpruned</i> risultante dalla divisione <i>top-down greedy</i> sui dati di training per il data set Hitters	60
6.5	Trainig, Cross-Validation e Test MSE sono mostrati in funzione del numero di nodi terminali nell'albero ridotto per il data set Hitters . Sono mostrate anche le standard error bars attorno all'errore stimato. Il minimo errore di cross-validazione si ha in un albero di dimensione tre.	60
6.6	Per il data set Heart . In alto: l' <i>unpruned tree</i> . In basso a sinistra: gli errori di cross-validation, training e test in funzione delle differenti misure dell'albero ridotto. In basso a destra: l'albero ridotto basandosi sul minimo errore di cross-validazione.	62
6.7	Sopra: un esempio di classificazione bi-dimensionale, nel quale il decision boundary reale è lineare, indicato dalle regioni colorate. Un approccio lineare classico (sinistra) outperforms il decision tree che applica split paralleli agli assi (destra). Sotto: abbiamo un decision boundary non lineare. Un modello lineare (sinistra) non riesce quindi a catturare il vero decision boundary, mentre un decision tree (destra) riesce.	63
6.8	Risultati del bagging e del random forest per il data set Heart . I test error rate sono in funzione di B.	64
6.9	Un plot della variable importance per il data set Heart . La variable importance è calcolata usando la diminuzione media del Gini index, espressa relativamente al massimo.	65
6.10	Risultati di un random forest per le 15-class gene expression data set con $p = 500$ predittori. Ogni linea colorata rappresenta un valore differente di m	66

6.11	Risultati dell'applicazione del boosting e random forest sul 15-class gene expression data set in maniera tale da predire <i>cancer vs normal</i> . Il test error è mostrato in funzione del numero di alberi. Per i due modelli di boosting $\lambda = 0.01$	68
7.1	Esempi di separating hyperplane.	69
7.2	Esempio di maximal margin classifier.	70
7.3	Due classi che non possono essere separate da un hyperplane. . .	71
7.4	Sinistra: due classi di osservazioni sono mostrate in blu e viola. Destra: aggiungendo una nuova osservazione il maximal margin hyperplane cambia da quello tratteggiato a quello continuo. . . .	71
7.5	Esempi di support vector classifier.	72
7.6	Esempi di support vector classifier usando quattro differenti valori del parametro di tuning C.	73
7.7	Esempio support vector classifier con un non-linear boundary. . .	74
7.8	Sinistra: un SVM con un kernel polinomiale di grado 3 è applicato a dati non-lineari. Destra: un SVM con un kernel radiale è applicato.	75
7.9	Curve ROC per il training set del data set Heart	76
7.10	Curve ROC per il test set del data set Heart	76
7.11	Confronto tra la funzione di Loss del SVM e della Logistic Regression.	77
8.1	Neural network, con un singolo hidden layer, per modellare una risposta quantitativa utilizzando quattro predittori.	78
8.2	Funzioni di attivazione.	79
8.3	Esempi di digits scritti a mano presi dal data set MNIST	79
8.4	Neural network con due hidden layers ed uscite multiple.	80
8.5	75 immagini prese dal database CIFAR100 , contenente una collezione di immagini naturali della vita quotidiana divisa in 100 classi.	82
8.6	Esempio di CNN applicata ad un'immagine cartone rappresentante una tigre.	82
8.7	Esempio di convoluzione applicato all'immagine di una tigre. L'immagine convoluta in alto sottolinea le linee verticali della tigre, quella in basso enfatizza le linee orizzontali della tigre.	84
8.8	Architettura di una deep CNN per il CIFAR100 classification task.	84
8.9	Classificazione di sei foto utilizzando la CNN resnet50 trainata sul <i>corpus imagenet</i>	85
8.10	Esempio di funzione non convessa di una singola variabile θ ; ci sono due soluzioni: una è il minimo locale ($\theta = -0.46$) e l'altro quello globale ($\theta = 1.02$).	86
8.11	Dropout Learning	88
8.12	Precisione del lasso e della neural network a due classi sui dati IMDb . Entrambi tendono ad overfittare,	89
8.13	Schema di una semplice RNN.	89
9.1	Le prime due componenti per i dati USArrest	92
9.2	90 osservazioni simulate in 3 dimensioni.	93

9.3	Sinistra: uno <i>scree plot</i> raffigurante la PVE delle quattro componenti principali del USArrests dataset. Destra: la proporzione cumulativa di varianza spiegata dalle quattro componenti principali nel USArrests dataset.	94
9.4	Confronto tra i grafici delle componenti principali per il dataset USArrests con variabili scalate e non.	95
9.5	Data set simulato contenente 150 osservazioni in uno spazio bi-dimensionale. Ogni pannello mostra i risultati del K-means clustering, per diversi valori di K. I colori indicano i cluster a cui ogni osservazione è stata assegnata (questi sono casuali e potrebbero cambiare da un fit ad un altro).	96
9.6	Progresso dell'algoritmo K-means (con K=3).	97
9.7	Sei diverse volte in cui è stato runnato l'algoritmo per K=3. La migliore soluzione è stata ottenuta quattro volte, con un objective value segnato in rosso (di 235.8).	98
9.8	45 osservazioni generate in uno spazio bi-dimensionale. Nella realtà ci sono 3 classi distinti (mostrate dai diversi colori) ma no tratteremo questo dataset come sconosciuto.	99
9.9	Dendogrammi ottenuti con un completo linkage utilizzando la distanza Euclidiana (Sinistra), linkage tagliato ad un altezza pari a 9 (Centro), linkage tagliato ad un altezza pari a 5 (Destra).	99
9.10	Un illustrazione che aiuta a capire come interpretare un dendogramma con 9 osservazioni in uno spazio bi-dimensionale.	100
9.11	Illustrazione dei primi passi dell'algoritmo di clustering gerarchico per i dati della figura 9.10.	101
9.12	Average, Complete e Single Linkage applicato ad un dataset di esempio. Average e Complete linkage ottengono dei cluster più bilanciati.	102
9.13	Sono mostrate 3 osservazioni con misurazioni su 20 diverse variabili. Osservazioni 1 e 3 hanno valori simili per ogni variabile, quindi hanno una piccola distanza Euclidiana. Però non sono correlati, quindi hanno una maggiore correlation-based distance. Al contrario, le osservazioni 1 e 2 hanno una grande distanza Euclidiana tra di loro ma sono fortemente correlati.	103

1 Statistical Learning

Supponiamo di essere stati assunti da un cliente per investigare sulla correlazione tra la pubblicità e le vendite di un particolare prodotto. Il data set **Advertising** consiste nei **sales** di quel prodotto in 200 mercati differenti, in corrispondenza al budget della pubblicità del prodotto in ogni mercato per tre diversi media: **TV**, **radio** e **newspaper**.

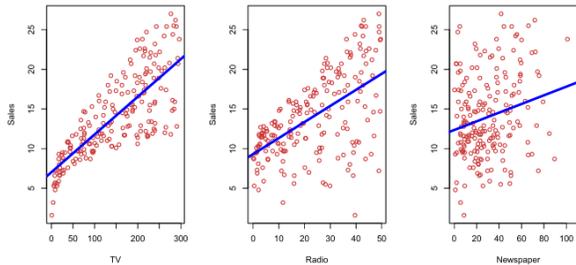


Figura 1.1: Il dataset **Advertising**.

Per il nostro cliente non è possibile incrementare direttamente le vendite del prodotto, quello che può fare è gestire il budget sulla pubblicità tra i 3 differenti media. Noi possiamo aiutarlo notando se effettivamente esiste una relazione tra pubblicità e vendite.

Nei problemi di statistical learning abbiamo gli ingressi che possono essere chiamati in diversi modi (predittori, variabili indipendenti, features) e le variabili di uscita, spesso chiamate variabili dipendenti o risposta e tipicamente indicata con Y . Solitamente osserviamo risposte quantitative Y avendo p predittori: X_1, X_2, \dots, X_p . Assumiamo che esiste una relazione tra Y e $X = (X_1, X_2, \dots, X_p)$ che scriviamo generalmente:

$$Y = f(X) + \epsilon \quad (1.1)$$

Dove f è una funzione di X , fissata ma sconosciuta, ed ϵ è un termine di errore che è indipendente da X ed ha media zero. Nella formula (1.1) quindi f rappresenta l'informazione sistematica che X provvede riguardo ad Y .

Se consideriamo l'immagine di sinistra della Figura 1.2, notiamo un grafico di **income** e **years of education** per 30 individui nel data set **Income**.

Dobbiamo stimare la f basandoci su i punti osservati. La f in realtà è nota, dato che **Income** è un data set simulato, ed è rappresentata dalla curva in blu dell'immagine a destra. Le linee verticali invece rappresentano ϵ .

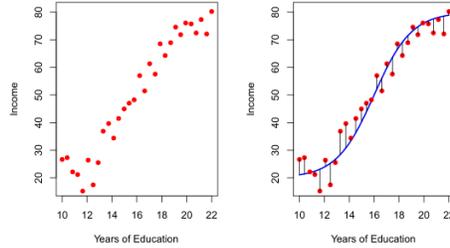


Figura 1.2: il data set **Income**.

Per statistical learning quindi intendiamo i vari approcci che esistono per stimare la f . Vogliamo stimare f per due motivi solitamente:

- **Prediction:** quando un set di input X è disponibile ma l'output Y non è facilmente ricavabile. Dato che il termine di errore ha media zero possiamo prevedere Y usando

$$\hat{Y} = \hat{f}(X) \quad (1.2)$$

dove \hat{f} rappresenta la nostra stima di f .

L'accuratezza di \hat{Y} come predittore di Y dipende da due quantità: *errore riducibile* e *irriducibile*. Generalmente \hat{f} non è una stima perfetta di f e questa in-accuratezza introduce un errore *riducibile*. Tuttavia non importa quanto bene stimiamo la f perché avremmo comunque sempre un errore *irriducibile* (ϵ). Questo consiste in variabili non misurate o da variazioni casuali non misurabili.

Se assumiamo che \hat{f} ed X siano fissate, l'unica variazione dipende da ϵ

$$E(Y - \hat{Y})^2 = E[f(X) + \epsilon - \hat{f}(X)]^2 = [f(X) - \hat{f}(X)]^2 + Var(\epsilon) \quad (1.3)$$

dove $E(Y - \hat{Y})^2$ rappresenta la media (o valore atteso) al quadrato della differenza tra il valore predetto e quello reale, la $Var(\epsilon)$ rappresenta l'errore *irriducibile* mentre il fattore $[f(X) - \hat{f}(X)]^2$ rappresenta un errore *riducibile*.

- **Inference:** quando vogliamo capire la relazione esistente tra Y ed X , quindi vogliamo stimare f ma non necessariamente dobbiamo fare predizioni per la Y .

Vogliamo scoprire in questo caso quali sono i predittori importanti e utili, quale relazione esiste tra i predittori e la Y e se questa può essere rappresentata da un'equazione lineare o è più complicata.

Assumeremo sempre di aver osservato un set di n differenti punti (ad esempio $n = 30$). Queste osservazioni vengono chiamate *training data* perché possono

essere usate per addestrare il nostro metodo di stima delle f . Un training data consiste in $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ con $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})^T$.

Il nostro obiettivo è trovare una funzione tale che $Y \approx \hat{f}(X)$ per ogni osservazione (X, Y) .

Parametric Methods

1. Assumiamo la forma di f , ad esempio possiamo pensare che essa sia lineare:

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p \quad (1.4)$$

2. Una volta scelto il modello, abbiamo bisogno di una procedura per fittare o addestrare il modello. Nell'esempio del modello lineare ciò si traduce nello stimare i parametri $\beta_0, \beta_1, \dots, \beta_p$, in modo tale che:

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p \quad (1.5)$$

Un potenziale svantaggio del approccio parametrico consiste nel fatto che il modello scelto da noi potrebbe non matchare la vera forma di f . Se il modello scelto da noi dista troppo dalla f reale, vuol dire che il modello stimato è scarso. Possiamo provare a risolvere questo problema scegliendo dei modelli più flessibili, ma fittare un modello più flessibile si traduce nello stimare un numero maggiore di parametri. Inoltre, l'uso di modelli complessi potrebbe portare ad un *overfitting* dei dati (in cui il modello segue anche gli errori)

Non-Parametric Methods

Non effettuiamo assunzioni esplicite della funzione f ma essa si cerca in maniera tale da essere più vicina ai data points senza essere però troppo rigida o ondulata. Questi modelli però soffrono di un peggiore svantaggio, ovvero, un numero elevato di osservazioni è necessario in modo tale da ottenere una stima accurata di f .

1.1 Trade-Off tra Accuratezza delle Predizioni e Interpretabilità del Modello

Scegliamo dei modelli restrittivi e più interpretabili se siamo interessati in *inference*. In questi casi un modello lineare potrebbe essere un'ottima scelta.

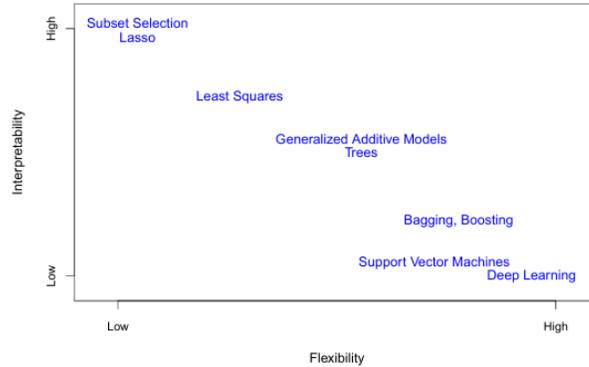


Figura 1.3: Trade-Off tra interpretabilità e flessibilità.

Generalmente quando aumenta la flessibilità di un metodo la sua interpretabilità decresce.

Cluster Analysis

L'obiettivo di quest'analisi è di accertarsi se le osservazioni appartengono a gruppi distinti.

In uno studio di mercato potremmo osservare varie caratteristiche per ogni consumatore come il guadagno della famiglia e le abitudini di shopping e determinare se esso ricade in differenti gruppi (quali potrebbero essere *spendaccione* o *taccagno*).

Un esempio di cluster è rappresentato dalla Figura 1.4, dove sono stati plot-tati 150 osservazioni con misurazioni in due variabili X_1 e X_2 . Ogni osservazione cade in uno di tre distinti gruppi (rappresentati con diversi colori e simboli).

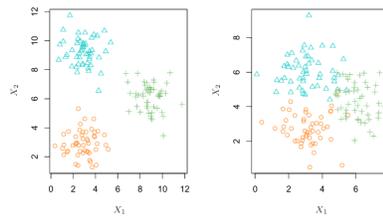


Figura 1.4: Clustering data set.

Nella pratica il nostro obiettivo è determinare a quale gruppo ogni osservazione appartiene.

Un metodo di clustering potrebbe non assegnare i punti sovrapposti nel corretto gruppo.

1.2 Regression Problems vs Classification Problems

Le variabili possono essere:

- *quantitative*, quando assumono un valore numerico (l'età o l'altezza di una persona, quando questa guadagna, il valore di una casa, etc.)
- *qualitative*, quando assumono valori in una o K differenti classi, o categorie, [lo stato di una persona (sposato, single), i brand di un prodotto acquistato (Nike, Adidas, Diadora), etc.].

Chiamiamo i problemi con una risposta quantitativa: *regression problems* e quelli con una risposta qualitativa *classification problems*.

1.3 Misurare la qualità di un Fit

Per valutare le performance di un metodo partendo da un data set fornito abbiamo bisogno di un modo per calcolare quanto bene le predizioni matchano i dati osservati. Nei problemi di regressione la misura usata comunemente è il *Mean Squared Error*:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}(x_i))^2 \quad (1.6)$$

L'MSE sarà piccolo se le risposte predette sono vicine a quelle reali, mentre sarà grande se differiscono sostanzialmente.

Generalmente siamo interessati alla precisione dei dati ottenuti applicando il metodo a data set mai visti prima e non a quelli di training.

Non siamo interessati quindi a sapere se $\hat{f}(x_i) \approx y_i$ ma se $\hat{f}(x_0) \approx y_0$, dove (x_0, y_0) è un'osservazione mai vista, quindi non usata nel training del metodo.

Se abbiamo un grande numero di osservazioni per il test, possiamo calcolare:

$$Ave \left(y_0 - \hat{f}(x_0) \right)^2 \quad (1.7)$$

la media al quadrato dell'errore di predizione per le osservazioni (x_0, y_0) . Vogliamo selezionare il modello che ha questa quantità minore.

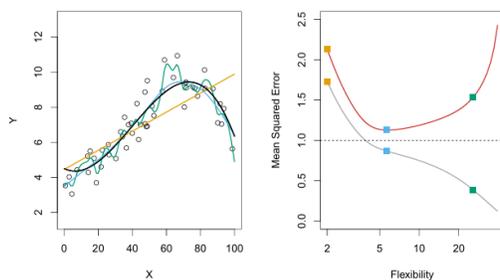


Figura 1.5: Sinistra: Dati simulati da f e tre stime per essa. Destra: training MSE vs test MSE.

Nella parte sinistra della Figura 1.5 abbiamo delle osservazioni generate da $Y = f(X) + \epsilon$ dove la vera f è la curva in nero. Le altre curve (arancione, blu e verde) illustrano tre possibili stime di f ottenute usando metodi incrementando il livello di flessibilità. La curva arancione usa la regressione lineare (relativamente inflessibile), la blu e la verde utilizzano delle *smoothing splines* con diversi livelli di smoothness. Man mano che il livello di flessibilità aumenta le curve fittano i dati osservati maggiormente. La curva più flessibile (verde) segue i dati molto bene ma confrontandola con la reale (nera) notiamo che non è un buon fit (perché troppo ondeggiante).

Nella parte destra della figura invece la curva grigia mostra il training MSE in funzione della flessibilità. I quadrati arancioni, blu e verde rappresentano l'MSE associato alla corrispondente curva della figura a sinistra.

La funzione reale è non-lineare, infatti l'arancione non stima bene f . La curva verde ha il MSE più basso.

Il test MSE è rappresentato nel grafico a destra in rosso, esso inizialmente scende all'aumentare della flessibilità, dopo un certo punto comincia a risalire. L'arancione e il verde hanno un valore di MSE alto entrambi quindi la curva in blu minimizza meglio il test MSE.

La linea tratteggiata orizzontale (ad 1.0) rappresenta l'errore irriducibile $Var(\epsilon)$, che corrisponde al minore test MSE ottenibile tra tutti i metodi.

Quando la flessibilità di un modello aumenta il training MSE tende a diminuire ma non è detto che diminuisca anche il test MSE. Se un metodo ha un valore basso di training MSE ed uno alto di test MSE allora stiamo overfittando i dati. Solitamente ci aspettiamo un training MSE minore di quello del test, quindi parliamo di overfitting quando modelli meno flessibili potrebbero produrre valori minori di test MSE.

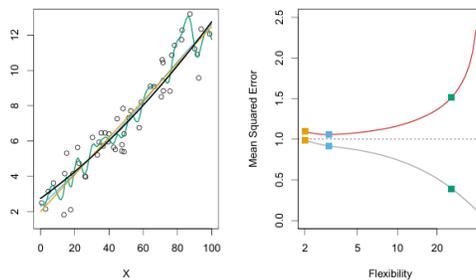


Figura 1.6: Sinistra: Dati simulati da f e tre stime per essa. Destra: training MSE vs test MSE.

In Figura 1.6 notiamo un altro esempio in cui la f è pressoché lineare. Nel grafico a destra notiamo che il training MSE è monotono decrescente man mano che aumenta la flessibilità, mentre il test MSE ha una forma simile ad una U. L'arancione in questo caso fitta molto meglio rispetto al verde.

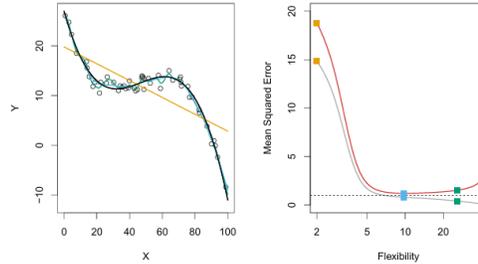


Figura 1.7: Sinistra: Dati simulati da f e tre stime per essa. Destra: training MSE vs test MSE.

In quest'ultimo esempio di Figura 1.7 notiamo come la f è fortemente non-lineare. Sia il training MSE che il test MSE decrescono rapidamente e solo verso la fine il test MSE torna a crescere lentamente.

1.4 Bias-Variance Trade-Off

Possiamo scrivere il test MSE, per un dato valore x_0 , come somma di tre quantità fondamentali:

$$E(y_0 - \hat{f}(x_0))^2 = \text{Var}(\hat{f}(x_0)) + [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\epsilon) \quad (1.8)$$

Il test MSE previsto potrebbe essere calcolato facendo la media di $E(y_0 - \hat{f}(x_0))^2$ rispetto a tutti i valori possibili di x_0 .

Per minimizzare il valore attesi di errore nel test dobbiamo scegliere un metodo che assuma *bassa varianza e bias basso*.

Per *varianza* si intende di quanto cambia \hat{f} se la stimiamo usando un training data set differente. I metodi più flessibili hanno una maggiore varianza. Per questi metodi quindi piccoli cambiamenti nel data set di training possono portare a grandi variazioni di \hat{f} .

Per *bias* si intende invece l'errore che è introdotto approssimando i problemi reali con modelli semplificati. Aumentando la flessibilità di un metodo quindi il bias diminuisce (in maniera più veloce rispetto all'aumento di varianza).

Il cambiamento di queste due quantità determina quindi se il test MSE incrementa o decresce.

In Figura 1.8 sono mostrati tre grafici in cui la curva blu rappresenta il bias al quadrato, l'arancione rappresenta la varianza, la linea orizzontale tratteggiata rappresenta la $\text{Var}(\epsilon)$, la curva rossa rappresenta il test MSE (la somma di queste 3 quantità) e infine la linea verticale tratteggiata rappresenta il livello di flessibilità corrispondente al più basso test MSE.

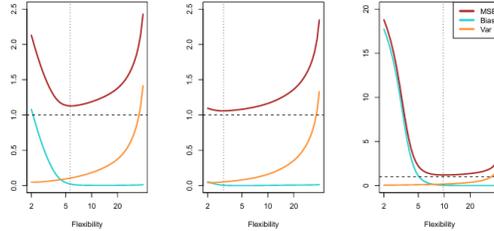


Figura 1.8: Bias, Varianza, MSE per tre diversi data set.

Notiamo come nel grafico a sinistra il bias decresce rapidamente, portando ad una decrescita anche del test MSE aspettato. Nel grafico al centro la f reale è quasi lineare, quindi c'è solo una piccola decrescita del bias ed il test MSE decresce pochissimo e successivamente cresce rapidamente ad un ulteriore aumento della flessibilità. Infine, nel grafico a destra all'aumentare della flessibilità abbiamo una decrescita del bias e del test MSE, perché la f è non-lineare, e l'aumento di varianza all'aumentare della flessibilità è basso.

Come già detto un buon modello dovrebbe avere basso bias e bassa varianza, ma questa cosa è difficile da fare per questo bisogna trovare un compromesso (trade-off).

E facilmente ottenibile un metodo con basso bias e varianza elevata (disegnando una curva che passa attraverso ogni osservazione di training) o un metodo con bassa varianza e bias elevato (fittando una linea orizzontale).

1.5 Classification Setting

Supponiamo di voler stimare f sulla base di osservazioni di training:

$\{(x_1, y_1), \dots, (x_n, y_n)\}$, in cui y_1, \dots, y_n sono qualitative.

L'approccio più comune per quantificare la precisione della nostra stima \hat{f} è il training *error rate* (la proporzione degli errori che vengono fatti applicando la stima alle osservazioni di training):

$$\frac{1}{n} \sum_{i=1}^n I(y_i \neq \hat{y}_i) \quad (1.9)$$

con \hat{y}_i la classe predetta per l' i -esima osservazione usando \hat{f} . $I(y_i \neq \hat{y}_i)$ invece rappresenta una variabile indicativa che è uguale ad 1 se $y_i \neq \hat{y}_i$ e 0 se $y_i = \hat{y}_i$, quindi se è uguale a zero l' i -esima osservazione è stata classificata correttamente, altrimenti è stata misclassificata.

Il test error rate associato ad un set di osservazioni della forma (x_0, y_0) è dato, invece, da:

$$Ave(I(y_0 \neq \hat{y}_0)) \quad (1.10)$$

Un buon classificatore è quello che ha il minore errore di test.

1.5.1 Bayes Classifier

Un semplice classificatore che assegna ogni osservazione alla classe più probabile, considerando i valori dei predittori dati.

Quindi, assegniamo un osservazione di test con un vettore di predittori x_0 alla classe j se la:

$$Pr(Y = j|X = x_0) \tag{1.11}$$

è la più grande.

In problemi aventi solo due classi, la classificazione di Bayes si traduce nell'assegnare Classe1 quando $Pr(Y = 1|X = x_0) > 0.5$ e Classe2 negli altri casi.

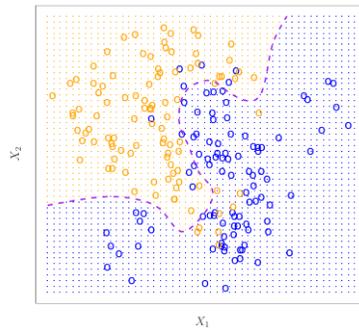


Figura 1.9: Data set consistente in 100 osservazioni appartenenti a due distinti gruppi (blu, arancione). La regione tratteggiata in viola indica il decision boundary del classificatore Bayes.

Nella Figura 1.9 i cerchi arancione e blu corrispondono alle osservazioni di training che appartengono a due classi differenti. Per ogni valore di X_1 ed X_2 esiste una differente probabilità che la risposta sia appartenente alla classe blu o a quella arancione.

La regione arancione rappresenta i punti per cui $Pr(Y = orange|X)$ è maggiore del 50%, mentre la regione blu indica i punti per cui la probabilità è minore de 50%.

Il classificatore Bayes produce il più basso test error rate, chiamato *Bayes error rate*:

$$1 - E(\max[Pr(Y = j|X)]) \tag{1.12}$$

Possiamo dire che il Bayes error rate è analogo all'errore irriducibile.

1.5.2 K-Nearest Neighbors

Quando non è possibile applicare il classificatore Bayes, perché non conosciamo la distribuzione condizionata di Y dato X , possiamo applicare il KNN (K-nearest neighbors).

Dato un numero intero positivo K ed un test di osservazioni x_0 , il classificatore KNN identifica i K punti nel training data che sono vicini ad x_0 (rappresentati da \mathcal{N}_0) e successivamente stima la probabilità condizionata:

$$Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j) \quad (1.13)$$

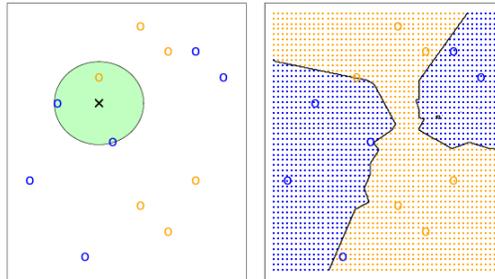


Figura 1.10: KNN, usando $K=3$.

Nel grafico a sinistra della Figura 1.10 è stato plottato un piccolo training set contenente sei osservazioni blu e sei arancioni. Supponendo di scegliere K pari a 3, il KNN identificherà tre osservazioni vicine alla croce (rappresentato con il cerchio in verde), in questo caso sono presenti due blu ed uno arancioni, quindi, segnerà che la croce appartiene alla classe blu. Nella parte a destra notiamo il decision boundary per $K=3$.

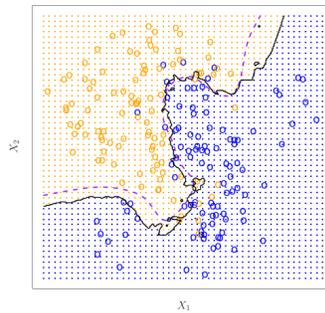


Figura 1.11: KNN, usando $K=10$.

In Figura 1.11 possiamo notare come il decision boundary ottenuto usando un KNN (con $K=10$) sia molto simile a quello ottenuto usando il classificatore Bayes. Anche i test error rate sono molto simili, usando il KNN si ha 0.1363 mentre usando il Bayes otteniamo 0.1304.

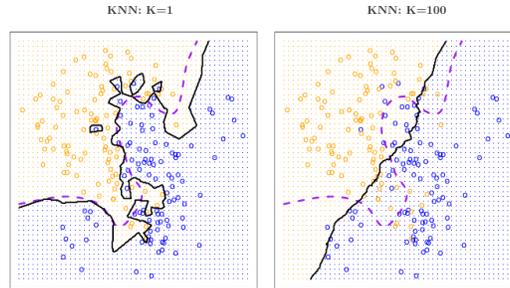


Figura 1.12: Decision boundary a confronto per diversi valori di K .

Possiamo notare come la scelta del K influenza molto il risultato del KNN.

In Figura 1.12 notiamo che per $K=1$ il decision boundary è troppo flessibile e non corrisponde con quello del Bayes, (corrisponde quindi ad un classificatore con basso bias ed alta varianza). All'aumentare di K il metodo diventa meno flessibile e produce decision boundary più lineari (corrispondenti a classificatori con bassa varianza e alto bias).

Quindi sia $K=1$ che $K=100$ non sono buoni predittori ed hanno test error rate rispettivamente di 0.1695 e 0.1925.

Come per la regressione non c'è una forte relazione tra training error rate e test error rate, infatti, per $K=1$ il training error rate è 0 ma il test error rate potrebbe essere abbastanza alto.

In Figura 1.13 sono plottati i test e training errors del KNN in funzione di $1/K$. All'aumentare di $1/K$ il metodo diventa più flessibile. Analogamente alla regressione il training error rate è monotono decrescente all'aumentare della flessibilità mentre il test error rate forma delle curve simili ad una U, decrescenti all'inizio e successivamente crescenti che overfittano quando il modello diventa eccessivamente flessibile.

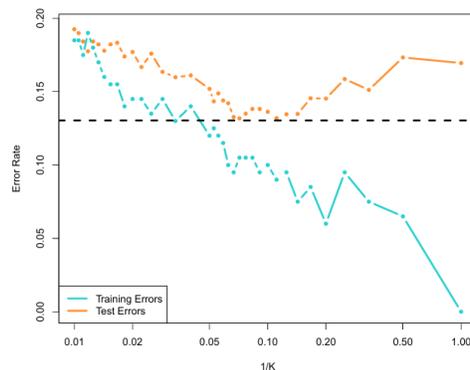


Figura 1.13: Confronto tra il test error rate ed il training error rate per il KNN.

2 Linear Regression

2.1 Simple Linear Regression

La simple linear regression è un approccio utilizzato per predire una risposta quantitativa Y sulla base di una singola variabile di predittori X .

Essa assume che ci sia una relazione approssimativamente lineare tra X ed Y , matematicamente:

$$Y \approx \beta_0 + \beta_1 X \quad (2.1)$$

dove β_0 e β_1 sono due costanti sconosciute che rappresentano l'intercetta e la pendenza, ma vengono chiamate anche parametri o coefficienti

2.1.1 Stimare i coefficienti

Stimare i coefficienti è necessario prima di poter effettuare predizioni.

Se $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ rappresentano n coppie di osservazioni; $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ rappresentano le predizioni per Y basate sull' i -esimo valore di X ; ed $e_i = y_i - \hat{y}_i$ rappresenta l' i -esimo residuo (differenza tra l' i -esimo valore della risposta osservata e l' i -esimo valore della risposta predetta dal nostro modello lineare), allora possiamo definire la *Residual Sum of Squares* come:

$$RSS = e_1^2 + e_2^2 + \dots + e_n^2 \quad (2.2)$$

o sostituendo:

$$RSS = (y_1 - \hat{\beta}_0 + \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 + \hat{\beta}_1 x_2)^2 + \dots + (y_n - \hat{\beta}_0 + \hat{\beta}_1 x_n)^2 \quad (2.3)$$

Scegliamo $\hat{\beta}_0$ ed $\hat{\beta}_1$ che minimizzano l' RSS :

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (2.4)$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \quad (2.5)$$

dove $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ ed $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ sono le sample means.

Dal grafico in Figura 2.1 notiamo come spendere \$1000 in più sulla pubblicità in TV è associato ad una vendita addizionale di 47.5 unità del prodotto, approssimativamente. In questo caso $\hat{\beta}_0 = 7.03$ ed $\hat{\beta}_1 = 0.0475$

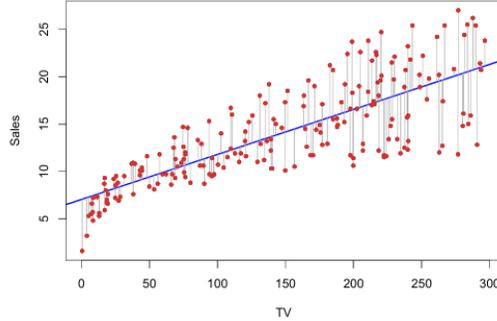


Figura 2.1: Least squares fit per la regressione di sales in TV per l'Advertising data set.

Possiamo sapere quanto sono vicini i valori $\hat{\beta}_0$ ed $\hat{\beta}_1$ rispetto a β_0 e β_1 , calcolando gli *standard error*:

$$SE(\hat{\beta}_0)^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right] \quad (2.6)$$

$$SE(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (2.7)$$

dove $\sigma^2 = Var(\epsilon)$. Dobbiamo assumere che gli errori ϵ_i per ogni osservazione abbiano una varianza comune σ^2 e siano incorrelati, affinché queste formule abbiano validità.

In generale la varianza non è conosciuta ma possiamo stimarla, questa stima è detta *Residual Standard Error*: $RSE = \sqrt{\frac{RSS}{n-2}}$.

Gli errori standard possono essere usati per computare intervalli di confidenza. Un intervallo di confidenza al 95% è definito come il range di valori per il quale esiste il 95% di probabilità che quel range contenga i valori reali (sconosciuti) dei parametri. Questo range è definito in termini di limiti superiore ed inferiore, computati dai sample dei dati.

Per la regressione lineare, l'intervallo di confidenza al 95% per β_1 approssimativamente ha la forma:

$$\hat{\beta}_1 \pm 2 \cdot SE(\hat{\beta}_1) \quad (2.8)$$

Esiste quindi il 95% di probabilità che l'intervallo:

$$[\hat{\beta}_1 - 2 \cdot SE(\hat{\beta}_1), \hat{\beta}_1 + 2 \cdot SE(\hat{\beta}_1)]$$

contenga il valore vero di β_1 . Analogamente per β_0 :

$$\hat{\beta}_0 \pm 2 \cdot SE(\hat{\beta}_0) \quad (2.9)$$

Gli standard error possono essere anche utilizzati per effettuare *hypothesis tests* sui coefficienti. Tra i più comuni abbiamo:

- *null hypothesis*: $H_0 : \beta_1 = 0$ (non c'è una relazione tra X ed Y).
- *alternative hypothesis*: $H_A : \beta_1 \neq 0$ (c'è una relazione tra X ed Y).

Infatti se β_1 è uguale a zero il modello diventa $Y = \beta_0 + \epsilon$, quindi X non è associata ad Y.

Per testare la null hypothesis dobbiamo determinare se $\hat{\beta}_1$ è sufficientemente lontana da zero, in modo da assicurarci che β_1 sia non nulla. Per verificare questo computiamo una *t-statistic*:

$$t = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)} \quad (2.10)$$

Chiamiamo *p-value* la probabilità di osservare un valore uguale a $|t|$ o maggiore (in valore assoluto).

Un piccolo p-value indica che esiste un'associazione tra i predittori e la risposta.

	Coefficient	Std. error	t-statistic	p-value
Intercept	7.0325	0.4578	15.36	< 0.0001
TV	0.0475	0.0027	17.67	< 0.0001

Tabella 2.1: Tabella di regressione per il data set **Advertising**.

La Tabella 2.1 provvede i dettagli del least square model per la regressione del numero di unità vendute nel budget pubblicitario TV per i data **Advertising**. Notiamo come i coefficienti $\hat{\beta}_0$ e $\hat{\beta}_1$ risultano relativamente grandi rispetto agli errori standard, quindi i t-statistic sono altrettanto grandi. Le probabilità di vedere questi valori, se fosse vero H_0 , sono nulle, quindi possiamo concludere che $\beta_0 \neq 0$ e $\beta_1 \neq 0$.

2.1.2 Valutazione della Precisione del Modello

La qualità di una regressione lineare è tipicamente valutata tramite due quantità correlate:

- **Residual Standard Error**

Anche se conosciamo la vera funzione di regressione (anche conoscendo i coefficienti β_0 e β_1), non siamo in grado di predire perfettamente Y da X.

L'RSE è una stima della deviazione standard di ϵ .

$$RSE = \sqrt{\frac{1}{n-2} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (2.11)$$

L'RSE è considerato una misura della *lack of fit* di un modello rispetto ai dati. Se l'RSE è basso, il modello fitta i dati molto bene.

- **R^2 Statistic**

Provvede una misura alternativa del fit. L' R^2 assume sempre valori compresi tra 0 ed 1, rendendolo indipendente dalla scala di Y.

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS} \quad (2.12)$$

dove $TSS = \sum(y_i - \bar{y})^2$ è la *Total Sum of Squares* ed $RSS = \sum(y_i - \hat{y}_i)^2$ la *Residual Sum of Squares*.

L' R^2 misura quindi la proporzione di variabilità di Y che può essere spiegata usando X.

Un R^2 vicino ad 1 indica una grande proporzione della variabilità nella risposta è espressa dalla regressione. Un numero vicino allo 0, invece, indica che la regressione non spiega molto della variabilità della risposta (perché il modello è sbagliato o la varianza dell'errore è elevata).

Sappiamo che la correlazione, definita come:

$$r = Cor(X, Y) = \sum_{i=1}^n \frac{(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{(x_i - \bar{x})^2(y_i - \bar{y})^2}} \quad (2.13)$$

descrive una misura della relazione lineare tra X ed Y. Nella regressione lineare semplice possiamo affermare che $r^2 = R^2$.

Quantity	Value
RSE	3.26
R^2	0.612
F-statistic	312.1

Tabella 2.2: RSE, R^2 ed F-statistic per la regressione lineare del numero di unità vedute in pubblicità TV per il data set **Advertising**.

Nella Tabella 2.2 possiamo notare che l'RSE è 3.26 (le vendite reali in ogni mercato si discostano da quelle approssimate di 3260 unità, in media. Sapere se questa quantità è accettabile o meno dipende dal contesto del problema. Nell'advertising data set il valore medio di vendite su tutti i mercati è di approssimativamente 14000 unità, quindi l'errore percentuale è: $3260/14000 = 23\%$. L' R^2 è 0.61, quindi circa due terzi della variabilità dei **sales** è spiegata da una regressione lineare in **TV**).

2.2 Multiple Linear Regression

Supponiamo di avere p predittori distinti, il modello di regressione lineare multipla ha la forma:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p \quad (2.14)$$

dove X_j rappresenta il j-esimo predittore e β_j quantifica l'associazione tra quella variabile e la risposta.

Nell'esempio di advertising, quindi otteniamo:

$$\text{sales} = \beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{radio} + \beta_3 \times \text{newspaper} + \epsilon \quad (2.15)$$

2.2.1 Stimare i coefficienti

Come per il caso della simple linear regression, i coefficienti $\beta_0, \beta_1, \dots, \beta_p$ sono sconosciuti e devono essere stimati. Possiamo fare predizioni usando la formula:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_p x_p \quad (2.16)$$

Scegliamo $\beta_0, \beta_1, \dots, \beta_p$ per minimizzare la residual sum of square:

$$RSS = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \dots - \hat{\beta}_p x_{ip})^2 \quad (2.17)$$

I valori di $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$ che minimizzano questa espressione sono i coefficienti stimate dei minimi quadrati.

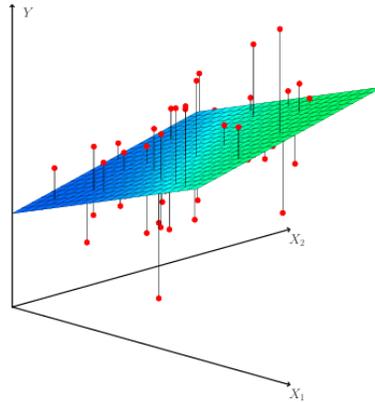


Figura 2.2: Least square fit per il data set `toy` con $p = 2$ predittori.

	Coefficient	Std. error	t-statistic	p-value
Intercept	2.939	0.3119	9.42	<0.0001
TV	0.046	0.0014	32.81	<0.0001
radio	0.189	0.0086	21.89	<0.0001
newspaper	-0.001	0.0059	-0.18	0.8599

Tabella 2.3: Coefficienti di regressione multipli stimati quando i budget pubblicitari di TV, radio e newspaper vengono usati per predire le vendite del prodotto usando il data set `Advertising`.

Nella Tabella 2.3 possiamo notare come spendere ulteriori 1000\$ per la pubblicità audio è associato con un aumento delle vendite di approssimativamente 189 unità.

La regressione multipla per stimare i coefficienti di TV e radio è molto simile a quella lineare. Invece la stima dei coefficienti di newspaper è pari a zero ed il corrispondente p-value non è significativo.

	TV	radio	newspaper	sales
TV	1.0000	0.0548	0.0567	0.7822
radio		1.0000	0.3541	0.5762
newspaper			1.0000	0.2283
sales				1.0000

Tabella 2.4: Matrice di correlazione tra TV, radio, newspaper e sales per il data set `Advertising`.

Notiamo nella Tabella 2.4 che la correlazione tra radio e newspaper è 0.35. Questo indica che i mercati con un alta pubblicità in newspaper tendono ad avere anche molta pubblicità radio. Ma in realtà dalla correlazione tra sales e newspaper possiamo vedere come questi ultimi non sono utilissimi per una maggiore vendita.

Rispondiamo ad alcune domande importanti:

1) Almeno uno dei predittori è utile a produrre la risposta?

Nella regressione multipla la null hypothesis diventa: $\beta_0 = \beta_2 = \dots = \beta_p = 0$; e l'alternative: *almeno un $\beta_j \neq 0$* .

L'hypotesis test è effettuato computando l'F-statistic:

$$F = \frac{(TSS - RSS)/p}{RSS/(n - p - 1)} \quad (2.18)$$

Quando non esiste una relazione tra la risposta ed i predittori l'F-statistic assume un valore vicino all'1. Se almeno un predittore è diverso da zero ci aspettiamo un valore di F maggiore di 1.

Quantity	Value
RSE	1.69
R^2	0.897
F-statistic	570

Tabella 2.5: Altri RSE, R^2 ed F-statistic per la regressione del numero di unità vedute in pubblicità TV per il data set `Advertising`.

Notiamo dalla Tabella 2.5 un F-statistic pari a 570, quindi contraddice la null hypothesis ed almeno uno degli advertising media è collegato ai sales.

L’F-statistic è utile solo quando p è piccolo (comparato con n). Se $p > n$ i coefficienti da stimare sono di più delle osservazioni e non possiamo neanche fittare il modello di regressione lineare multipla usando i minimi quadrati, quindi non possiamo usare di conseguenza l’F-statistic.

2) Tutti i predittori aiutano ad esprimere l’Y, o solamente un subset di essi è utile?

Può capitare che tutti i predittori siano associati con la risposta, ma non è sempre così. Per determinare quali predittori sono associati ad Y, in modo tale da fittare un modello contenente solo questi, si usa la *variable selection*.

3) Quanto bene il modello fitta i dati?

Due misure numeriche per valutare il model fit sono l’RSE e l’ R^2 .

Risulta però che l’ R^2 aumenta quando più variabili sono aggiunte al modello, anche se queste variabili sono poco associate alla risposta. Inoltre, i modelli con più variabili hanno un RSE maggiore se l’RSS decresce in maniera relativamente più bassa rispetto all’aumento di p .

2.3 Qualitative Predictors

Fin’ora abbiamo assunto le variabili per la regressione lineare tutte quantitative ma in pratica potrebbero capitare predittori qualitativi.

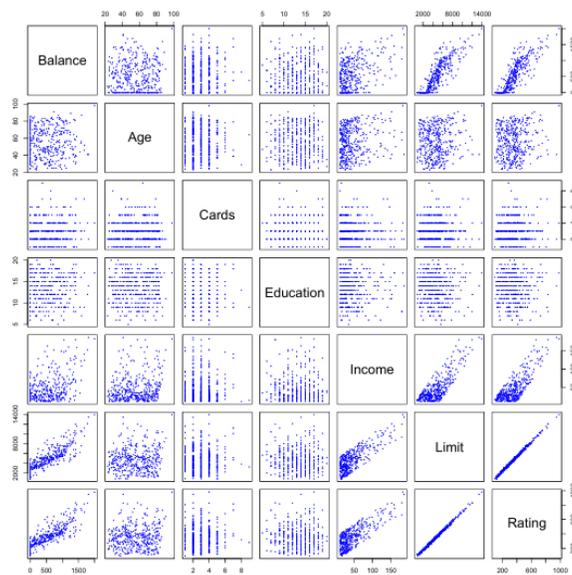


Figura 2.3: Scatterplots del data set Credit contenente informazioni per un numero di potenziali customers.

Il **Credit** data set registra le seguenti variabili per un certo numero di possessori della carta: **balance** (media del debito della carta per ogni individuo), **age**, **cards** (numero di carte di credito), **education** (anni di educazione), **income** (in milioni di dollari), **limit**, **rating**.

Inoltre abbiamo anche delle variabili qualitative: **own** (house ownership), **student** (status dello studente), **status** (stato matrimoniale), **region** (East, West o South).

2.3.1 Predittori con solo Due Livelli

Supponiamo di voler investigare nelle differenze di balance tra le persone che posseggono una casa e quelle che non la posseggono, ignorando altre variabili per il momento.

Se un predittore qualitativo (anche detto *fattore*) ha solo due livelli (o possibili valori) allora è molto semplice incorporarlo nel modello di regressione. Creiamo un indicatore (*dummy variable*) che assume due possibili valori numerici:

$$x_i = \begin{cases} 1 & \text{se la persona possiede una casa} \\ 0 & \text{se la persona non possiede una casa} \end{cases} \quad (2.19)$$

ed usiamo questa variabile come predittore dell'equazione di regressione.

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{se l'i-esima persona possiede una casa} \\ \beta_0 + \epsilon_i & \text{se l'i-esima persona non possiede una casa} \end{cases}$$

dove β_0 può essere interpretato come il credito medio di quelli che non possiedono una casa, mentre $\beta_0 + \beta_1$ quello di chi possiede una casa. Così facendo quindi β_1 rappresenta la differenza di balance medio tra i possessori di casa ed i non.

	Coefficient	Std. error	t-statistic	p-value
Intercept	509.80	33.13	15.389	<0.0001
own [Yes]	19.73	46.05	0.429	0.6690

Tabella 2.6: Stima dei coefficienti associati con la regressione di balance in own nel data set **Credit**.

Come si nota dalla Tabella 2.6 il debito medio per i non possessori di casa è stimato essere 509.80\$, mentre si stima che i possessori di casa posseggano un debito addizionale di 19.73\$ ($509.80 + 19.73 = 529.53$ \$).

Notiamo che il p-value per la dummy variable è alto, quindi, non esiste un'evidenza statistica che il balance medio sia basato sulla possessione delle case.

2.3.2 Predittori con Più di Due Livelli

Se un predittore ha più di due livelli, una singola dummy variable non può rappresentare tutti i possibili valori. Creiamo quindi più dummy variables. Per la variabile region ad esempio:

$$x_{i1} = \begin{cases} 1 & \text{se la persona è del South} \\ 0 & \text{se la persona non è del South} \end{cases} \quad (2.20)$$

$$x_{i2} = \begin{cases} 1 & \text{se la persona è del West} \\ 0 & \text{se la persona non è del West} \end{cases} \quad (2.21)$$

Otteniamo, quindi, il modello:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{se l'i-esima persona è del South} \\ \beta_0 + \beta_2 + \epsilon_i & \text{se l'i-esima persona è del West} \\ \beta_0 + \epsilon_i & \text{se l'i-esima persona è dell'East} \end{cases}$$

dove β_0 è interpretato come il balance medio per gli individui provenienti da East, β_1 è interpretato come la differenza di balance medio per gli individui provenienti da South rispetto a quelli provenienti da East, β_2 è interpretato come la differenza di balance medio per gli individui provenienti da West rispetto a quelli provenienti da East,

Ci sono sempre un numero minore di dummy variables rispetto al livello. Il livello ottenuto senza usare la dummy variable viene chiamato *baseline*.

	Coefficient	Std. error	t-statistic	p-value
Intercept	531.00	46.32	11.464	<0.0001
region [South]	-12.50	56.68	-0.221	0.8260
region [West]	-18.69	65.02	-0.287	0.7740

Tabella 2.7: Stima dei coefficienti associati con la regressione di balance in region nel data set `Credit`.

Nella Tabella 2.7 notiamo che il balance stimato per la baseline (East) è di 531.00\$. Quelli del South avranno un debito minore di 12.50\$ rispetto a quelli in East, e quelli del West di 18.69\$ in meno rispetto a quelli in East.

Notiamo però come i p-value associati alla stima dei coefficienti per le due variabili dummy risultano alti, suggerendo una non evidenza statistica in differenza di balance medio tra South ed East e tra West ed East.

Ciò che scegliamo come baseline è arbitrario, ma i coefficienti ed i p-value relativi dipendono da questa scelta.

Usando un F-test per testare l'ipotesi nulla otteniamo un p-value di 0.96, ciò conferma che non esiste una relazione tra balance e region.

Possiamo usare le dummy variable anche in presenza di problemi aventi sia variabili quantitative che qualitative.

2.4 Estensione del Modello Lineare

I modelli di regressione lineare standard forniscono risultati interpretabili che spesso funzionano bene nei problemi reali. Essi però effettuano delle assunzioni restrittive che nella pratica spesso vengono violate. Due di queste:

- *additivity*: l'associazione tra un predittore X_j e la risposta Y non dipende dai valori degli altri predittori;
- *linearity*: il cambio della risposta Y associato con il cambio di un unità di X_j è costante, indipendentemente dal valore di X_j .

2.4.1 Rimuovere l'Additive Assumption

Supponiamo che spendere soldi in pubblicità radio incrementa l'efficacia della pubblicità TV, quindi il coefficiente per la TV dovrebbe crescere all'aumentare di radio. Un budget di 100000\$ speso metà in radio e metà in TV, potrebbe far crescere i sales meglio rispetto ad allocare tutto il budget in TV o radio.

Questo in marketing è conosciuto come *synergy effect* ed in statistica come *interaction effect*.

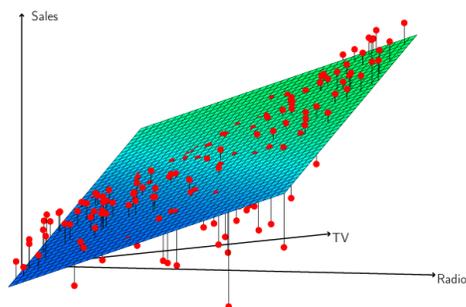


Figura 2.4: Per il data set **Advertising**, linear regression fit di sales usando TV e radio come predittori.

Un modo per estendere il modello è quello di includere un ulteriore predittore chiamato *interaction term* (costruito computando il prodotto tra due predittori), ad esempio:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon$$

Nel caso del data set advertising diventa:

$$\begin{aligned} Y &= \beta_0 + \beta_1 \times TV + \beta_2 \times radio + \beta_3 \times (radio \times TV) + \epsilon \\ &= \beta_0 + (\beta_1 + \beta_3 \times radio) \times TV + \beta_2 \times radio + \epsilon \end{aligned} \quad (2.22)$$

dove β_3 rappresenta l'incremento nell'efficienza della pubblicità TV associata ad un incremento unitario della pubblicità radio.

	Coefficient	Std. error	t-statistic	p-value
Intercept	6.7502	0.248	27.23	<0.0001
TV	0.0191	0.002	12.70	<0.0001
radio	0.0289	0.009	3.24	0.0014
TV×radio	0.0011	0.000	20.73	<0.0001

Tabella 2.8: Coefficienti stimati associati al modello dell'equazione (2.22).

Nella Tabella 2.8 notiamo come il modello che include il termine di interazione è superiore rispetto a quello che contiene solo i *main effects*, infatti il p-value associato a **TV×radio** è molto basso.

I coefficienti stimati nella Tabella 2.8 suggeriscono che un incremento di 1000\$ in pubblicità TV è associato ad un incremento dei sales pari a $(\hat{\beta}_1 + \hat{\beta}_3 \times radio) \times 1000 = 19 + 1.1 \times radio$, unità, ed un incremento di 1000\$ in pubblicità radio è associato ad un incremento dei sales pari a $(\hat{\beta}_2 + \hat{\beta}_3 \times TV) \times 1000 = 29 + 1.1 \times TV$, unità.

Potrebbe capitare che l'interazione tra due termini fornisca un p-value basso anche se uno dei due termini ha un p-value elevato.

Per il *principio gerarchico* se introduciamo un'interazione in un modello allora dobbiamo includere anche i main effects (anche se i p-value associati ai loro coefficienti non sono significativi).

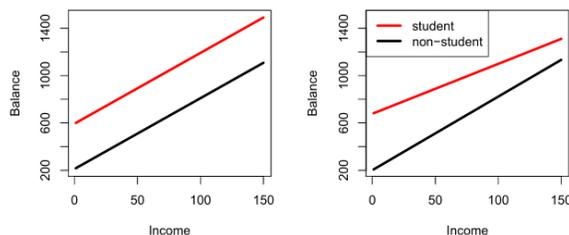


Figura 2.5: Relazione tra balance ed income per il data set **Credit**.

Considerando il data set **Credit** supponendo di voler predire **balance** usando l'**income** (variabile quantitativa) e lo **student** (variabile qualitativa), in assenza del termine di interazione, il modello assume la forma:

$$balance_i \approx \beta_0 + \beta_1 \times income_i + \begin{cases} \beta_2 & \text{se l'-esima persona è uno studente} \\ 0 & \text{se l'-esima persona non è uno studente} \end{cases} \quad (2.23)$$

Notiamo che in questo modo stiamo fittando due linee parallele, illustrate nel grafico a sinistra della Figura 2.5, dove possiamo notare che non esiste un

interazione tra student ed income. Se invece consideriamo:

$$balance_i \approx \beta_0 + \beta_1 \times income_i + \begin{cases} \beta_2 + \beta_3 \times income_i & \text{se è uno studente} \\ 0 & \text{se non è uno studente} \end{cases} \quad (2.24)$$

In questo modo dal grafico a destra della Figura 2.5 notiamo come esiste un'interazione tra income e student.

2.4.2 Rimuovere la Linear Assumption

La *polynomial regression* è un semplice modo utilizzato per estendere il modello lineare a realizzare relazioni non lineari.

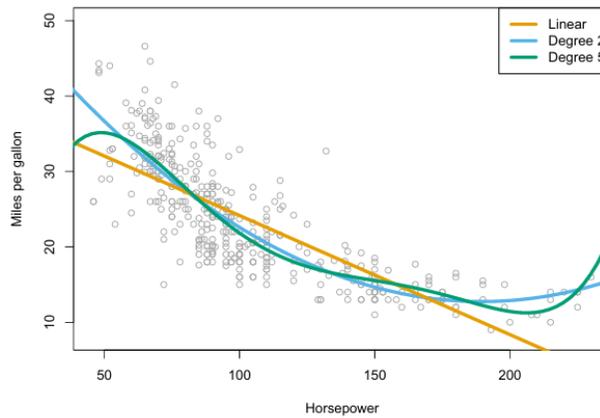


Figura 2.6: Relazione tra mpg ed horsepower per un certo numero di macchine del data set **Auto**.

Nella Figura 2.6 la linea arancione rappresenta il fitting utilizzando la relazione lineare, ma il modello sembra avere una forma quadratica.

Il modello potrebbe avere la seguente forma:

$$mpg = \beta_0 + \beta_1 \times horsepower + \beta_2 \times horsepower^2 + \epsilon \quad (2.25)$$

rappresentato nel grafico dalla curva blu. Dalla Tabella 2.9 notiamo che i p-values sono significativi, quindi includere il termine $horsepower^2$ porta ad un miglioramento del modello.

	Coefficient	Std. error	t-statistic	p-value
Intercept	56.9001	1.8004	31.6	<0.0001
horsepower	-0.4662	0.0311	-15.0	<0.0001
horsepower ²	0.0012	0.0001	10.1	<0.0001

Tabella 2.9: Stima dei coefficienti associati alla regressione di mpg in horsepower ed $horsepower^2$ per il data set **Auto**.

3 Classification

Nei problemi di classificazione anche abbiamo un set di osservazioni di training $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ che usiamo per costruire il classificatore.

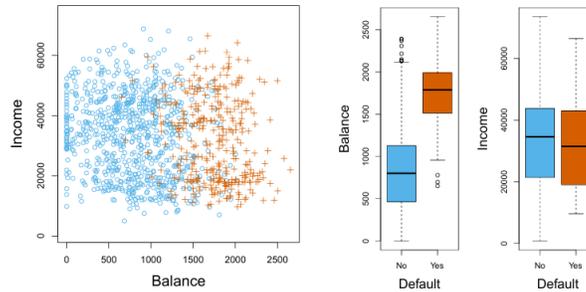


Figura 3.1: Default data set.

Nel grafico a sinistra della Figura 3.1 sono plottati l'income annuale ed il balance mensile di credito per un subset di 10000 individui.

Gli individui che hanno effettuato default (mostrati in rosso) tendono ad avere un balance di credito maggiore di quelli che non lo hanno fatto (mostrati in blu).

A destra della Figura 3.1 sono mostrati due box plot, il primo mostrante la distribuzione di balance divisa dalla variabile binaria default ed il secondo un grafico simile per l'income. Notiamo come esiste una relazione pronunciata tra il predittore balance e la risposta default.

Nella realtà la relazione tra predittore e risposta non è accentuata, ma per illustrare le procedure di classificazione faremo esempi esagerati.

Supponiamo di voler predire la condizione medica di un paziente in pronto soccorso sulla base dei suoi sintomi. Avendo, ad esempio, tre possibili diagnosi: *stroke*, *drug overdose* ed *epileptic seizure*. Potremmo codificare questi valori come una risposta quantitativa:

$$Y = \begin{cases} 1 & \text{stroke} \\ 2 & \text{drug overdose} \\ 3 & \text{epileptic seizure} \end{cases} \quad (3.1)$$

Tramite questa potremmo fittare un modello lineare per predire Y sulla base di un set di predittori X_1, X_2, \dots, X_p . Ma così facendo la relazione tra *stroke* e *drug overdose* è la stessa di quella tra *drug overdose* ed *epileptic seizure*.

Quindi non è possibile convertire risposte qualitative con più di due livelli in risposte quantitative utilizzabili in regressioni lineari.

Per le risposte qualitative a due livelli (*binarie*) possiamo utilizzare una dummy variabile:

$$Y = \begin{cases} 0 & \text{stroke} \\ 1 & \text{drug overdose} \end{cases} \quad (3.2)$$

Quindi fittare una regressione lineare e predire drug overdose se $\hat{Y} > 0.5$ altrimenti stroke.

Potrebbe capitare però che le stime capolino fuori all'intervallo $[0,1]$, rendendo difficile interpretare le probabilità.

3.1 Logistic Regression

Considerando il data set `Default`, la risposta `default` può ricadere in due categorie (`Yes` o `No`). Tramite la logistic regression si valuta la probabilità che Y appartenga ad una particolare categoria, ad esempio dato il `balance` possiamo scrivere: $Pr(\text{default} = \text{Yes} | \text{balance})$, che noi possiamo abbreviare con $p(\text{balance})$, assumerà valori compresi tra 0 ed 1. Possiamo predire `default = Yes` quando $p(\text{balance}) > 0.5$; o se un'azienda desidera essere prudente nel predire chi è a rischio di default, sceglierà un threshold più basso, ad esempio $p(\text{balance}) > 0.1$.

3.1.1 Logistic Model

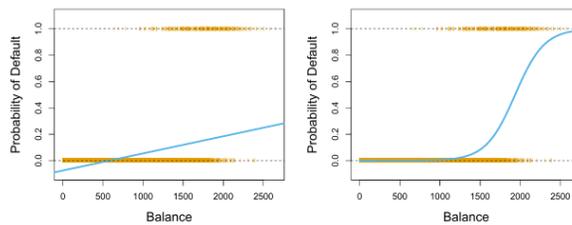


Figura 3.2: Classification usando il data set `Default`.

Nel grafico a sinistra della Figura 3.2 è rappresentata il modello ottenuto applicando una regressione lineare per predire `default = Yes`, usando il `balance`.

Notiamo che per i `balance` vicini allo zero otteniamo valori di probabilità negativi, analogamente per valori elevati di `balance` otterremo valori maggiori di 1. Per risolvere questo problema usiamo la funzione logistica:

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} \quad (3.3)$$

Il fit della regressione logistica è mostrato nel grafico a destra della Figura 3.2. Notiamo come essa produce una curva ad S.

Manipolando l'equazione (3.3), ricaviamo:

$$\frac{p(X)}{1-p(X)} = e^{\beta_0 + \beta_1 X} \quad (3.4)$$

dove la quantità $p(X)/[1-p(X)]$ viene chiamata *odds*, ed assume valore tra 0 ed ∞ . Applicando il logaritmo ad ambo i membri:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X \quad (3.5)$$

3.1.2 Stimare i coefficienti di regressione

Per conoscere β_0 e β_1 utilizziamo un metodo chiamato *maximum likelihood*.

In questo modo cerchiamo di trovare $\hat{\beta}_0$ ed $\hat{\beta}_1$ in maniera tale che $p(X)$ fornisca un numero vicino ad uno per le persone che hanno effettuato default e vicino allo zero per quelle che non l'ho hanno fatto.

Questo viene formalizzato dalla *likelihood function*:

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1-p(x_{i'})) \quad (3.6)$$

Le stime $\hat{\beta}_0$ ed $\hat{\beta}_1$ sono scelte in modo da massimizzare questa funzione.

	Coefficient	Std. error	z-statistic	p-value
Intercept	-10.6513	0.3612	-29.5	<0.0001
balance	0.0055	0.0002	24.9	<0.0001

Tabella 3.1: Coefficienti stimati che prevedono la probabilità di default usando il balance per il data set `Default`.

Notiamo dalla Tabella 3.1 come $\hat{\beta}_1 = 0.0055$, questo indica che un incremento di balance è associato con un incremento nella probabilità di default.

La z-statistic ha lo stesso ruolo del t-statistic. La z-statistic associata a β_1 è $\hat{\beta}_1/SE(\hat{\beta}_1)$.

3.1.3 Making Prediction

La probabilità di default per un individuo con un balance pari a 1000\$ è:

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X}} = \frac{e^{-10.6513 + 0.0055 \times 1000}}{1 + e^{-10.6513 + 0.0055 \times 1000}} = 0.00576$$

Che è minore dell'1%. Se consideriamo un individuo con un balance pari a 2000\$ $\hat{p}(X) = 0.586$ quindi 58.6%.

Per fittare un modello che usa lo stato dello studente come un predittore, creiamo semplicemente una variabile fantoccio che assume valore 1 se è uno studente e 0 se non lo è.

	Coefficient	Std. error	z-statistic	p-value
Intercept	-3,5041	0.0707	-49.55	<0.0001
student [Yes]	0.4049	0.1150	3.52	0.0004

Tabella 3.2: Coefficienti stimati che prevedono la probabilità di default usando lo student status per il data set `Default`.

Dalla Tabella 3.2 notiamo che il coefficiente associato con la variabile fantoccio è positivo ed il p-value associato è significante. Quindi gli studenti hanno una probabilità di default maggiore dei non-student.

$$\widehat{Pr}(\text{default} = \text{Yes} | \text{student} = \text{Yes}) = \frac{e^{-3.5041+0.4049 \times 1}}{1 + e^{-3.5041+0.4049 \times 1}} = 0.0431$$

$$\widehat{Pr}(\text{default} = \text{Yes} | \text{student} = \text{No}) = \frac{e^{-3.5041+0.4049 \times 0}}{1 + e^{-3.5041+0.4049 \times 0}} = 0.0292$$

3.1.4 Multiple Logistic Regression

Possiamo estendere la simple logistic regression come segue:

$$\log \left(\frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p \quad (3.7)$$

che può essere riscritta in questa maniera:

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}} \quad (3.8)$$

	Coefficient	Std. error	z-statistic	p-value
Intercept	-10.8690	0.4923	-22.08	<0.0001
balance	0.0057	0.0002	24.74	<0.0001
income	0.0030	0.0082	0.37	0.7115
student [Yes]	-0.6468	0.2362	-2.74	0.0062

Tabella 3.3: Coefficienti stimati con la regressione logistica per il data set `Default`.

Notiamo dalla Tabella 3.3 che i p-value associati con il balance e lo student status sono piccoli (quindi associati alla probabilità di default). Però, i coefficienti della variabile fantoccio in questo caso sono negativi (indicando che per gli studenti è meno probabile il default) mentre nella Tabella 3.2 risultavano positivi.

La Figura 3.3 mostra una illustrazione grafica di questo paradosso. Le linee rossa e blu indicano i rates medi di default per gli studenti e non-studenti, rispettivamente, in funzione del credit card balance.

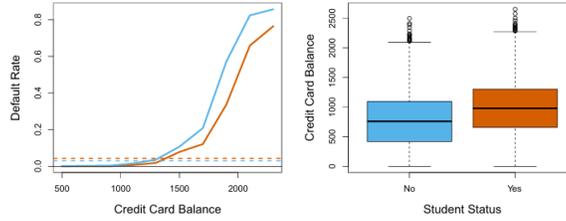


Figura 3.3: Confounding nel data set `Default`.

Il coefficiente negativo per gli studenti nella regressione logistica multipla sta ad indicare che per un valore fisso di balance ed income, uno studente è meno propenso al default di un non-studente. Infatti, dal grafico a sinistra osserviamo che il default rate per gli studenti è inferiore a quello dei non studenti per ogni valore di balance. Ma le linee tratteggiate orizzontali (che indicano la media del default rate per gli studenti e non studenti su tutti i valori di balance ed income) dimostrano il contrario.

Il grafico a destra fornisce una spiegazione per questa discrepanza. Le variabili student e balance sono correlate (gli studenti tendono ad avere maggiori livelli di debito e credit card balance, il che è associato con una probabilità maggiore di default).

Se un'azienda di carte di credito sta cercando di determinare a chi dovrebbero offrire denaro: uno studente è più rischioso di un non-studente se non esistono informazioni riguardanti il balance; ma uno studente è meno rischioso di un non-studente con lo stesso credit card balance!

Questo fenomeno è conosciuto come *confounding*.

3.1.5 Multinomial Logistic Regression

È possibile estendere la two-class logistic regression a $K > 2$ classi. Questa estensione è conosciuta come *multinomial logistic regression*. Per effettuarla, selezioniamo inizialmente una singola classe che funge da *baseline*, senza perdere generalità, selezioniamo la K -esima classe per questo ruolo:

$$Pr(Y = k | X = x) = \frac{e^{\beta_{k_0} + \beta_{k_1} x_1 + \dots + \beta_{k_p} x_p}}{1 + \sum_{l=1}^{K-1} e^{\beta_{l_0} + \beta_{l_1} x_1 + \dots + \beta_{l_p} x_p}} \quad (3.9)$$

con $k = 1, \dots, K - 1$.

3.2 Generative Models per la Classification

Supponiamo di voler classificare un'osservazione in una classe, avendo K classi ($K \geq 2$) disponibili. Quindi la risposta qualitativa Y può assumere K possibili valori distinti e disordinati.

Con π_k rappresentiamo la probabilità a *priori* che un'osservazione scelta randomicamente provenga dalla k-esima classe, ed $f_k(X) = Pr(X|Y = k)$ denota la *funzione densità* di X per un'osservazione che proviene dalla k-esima classe.

Se $f_k(X)$ è relativamente grande allora l'osservazione della k-esima classe con alta probabilità ha $X \approx x$, se invece $f_k(X)$ è piccolo $X \approx x$ è improbabile.

Per il *teorema di Bayes*:

$$p_k(x) = Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)} \quad (3.10)$$

Questa è detta *posterior probability* ed è la probabilità che l'osservazione appartenga alla k-esima classe, dato il valore del predittore per quella osservazione.

Stimare π_k è semplice rispetto ad $f_k(x)$. Di seguito elenchiamo alcune classifiers utilizzando diverse stime di $f_k(x)$.

3.2.1 Linear Discriminant Analysis per $p = 1$

Supponendo di avere un solo predittore. Per stimare $f_k(x)$ assumiamo che essa sia una *Gaussiana*:

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{(x - \mu_k)^2}{2\sigma_k^2}\right) \quad (3.11)$$

con μ_k la media e σ_k^2 la varianza, per la k-esima classe. Assumiamo che $\sigma_1^2 = \dots = \sigma_k^2$, che per semplicità chiamiamo solo σ^2 . Sostituendo quindi la (3.11) nella (3.10) otteniamo:

$$p_k(x) = Pr(Y = k|X = x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^K \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)} \quad (3.12)$$

Il classificatore Bayes assegnerà, quindi, un'osservazione $X = x$ alla classe con $p_k(x)$ maggiore; che risulta equivalente ad assegnare l'osservazione alla classe con:

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \quad (3.13)$$

maggiore. Se $K = 2$ e $\pi_1 = \pi_2$ allora il Bayes classifier assegnerà l'osservazione alla classe 1 quando $2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2$ ed alla classe 2 altrimenti.

Il decision boundary del Bayes è il punto per il quale $\delta_1 = \delta_2$, ovvero:

$$x = \frac{\mu_1^2 - \mu_2^2}{2(\mu_1 - \mu_2)} = \frac{\mu_1 + \mu_2}{2} \quad (3.14)$$

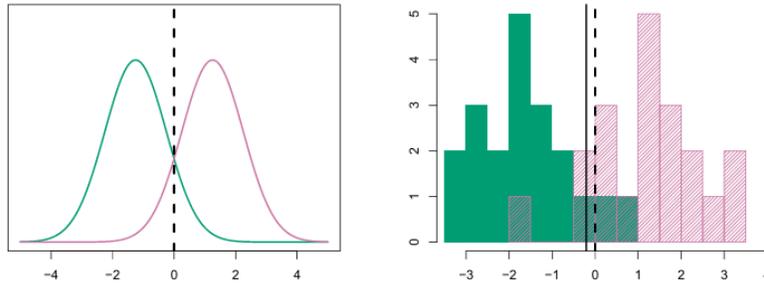


Figura 3.4: Due funzioni di densità *normali* e divisione di 20 osservazioni nelle due classi.

Nel grafico a destra della Figura 3.4 sono mostrate due funzioni gaussiane $f_1(x)$ e $f_2(x)$, rappresentanti due distinte classi. La media e la varianza per queste funzioni sono: $\mu_1 = -1.25$, $\mu_2 = 1.25$, $\sigma_1^2 = \sigma_2^2 = 1$. Tra le due densità è presente overlap, quindi dato $X = x$ abbiamo un'incertezza sulla classe di appartenenza dell'osservazione. Se assumiamo che l'osservazione è egualmente probabile che appartenga ad una classe o all'altra ($\pi_1 = \pi_2 = 0.5$) allora il Bayes classifier assegnerà l'osservazione alla classe 1 quando $x < 0$ e alla 2 altrimenti.

Tramite la *Linear Discriminant Analysis* (LDA) si approssima il classificatore Bayes sostituendo nella (3.13) le stime:

$$\hat{\pi}_k = \frac{n_k}{n} \quad (3.15)$$

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i \quad (3.16)$$

$$\hat{\sigma}^2 = \frac{1}{n - K} \sum_{k=1}^K \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2 \quad (3.17)$$

con n numero totale delle osservazioni di training ed n_k il numero di osservazioni di training nella k -esima classe.

Il grafico a destra della Figura 3.4 mostra un istogramma per un sample casuale di 20 osservazioni per ogni classe (in questo caso, quindi, $n_1 = n_2 = 20 \rightarrow \hat{\pi}_1 = \hat{\pi}_2$).

Il Bayes decision boundary è mostrato dalla linea verticale tratteggiata, mentre quello dell'LDA è mostrato da quella continua (che è leggermente spostata a sinistra). Tutti i punti a sinistra di questa linea saranno assegnati alla classe verde, mentre quelli a destra saranno assegnati alla classe viola.

Il Bayes error rate è 10.6% e l'LDA test error rate è 11.1%, quindi possiamo dire che quest'ultimo performa bene su questo data set.

3.2.2 Linear Discriminant Analysis per $p > 1$

Per estendere il classificatore LDA nel caso di più predittori $X = (X_1, \dots, X_p)$ assumeremo una distribuzione Gaussiana multivariata.

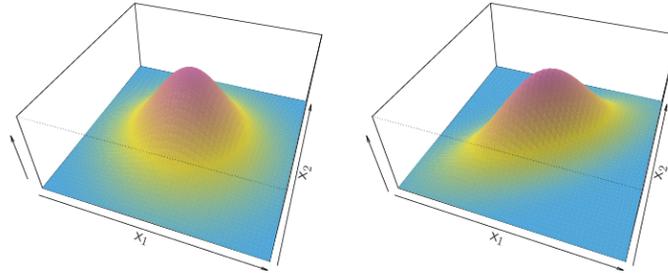


Figura 3.5: Esempi di distribuzione Gaussiana multivariata con $p = 2$.

Nel grafico a sinistra della Figura 3.5 si ha $Cor(X_1, X_2) = 0$ e $Var(X_1) = Var(X_2)$ ottenendo una forma a *campana*. Questa campana risulterà distorta se i predittori sono correlati o hanno varianze diverse, come si nota dalla figura a destra.

Per indicare che una variabile casuale p -dimensionale X ha una distribuzione Gaussiana multivariata scriviamo: $X \sim N(\mu, \Sigma)$.

Con $E(X) = \mu$: la media di X ; e $Cov(X) = \Sigma$: la matrice covarianza ($p \times p$) di X . La densità Gaussiana multivariata è definita come:

$$f(x) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right) \quad (3.18)$$

La funzione densità per la k -esima classe è $f_k(X = x)$, quindi, il classificatore Bayes assegna un osservazione $X = x$ alla classe con:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k \quad (3.19)$$

Nella Figura 3.6 le tre ellissi rappresentano le regioni contenenti il 95% della probabilità di appartenere ad ognuna delle tre classi. Le linee tratteggiate rappresentano i Bayes decision boundaries [per cui $\delta_k(x) = \delta_l(x)$, con $k \neq l$]. Le linee continue rappresentano invece i decision boundaries dell'LDA. Anche in questo caso sono molto vicini ed i test error rate per i classificatori Bayes ed LDA sono rispettivamente 0.0764 e 0.0770; quindi LDA ha buone performance su questi dati.

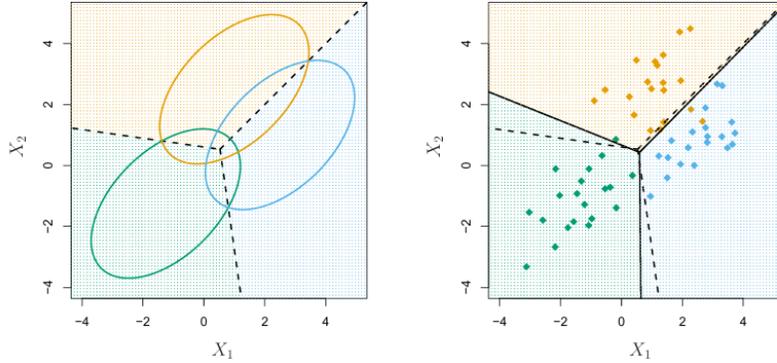


Figura 3.6: Tre classe Gaussianhe sono mostrate con un class-specific mean vector ed una matrice covarianza comune.

Applicando l'LDA sul `Default` data set possiamo predire se un individuo effettuerà default o meno. Per 10000 training samples si ottiene un training error di 2.75% che sembra basso, ma bisogna fare attenzione perché:

1. il training error rate è più basso del test error rate, solitamente;
2. il banale *null* classifier avrà un error rate leggermente maggiore dell'LDA training error rate.

Quindi un classificatore binario, come questo, potrà fare due tipi di errori: assegnare un individuo alla classe "no default" pure avendolo fatto, oppure assegnare un individuo alla categoria "default" quando non lo fa. Certe volte è necessario determinare quale dei seguenti errori è stato commesso.

		<i>True</i>	<i>default status</i>	
		No	Yes	Total
<i>Predicted</i>	No	9644	252	9896
<i>default status</i>	Yes	23	81	104
Total		9667	333	10000

Tabella 3.4: Una confusion matrix per comparare le predizione LDA ai true default statuses per 10000 osservazioni di training nel data set `Default` (gli elementi diagonali della matrice rappresentano gli individui con default status predicato correttamente, mentre quelli off-diagonals rappresentano gli elementi misclassificati).

Dalla tabella notiamo che LDA ha previsto un totale di 104 persone che effettueranno default, di queste persone 81 hanno realmente effettuato default mentre le restanti 23 no. Quindi solamente a 23 su 9667 individui era stata assegnata una label sbagliata (sembra un basso error rate).

Però, di 333 persone che defaulted, 252 non sono stati considerati dall'LDA. Quindi l'error rate totale è basso, ma l'error rate tra le persone che hanno fatto default è molto alto (un error rate di $252/333 = 75.7\%$ è inaccettabile).

La *sensitivity* (in questo caso) è la percentuale di individui che realmente hanno fatto default e sono stati identificati: $(1 - 252/333)*100 = 24.3\%$.

La *specificity* è la percentuale di non-defaulters che sono stati identificati correttamente: $(1 - 23/9667)*100 = 99.8\%$

È possibile modificare l'LDA in modo tale da sviluppare un classificatore migliore.

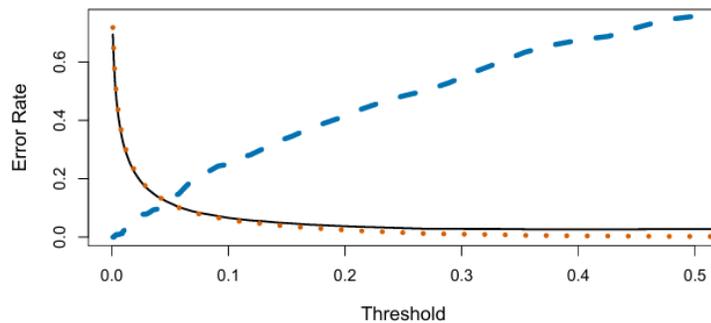


Figura 3.7: Per il data set `Default` sono mostrati alcune Error Rate in funzione del Threshold.

Utilizzando un threshold di 0.5 si minimizza l'overall error rate (mostrato nella Figura 3.7 dalla linea nera), questo potevamo aspettarcelo dato che il Bayes (avente l'overall error rate più basso) utilizza lo stesso valore di threshold. Ma usando un threshold di 0.5 l'error rate per gli individui che hanno effettuato default è molto alto (mostrato dalla linea blu tratteggiata). Se abbassassimo il threshold questo si abbasserebbe, ma aumenterebbe l'error rate degli individui che non hanno effettuato default (mostrato dalla linea arancione punteggiata). Quindi il nostro obiettivo è quello di trovare un trade-off.

La curva ROC (*Receiver Operating Characteristics*) è un grafico usato per mostrare contemporaneamente due tipi di errori per ogni possibile valore di threshold. La performance complessiva del classificatore è data dall'AUC (*Area Under the Curve*). Maggiore è l'AUC e migliore sarà il classificatore, quindi una curva ROC ideale abbraccerà l'angolo in alto a sinistra.

Per la Figura 3.8 l'AUC è 0.95 che molto vicino al massimo 1.0, quindi è considerato molto buono. La curva ROC per la logistic regression è virtualmente indistinguibile da quella dell'LDA.

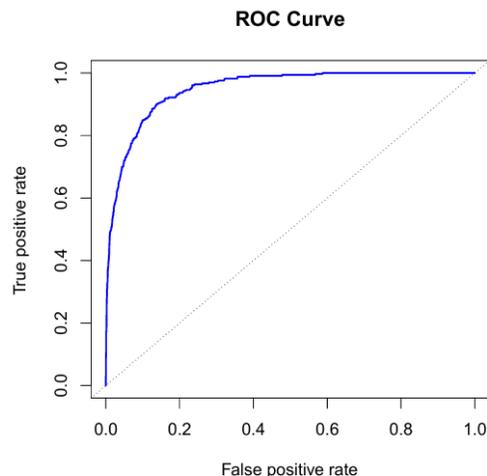


Figura 3.8: Curva ROC per il classificatore LDA sul data set `Default`. La linea tratteggiata rappresenta il "no information" classifier.

3.2.3 Naive Bayes

Sappiamo che stimare $\hat{\pi}_k$ è semplice (come proporzione di osservazioni di training appartenenti all k -esima classe per $k = 1, \dots, K$).

Il naive Bayes stima $f_k(x)$ in maniera diversa, invece di assumere che questa funzione appartenga ad una famiglia di distribuzioni (nel caso dell'LDA la normale Gaussiana), facciamo una singola assunzione: ***fino alla k -esima classe tutti i predittori sono indipendenti***:

$$f_k(x) = f_{k_1}(x_1) \times f_{k_2}(x_2) \times \dots \times f_{k_p}(x_p) \quad (3.20)$$

dove f_{k_j} è la funzione densità del j -esimo predittore tra le osservazioni della k -esima classe.

Stimare una funzione di densità p -dimensionale è difficile, perché dovremmo considerare sia la *distribuzione marginale* (la distribuzione di ogni predittore da solo) che la *distribuzione congiunta* (l'associazione tra i differenti predittori). Assumendo che i predittori sono indipendenti nella classe non consideriamo quest'ultima.

Il naive Bayes introduce un po di bias, ma riduce la varianza, portando ad un classificatore che lavora bene in pratica (per via del bias variance trade-off).

La posterior probability è:

$$Pr(Y = k|X = x) = \frac{\pi_k \times f_{k_1}(x_1) \times f_{k_2}(x_2) \times \dots \times f_{k_p}(x_p)}{\sum_{l=1}^K \pi_l \times f_{l_1}(x_1) \times f_{l_2}(x_2) \times \dots \times f_{l_p}(x_p)} \quad (3.21)$$

con $k = 1, \dots, K$.

Consideriamo il classificatore naive Bayes per il dataset `toy` con $p = 3$ predittori (i primi due quantitativi ed il terzo qualitativo, a tre livelli) e $K = 2$ classi.

Supponendo che $\hat{\pi}_1 = \hat{\pi}_2 = 0.5$, le funzioni densità stimate \hat{f}_{kj} (per $k = 1, 2$ e $j = 1, 2, 3$) sono mostrate in Figura 3.9. Supponiamo di voler classificare una nuova osservazione $x^* = (0.4, 1.5, 1)^T$, otteniamo: $\hat{f}_{11}(0.4) = 0.368$, $\hat{f}_{12}(1.5) = 0.484$, $\hat{f}_{13}(1) = 0.226$; $\hat{f}_{21}(0.4) = 0.030$, $\hat{f}_{22}(1.5) = 0.130$, $\hat{f}_{23}(1) = 0.616$. Inoltre le posterior probability stimate sono: $Pr(Y = 1|X = x^*) = 0.944$; $Pr(Y = 2|X = x^*) = 0.056$.

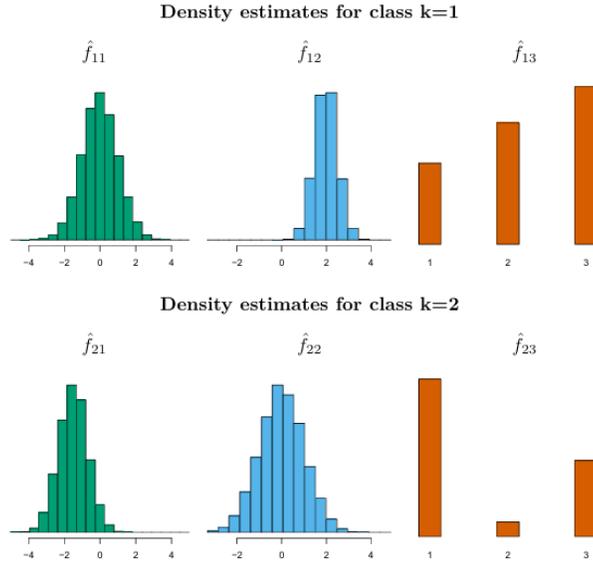


Figura 3.9: Esempio toy.

Settando K come *baseline* class assegneremo un'osservazione alla classe che massimizza:

$$\begin{aligned}
 & \log \left(\frac{Pr(Y = k|X = x)}{Pr(Y = K|X = x)} \right) = \log \left(\frac{\pi_k f_k(x)}{\pi_K f_K(x)} \right) = \\
 & = \log \left(\frac{\pi_k \prod_{j=1}^p f_{kj}(x_j)}{\pi_K \prod_{j=1}^p f_{Kj}(x_j)} \right) = \log \left(\frac{\pi_k}{\pi_K} \right) + \sum_{j=1}^p \log \left(\frac{f_{kj}(x_j)}{f_{Kj}(x_j)} \right) = \\
 & = a_k + \sum_{j=1}^p g_{kj}(x_j) \tag{3.22}
 \end{aligned}$$

per $k = 1, \dots, K$. Dove $a_k = \log(\frac{\pi_k}{\pi_K})$ e $g_{kj}(x_j) = \log(\frac{f_{kj}(x_j)}{f_{Kj}(x_j)})$. Quindi assume la forma di un *generalized additive model*.

4 Resampling Methods

Questi metodi vengono utilizzati per generare nuovi punti di dati nel dataset prendendo in maniera casuale data point già esistenti. È utile a creare nuovi dataset per addestrare modelli e per stimare le proprietà di un dataset quando questo è sconosciuto, difficile da stimare o ancora se la dimensione dei sample nel dataset è piccola.

4.1 Cross-Validation

Il test error può essere facilmente calcolato se è disponibile un dataset di partenza. Nella cross-validation consideriamo una classe di metodi che stimano il test error rate *trattenendo* un subset di osservazioni di training dal processo di fitting e successivamente applicando i metodi di statistical learning su queste osservazioni.

4.1.1 Validation Set

Questo approccio è molto semplice per stimare i test error associati ad un particolare metodo di statistical learning su un set di osservazioni.

Consiste nel dividere in maniera casuale il set di osservazioni in due parti: *training set* e *validation set* (anche detto hold-out set). Il modello viene fittato inizialmente nel training set e quest'ultimo viene usato per predire la risposta per le osservazioni del validation set. Il risultante validation set error rate provvede una stima del test error rate.



Figura 4.1: Esempio schematico del validation set approach. Un set di n osservazioni viene diviso casualmente in training set (in blu, contenente le osservazioni 7, 22, 13, ed altre...) ed un validation set (in beige, contenente l'osservazione 91, ed altre...).

Dividiamo randomicamente le 392 osservazioni del data set **Auto** in due set (training set di 196 data points ed un validation set contenente le restanti 196 osservazioni). Il grafico a sinistra nella Figura 4.2 mostra il validation set error risultante dal fitting di vari modelli regressivi sui sample di training e valutando le loro performance sui validation sample (usando il MSE come misura per il validation test error). Per creare questo grafico è stato diviso casualmente in due parti il data set.

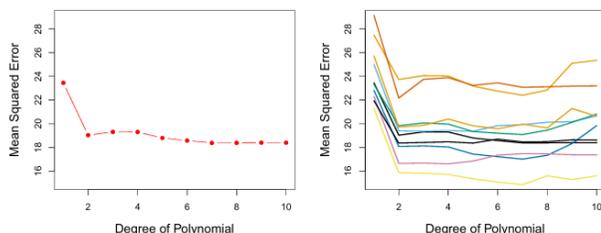


Figura 4.2: Validation set approach sul data set `Auto`.

Se effettuiamo questa divisione più volte otteniamo diverse stime per il test MSE. Nel grafico a destra notiamo 10 differenti curve di validation set MSE per il data set `Auto`, prodotte slittando casualmente le osservazioni dieci volte.

Notiamo che in tutte le curve i modelli con un termine quadratico hanno un validation set MSE minore rispetto a quelli con solo termini lineari. Inoltre notiamo che non otteniamo un maggiore vantaggio inserendo termini cubici o superiori nel modello. In ogni caso possiamo concludere che il fitting lineare non è adeguato per questi dati.

Il validation set approach quindi è molto semplice concettualmente ed è facile da implementare, ma ha due inconvenienti:

1. le stime del validation test error rate possono essere altamente variabili, a seconda di quali osservazioni ricadono nel validation set e quali nel training set (come si può notare anche dal grafico a destra della Figure 4.2);
2. con questo approccio inoltre solo un sottoinsieme di osservazioni (quelle incluse nel training set) vengono usate per fittare il modello.

4.1.2 Leave-One-Out Cross-Validation

la LOOCV cerca di risolvere questi inconvenienti. Questo metodo consiste sempre nel dividere il set di osservazioni in due parti, ma, invece di crearli di dimensione comparabile, una singola osservazione (x_1, y_1) è usata per il validation set e le restanti $\{(x_2, y_2), \dots, (x_n, y_n)\}$ formano il training set. Il metodo di statistical learning è fittato sulle $n-1$ osservazioni di training ed una predizione \hat{y} è fatta per l'osservazione esclusa, usando il valore x_1 .

Dato che (x_1, y_1) non è stata usata per fittare il modello, $MSE_1 = (y_1 - \hat{y}_1)^2$ fornisce una stima imparziale per il test error. MSE_1 comunque rimane una scarsa stima perché è altamente variabile, dato che è basata su una sola osservazione.

Possiamo quindi ripeterla n volte ottenendo diversi errori: MSE_1, \dots, MSE_n .

la stima LOOCV per il test MSE sarà la media di questi n errori di test:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^n MSE_i \quad (4.1)$$

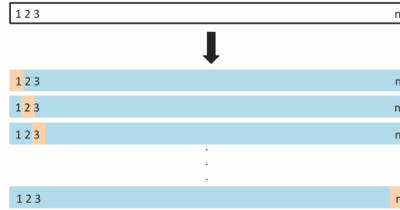


Figura 4.3: Esempio schematico del LOOCV. Un set di n data points viene diviso nel training set (in blu, contenente tutte le osservazioni tranne 1) ed il validation set (in beige, contenente la restante osservazione).

Il LOOCV ha come vantaggi: un minore bias e la non randomicità nella divisione training/validation set.

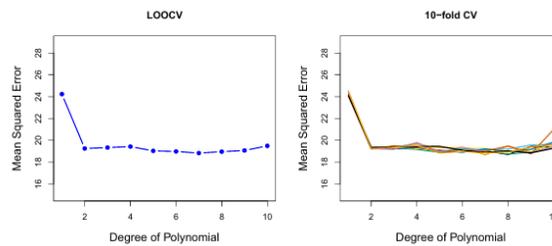


Figura 4.4: LOOCV sul data set Auto.

LOOCV potrebbe essere costoso da implementare, dato che il modello deve essere fittato n volte. Potrebbe essere molto lento da fittare inoltre, se n è grande.

Possiamo rendere il costo del LOOCV lo stesso di un singolo fit di modello:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{y}_i}{1 - h_i} \right)^2 \quad (4.2)$$

dove \hat{y}_i è l'i-esimo valore fittato dal least squares fit originale ed h_i è la leva (numero compreso tra $1/n$ ed 1 che riflette quanto ogni osservazione influenza il suo stesso fit).

4.1.3 K-Fold Cross-Validation

Questo approccio consiste nel dividere in maniera casuale il set di osservazioni in k gruppi (anche detti *fold*) di dimensione approssimativamente simile.

La stima del k-fold CV è effettuata come segue:

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^k MSE_i \quad (4.3)$$

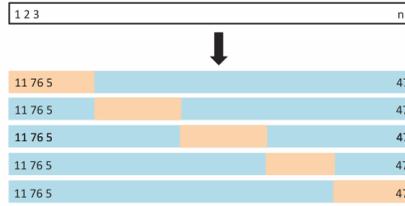


Figura 4.5: Esempio schematico del 5-fold.

Il LOOCV quindi è un caso particolare di k-fold nel quale impostiamo $k=n$. Solitamente il k-fold CV viene effettuato utilizzando $k=5$ o $k=10$. In Figura 4.6 sono stati plottati i valori stimati e veri dei test error rates. Il true test MSE è plottato in blu, la linea tratteggiata nera rappresenta il valore stimato con il LOOCV mentre la linea arancione rappresenta la stima mediante la 10-fold CV. In tutti e tre i grafici i due metodi di cross-validation stimano in maniera simile. Notiamo come ogni curva di CV riesce ad identificare il corretto livello di flessibilità (esso corrisponde al minor valore del test MSE, nei grafici li abbiamo segnati tramite delle croci).

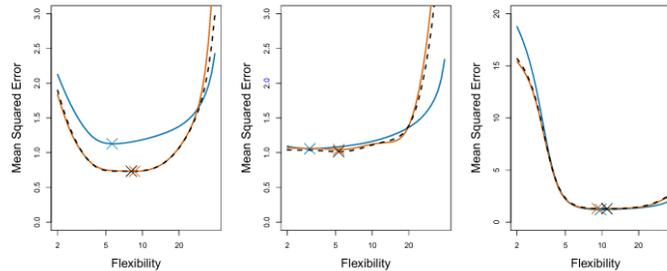


Figura 4.6: Stima e valore vero del test MSE.

4.1.4 Cross-Validation nei problemi di Classificazione

Quando la Y è qualitativa, invece di usare il MSE per quantificare il test error, usiamo il numero di osservazioni classificate in maniera errata,

In questo modo, il LOOCV per i classification problems diventa:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^n Err_i \quad (4.4)$$

dove $Err_i = I(y_i \neq \hat{y}_i)$. Analogamente vengono definiti anche il k-fold CV ed il validation set error rates.

4.2 Bootstrap

Il bootstrap può essere usato per quantificare l'incertezza associata ad una stima data o ad un metodo statistico.

Esso viene usato su un computer per emulare nuovi set di sample, in modo tale da stimare la variabilità di $\hat{\alpha}$ senza generare nuovi sample. Quindi tramite il bootstrap riusciamo ad ottenere data set distinti semplicemente campionando ripetitivamente le osservazioni dal data set originale, invece di ottenere un nuovo data set indipendente ogni volta.

In Figura 4.7 su un semplice data set, chiamato Z (contenente soltanto $n=3$ osservazioni) viene applicato questo approccio. Vengono, quindi, scelti in maniera casuale n osservazioni dal data set in maniera tale da produrre un bootstrap data set, Z^{*1} . Il sampling è effettuato per sostituzione, ciò vuol dire che la stessa osservazione può verificarsi più di una volta nel bootstrap data set.

Nel nostro caso, Z^{*1} contiene la terza osservazione due volte, la prima osservazione una volta ed ha zero istanze della seconda osservazione.

Quando un'osservazione è contenuta in Z^{*1} , anche i suoi valori di X ed Y sono inclusi.

Usiamo Z^{*1} per produrre una nuova stima bootstrap per α che chiamiamo $\hat{\alpha}^{*1}$. Questa procedura è ripetuta B volte in maniera tale da produrre B differenti bootstrap data set, $Z^{*1}, Z^{*2}, \dots, Z^{*B}$ e B corrispondenti stime $\hat{\alpha}^{*1}, \hat{\alpha}^{*2}, \dots, \hat{\alpha}^{*B}$.

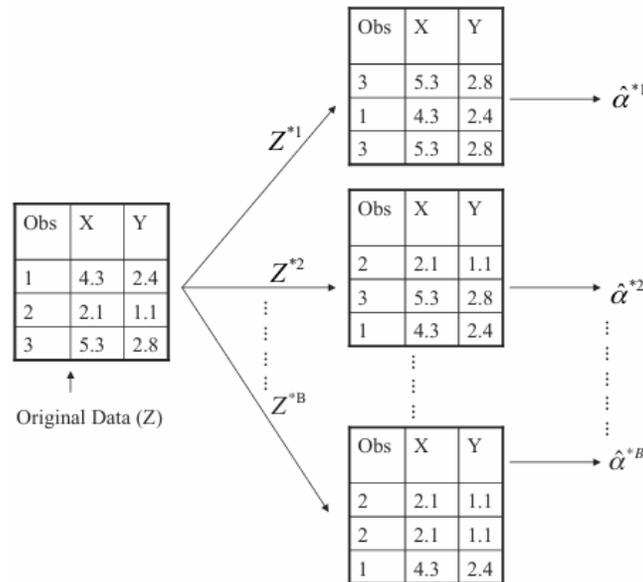


Figura 4.7: Esempio schematico del bootstrap approach su un piccolo sample, contenente $n=3$ osservazioni.

5 Linear Model Selection and Regularization

Il modello lineare standard:

$$Y = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p + \epsilon \quad (5.1)$$

nei modelli di regressione è comunemente usato per descrivere la relazione tra la risposta Y ed il set di variabili X_1, \dots, X_p . Invece di usare la procedura dei minimi quadrati, vedremo alcune procedure che producono migliori *Prediction Accuracy* e *Model Interpretability*.

Esistono molte metodi alternativi di fitting, invece di usare il least squares. Ecco tre importanti classi di metodi:

- *Subset Selection*: consiste nel identificare un subset di predittori che pensiamo possano essere collegati alla risposta. Successivamente, fittiamo un modello usando il least squares ma su un set di variabili ridotto.
- *Shrinkage*: consiste nel fittare un modello utilizzando tutti i suoi p predittori. Tuttavia, i coefficienti stimati sono rimpiccioliti a zero relativo rispetto alle stime dei minimi quadrati. Questo metodo è anche conosciuto come *regularization* e porta ad una riduzione della varianza.
- *Dimension Reduction*: consiste nel proiettare i predittori in un sottospazio M -dimensionale, dove $M < p$. Questo è effettuato calcolando M diverse combinazioni lineari delle variabili. Successivamente, queste M proiezioni sono usati come predittori per fittare il modello lineare di regressione per minimi quadrati.

5.1 Subset Selection Methods

5.1.1 Best Subset Selection

Fittiamo una least squares regression diversa per ogni possibile combinazione dei p predittori. Quindi, fittiamo p modelli che contengono esattamente un predittore, $\binom{p}{2} = p(p-1)/2$ modelli contenenti esattamente due predittori, e così via. Successivamente, guardiamo tutti i risultati e identifichiamo il modello migliore. Nell'Algoritmo 5.1 tramite lo Step 2 si identifica il modello migliore, riducendo il problema di doverlo selezionare tra 2^p diversi modelli, scegliendolo tra $p+1$ possibili modelli ogni iterazione. Bisogna fare attenzione nella scelta del modello perché un RSS basso ed un alto R^2 indicano un modello con un *training* error basso, e noi dobbiamo scegliere il più basso *test* error. Rischiamo di selezionare un modello che includa tutte le variabili, per questo motivo viene eseguito anche lo Step 3.

Nella Figura 5.1 notiamo un'applicazione del best subset selection. Sono mostrate l'RSS e l' R^2 per ogni modello in funzione del numero di variabili. Le curve rosse collegano i migliori modelli. Si può vedere come all'aumentare del numero di variabili l'RSS e l' R^2 migliorano.

Algoritmo 5.1 *Best subset selection*

- 1) Se \mathcal{M}_0 rappresenta il *null model* (contenente zero predittori). Questo prevede la media dei sample per ogni osservazione.
 - 2) Per $k=1,2,\dots,p$:
 - a) fitta tutti i $\binom{p}{k}$ modelli che contengono esattamente k predittori.
 - b) sceglie il migliore tra questi $\binom{p}{k}$ modelli, e chiamalo \mathcal{M}_k (la scelta avviene prendendo il più piccolo RSS o il maggiore R^2).
 - 3) Seleziona un singolo modello tra $\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_p$, utilizzando il prediction error su un set di validazione, C_p (AIC), BIC, o adjusted R^2 . Oppure usiamo il metodo di cross-validazione.
-

Ogni punto della figura corrisponde ad una least square regression utilizzando differenti subset dei 10 predittori del data set **Credit**.

La best subset selection ha delle limitazioni computazionali, perché il numero dei possibili modelli che devono essere considerati cresce rapidamente all'aumentare di p . Quindi risulta impossibile da calcolare per valori di $p \geq 40$ (anche utilizzando computer moderni e veloci).

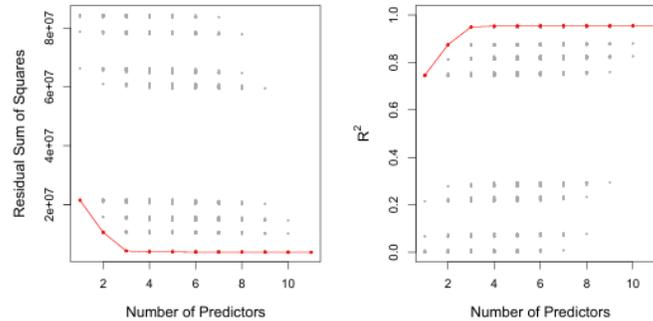


Figura 5.1: RSS ed R^2 in funzione del numero di predittori per il data set **Credit**.

5.1.2 Stepwise Selection

È un'alternativa al best subset selection.

Forward Stepwise Selection considera un numero di set del modello minore (e non tutti i 2^p possibili modelli come il best subset selection). Essa inizia con un modello contenente zero predittori, ed i predittori vengono aggiunti uno alla volta fino a raggiungere il numero di predittori del modello.

Nella forward stepwise selection viene fittato un modello nullo ed altri $p - k$ modelli (con $k=0, \dots, p-1$). Questo si traduce in un totale di $1 + \sum_{k=0}^{p-1} \binom{p-k}{1} = 1 +$

$p(p+1)/2$ modelli. Se consideriamo $p = 20$ notiamo una differenza sostanziale, la best subset selection richiederà un fitting di 1048576 modelli mentre la forward stepwise selection solamente 211 modelli.

Algoritmo 5.2 *Forward stepwise selection*

- 1) \mathcal{M}_0 rappresenta il *null model* (contenente zero predittori).
 - 2) Per $k=1,2,\dots,p-1$:
 - a) consideriamo tutti i $p - k$ modelli che fanno crescere i predittori in \mathcal{M}_k con un predittore addizionale.
 - b) sceglie il migliore tra questi $p - k$ modelli e lo chiama \mathcal{M}_{k+1} . Questo è scelto se ha il più basso RSS o il più alto R^2 .
 - 3) Seleziona un singolo modello tra $\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_p$, utilizzando il prediction error su un set di validazione, C_p (AIC), BIC, o adjusted R^2 . Oppure usiamo il metodo di cross-validazione.
-

Anche se la forward stepwise selection performa bene in pratica, non è garantito che riesca a trovare il miglior modello tra tutti i 2^p possibili. Infatti, supponendo un dataset di $p=3$ predittori, può capitare che il modello migliore considerando una sola variabile contenga X_1 e quello considerando due variabili contenga X_2 ed X_3 . Usando la forward stepwise selection il modello a due variabile conterrà X_1 ed un'altra variabile (e non corrisponde con il migliore).

Dalla Tabella 5.1 si nota meglio questo fenomeno. I modelli ad una, due e tre variabili sono identici utilizzando le due selection; il modello a quattro variabili per il best subset sostituisce cards al rating, mentre nella forward stepwise rimane invariato e si aggiunge semplicemente limit.

# Variables	Best subset	Forward stepwise
One	rating	rating
Two	rating, income	rating, income
Three	rating, income, student	rating, income, student
Four	cards, income, student, limit	rating, income, student, limit

Tabella 5.1: I primi quattro modelli selezionati con il best subset e la forward stepwise per il data set **Credit**.

Dalla Figura 5.1 notiamo che non c'è molta differenza tra i metodi a tre e quattro variabili in termini di RSS, quindi entrambi i modelli a quattro variabili sono accettabili.

Backward Stepwise Selection considera un modello full least squares, contenente tutti i p predittori che vengono rimossi iterativamente uno alla volta, scegliendo il meno utile.

Algoritmo 5.3 *Backward stepwise selection*

- 1) Se \mathcal{M}_p rappresenta il *full model* (contenente tutti i p predittori).
 - 2) Per $k=p, p-1, \dots, 1$:
 - a) considera tutti i k modelli che contengono tutti i predittori meno uno in \mathcal{M}_k (per un totale di $k-1$ predittori).
 - b) scegli il migliore tra questi k modelli, e chiamalo \mathcal{M}_{k-1} (la scelta avviene prendendo il più piccolo RSS o il maggiore R^2).
 - 3) Seleziona un singolo modello tra $\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_p$, utilizzando il prediction error su un set di validazione, C_p (AIC), BIC, o adjusted R^2 . Oppure usiamo il metodo di cross-validazione.
-

Similmente alla forward stepwise selection, la ricerca avviene solamente tra $1 + p(p+1)/2$ modelli, quindi può essere utilizzato nel caso in cui p fosse troppo grande per applicare la best subset selection.

La backward stepwise selection può essere applicata però solo se il numero di samples n è maggiore del numero di variabili p . La forward stepwise, invece, può essere usata anche se $n < p$.

5.1.3 Scegliere il Modello Ottimale

Per conoscere quale dei tre seleziona il modello migliore basandosi sul test error, dobbiamo prima di tutto stimare quest'ultimo. Conosciamo due approcci:

1. possiamo stimare indirettamente il test error facendo degli *aggiustamenti* al training error in modo tale da raggiungere il bias che porta all'overfitting
2. possiamo stimare direttamente il test error utilizzando un validation set o una cross-validation.

Le tecniche utilizzate per *aggiustare* il training error per la dimensione del modello sono: C_p , AIC (*Akaike Information Criterion*), BIC (*Bayesian Information Criterion*) e *adjusted R^2* .

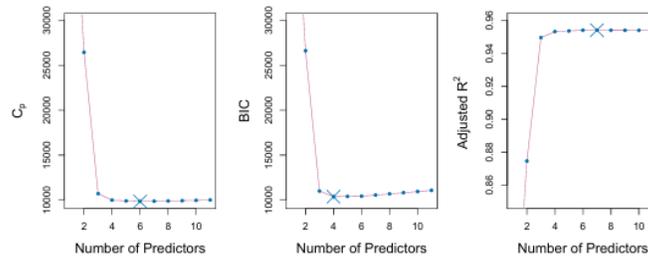


Figura 5.2: C_p , BIC ed adjusted R^2 per il migliore modello di ogni dimensione prodotto dalla best subset selection sul data set **Credit**.

Per un least squares model contenente d predittori, il C_p stimato del test MSE si calcola come:

$$C_p = \frac{1}{n}(RSS - 2d\hat{\sigma}^2) \quad (5.2)$$

con $\hat{\sigma}^2$ una stima della varianza del errore ϵ associato con ogni risposta misurata (tipicamente utilizzano il modello contenente tutti i predittori). Praticamente il C_p introduce una penalità di $2d\hat{\sigma}^2$ al training RSS in modo tale da adattarlo, sapendo che il training error tende a sottostimare il test error. Chiaramente, questa penalità incrementa all'aumentare del numero di predittori nel modello.

Il C_p assume valore piccolo per i modelli che hanno un basso test error, quindi sceglieremo come modello migliore quello con il C_p minore.

Dalla Figura 5.2 il C_p seleziona il modello a sei variabili (contenente i predittori `income`, `limit`, `rating`, `cards`, `age`, `student`).

$$AIC = -2 \cdot \log(L) + 2d = \frac{1}{n}(RSS - 2d\hat{\sigma}^2) \quad (5.3)$$

Con L indichiamo il valore massimo della likelihood function per il modello stimato. Nel caso di modelli lineari standard con errore Gaussiano, l'AIC assume la seconda equazione identica a quella del C_p .

Per il least squares model con d predittori il BIC è dato da:

$$BIC = \frac{1}{n}(RSS + \log(n)d\hat{\sigma}^2) \quad (5.4)$$

con n numero di osservazioni. Anche il BIC (come il C_p e l'AIC) assume valore basso per i modelli che hanno un test error minore. Esso generalmente applica una penalità maggiore rispetto al C_p , infatti notiamo dalla Figura 5.2 che sceglie un modello contenente solamente quattro predittori (`income`, `limit`, `cards`, `student`).

$$Adjusted \ R^2 = 1 - \frac{RSS/(n-d-1)}{TSS/(n-1)} \quad (5.5)$$

Al contrario del C_p , AIC e BIC, un elevato valore di adjusted R^2 indica un modello con un basso test error. Nella Figura 5.2 notiamo come viene scelto un modello a sette variabili (dove viene aggiunto `own` al modello selezionato dall'AIC e dal C_p).

In alternativa possiamo direttamente stimare il test error usando i metodi di cross-validation o di validation set.

I validation errors sono stati calcolati selezionando casualmente tre quarti delle osservazioni come training set e le rimanenti come validation set. Gli errori di cross-validation invece sono stati calcolati usando $k=10$ folds. Dalla Figura 5.3 notiamo come entrambi i modelli suggeriscono modelli a sei variabili.

Potremmo scegliere il modelli usando la *one-standard-error rule*, calcolando inizialmente lo standard error del test MSE stimato per ciascuna dimensione del modello e successivamente selezionare il modello con il minore numero di predittori tra quelli aventi test error entro un errore standard dal punto più basso della curva.

Utilizzando questo approccio il validation set e la cross validation avrebbero scelto un modello a tre variabili.

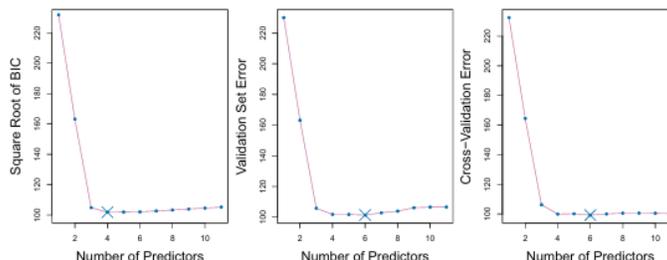


Figura 5.3: BIC, validation set error, cross-validation error per il data set Credit.

5.2 Shrinkage Methods

5.2.1 Ridge Regression

Sappiamo che il least squares fit stima i coefficienti β_0, \dots, β_p considerando i valori che minimizzano:

$$RSS = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2$$

Nella ridge regression vengono stimati i coefficienti $\hat{\beta}^R$, come valori che minimizzano:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = RSS + \lambda \sum_{j=1}^p \beta_j^2 \quad (5.6)$$

Il secondo termine è chiamato *shrinkage penalty* ed è piccolo quando i coefficienti sono vicini a zero. Il *tuning parameter* (λ) serve a regolare l'impatto dei due termini sulla stima dei coefficienti di regressione. Quando $\lambda = 0$ il termine di penalità non ha effetto e la ridge regression produrrà la stima dei minimi quadrati. Quando $\lambda \rightarrow \infty$ l'impatto della shrinkage penalty cresce ed le stime dei coefficienti della ridge regression tendono a zero.

La ridge regression produce un set differente di coefficienti stimati per ogni valore di lambda (a differenza della least squares), quindi è necessario scegliere un buon valore di λ .

Dalla formula notiamo che la shrinkage penalty è applicata solo a β_1, \dots, β_p e non all'intercetta β_0 . Se assumiamo che le variabili siano state centrate in maniera tale da avere media zero prima di effettuare la ridge regression, allora l'intercetta stimata avrà la forma: $\hat{\beta}_0 = \bar{y} = \sum_{i=1}^n y_i / n$.

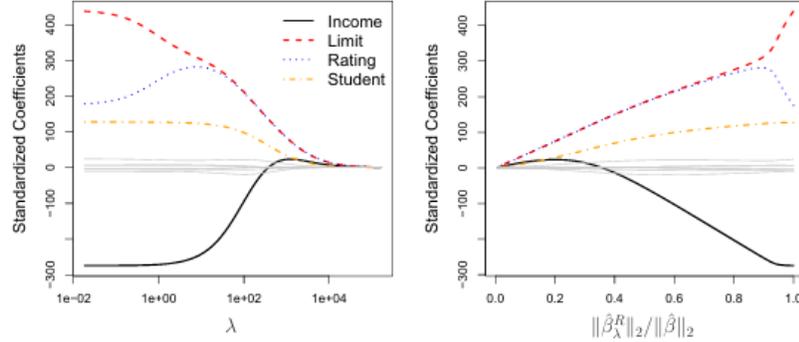


Figura 5.4: Stime dei coefficienti utilizzando la ridge regression per il data set Credit.

Nel grafico a sinistra della Figura 5.4 ogni curva corrisponde alla stima del coefficiente utilizzando la ridge regression per una delle dieci variabili plottate in funzione di λ (ad esempio, la linea continua nera rappresenta la stima con ridge regression per il coefficiente **income**). Si può vedere come quando lambda è uguale a zero la stima dei coefficienti è uguale a quella ottenuta utilizzando il metodo least squares, ma appena lambda aumenta i coefficienti tendono a zero.

Il grafico a destra nella Figura 5.4 rappresenta gli stessi coefficienti di quello a sinistra ma in funzione di $\|\hat{\beta}_\lambda^2\|_2 / \|\hat{\beta}\|_2$, dove $\hat{\beta}$ rappresenta il vettore dei coefficienti stimati usando i minimi quadrati e $\|\beta\|_2 = \sqrt{\sum_{j=1}^p \beta_j^2}$ è la *norma* ℓ_2 , la quale misura la distanza di β da zero. All'aumentare di λ , la $\|\hat{\beta}_\lambda^2\|_2$ decresce. Possiamo pensare l'asse x, in questo caso, come la quantità di rimpicciolimento verso lo zero (un valore basso indica che i coefficienti stimati sono stati rimpiccioliti molto vicino allo zero). La quantità $\|\hat{\beta}_\lambda^2\|_2 / \|\hat{\beta}\|_2$ varia da 1 (quando lambda è pari a zero) a 0 (quando lambda è infinito).

I coefficienti stimati con il metodo dei minimi quadrati sono *scale equivalent*, ovvero non importa come è scalato il j-esimo predittore, $X_j \hat{\beta}_j$ rimmarrà sempre lo stesso (moltiplicare X_j per una costante c porta a scalare i coefficienti stimati ai minimi quadrati di un fattore $1/c$). I coefficienti stimati con la ridge regression invece possono cambiare *sostanzialmente* se un predittore viene moltiplicato per una costante. Quindi prima di applicare la ridge regression è necessario *standardizzare i predittori*, tramite la formula:

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n} \sum_{i=1}^n (x_{ij} - \bar{x}_j)^2}} \quad (5.7)$$

in questo modo avranno tutti la stessa scala.

Tramite la ridge regression si riscontrano vantaggi nel trade-off tra bias e varianza (rispetto alla least squares). All'aumentare di λ la flessibilità della

ridge regression diminuisce, portando ad una diminuzione della varianza ed un aumento del bias.

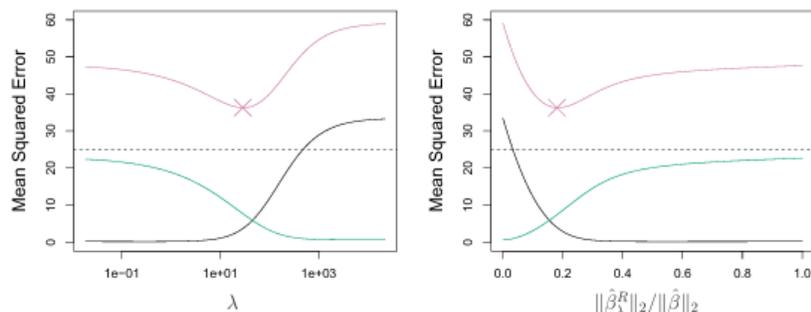


Figura 5.5: Squared bias (nero), varianza (verde) e test MSE (rosa) per la ridge regression. La linea orizzontale tratteggiata indica il minimo MSE possibile, la X in rosa indica il modello di ridge regression che ha MSE minore.

Nel grafico a sinistra della Figure 5.5 notiamo questo fenomeno in un data set contenente $p = 45$ predittori e $n = 50$ osservazioni. Per $\lambda = 0$ si ha varianza alta e bias nullo, all'aumentare di λ la varianza si riduce ed un incremento del bias. Nel grafico a destra le stesse curve sono mostrate in funzione della norma ℓ_2 dei coefficienti stimati con la ridge regression diviso la norma ℓ_2 dei coefficienti stimati con il least squares.

La ridge regression funziona meglio in situazioni in cui le stime con i minimi quadrati hanno varianza elevata.

5.2.2 The Lasso

Nella ridge regression però la penalità $\lambda \sum \beta_j^2$ riduce tutti i coefficienti verso lo zero, ma nessuno sarà mai uguale a zero prima di $\lambda = \infty$.

Questo non è un problema per la precisione delle previsioni ma può creare difficoltà nell'interpretare il modello, specialmente se il numero di variabili è elevato. Se consideriamo il data set **Credit**, sappiamo che il miglior modello include le variabili **income**, **limit**, **rating**, **student**. Volendo utilizzare solo quest'ultimi, non possiamo applicare la ridge regression perché il modello generato avrà comunque tutti e dieci i predittori del data set. Tramite il *lasso* si risolve questo problema, i coefficienti $\hat{\beta}_\lambda^L$ minimizzano la quantità:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| = RSS + \lambda \sum_{j=1}^p |\beta_j| \quad (5.8)$$

A differenza della ridge regression, il lasso utilizza la norma ℓ_1 che è data da $\|\beta\| = \sum |\beta_j|$. Quindi quando lambda è sufficientemente grande alcuni coefficienti stimati potrebbero tendere a zero.

Dalla Figura 5.6 notiamo come quando $\lambda = 0$ si ha il least squares fit, per λ sufficientemente grande si ha il modello nullo. Dal grafico a destra possiamo notare che inizialmente il lasso genera un modello contenente solo il **rating**, successivamente sono aggiunti **student** e **limit**, poco dopo viene aggiunto anche **income**. Al contrario la ridge regression includeva subito tutte le variabili insieme.

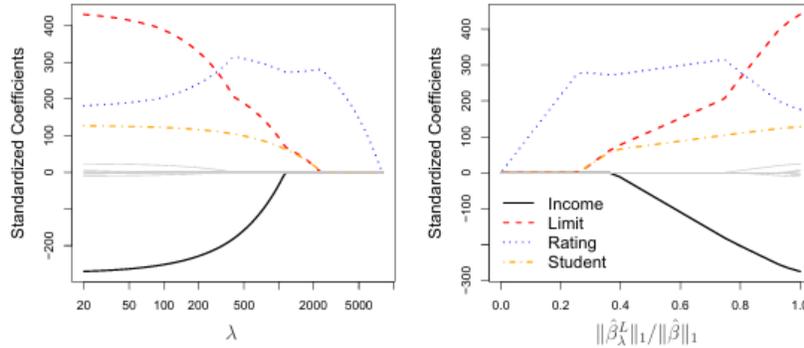


Figura 5.6: Stime dei coefficienti utilizzando il lasso per il data set Credit.

Dalla Figura 5.7 possiamo vedere come se s è sufficientemente grande, allora le regioni di vincolo conteranno $\hat{\beta}$ (quindi le stime con la ridge regression ed il lasso saranno uguali a quelle con i least squares). Tutti i punti su una particolare ellisse hanno lo stesso valore di RSS, man mano che le ellissi si allontanano dal coefficiente stimato con i minimi quadrati, la RSS aumenta.

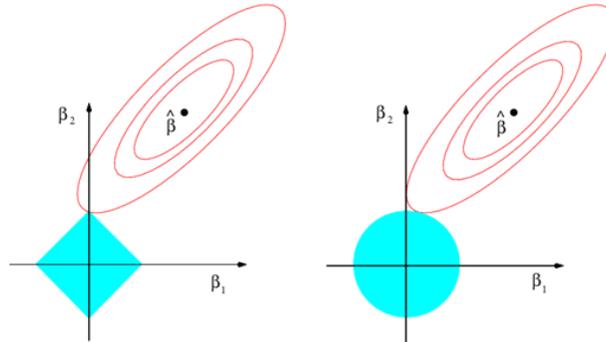


Figura 5.7: Contorni degli errori e delle funzioni vincolari per il lasso (a sinistra) e la ridge regression (a destra). Le aree blu sono le regioni di vincolo, considerando $p = 2$, $(|\beta_1| + |\beta_2| \leq s$ e $\beta_1^2 + \beta_2^2 \leq s)$, mentre le ellissi rosse sono i contorni del RSS.

Il lasso quindi ha chiaramente il vantaggio di produrre modelli più interpretabili (che includono meno variabili), rispetto alla ridge regression, però per sapere quale metodo porta ad ottenere una migliore accuratezza nelle predizioni bisognerebbe sapere il numero di predittori utilizzati.

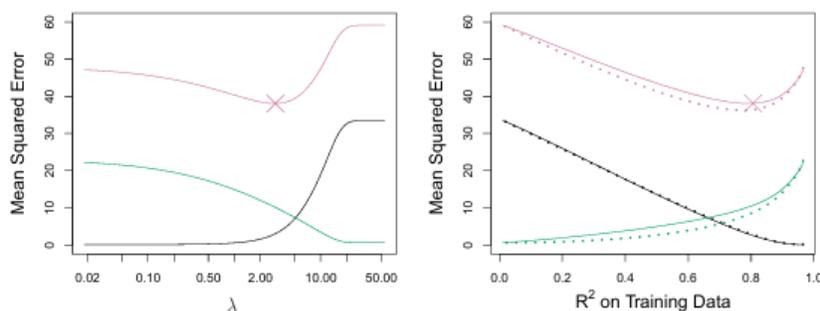


Figura 5.8: Squared bias (nero), varianza (verde) e test MSE (rosa) per il lasso. La linea orizzontale tratteggiata indica il minimo MSE possibile, la X in rosa indica il modello di lasso che ha MSE minore.

ella Figura 5.8 sono stati utilizzati tutti i predittori (nessuno dei coefficienti vale zero) e possiamo vedere dal grafico a destra che la ridge regression (linee punteggiate) ha una minore varianza e di conseguenza un minor MSE rispetto al lasso (linee continue).

Nella Figura 5.9 invece la risposta è in funzione di soli due predittori e possiamo notare dal grafico a destra come in questo caso il lasso outperforma la ridge regression in termini di bias, varianza ed MSE.

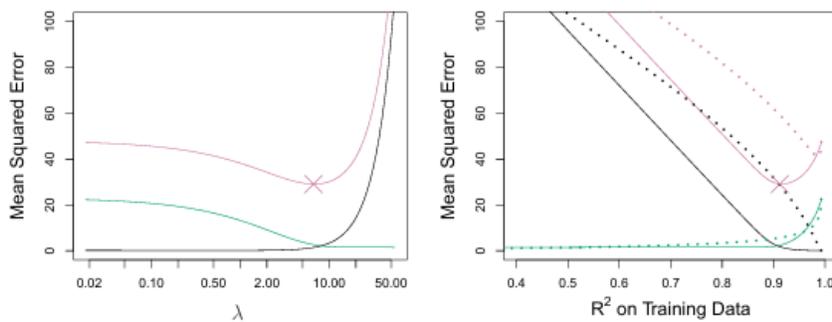


Figura 5.9: Squared bias (nero), varianza (verde) e test MSE (rosa) per il lasso. La linea orizzontale tratteggiata indica il minimo MSE possibile, la X in rosa indica il modello di lasso che ha MSE minore.

5.2.3 Selezionare i Parametri di Tuning

Scegliamo una griglia di λ valori e calcoliamo l'errore di cross-validazione per ogni valore di λ , sceglieremo come parametro di tuning quello che ha un errore di cross-validazione minore. Successivamente si re-fitta il modello usando tutte le osservazioni disponibili ed il valore scelto di λ .

Nella Figura 5.10 la linea verticale tratteggiata indica il valore selezionato per λ . In questo caso notiamo che è relativamente basso, stando ad indicare che il fit ottimale include solo una piccola parte di shrinkage, quindi potremo semplicemente applicare la least square.

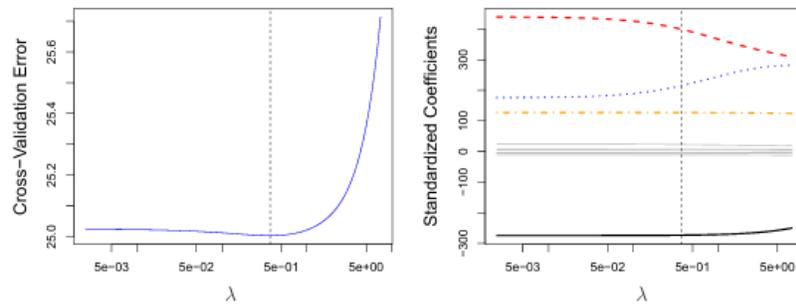


Figura 5.10: Scelta di λ utilizzando la LOOCV sulla ridge regression per il data set **Credit**.

Nel grafico a destra della Figura 5.11 notiamo l'errore di cross validazione mentre a destra sono mostrate le stime dei coefficienti. Le linee verticali tratteggiate indicano il punto in cui l'errore di cross-validazione è minore. Le due linee colorate nella figura a destra rappresentano gli unici due predittori collegati alla risposta, in grigio sono rappresentati quelli non correlati (questi vengono spesso chiamate rispettivamente variabili di *segnale* e di *rumore*).

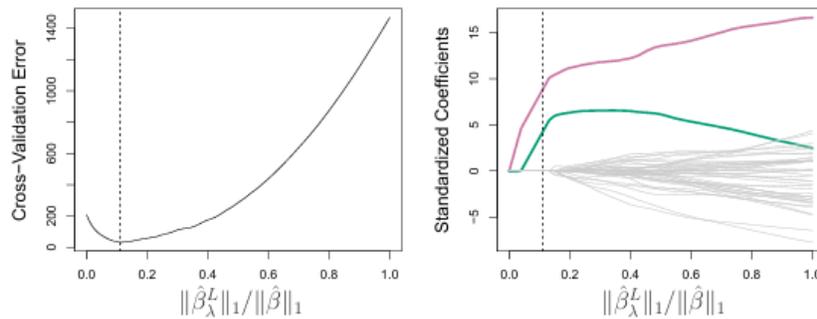


Figura 5.11: 10-folds cross-validation applicata sul lasso utilizzando i dati simulati in Figura 5.9.

5.3 Dimension Reduction Methods

In questo approccio *trasformiamo* i predittori e poi fittiamo i minimi quadrati usando quest'ultimi.

Se Z_1, Z_2, \dots, Z_M rappresentano $M < p$ combinazioni lineari dei nostri predittori originali per alcune costanti $\phi_{1m}, \phi_{2m}, \dots, \phi_{pm}$, con $m=1, \dots, M$:

$$Z_m = \sum_{j=1}^p \phi_{jm} X_j \quad (5.9)$$

allora possiamo fittare il modello di regressione:

$$y_i = \theta_0 + \sum_{m=1}^M \theta_m z_{im} + \epsilon_i \quad (5.10)$$

per $i=1, \dots, n$; usando i minimi quadrati. I coefficienti di regressione in questo caso sono dati da $\theta_0, \theta_1, \dots, \theta_M$.

Viene chiamata *dimension reduction* perché il problema si riduce dallo stimare $p+1$ coefficienti a $M+1$ coefficienti con $M < p$. I coefficienti quindi sono limitati, ed assumono la seguente forma:

$$\beta_j = \sum_{m=1}^M \theta_m \phi_{jm} \quad (5.11)$$

per $M \ll p$ la varianza si riduce significativamente, se $M = p$ non cambia nulla.

5.3.1 Principal Components Regression

PCA (*Principal Components Analysis*) è una tecnica utilizzata per ridurre la dimensione $n \times p$ di una matrice data \mathbf{X} .

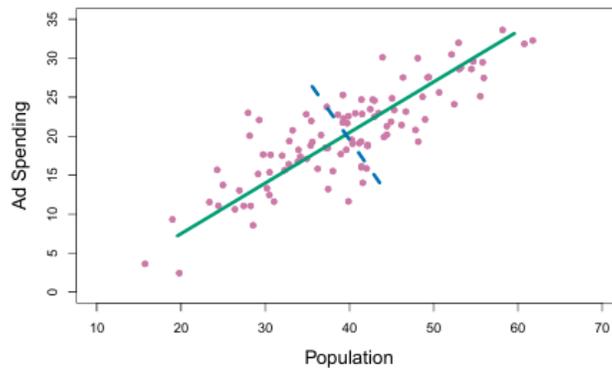


Figura 5.12: La *population size* (pop) e l'*ad spending* (ad) per 100 differenti città è rappresentato con i pallini rosa. La linea verde continua indica la prima componente principale e la linea blu tratteggiata la seconda componente principale.

La direzione della prima componente principale è quella per cui si ha maggiore variabilità dei dati, cioè se proiettiamo 100 osservazioni su questa linea (come mostrato nella Figura 5.13 a sinistra) allora le osservazioni risultanti da questa proiezione avranno la maggior varianza possibile. Proiettare un punto su una linea corrisponde a trovare la posizione sulla linea che è più vicina al punto.

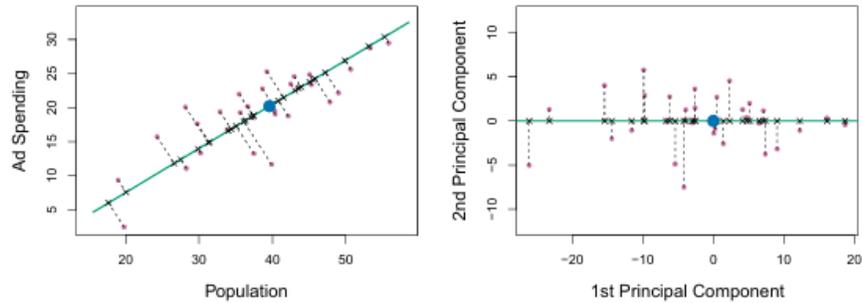


Figura 5.13: Un subset dell’advertising data. Il punto in blu è $(\overline{pop}, \overline{ad})$ ovvero il pop medio e l’ad medio rispetto a tutti i valori.

I valori $z_{11}, z_{21}, \dots, z_{n1}$ sono detti *principal component scores* e li possiamo vedere nel grafico a destra della Figura 5.13.

Generalmente, si potrebbero costruire p componenti principali distinte. La seconda componente principale Z_2 è una combinazione lineare delle variabili non correlate a Z_1 ed ha la varianza maggiore. Dalla Figura 5.12 come questa condizione di avere zero correlazione tra Z_1 e Z_2 si traduce nel fatto che queste due direzioni risultano *perpendicolari* tra loro.

La Figura 5.15 suggerisce che è necessaria solamente la prima componente principale per rappresentare accuratamente il pop ed i budgets ad.

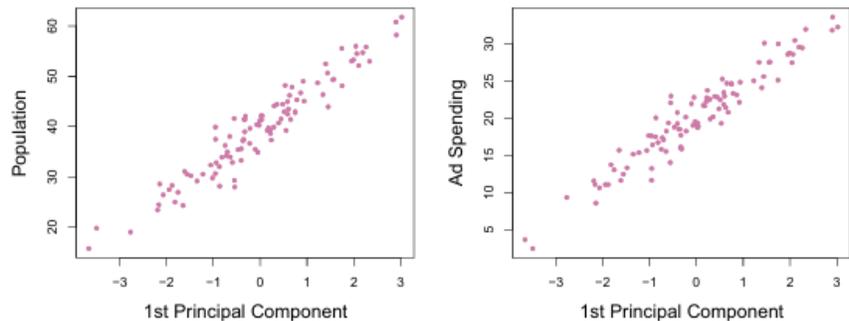


Figura 5.14: Plots dei first principal component scores z_{i1} in relazione a pop e ad. Notiamo una forte relazione.

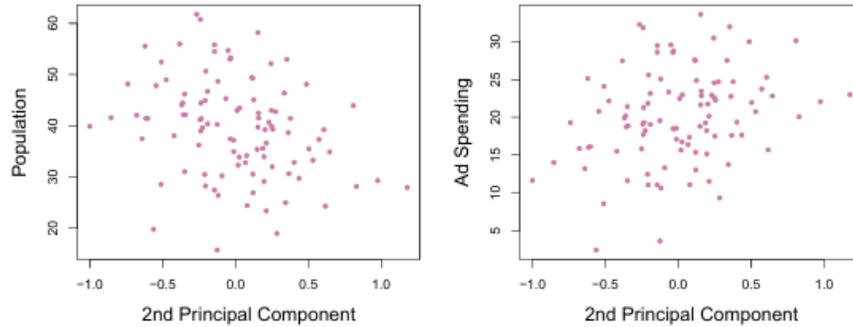


Figura 5.15: Plots dei first principal component scores z_{i2} in relazione a pop e ad. Notiamo una relazione debole.

La PCR (*Principal Components Regression*) consiste nel costruire le prime M componenti principali Z_1, \dots, Z_M ed usare queste come predittori nel modello di regressione lineare fittato usando i minimi quadrati.

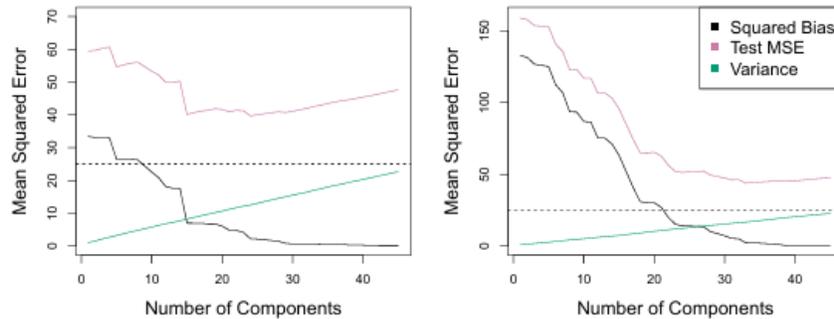


Figura 5.16: PCR applicata ai dati simulati della Figura 5.8 (sinistra) e 5.9 (destra).

Ricordiamo che per generare questi dati sono stati usati $n = 50$ osservazioni e $p = 45$ predittori. Notiamo che la PCR non performa bene quanto la ridge regression o il lasso; questo perché i dati sono stati generati in maniera tale che molte componenti principali siano necessarie per modellare adeguatamente la risposta. Il PCR performa meglio nei casi in cui poche componenti principali sono sufficienti a capire la relazione con la risposta. Solitamente il numero di componenti principali (M) è scelto applicando la cross-validazione.

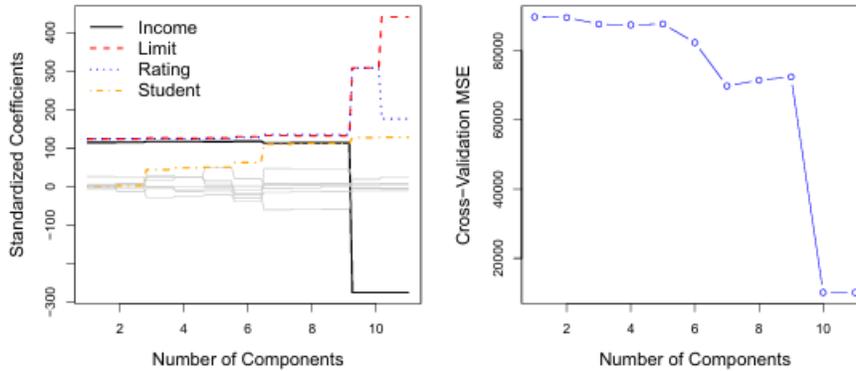


Figura 5.17: PCR applicata ad data set **Credit**

Il grafico a destra nella Figura 5.17 mostra l'errore di cross-validazione ottenuto in funzione di M . Questo è basso per $M = 10$, che corrisponde quasi ad una nessuna riduzione ($M = 11$ è equivalente a performare la classica least squares).

5.3.2 Partial Least Squares

Con la PCR non siamo sicuri che la direzione che meglio esprime i predittori sia la stessa per predire la risposta (perché questa è identificata in maniera *unsupervised*). Tramite la PLS (*Partial Least Squares*), un'alternativa *supervised*, si fa uso della risposta Y per identificare il nuovo set di features Z_1, \dots, Z_M .

Dopo aver standardizzato i p predittori, la PLS calcola la prima direzione Z_1 settando ogni ϕ_{j1} nell'equazione (5.9) pari al coefficiente della regressione lineare semplice di Y in X_j .

Quindi, nel calcolo di $Z_1 = \sum_{j=1}^p \phi_{j1} X_j$, la PLS dà maggiore peso alle variabili che sono maggiormente correlate alla risposta.

6 Tree-Based Methods

6.1 Regression Trees

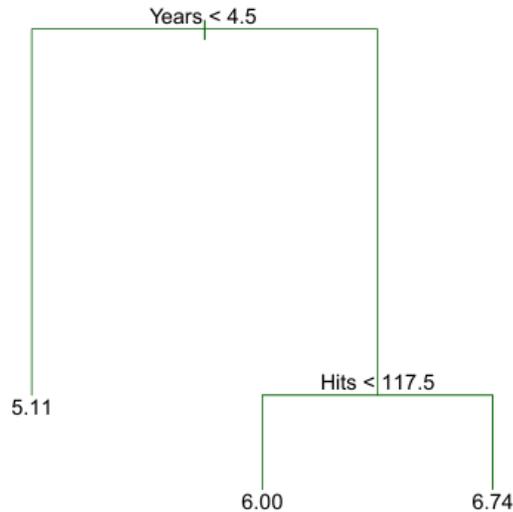


Figura 6.1: Un regression tree per predire il log salary di un baseball player per il data set `Hitters`.

Possiamo predire il `Salary` di un giocatore di baseball basandoci su `Years` (il numero di anni in cui ha giocato in una major leagues) e `Hits` (il numero di hits effettuate lo scorso anno). Per iniziare rimuoviamo le osservazioni che non includono valori di `Salary` e log-trasformiamo il `Salary` in maniera tale che la sua distribuzione assuma una forma a campana (sapendo che il `Salary` è misurato in milioni di dollari).

Nella Figura 6.1 è mostrato un regression tree per fittare questi dati. Esso consiste in una serie di regole di divisione, partendo dall'alto dell'albero. La prima divisione assegna un osservazione avente $Years < 4.5$ al ramo di sinistra, quindi, il salario previsto per questi giocatori sarà dato dalla media dei valori di risposta per il giocatori nel data set aventi $Years < 4.5$. Invece, i giocatori aventi $Years \geq 4.5$ sono assegnati al ramo di destra e suddivisi per i numero di `Hits`. Complessivamente, l'albero suddivide il giocatore in tre regioni di spazio:

$$R_1 = \{X | Years < 4.5\},$$

$$R_2 = \{X | Years \geq 4.5, Hits < 117.5\}, \quad R_3 = \{X | Years \geq 4.5, Hits \geq 117.5\}.$$

I salari previsti per questi tre gruppi sono rispettivamente:

$$1000 \times e^{5.107} = 165174\$, \quad 1000 \times e^{5.999} = 402834\$, \quad 1000 \times e^{6.740} = 845346\$.$$

Le regioni R_1, R_2, R_3 sono chiamate *nodi terminali* o *foglie*, dell'albero. I punti in cui un predictor space è diviso sono chiamati *nodi interni*. Invece, i segmenti dell'albero che connettono i nodi vengono detti *rami*.

L'albero in Figura 6.1 può essere interpretato nel seguente modo: Years è il fattore più importante per determinare il Salary, giocatori con meno esperienza guadagnano di meno rispetto a quelli professionisti. Se un giocatore è meno esperto il numero di hits effettuate l'anno precedente non è caratterizzante nel guadagno. Per i giocatori che sono stati in leagues importanti per 5 o più anni, invece, il numero di hits influenza il salario. Ovviamente l'albero in questa figura è un over-semplificazione della relazione tra Hits, Years e Salary; ma ha una facile interpretazione ed una bella rappresentazione grafica.

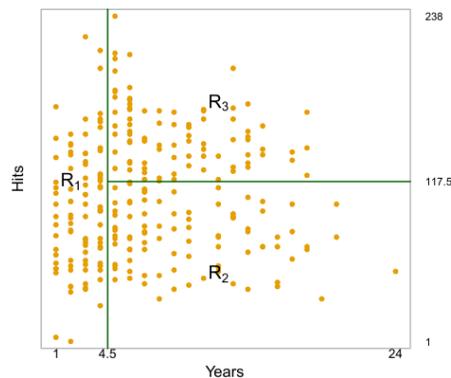


Figura 6.2: La regione di partizione dell'albero per il data set *Hitters* con il regression tree illustrato in Figura 5.1.

Prediction via Stratification of the Feature Space

Per costruire un regression tree principalmente seguiamo due steps:

1. dividiamo lo spazio dei predittori (il set dei possibili valori assunti da X_1, \dots, X_p) in J regioni distinte R_1, \dots, R_j .
2. per ogni osservazione che ricade nella regione R_j facciamo le stesse predizioni (che sono la media dei valori della risposta per le osservazioni di training in R_j).

Supponiamo che nello Step 1 otteniamo due regioni R_1, R_2 e che la media delle osservazioni di training nella prima regione è 10, mentre nella seconda 20.

Allora per un osservazione $X = x$, se $x \in R_1$ prevediamo un valore 10, se $x \in R_2$ prevediamo un valore 20.

In teoria le regioni possono avere qualsiasi forma, ma noi scegliamo di dividere lo spazio dei predittori in rettangoli multi-dimensionali (anche detti *boxes*) per

semplicità. L'obiettivo è quello di trovare dei boxes R_1, \dots, R_J che minimizzano la RSS data da:

$$\sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \quad (6.1)$$

Non è computazionalmente possibile considerare tutte le possibili partizioni dello spazio di feature in J boxes. Per questa ragione, eseguiamo un approccio *top-down, greedy* conosciuto come *recursive binary splitting*.

È *top-down* perché inizia dal sopra dell'albero e successivamente divide lo spazio dei predittori. È *greedy* perché ad ogni step del processo di costruzione dell'albero, la migliore divisione è fatta in quello step.

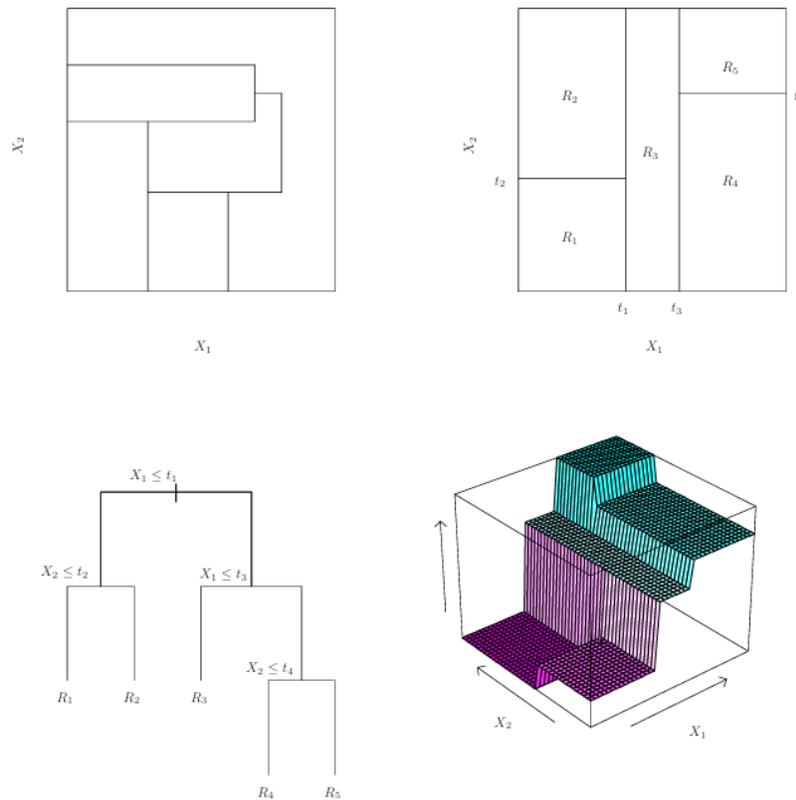


Figura 6.3: In alto a sinistra: una partizione bi-dimensionale di un feature space che non può uscire dal recursive binary splitting. In alto a destra: l'output di un recursive binary splitting su un esempio bi-dimensionale. In basso a sinistra: un albero che corrisponde alla partizione del pannello in alto a destra. In basso a destra: Un *perspective plot* del *prediction surface* corrispondente a quell'albero.

Tree Pruning

Con il processo descritto fin'ora però è possibile overfitting dei dati ed un albero troppo complesso. Un albero più piccolo, con meno divisioni potrebbe portare ad ottenere una varianza più bassa ed una migliore interpretazione al costo di un po' di bias.

Un'alternativa è quella di creare un albero grande T_0 e poi ridurlo in modo tale da ottenere un *subtree* che porti ad un minor test error rate. Per stimare quest'ultimo senza considerare ogni subtree possibile, ma considerando una sequenza di alberi indicizzati da un tuning parameter (non-negativo) α , utilizziamo il *cost complexity pruning* (anche detto *weakest link pruning*).

Ad ogni valore di α corrisponde un sotto albero $T \subset T_0$ che rende minore la quantità:

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \quad (6.2)$$

Dove $|T|$ indica il numero di nodi terminali dell'albero T , R_m è il rettangolo (che è il subset dello spazio dei predittori) corrispondente all' m -esimo nodo terminale ed \hat{y}_{R_m} è la risposta predetta associata con R_m (ovvero è la media delle osservazioni di training in R_m).

Il parametro di tuning α controlla il trade-off tra la complessità ed il fit dei dati di training. Quando $\alpha = 0$ il subtree T sarà uguale a T_0 , all'aumentare di α i rami vengono ridotti.

Algoritmo 6.1 *Building a Regression Tree*

- 1) Usare la recursive binary splitting per creare un grande albero sui dati di training, fermandosi solo quando ogni nodo terminale ha meno di un numero minimo di osservazioni.
 - 2) Applicare il cost complexity pruning all'albero, in maniera tale da ottenere una sequenza di subset in funzione di α .
 - 3) Usare la cross-validazione per scegliere α . Quindi per $k = 1, \dots, K$:
 - a) ripeti gli Step 1 e 2 su tutto il training data al meno del k -esimo fold.
 - b) valutare il mean squared prediction error sui dati del k -esimo fold lasciato fuori, in funzione di α .Fare la media dei risultati per ogni valore di α e scegliere il valore che minimizza l'errore medio.
 - 4) Ritorna il subtree dallo Step 2 che corrisponde al valore scelto di α .
-

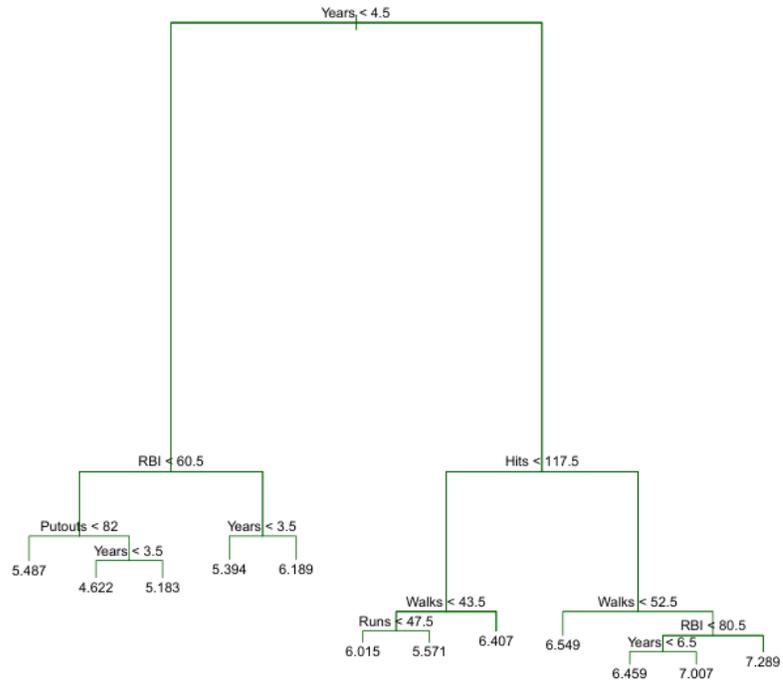


Figura 6.4: Albero di regressione *unpruned* risultante dalla divisione *top-down greedy* sui dati di training per il data set **Hitters**.

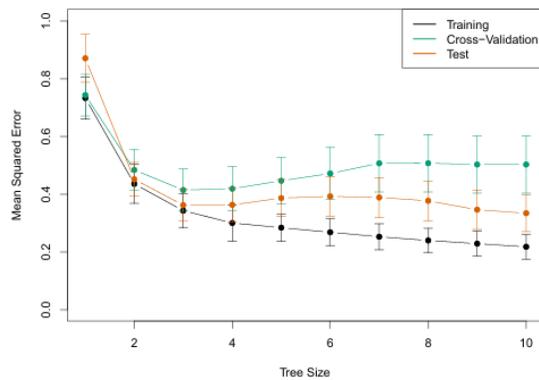


Figura 6.5: Trainig, Cross-Validation e Test MSE sono mostrati in funzione del numero di nodi terminali nell'albero ridotto per il data set **Hitters**. Sono mostrate anche le standard error bars attorno all'errore stimato. Il minimo errore di cross-validazione si ha in un albero di dimensione tre.

Le Figure 6.4 e 6.5 mostrano i risultati del fitting e *pruning* di un regression tree per il data set **Hitters**, usando nove delle features. Inizialmente dividiamo casualmente il data set, in 132 osservazioni di training e 131 di test. Successivamente, costruiamo un albero di regressione sui dati di training e variamo α per creare i subtree con un numero diverso di nodi terminali. Infine, performiamo la 6-fold cross-validation per stimare il MSE di cross-validazione degli alberi in funzione di α (abbiamo scelto la 6-fold perché 132 è un multiplo esatto di sei).

6.2 Classification Trees

Simili ai regression tree, con la differenza che viene predetta una risposta qualitativa invece che una quantitativa. Per i classification trees prevediamo che ogni osservazione appartenga alla *most commonly occurring class* di training. Nell'interpretare gli alberi di classificazione siamo spesso interessati anche alla proporzioni della classe in cui ricadono le osservazioni di training. Anche per i classification tree usiamo la recursive binary splitting per costruire gli alberi.

Però, nei problemi di classificazione non può essere usato la RSS come criterio per effettuare gli split, in alternativa usiamo il *classification error rate* (che è la frazione di osservazioni di training nella regione che non appartengono alla classe più comune):

$$E = 1 - \max_k(\hat{p}_{mk}) \quad (6.3)$$

dove \hat{p}_{mk} è la proporzione di osservazioni di training nell' m -esima regione della k -esima classe. Anche questo errore però non è abbastanza sensitivo per la costruzione degli alberi, quindi, è preferibile usare altre due misure.

Il *Gini index*:

$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}) \quad (6.4)$$

la misura della varianza totale in tutte le K classi, o anche la misura dei node *purity*. Un valore basso indica che un nodo contiene osservazioni predominanti di una singola classe. Un'alternativa a questo è l'*entropia*:

$$D = - \sum_{k=1}^K \hat{p}_{mk} \log(\hat{p}_{mk}) \quad (6.5)$$

Dato che $0 \leq \hat{p}_{mk} \leq 1$ allora $0 \leq -\hat{p}_{mk} \log \hat{p}_{mk}$. L'entropia assumerà un valore vicino allo zero se tutti i \hat{p}_{mk} sono vicini allo zero o all'1.

Come il Gini index, l'entropia assumerà valori piccoli se l' m -esimo nodo è puro. Questi due metodi sono usati per effettuare il *pruning* dell'albero, mentre il classification error rate è usato per valutare la precisione delle predizioni dell'albero finale.

I dati della Figura 6.6 contengono un outcome binario **HD** per 303 pazienti che si sono presentati con un dolore al petto. Un valore in uscita **Yes** indica la presenza di problemi al cuore basandosi su test angiografici, mentre **No** rappresenta zero problemi al cuore. Nel data set sono presenti 13 predittori.

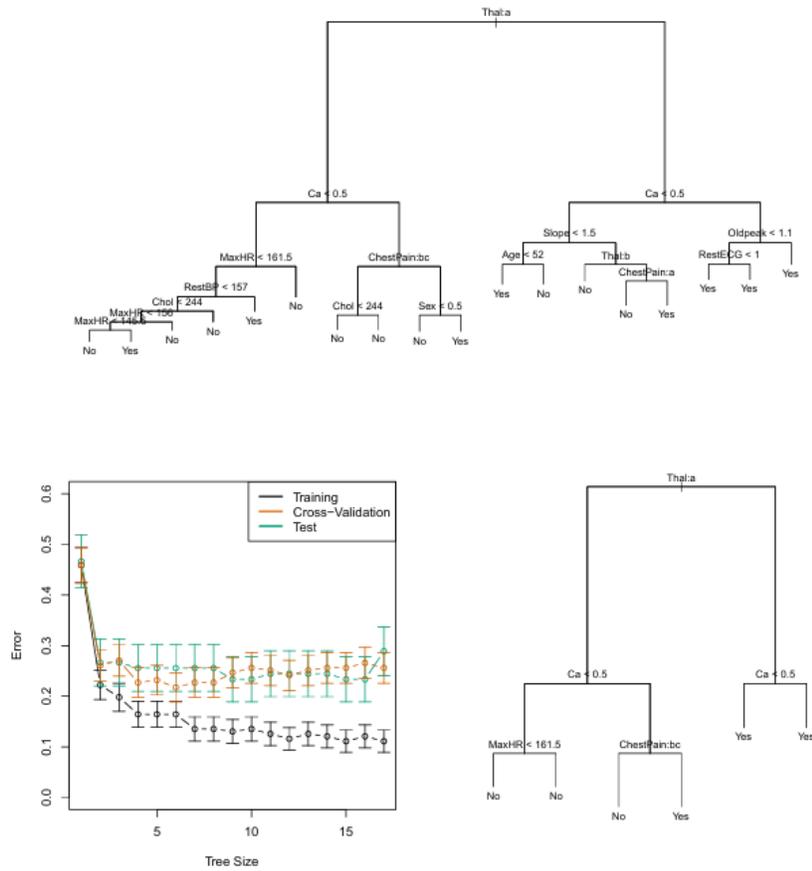


Figura 6.6: Per il data set **Heart**. In alto: l'*unpruned tree*. In basso a sinistra: gli errori di cross-validation, training e test in funzione delle differenti misure dell'albero ridotto. In basso a destra: l'albero ridotto basandosi sul minimo errore di cross-validation.

Possiamo notare un caratteristica sorprendente: alcuni split portano a due nodi terminali con lo stesso valore predetto. Questa divisione viene comunque effettuata per aumentare la node purity.

6.3 Vantaggi e Svantaggi degli Alberi

Vantaggi:

- sono facili da spiegare;
- possono essere rappresentati graficamente ed essere interpretati più facilmente da in non-esperti (specialmente se sono piccoli);
- possono lavorare con predittori qualitativi senza il bisogno di creare variabili fantoccio.

Svantaggi:

- hanno un livello di precisione più basso rispetto ad altri approcci di classificazione e regressione visti;
- possono essere non-robusti (quindi, un piccolo cambio nei dati potrebbe portare ad un grande cambio nella stima finale).

Le performance degli alberi possono aumentare se aggregiamo più decision tree, utilizzando i metodi che seguono.

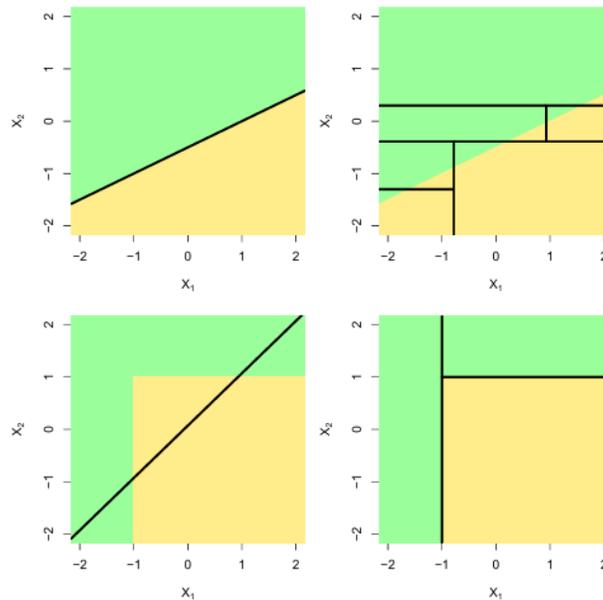


Figura 6.7: Sopra: un esempio di classificazione bi-dimensionale, nel quale il decision boundary reale è lineare, indicato dalle regioni colorate. Un approccio lineare classico (sinistra) outperforma il decision tree che applica split paralleli agli assi (destra). Sotto: abbiamo un decision boundary non lineare. Un modello lineare (sinistra) non riesce quindi a catturare il vero decision boundary, mentre un decision tree (destra) riesce.

6.4 Bagging

Alberi che soffrono di alta varianza portano ad ottenere risultati differenti ogni volta che vengono divisi i dati di training in maniera casuale in due parti. Al contrario, un albero con varianza bassa porterà a risultati simili se applicato ripetutamente a data sets distinti. *Aggregation* o *bagging* è una procedura general-purpose per ridurre la varianza dei metodi statistici.

Dato un set di n osservazioni indipendenti Z_1, \dots, Z_n ognuna con varianza σ^2 , la varianza della media delle osservazioni \bar{Z} è data da σ^2/n .

Un modo per ridurre la varianza ed aumentare la precisione del set di test è quello di prendere molte osservazioni di training dalla popolazione, costruire previsioni separate usando ogni set di training e fare la media delle previsioni risultanti. Quindi, dovremmo calcolare $\hat{f}^1(x), \hat{f}^2(x), \dots, \hat{f}^B(x)$ usando B diversi training sets e fare la media di quest'ultimi per ottenere un singolo modello statistico a bassa varianza. Questo, però, non è pratico perché non abbiamo generalmente accesso a multipli training sets. Potremmo generare B differenti training data sets utilizzando il bootstrap e successivamente addestrare il metodo sul b -esimo training set creato tramite bootstrap in maniera tale da ottenere $\hat{f}^{*b}(x)$. Infine, possiamo fare la media di tutte le previsioni per ottenere:

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(x) \quad (6.6)$$

Per una data osservazione di test possiamo registrare la classe predetta da ognuno degli B alberi e fare un *majority vote*: la previsione complessiva è la classe che si verifica più comunemente tra le B previsioni.

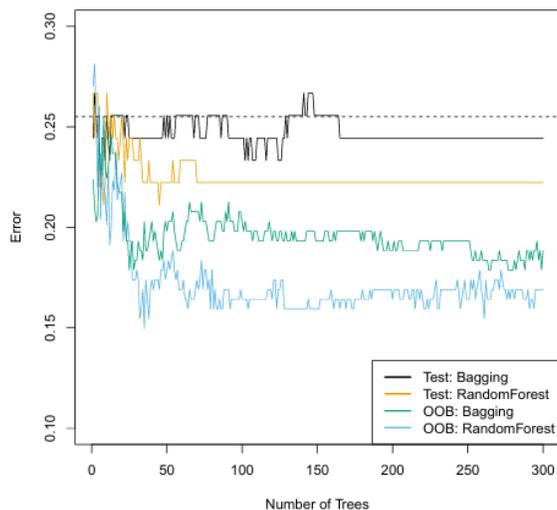


Figura 6.8: Risultati del bagging e del random forest per il data set Heart. I test error rate sono in funzione di B .

Dalla Figura 6.8 notiamo come il Bagging test error (linea nera continua) è leggermente inferiore in questo caso rispetto al test error rate ottenuto da un singolo albero (linea tratteggiata). Nel bagging usare un valore grande di B non porta ad overfitting, usare un $B = 100$ è sufficiente ad ottenere buone performance in questo esempio.

Stima dell'Out-of-Bag Error

Esiste un metodo per stimare il test error di un modello di bagging senza il bisogno di performare la cross-validazione o il validation set. Ogni bagged tree fa uso di circa due terzi delle osservazioni, il restante un terzo delle osservazioni sono dette OOB (*out-of-bag*). Possiamo predire la risposta per l' i -esima osservazione usando ogni albero in cui quella osservazione era OOB. Questo porta ad avere circa $B/3$ predizioni per l' i -esima osservazione. Per ottenerne solo una possiamo fare la media delle risposte predette (se l'obiettivo è la regressione) o possiamo scegliere con il majority vote (se l'obiettivo è la classificazione). L'errore di OOB risultante è una valida stima del test error per i modelli bagged ed è particolarmente conveniente quando effettuiamo il bagging su data set grandi (per i quali la cross-validazione sarebbe computazionalmente onerosa). Nella Figura 6.8 notiamo come quando B è sufficientemente grande l'OOB error è virtualmente equivalente al leave-one-out cross-validation error.

Variable Importance Measures

Dopo aver effettuato il bagging il modello risultante può essere difficile da interpretare (così facendo si va a perdere uno dei vantaggi dell'usare gli alberi).

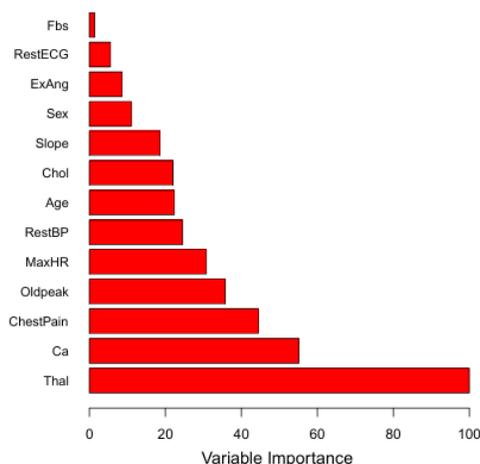


Figura 6.9: Un plot della variable importance per il data set Heart. La variable importance è calcolata usando la diminuzione media del Gini index, espressa relativamente al massimo.

Dalla Figura 6.9 notiamo come le variabili con la più grande diminuzione media del Gini index (un valore alto indica un importante predittore) sono **Thal**, **Ca** e **ChestPain**.

6.5 Random Forest

Nel random forest si hanno dei miglioramenti rispetto al bagging grazie ad una piccola modifica che *de-correla* l'albero. Quando costruiamo i decision tree (sui sample di bootstrap, come per il bagging), un sample casuale di m predittori è scelto come candidato per la divisione considerando il set di p predittori. Tipicamente scegliamo $m \approx \sqrt{p}$ (quindi, scegliamo 4 di 13 predittori per il data set **Heart**). Supponendo di avere un predittore caratterizzante nel data set, ed un numero di altri predittori importanti. Allora nella collezione di bagged trees, molti o tutti gli alberi useranno il predittore caratterizzante come prima divisione. Quindi, tutti i bagged trees risultano simili e le previsioni saranno altamente correlate. La differenza tra il bagging ed il random forest persiste nella scelta della dimensione del subset m . Se un random forest ha $m = p$ praticamente è un bagging. Dalla Figura 6.8 notiamo come un random forest considerando $m = \sqrt{p}$ porta ad ottenere un test error ed un OOB error inferiore al bagging. Usare un valore di m basso nella costruzione di un random forest è tipicamente utile quando abbiamo un alto numero di predittori correlati.

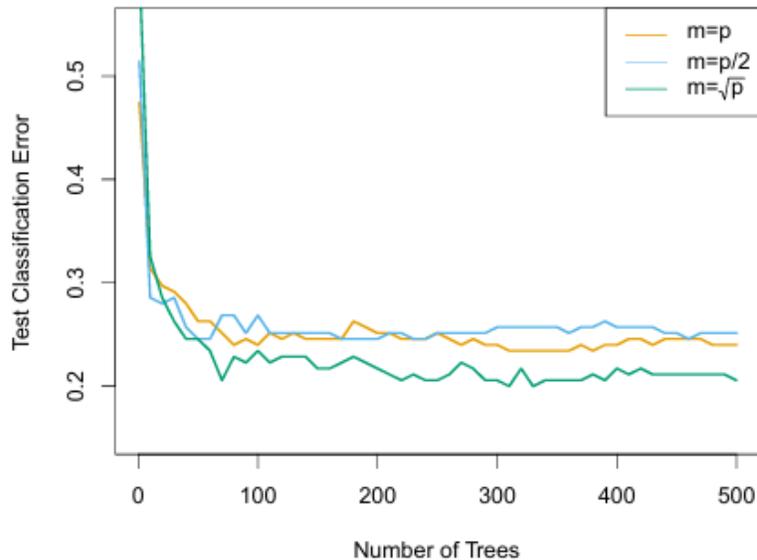


Figura 6.10: Risultati di un random forest per le 15-class gene expression data set con $p = 500$ predittori. Ogni linea colorata rappresenta un valore differente di m .

Applicando il random forest ad un high-dimensional biological data set che consiste in misurazioni di espressione di 4718 geni su tessuti campione di 349 pazienti. Ci sono 20000 geni negli umani, ed i geni individuali hanno differenti livelli di attività, o espressione, ed in particolare celle, tessuti e condizioni biologiche. Per questo data set ad una paziente può essere assegnata una label qualitativa con 15 differenti livelli: *normale* o 1 di 14 differenti tipi di *cancro*. L'obiettivo è quello di usare il random forest per prevedere il tipo di cancro basandoci sui 500 geni che hanno il valore maggiore di varianza nel training set. Dividiamo casualmente le osservazioni in training e test set ed applichiamo il random forest al training set per tre differenti valori di m . L'error rate di un single tree è di 45.7% ed il null rate è 75.4%. Usando 400 alberi è sufficiente a fornire buone performance.

6.6 Boosting

Nel boosting gli alberi sono costruiti *sequenzialmente*: ogni albero è costruito usando le informazioni dell'albero costruito precedentemente. Il boosting non include il bootstrap ma ogni albero è fittato su una versione modificata del data set originale.

Algoritmo 6.2 *Boosting per i Regression Trees*

- 1) Impostare $\hat{f}(x) = 0$ ed $r_i = y_i$ per ogni i del training set.
- 2) Per $b = 1, 2, \dots, B$ ripeti:
 - a) fitta un albero \hat{f}^b con d divisioni ($d + 1$ nodi terminali) nei training data (X, r) .
 - b) Aggiorna \hat{f} aggiungendo una versione ristretta del nuovo albero:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x) \quad (6.7)$$

- c) Aggiorna i residui:

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i) \quad (6.8)$$

- 3) In output il modello boosted:

$$\hat{f}(x) = \sum_{b=1}^B \lambda \hat{f}^b(x) \quad (6.9)$$

Il boosting ha tre parametri di tuning:

1. il numero di alberi B (diversamente dal bagging e dal random forest, il boosting può overfittare se B è troppo grande), scelto tramite cross-validazione.
2. il parametro di shrinkage λ , controlla il rate con cui impara il boosting (è un numero positivo piccolo, perché è uno *slow learner*). Tipicamente assume valore 0.01 oppure 0.001, la scelta dipende dal problema e da B .
3. il numero di divisioni d in ogni albero, controlla la complessità del boosting. Spesso $d = 1$ è un buon valore (in questo caso l'albero è uno *stump*, ceppo, che consiste in una sola divisione. Generalmente d quindi è l'*interaction depth* e controlla l'ordine di interazione delo modello boostato, dato che d divisioni possono includere al massimo d variabili.

In Figura 6.11 notiamo come un singolo ceppo con un'interaction depth 1 performa meglio del modello con depth 2 e del random forest.

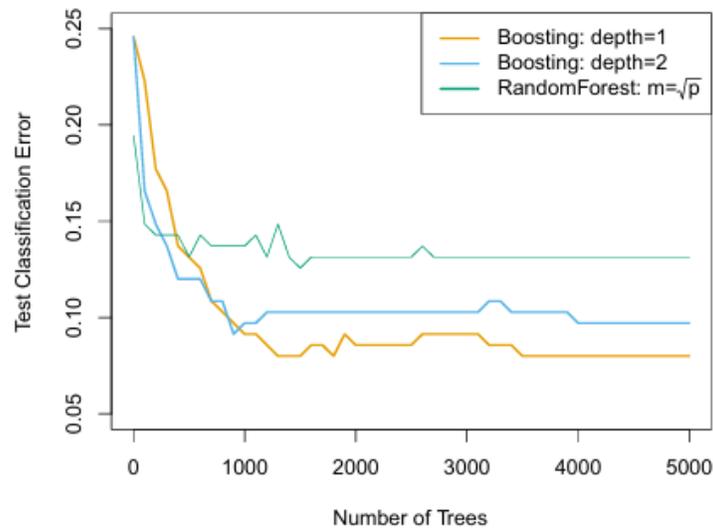


Figura 6.11: Risultati dell'applicazione del boosting e random forest sul 15-class gene expression data set in maniera tale da predire *cancer vs normal*. Il test error è mostrato in funzione del numero di alberi. Per i due modelli di boosting $\lambda = 0.01$.

7 Support Vector Machines

La SVM (*Support Vector Machine*) è considerata una tra i migliori classificatori "out of the box". Essa è una generalizzazione del *maximal margin classifier*.

7.1 Maximal Margin Classifier

In uno spazio p -dimensionale, un *hyperplane* è un sottospazio piatto affine¹ di dimensione $p - 1$. Matematicamente è definito dall'equazione:

$$\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p = 0 \quad (7.1)$$

Se un punto $X = (X_1, X_2, \dots, X_p)^T$ in uno spazio p -dimensionale soddisfa l'equazione (7.1) allora si poggia sull'hyperplane.

Dal grafico a sinistra della Figura 7.1 possiamo notare come è possibile costruire un hyperplane che separa le osservazioni di training. Possiamo chiamare le osservazioni della classe blu $y_i = 1$ e quelle della classe viola $y_i = -1$. Un *separating hyperplane* segue le seguenti proprietà: (Chiamando $f(x) = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}$), se $y_i = 1$ allora $f(x) > 0$, se $y_i = -1$ allora $f(x) < 0$. Un'altra proprietà dei separating hyperplane è: $y_i f(x) > 0 \quad \forall i = 1, \dots, n$. Il separating hyperplane, se esiste, può essere utilizzato come classifier (un'osservazione è assegnata ad una data classe a seconda di come è situata rispetto all'hyperplane). Se vogliamo classificare un'osservazione x^* basandoci sul segno di $f(x^*) = \beta_0 + \beta_1 x_1^* + \dots + \beta_p x_p^*$, diremo che se $f(x^*)$ è positivo allora assegnamo l'osservazione di test alla classe 1, se $f(x^*)$ è negativo, l'assegnamo alla classe -1. Nella Figura 7.1 notiamo un esempio di questo classificatore (notiamo che il decision boundary è lineare).

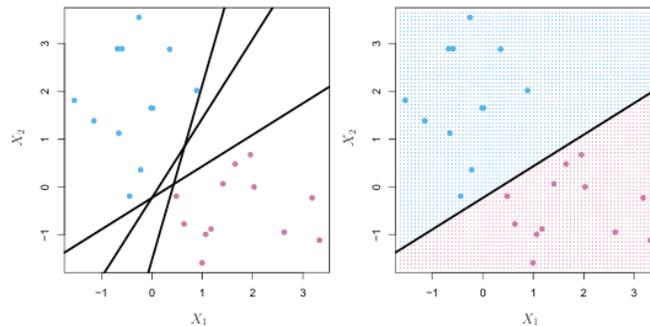


Figura 7.1: Esempi di separating hyperplane.

In generale, un dato può essere separato utilizzando un infinito numero di hyperplane, perché ogni hyperplane può essere leggermente spostato o ruotato senza entrare in contatto con nessuna delle osservazioni.

¹*affine*: che non passa dall'origine

Un metodo per scegliere uno tra questi infiniti hyperplane è il *maximal margin hyperplane* (anche conosciuto come *optimal separating hyperplane*) per il quale si sceglie il separating hyperplane che il più distante dalle osservazioni di training.

Quindi, se chiamiamo *margin* la più piccola distanza (perpendicolare) tra un hyperplane e le osservazioni, con il maximal margin hyperplane scegliamo come separating hyperplane quello avente margine maggiore.

Quando classifichiamo usando il maximal margin hyperplane otteniamo un metodo conosciuto come *maximal margin classifier*; il quale è spesso funzionale, ma potrebbe portare ad overfitting nel caso in cui p sia troppo grande.

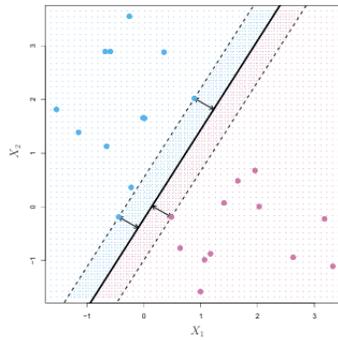


Figura 7.2: Esempio di maximal margin classifier.

Dalla Figura 7.2 notiamo come tre osservazioni di training risultano equidistanti dal maximal margin hyperplane e si appoggiano sulle linee tratteggiate (le quali indicano la larghezza del margine). Queste tre osservazioni sono conosciute come *support vectors* dato che sono vettori in uno spazio p -dimensionale (nella figura $p = 2$) e "supportano" il maximal margin hyperplane, nel senso che se spostiamo questi punti anche il margine si sposta.

Il margine dipende solo dai support vectors e non dalle altre osservazioni.

Volendo costruire il maximal margin hyperplane basata su un set di n osservazioni di training $x_1, \dots, x_n \in \mathbb{R}^p$ e associata alle classi $y_1, \dots, y_n \in \{-1, 1\}$, il *maximal margin hyperplane* è la soluzione al problema di ottimizzazione:

$$\max_{\beta_0, \beta_1, \dots, \beta_p, M} M \quad (7.2)$$

$$\text{soggetto a } \sum_{j=1}^p \beta_j^2 = 1 \quad (7.3)$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n \quad (7.4)$$

In questo modo ogni osservazione sarà nel lato corretto dell'hyperplane, se M è positivo. Il problema di ottimizzazione, quindi, sceglie i β_0, \dots, β_p per massimizzare M , quest'ultimo rappresenta il margine dell'hyperplane.

Se non esiste un separating hyperplane il problema di ottimizzazione (7.2)-(7.4) non ha soluzione con $M > 0$, un esempio è mostrato nella Figura 7.3. In questo le due classi non possono essere separate correttamente.

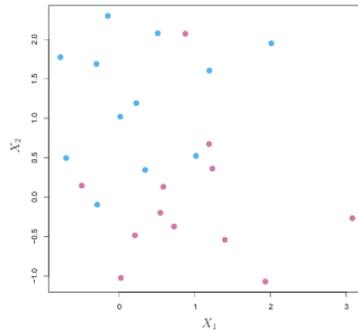


Figura 7.3: Due classi che non possono essere separate da un hyperplane.

Possiamo, però, estendere il concetto di separating hyperplane in modo tale da ottenerne uno che riesce quasi a dividere le classi, questo viene detto *soft margin*.

7.2 Support Vector Classifiers

Notiamo in Figura 7.4 come una l'aggiunta di una singola osservazione porta ad un cambiamento drastico del maximal margin hyperplane, ed inoltre il nuovo maximal margine hyperplane non è soddisfacente perché il margine è troppo stretto. Questo ci suggerisce che è possibile un overfitting dei dati utilizzando questo metodo. Potrebbe essere utile misclassificare alcune osservazioni di training in maniera tale da ottenere una migliore classificazione delle restanti osservazioni.

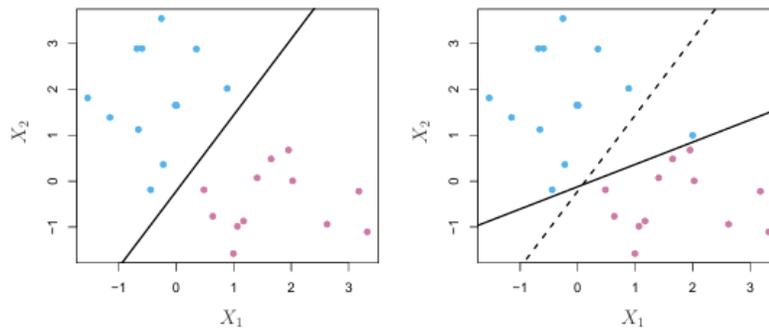


Figura 7.4: Sinistra: due classi di osservazioni sono mostrate in blu e viola. Destra: aggiungendo una nuova osservazione il maximal margin hyperplane cambia da quello tratteggiato a quello continuo.

Il *support vector classifier* (anche conosciuto come *soft margin classifier*) permette ad alcune osservazioni di essere nel lato sbagliato dell'hyperplane. (Il margine quindi viene chiamato *soft* perché può essere violato da alcune osservazioni).

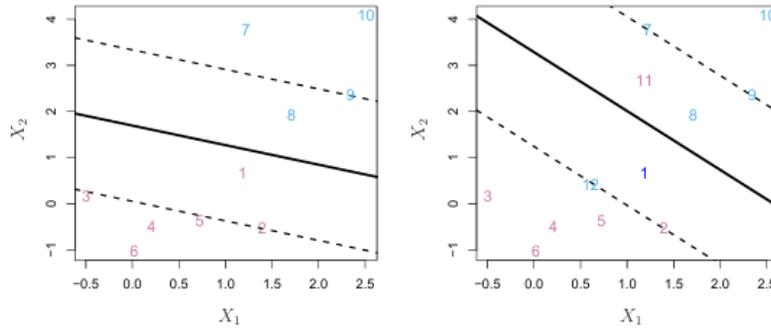


Figura 7.5: Esempi di support vector classifier.

L'hyperplane in questo caso sarà la soluzione al problema di ottimizzazione:

$$\max_{\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n, M} M \quad (7.5)$$

$$\text{soggetto a } \sum_{j=1}^p \beta_j^2 = 1 \quad (7.6)$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \quad (7.7)$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^n \epsilon_i \leq C \quad (7.8)$$

dove C è un parametro di tuning non negativo ed $\epsilon_1, \dots, \epsilon_n$ sono le *slack variables* (le quali permettono ad osservazioni individuali di essere sul lato sbagliato (o sul margine sbagliato) dell'hyperplane).

Se $\epsilon_i = 0$ allora l' i -esima osservazione è nel lato corretto del margine. Se $\epsilon_i > 0$ allora l' i -esima osservazione è nel lato sbagliato del margine. Se $\epsilon_i > 1$ allora l' i -esima osservazione è nel lato sbagliato dell'hyperplane.

C determina il numero e la severità di queste violazioni di margine ed hyperplane che può essere tollerato.

Se $C = 0$ allora non sono possibili violazioni, $\epsilon_1 = \dots = \epsilon_n = 0$, siamo nel caso del maximal margin hyperplane. Se $C > 0$ non più di C osservazioni possono essere nel lato sbagliato dell'hyperplane. All'aumentare di C aumenta il margine, analogamente quando C decresce il margine si restringe.

C è generalmente scelto tramite cross-validazione e controlla il trade-off tra bias e varianza. Nella Figura 7.6 in alto a sinistra il valore di C è elevato, quindi, il classificatore ha bassa varianza e potenzialmente alto bias; nel grafico in basso

a destra invece abbiamo un valore di C basso (abbiamo, quindi, meno support vectors) il bias sarà anch'esso basso mentre la varianza risulta alta.

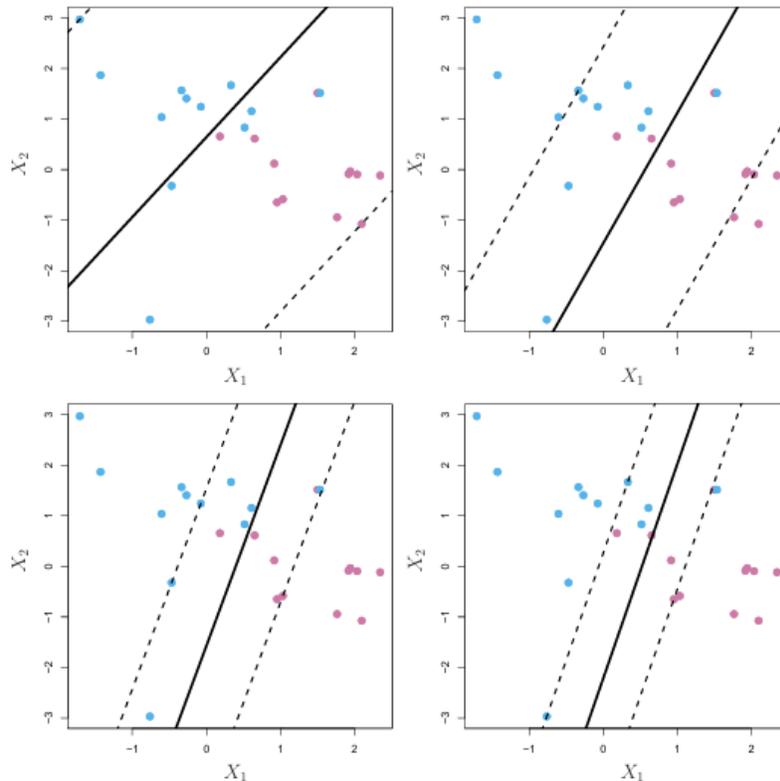


Figura 7.6: Esempi di support vector classifier usando quattro differenti valori del parametro di tuning C.

7.3 Support Vector Machines

Il support vector classifier ha un decision boundary lineare, se consideriamo infatti i dati in Figura 7.7 quest'ultimo risulta inutile.

Il *Support Vector Machine* (SVM) è un'estensione del support vector classifier in cui il feature space è ingrandito usando i *kernels*.

L'*inner product* di due osservazioni $x_i, x_{i'}$ è dato da:

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^p x_{ij} x_{i'j} \quad (7.9)$$

Il linear support vector classifier può essere rappresentato come:

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x_i, x \rangle \quad (7.10)$$

dove sono presenti n parametri α_i , $i=1, \dots, n$; uno per ogni osservazione di training. Per stimare questi parametri $\alpha_1, \dots, \alpha_n$ e β_0 abbiamo bisogno di $\binom{n}{2} = n(n-1)/2$ inner products $\langle x_i, x_{i'} \rangle$ tra tutte le coppie di osservazioni di training. Se un'osservazione non è un support vector allora $\alpha_i = 0$, solo i support vectors hanno un α_i diverso da zero.

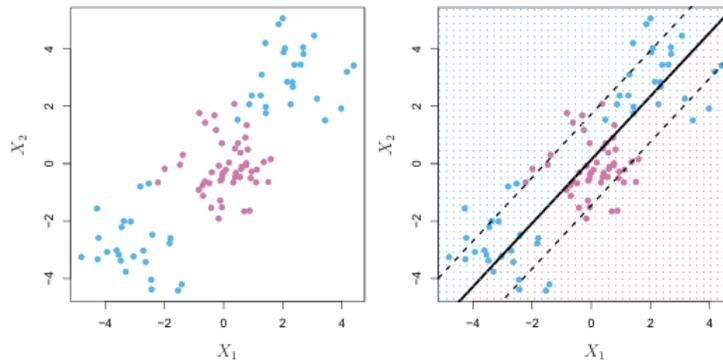


Figura 7.7: Esempio support vector classifier con un non-linear boundary.

Se chiamiamo \mathcal{S} l'insieme degli indici di questi punti di supporto, possiamo riscrivere ogni solution function come:

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x_i, x \rangle \quad (7.11)$$

Possiamo sostituire l'inner product con una sua *generalizzazione*, in cui K è una funzione (che chiameremo *kernel*) e quantifica la similarità di due osservazioni.

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^p x_{ij} x_{i'j}\right)^d \quad (7.12)$$

Questa funzione è conosciuta come *kernel polinomiale* di grado d (con d intero positivo). Usando $d > 1$ porta ad avere un decision boundary più flessibile. Quando il support vector classifier è combinato con un kernel non-lineare, il classificatore risultante è noto come *support vector machine*.

La funzione non-lineare ha la forma:

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i) \quad (7.13)$$

Un'altro kernel popolare è quello *radiale*, che ha la forma:

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2) \quad (7.14)$$

in cui γ è una costante positiva.

Quando una osservazione di test data $x^* = (x_1^*, \dots, x_p^*)^T$ è lontana dalle osservazioni di training x_i (in termini di distanza Euclidiana) allora $\sum_{j=1}^p (x_j^* - x_{ij})^2$ sarà grande, quindi $K(x^*, x_i) = \exp(-\gamma \sum_{j=1}^p (x_j^* - x_{ij})^2)$ sarà piccolo.

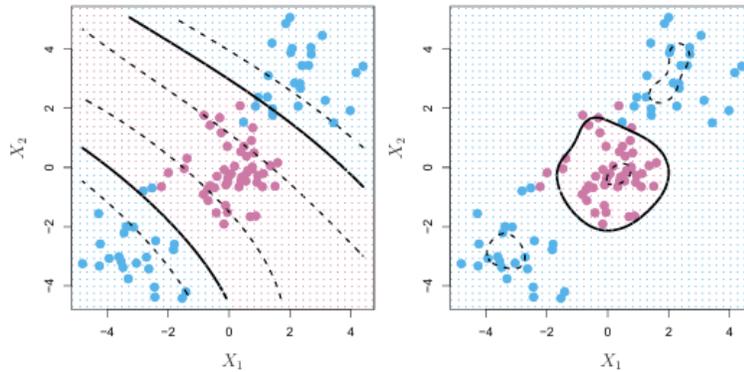


Figura 7.8: Sinistra: un SVM con un kernel polinomiale di grado 3 è applicato a dati non-lineari. Destra: un SVM con un kernel radiale è applicato.

Usando i kernels abbiamo vantaggi dal punto di vista computazionale, perché abbiamo bisogno di calcolare solamente $K(x_i, x_{i'})$ per tutte le $\binom{n}{2}$ distinte coppie i, i' . Per il radial kernel però il feature space è *implicito* e dimensionalmente infinito, quindi non possiamo calcolarlo!

Compariamo l'SVM all'LDA sul data set **Heart**. Dopo aver rimosso 6 osservazioni mancanti, il data set consiste in 297 soggetti, che dividiamo casualmente in 207 osservazioni di training e 90 di test.

Nel grafico a sinistra della Figura 7.9 sono confrontati LDA e support vector classifier (che è equivalente ad un SVM con un kernel polinomiale di grado $d = 1$) sui dati di training. Entrambi i classificatori sembrano ottimi, forse è leggermente meglio il support vector classifier. Nel grafico a destra invece sono comparati il support vector classifier con vari SVMs a kernel radiale aventi valori differenti di γ . All'aumentare di γ il fit diventa meno lineare e la curva ROC migliora. Usando un $\gamma = 10^{-1}$ si ottiene una curva ROC quasi perfetta.

Nella Figura 7.10 sono stati effettuati gli stessi confronti sulle osservazioni di test. Ancora una volta il support vector classifier sembra avere un piccolo vantaggio sull'LDA. Invece, l'SVM avente $\gamma = 10^{-1}$ questa volta è outperformato da tutte le restanti curve, questo conferma che un metodo più flessibile

produce i più bassi errori di training ma non porta necessariamente a migliori performance sui dati di test.

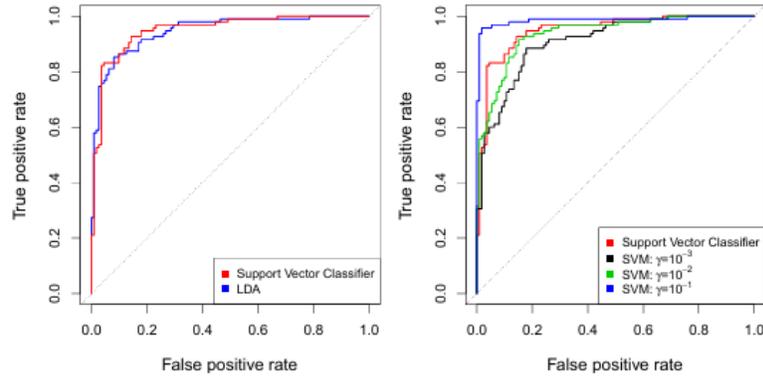


Figura 7.9: Curve ROC per il training set del data set **Heart**.

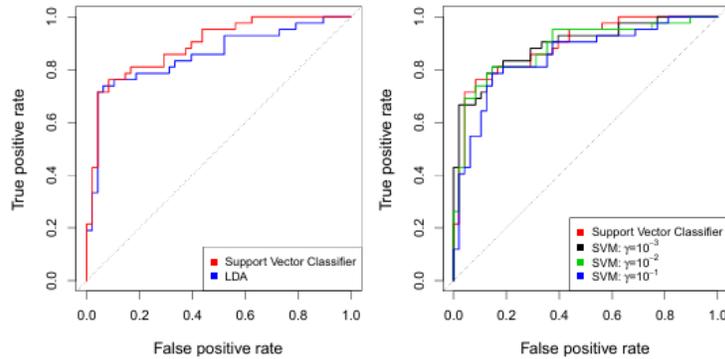


Figura 7.10: Curve ROC per il test set del data set **Heart**.

7.4 SVMs con più di Due Classi

Il concetto di separating hyperplane non supporta più di due classi.

7.4.1 One-Versus-One (OVO) Classification

Supponendo di voler performare una classificazione utilizzando SVMs, ma sono presenti $K > 2$ classi. Un approccio *one-versus-one* (anche conosciuto come *all-pairs*) costruisce $\binom{K}{2}$ SVMs, ognuno che compara una coppia di classi. La classificazione finale è effettuata aggiudicando l'osservazione di test alla classe a cui questa è stata maggiormente assegnata nelle classificazioni a coppie.

7.4.2 One-Versus-All (OVA) Classification

L'approccio *one-versus-all* (anche detto *one-versus-rest*) è un'altra alternativa utilizzata per applicare SVMs nel caso in cui sono presenti $K > 2$ classi. Fittiamo K SVMs, ogni volta comparando una delle K classi alle restanti $K - 1$ classi. Se $\beta_{ok}, \beta_{1k}, \dots, \beta_{pk}$ denotano i parametri risultanti fittando un SVM comparando la k -esima classe alle altre e x^* denota un'osservazione di test, assegniamo l'osservazione alla classe che ha il più grande $\beta_{ok}, \beta_{1k}x_1^*, \dots, \beta_{pk}x_p^*$.

7.5 Relazioni con la Logistic Regression

Esistono connessioni tra SVMs ed altri metodi classici di statistica.

I criteri (7.5)-(7.8) possono essere riscritti per fittare il support vector classifier $f(x) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p$ come:

$$\min_{\beta_0, \beta_1, \dots, \beta_p} \left\{ \sum_{i=1}^p \max[0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^p \beta_j^2 \right\} \quad (7.15)$$

con λ parametro di tuning positivo. Se λ è grande β_1, \dots, β_p sono piccoli, quindi sono tollerate più violazioni di margine ed il classificatore risulterà avere bassa varianza ed alto bias. Se λ è piccolo allora possono accadere solo poche violazioni di margine ed il classificatore risultate ha basso bias ed alta varianza. L'equazione (7.5) assume una forma "Loss + Penalty".

La funzione di loss delle SVMs viene chiamata *hinge loss* ed è mostrata nella Figura 7.11 a confronto con la funzione di loss della regressione logistica. Se le classi sono ben separate la SVMs performa meglio; in regimi in cui è presente overlap è preferibile l'utilizzo della regressione logistica.

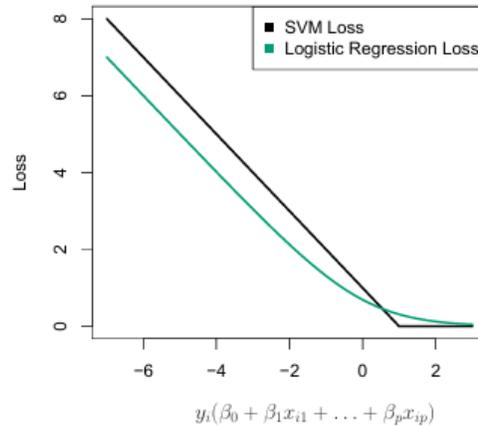


Figura 7.11: Confronto tra la funzione di Loss del SVM e della Logistic Regression.

8 Deep Learning

8.1 Single Layer Neural Networks

Una rete neurale prende come input un vettore $X = (X_1, X_2, \dots, X_p)$, di p variabili, e costruisce una funzione non lineare $f(X)$ per predire la risposta Y .

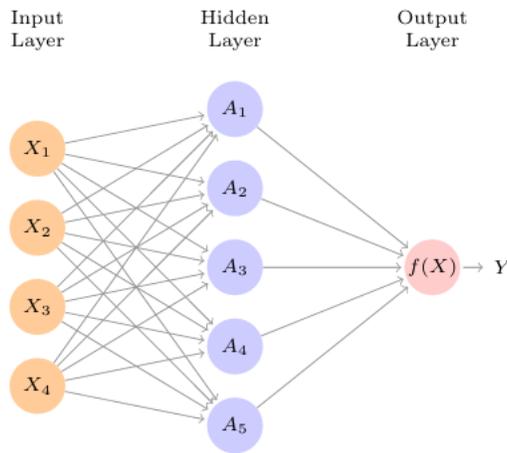


Figura 8.1: Neural network, con un singolo hidden layer, per modellare una risposta quantitativa utilizzando quattro predittori.

Le quattro features X_1, X_2, X_3, X_4 formano l'*input layer*. Le frecce indicano che ogni input può andare in ogni K *hidden units* (in figura $K = 5$). Il neural network model ha la seguente forma:

$$f(X) = \beta_0 + \sum_{k=1}^K \beta_k h_k(X) \quad (8.1)$$

dove

$$h_k(X) = A_k = g(w_{k0} + \sum_{j=1}^p w_{kj} X_j) \quad (8.2)$$

in cui $g(z)$ è una *funzione di attivazione* non lineare, specificata in anticipo.

La funzione di attivazione *sigmoid*:

$$g(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}} \quad (8.3)$$

La funzione di attivazione ReLU (*Rectified Linear Unit*):

$$g(z) = (z)_+ = \begin{cases} 0 & \text{se } z < 0 \\ z & \text{altrimenti} \end{cases} \quad (8.4)$$

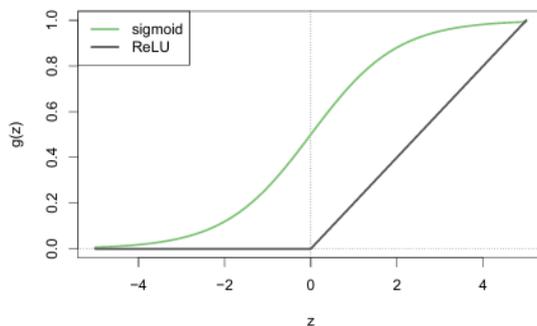


Figura 8.2: Funzioni di attivazione.

Per ottenere una risposta quantitativa è usato tipicamente lo squared-error loss, quindi i parametri sono scelti per minimizzare:

$$\sum_{i=1}^n (y_i - f(x_i))^2 \quad (8.5)$$

8.2 Multilayer Neural Networks

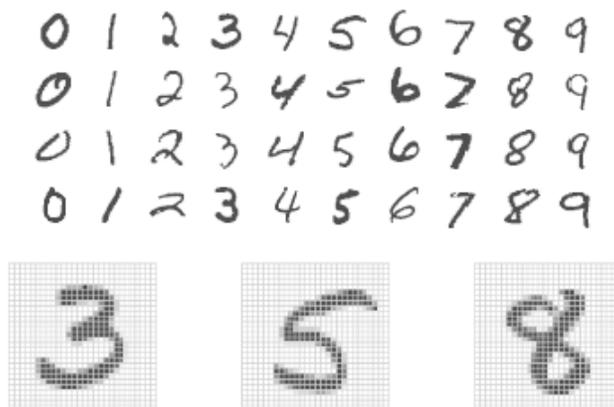


Figura 8.3: Esempi di digits scritti a mano presi dal data set MNIST.

L'idea è quella di costruire un modello per classificare le immagini nella loro classe di digit (0-9) corretta. Ogni immagine ha $p = 20 \times 28 = 784$ pixels, ognuno di questi è un 8-bit grayscale value compreso tra 0 e 255 rappresentante il valore del digit scritto in quel quadrato piccolo. Questi pixels sono conservati in un vettore di input X . L'output è l'etichetta della classe, rappresentata da un

vettore di 10 variabili fantoccio $Y = (Y_0, Y_1, \dots, Y_9)$, aventi uno nella posizione della label corrispondente e zero altrove. Ci sono 60000 immagini di training e 10000 di test.

La Figura 8.4 rappresenta una rete neurale adatta per risolvere questa digit-classification.

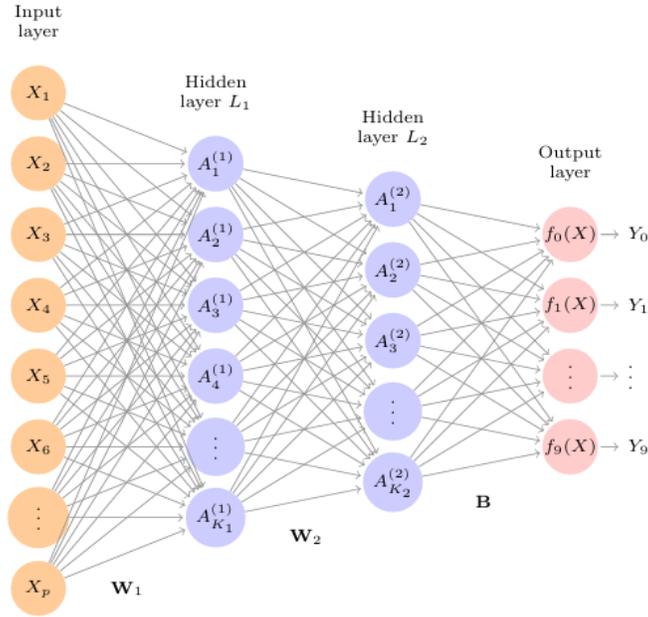


Figura 8.4: Neural network con due hidden layers ed uscite multiple.

Il primo hidden layer è:

$$A_k^{(1)} = h_k^{(1)}(X) = g(w_{k0}^{(1)} + \sum_{j=1}^p w_{kj}^{(1)} X_j) \quad (8.6)$$

per $k = 1, \dots, K_1$. Il secondo hidden layer è:

$$A_\ell^{(2)} = h_\ell^{(2)}(X) = g(w_{\ell 0}^{(2)} + \sum_{k=1}^{K_1} w_{\ell k}^{(2)} A_k^{(1)}) \quad (8.7)$$

con $\ell = 1, \dots, K_2$.

Nella Figura 8.4 la notazione \mathbf{W}_1 rappresenta l'intera matrice dei pesi che passano dall'input layer al primo hidden layer L_1 . Questa matrice avrà $785 \times 256 = 200960$ elementi (i termini sono 785 e non 784 perché teniamo conto dell'intercetta, anche detta *bias term*). La dimensione della matrice dei pesi \mathbf{W}_2 è $257 \times 128 = 32896$. Perché per il problema MNIST l'input layer, il primo

ed il secondo hidden layer hanno rispettivamente $p = 784, K_1 = 256, K_2 = 128$ unità. L'output layer ha 10 unità. Questa rete neurale ha 235146 parametri (che vengono detti *weights*).

Per computare dieci differenti modelli lineari simili al nostro modello singolo:

$$Z_m = \beta_{m0} + \sum_{\ell=1}^{K_2} \beta_{m\ell} h_{\ell}^{(2)}(X) \quad (8.8)$$

per $m = 0, 1, \dots, 9$. La matrice \mathbf{B} conserva tutti i $129 \times 10 = 1290$ weights.

Se queste risposte quantitative fossero tutte separate basterebbe settare per ognuna $f_m(X) = Z_m$. Invece, vogliamo che la nostre stime rappresentino le class probabilities; quindi, usiamo la funzione di attivazione *softmax*:

$$f_m(X) = Pr(Y = m|X) = \frac{e^{Z_m}}{\sum_{\ell=0}^9 e^{Z_{\ell}}} \quad (8.9)$$

Per addestrare questa rete, dato che la risposta è qualitativa, cerchiamo una stima dei coefficienti che minimizza la multinomial log-likelihood negativa:

$$-\sum_{i=1}^n \sum_{m=0}^9 y_{im} \log(f_m(x_i)) \quad (8.10)$$

conosciuta come *cross-entropia*.

Dalla Tabella 8.1 vediamo come le neural networks hanno risultati migliori rispetto ai modelli lineari.

Method	Test Error
Neural Networks + Ridge Regularization	2.3%
Neural Networks + Dropout Regularization	1.8%
Multinomial Logistic Regression	7.2%
Linear Discriminant Analysis	12.7%

Tabella 8.1: Test error rate per le neural networks con due forme di regolarizzazione e per la multinomial logistic regression e la linear discriminant analysis sul data set MNIST.

Le due regolarizzazioni (ridge e dropout) sono necessarie ad evitare l'overfitting.

8.3 Convolutional Neural Networks

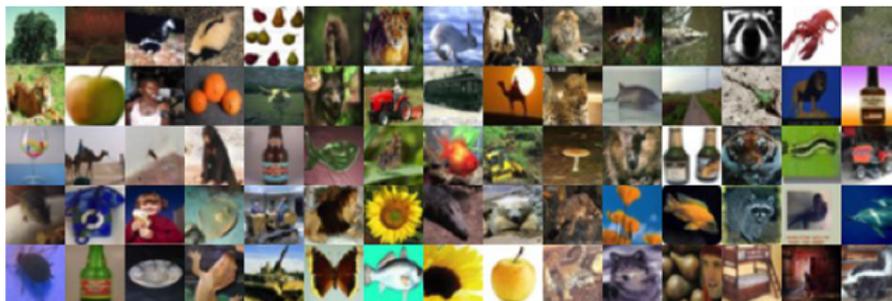


Figura 8.5: 75 immagini prese dal database CIFAR100, contenente una collezione di immagini naturali della vita quotidiana divisa in 100 classi.

Questo database consiste in 60000 immagini etichettate in base a 20 superclassi ognuna avente 5 classi. Ogni immagine ha una risoluzione di 32 x 32 pixels, con tre numeri 8-bit rappresentati rosso, verde e blu. I numeri per ogni immagine è organizzata in un array 3-dimensionale chiamato *feature map*. I primi due assi sono spaziali (entrambi 32-dimensionali) ed il terzo è il *channel axis* rappresentante i tre colori. Il training set è di 50000 immagini ed il test set di 10000.

Le CNNs (*Convolutional Neural Networks*) emulano il modo di classificare le immagini degli umani, riconoscendo features specifiche o patterns nell'immagine che distinguono ogni oggetto o classe particolare.

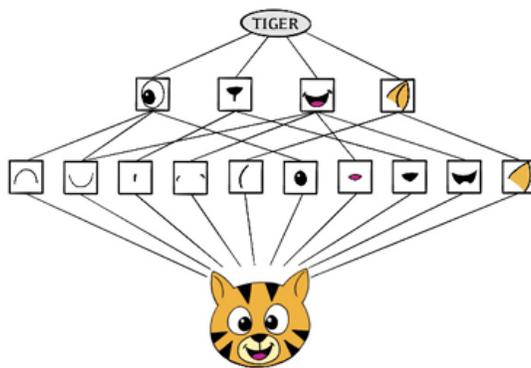


Figura 8.6: Esempio di CNN applicata ad un'immagine cartone rappresentante una tigre.

La rete inizialmente identifica delle features di basso livello dalla immagine di input (come angoli, colori) che vengono successivamente combinate per formarne

altre di maggior livello (come parti dell'orecchio, occhi, etc...). La presenza di queste features di alto livello contribuisce ad ottenere un corretta classe in output.

8.3.1 Convolution Layers

Un *convolution layer* è formato da un grande numero di *filtri di convoluzione*, ognuno che determina se una particolare feature è presente in un immagine. L'operazione della *convoluzione* consiste nel moltiplicare ripetutamente elementi di una matrice ed aggiungere i risultati.

Consideriamo per esempio un'immagine 4 x 3:

$$\text{Immagine Originale} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \\ j & k & l \end{bmatrix}$$

Se usiamo un filtro 2 x 2:

$$\text{Convolution Filter} = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$$

Otteniamo:

$$\text{Immagine Convoluta} = \begin{bmatrix} a\alpha + b\beta + d\gamma + e\delta & b\alpha + c\beta + e\gamma + f\delta \\ d\alpha + e\beta + g\gamma + h\delta & e\alpha + f\beta + h\gamma + i\delta \\ g\alpha + h\beta + j\gamma + k\delta & h\alpha + i\beta + k\gamma + l\delta \end{bmatrix}$$

L'elemento in alto a sinistra è ottenuto moltiplicando ogni elemento del filtro 2 x 2 con il corrispondente elemento della porzione di immagine 2 x 2 in alto a sinistra, ed aggiungendo i risultati ottenuti. Gli altri elementi sono ottenuti in maniera simile. Se una sottomatrice 2 x 2 dell'immagine originale assomiglia al filtro di convoluzione allora avrà un valore grande; altrimenti avrà un valore piccolo. La Figura 8.7 illustra l'applicazione di due filtri di convoluzione ad un'immagine 192 x 179 rappresentante una tigre. Ogni filtro di convoluzione è un'immagine 15 x 15 contenente per lo più zero (nero), con sottili strisce di uno (bianco) orientate verticalmente ed orizzontalmente all'interno dell'immagine.

In un convolution layer usiamo molti filtri per catturare una varietà di angoli e forme orientate in maniera diversa nell'immagine.

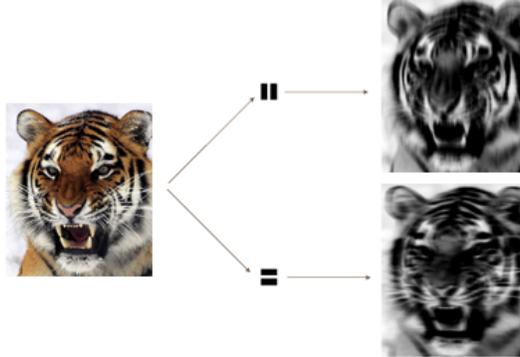


Figura 8.7: Esempio di convoluzione applicato all'immagine di una tigre. L'immagine convoluta in alto sottolinea le linee verticali della tigre, quella in basso enfatizza le linee orizzontali della tigre.

8.3.2 Pooling Layers

Un *pooling layer* è un modo per condensare un'immagine grande in una piccola riassuntiva. Esistono diverse tecniche di pooling, il *max pooling* riassume ciascun blocco di pixels 2×2 in un'immagine usando il massimo valore di quel blocco. In questo modo si riduce la grandezza dell'immagine di un fattore o due in ogni direzione ed allo stesso tempo si ha *invarianza della posizione*.

Un esempio di max pooling:

$$\text{Max pool} \begin{bmatrix} 1 & 2 & 5 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 1 & 3 & 4 \\ 1 & 1 & 2 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 5 \\ 2 & 4 \end{bmatrix}$$

8.3.3 Architettura di una CNN

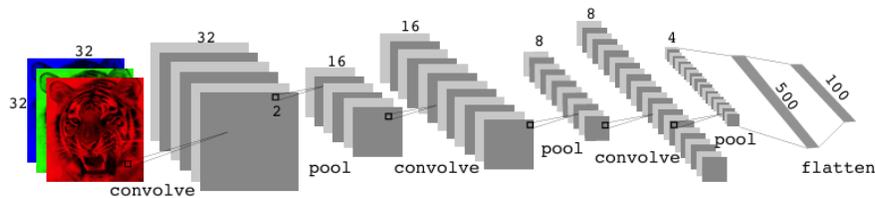


Figura 8.8: Architettura di una deep CNN per il CIFAR100 classification task.

Come input layer notiamo una feature map tridimensionale di un'immagine a colori, dove le channel axis rappresentano ogni colore con una 32×32 feature

map bidimensionale. Ogni filtro di convoluzione produce un nuovo channel al primo hidden layer, ognuno è una 32 x 32 feature map. Avremo quindi una "nuova immagine" (una feature map con più canali rispetto ai canali per i tre colori in input). Questo è seguito da un max-pool layer che riduce la dimensione della feature map in ogni canale di un fattore 4 (due in ogni dimensione). Questa sequenza di convoluzione è poolin è ripetuta per i successivi due layers (fino a quando ogni feature map channel è formata da solo pochi pixels in ogni dimensione). A questo punto le feature maps tridimensionali sono *appiattite* ed unite in uno o più layers prima di raggiungere l'output layer.

8.3.4 Risultati Utilizzando un Pretrained Classifier

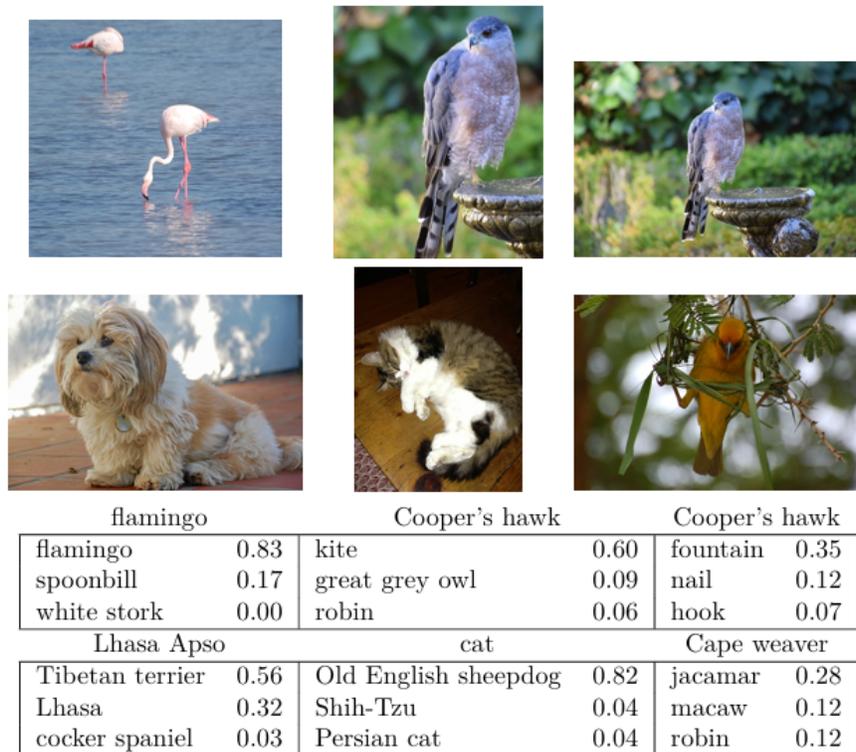


Figura 8.9: Classificazione di sei foto utilizzando la CNN `resnet50` trainata sul *corpus imagenet*.

È stato usato un classificatore pre-addestrato a livello industriale per predire la classe di queste immagini. La CNN classifica abbastanza bene l'hawk nella seconda immagine, ma una volta zoomato nella terza immagine viene confuso come fontana.

8.4 Fittare una Neural Network

Data un'osservazione $(x_i, y_i), i = 1, \dots, n$; possiamo fittare il modello risolvendo il problema dei minimi quadrati non lineare:

$$\min_{\{w_k\}_1^K, \beta} \frac{1}{2} \sum_{i=1}^n (y_i - f(x_i))^2 \quad (8.11)$$

dove

$$f(x_i) = \beta_0 + \sum_{k=1}^K \beta_k g(w_{k0} + \sum_{j=1}^p w_{kj} x_{ij}) \quad (8.12)$$

Il problema è non convesso nei parametri (quindi esistono soluzioni multiple).

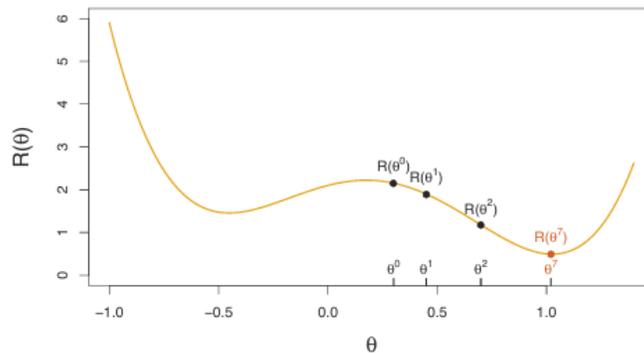


Figura 8.10: Esempio di funzione non convessa di una singola variabile θ ; ci sono due soluzioni: una è il minimo locale ($\theta = -0.46$) e l'altro quello globale ($\theta = 1.02$).

Per evitare l'overfitting sono possibili due strategie quando fittiamo le reti neurali:

- *Slow Learning*: il modello è fittato in modo iterativo lentamente, usando il *gradient descent*. Quando viene rilevato overfitting il processo si interrompe.
- *Regularization*: sono imposte delle penalità sui parametri.

Supponendo di rappresentare i parametri in un vettore lungo θ :

$$R(\theta) = \frac{1}{2} \sum_{i=1}^n (y_i - f_{\theta}(x_i))^2 \quad (8.13)$$

L'idea del gradient descent è semplice:

1. iniziamo ad ipotizzare θ^0 per tutti i parametri in θ , settando $t = 0$.

2. iteriamo finché (8.13) fallisce a decrescere:

- a) trova un vettore δ che riflette un piccolo cambiamento di θ in modo tale che $\theta^{t+1} = \theta^t + \delta \rightarrow R(\theta^{t+1}) < R(\theta^t)$.
- b) impostare $t \leftarrow t + 1$.

8.4.1 Backpropagation

Il *gradiente* di $R(\theta)$, calcolato ad un qualche valore corrente $\theta = \theta^m$, è il vettore delle derivate parziali in quel punto:

$$\nabla R(\theta^m) = \left. \frac{\partial R(\theta)}{\partial \theta} \right|_{\theta=\theta^m} \quad (8.14)$$

Questo dà la direzione nello spazio θ in cui $R(\theta)$ *incrementa* più rapidamente. L'idea del gradiente discendente è quella di muovere leggermente θ nella direzione *opposta* (dato che vogliamo scendere):

$$\theta^{m+1} \leftarrow \theta^m - \rho \nabla R(\theta^m) \quad (8.15)$$

Per un valore abbastanza basso di ρ (*learning rate*), lo step decrescerà l'obiettivo $R(\theta)$. Se il vettore gradiente è zero, allora è possibile che siamo arrivati al minimo dell'obiettivo. La *chain rule*:

$$R_i(\theta) = \frac{1}{2} \left(y_i - \beta_0 - \sum_{k=1}^K \beta_k g \left(w_{k0} + \sum_{j=1}^p w_{kj} x_{ij} \right) \right)^2 \quad (8.16)$$

Per semplificare l'espressione chiamiamo $z_{ik} = w_{k0} + \sum_{j=1}^p w_{kj} x_{ij}$. Calcoliamo prima la derivata rispetto a β_k :

$$\frac{\partial R_i(\theta)}{\partial \beta_k} = \frac{\partial R_i(\theta)}{\partial f_\theta(x_i)} \cdot \frac{\partial f_\theta(x_i)}{\partial \beta_k} = -(y_i - f_\theta(x_i)) \cdot g(z_{ik}) \quad (8.17)$$

E successivamente quella rispetto a w_{kj} :

$$\begin{aligned} \frac{\partial R_i(\theta)}{\partial w_{kj}} &= \frac{\partial R_i(\theta)}{\partial f_\theta(x_i)} \cdot \frac{\partial f_\theta(x_i)}{\partial g(z_{ik})} \cdot \frac{\partial g(z_{ik})}{\partial z_{ik}} \cdot \frac{\partial z_{ik}}{\partial w_{kj}} = \\ &= -(y_i - f_\theta(x_i)) \cdot \beta_k \cdot g'(z_{ik}) \cdot x_{ij} \end{aligned} \quad (8.18)$$

Possiamo notare come entrambe le equazioni contengano il termine residuo $y_i - f_\theta(x_i)$.

8.4.2 Dropout Learning

Inspirato dal random forest, si vuole casualmente rimuovere una frazione ϕ dell'unità in un layer quando fittiamo il modello. Questo è fatto separatamente

ogni volta che viene processata un'osservazione di training. Le unità sopravvissute devono coprire quelle mancanti e i loro pesi sono scalati di un fattore di $1/(1 - \phi)$ per compensare. In questo modo si previene che i nodi siano troppo specializzati, per questo viene visto come una forma di regolarizzazione.

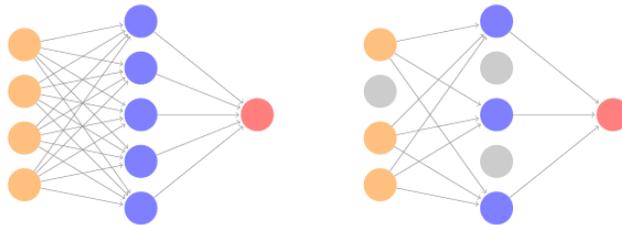


Figura 8.11: Dropout Learning

8.5 Document Classification

Considerando i ratings **IMDb** (Internet Movie Database), brevi documenti contenenti recensioni su films. La risposta in questo caso sarà il **sentiment** della review (che può essere *positivo* o *negativo*).

Ogni recensione può essere differente in lunghezza, includere slang o avere errori di spelling. Abbiamo bisogno di un metodo per *featurizzare* questo tipo di documenti.

Ecco un esempio di recensione negativa:

This has to be one of the worst film of the 1990s. When my friends & I were watching this film (being the target audience it was aimed at) we just sat and watched the first half an hour with our jaws touching the floor at how bad it really was. The rest of the time, everyone else in the just started talking to each other, leaving or generally crying into their popcorn...

Il metodo di featurizzazione più semplice e comune è la *bag-of-words*. Questo trae informazioni sul documento dalle parole utilizzate, ignorando il contesto, ad esempio considerando un *bag-of-n-grams* verrà valutata la co-occorrenza di due parole consecutive distinte ("blissfully long" viene preso come aspetto positivo di un film, "blissfully short" come negativo).

Si potrebbe registrare la frequenza delle parole. Dividendo un validation set di dimensione 2000 sulle 25000 osservazioni di training e fittando due model sequences:

- una logistic regression lasso usando il package **glmnet**
- una neural network a due classi con due hidden layers, ciascuno di 16 ReLU units

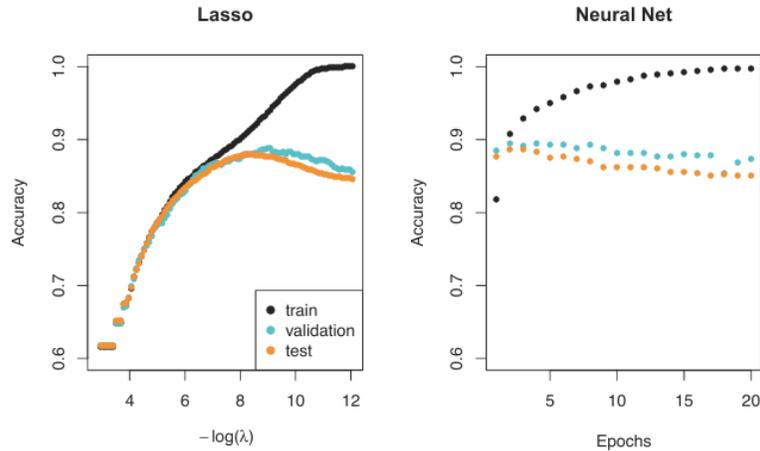


Figura 8.12: Precisione del lasso e della neural network a due classi sui dati IMDb. Entrambi tendono ad overfittare,

8.6 Recurrent Neural Networks

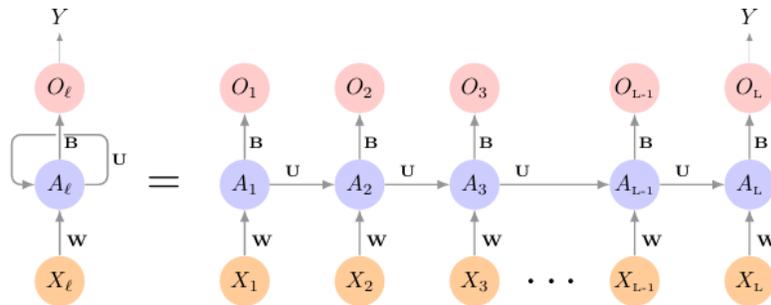


Figura 8.13: Schema di una semplice RNN.

Molti dati risultano sequenziali in natura, ad esempio:

- Documenti quali recensioni di libri e film, articoli di giornale o tweets. La sequenza e relativa posizione delle parole in questi documenti cattura la narrativa, il tema ed il tono.
- Serie nel tempo di pioggia, velocità del vento, qualità dell'aria. In maniera tale da riuscire a prevedere il meteo giorni prima.
- RegISTRAZIONI di suoni. Per trascrizioni di testo o traduzioni di discorsi.

In una RNN (*Recurrent Neural Network*) in input abbiamo l'oggetto X che è una *sequenza*. Consideriamo l'insieme di recensioni IMDb. Ogni documento può essere rappresentato come una sequenza di L parole, quindi $X = \{X_1, X_2, \dots, X_L\}$, ogni X_ℓ rappresenta una parola. L'uscita Y potrebbe anch'essa essere una sequenza, ma solitamente è uno scalare.

La sequenza dell'hidden-layer è:

$$A_{\ell k} = g \left(w_{k0} + \sum_{j=1}^p w_{kj} X_{\ell j} + \sum_{s=1}^K u_{ks} A_{\ell-1, s} \right) \quad (8.19)$$

La previsione dell'uscita invece è:

$$O_\ell = \beta_0 + \sum_{k=1}^K \beta_k A_{\ell k} \quad (8.20)$$

Considerando n coppie di ingresso sequenza/risposta (x_i, y_i) i parametri sono trovati minimizzando i minimi quadrati:

$$\sum_{i=1}^n (y_i - o_{iL})^2 = \sum_{i=1}^n \left(y_i - \left(\beta_0 + \sum_{k=1}^K \beta_k g \left(w_{k0} + \sum_{j=1}^p w_{kj} x_{iLj} + \sum_{s=1}^K u_{ks} a_{i, L-1, s} \right) \right) \right)^2 \quad (8.21)$$

9 Unsupervised Learning

Non siamo interessati alla previsione di una variabile di risposta Y , ma il nostro obiettivo è quello di fare scoperte interessanti riguardo alle misure X_1, X_2, \dots, X_p .

Gli esercizi diventano più soggettivi e non esiste un singolo obiettivo per l'analisi. Nell'unsupervised learning non c'è un modo per controllare se il lavoro fatto è corretto, perché non esiste una vera risposta.

9.1 Principal Component Analysis

Con PCA (*Principal Component Analysis*) ci riferiamo al processo per il quale calcoliamo le componenti principali ed un successivo utilizzo di queste per comprendere i dati. La PCA in un approccio unsupervised include un set di features, ma nessuna risposta associata.

Supponiamo di voler visualizzare n osservazioni con misurazioni su un set di p features per un'analisi esplorativa di dati. Potremmo fare ciò esaminando scatterplot bi-dimensionali dei dati, ognuno contenente n misure delle osservazioni su due delle features. Ci sono quindi: $\binom{p}{2} = p(p-1)/2$ scatterplots. [Ad esempio, con $p = 10$ si avranno 45 scatterplots]. Se p è molto grande esistono metodi migliori; in particolare, vogliamo cercare una rappresentazione dei dati bassa dimensionalmente che riesca a catturare più informazioni possibili.

La PCA cerca un piccolo numero di dimensioni che siano il più interessanti possibile, dove *interessante* indica di quanto le osservazioni variano in ogni dimensione. Ogni dimensione trovata dalla PCA è una combinazione lineare delle p features.

La *prima componente principale* di un set di features X_1, X_2, \dots, X_p è quella che normalizza la loro combinazione lineare in modo tale da avere la più alta varianza:

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \dots + \phi_{p1}X_p \quad (9.1)$$

Per *normalizzato* si intende $\sum_{j=1}^p \phi_{j1}^2 = 1$. Chiamiamo *loadings* (della prima componente principale) gli elementi $\phi_{11}, \dots, \phi_{p1}$. Questi vengono scelti in maniera tale che la loro somma dei quadrati sia uguale ad 1, perché se questi elementi fossero arbitrariamente grandi in valore assoluto potrebbero portare ad un'altrettanta alta varianza.

Siccome siamo interessati solamente alla varianza, assumiamo che ogni variabile in X sia centrata per avere media nulla.

Il vettore di loading della prima componente principale risolve il problema di ottimizzazione:

$$\max_{\phi_{11}, \dots, \phi_{p1}} \left\{ \frac{1}{n} \sum_{i=1}^n \left(\sum_{j=1}^p \phi_{j1} x_{ij} \right)^2 \right\} \quad (9.2)$$

Chiamiamo *scores* della prima componente principale:

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \dots + \phi_{p1}x_{ip}$$

Dopo aver determinato la prima componente principale, possiamo trovare la seconda componente principale, Z_2 . La quale è la combinazione lineare di X_1, X_2, \dots, X_p che ha varianza massima rispetto a tutte le combinazioni lineari che sono *non correlate* a Z_1 . Gli scores della seconda componente principale sono della forma:

$$z_{i2} = \phi_{12}x_{i1} + \phi_{22}x_{i2} + \dots + \phi_{p2}x_{ip}$$

Una volta calcolate le componenti principali possiamo graficarle.

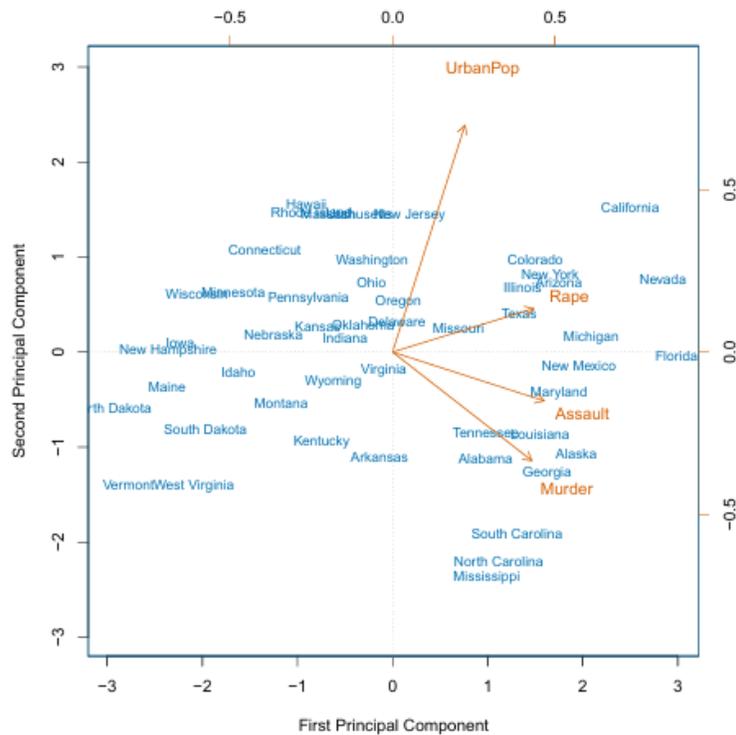


Figura 9.1: Le prime due componenti per i dati USArrest.

Per tutti i 50 stati degli US il dataset contiene il numero di arresti per 100000 residenti per ognuno dei seguenti crimini: **Assault**, **Murder**, **Rape**. È stata registrata anche l'**UrbanPop** (la percentuale di popolazione in ogni stato che vive in aree urbane). I vettori score delle componenti principali hanno lunghezza $n = 50$ ed i loading vector invece hanno $p = 4$. La PCA viene eseguita dopo aver standardizzato ogni variabile in modo da avere media zero e deviazione standard unitaria.

Nella Figura 9.1 notiamo come il primo vettore di loading ha approssimativamente un peso uguale di Assault, Murder e Rape ma un peso minore in

UrbanPop. Il secondo loading vector invece al contrario ha maggior peso su UrbanPop e meno sulle altre 3. Questo indica che le variabili legate al crimine sono correlate tra loro ma non con la variabile UrbanPop.

Notiamo dalla Figura 9.1 anche come gli stati con alti score positivi sulla prima componente (come California, Nevada, Florida) hanno alti tassi di crimine, mentre stati (come il North Dakota) che hanno scores negativi sulla prima componente hanno un minore tasso di crimine.

California ha anche un alto score sulla seconda componente, ciò indica un alto livello di urbanizzazione, mentre altri stati come il Mississippi hanno un basso livello.

Stati che hanno entrambe le componenti vicine allo zero, come l'Indiana, hanno approssimativamente un livello nella media di crimine ed urbanizzazione.

	PC1	PC2
Murder	0.5358995	-0.4181809
Assault	0.5831836	-0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

Tabella 9.1: I loading vectors ϕ_1 e ϕ_2 per il dataset USArrest

9.1.1 Un'altra Interpretazione delle Componenti Principali

Nel grafico a sinistra delle Figura 9.2 le prime tre componenti principali del data set abbracciano l'hyperplane tri-dimensionale più vicino alle n osservazioni in termini di distanza media quadrata Euclidiana.

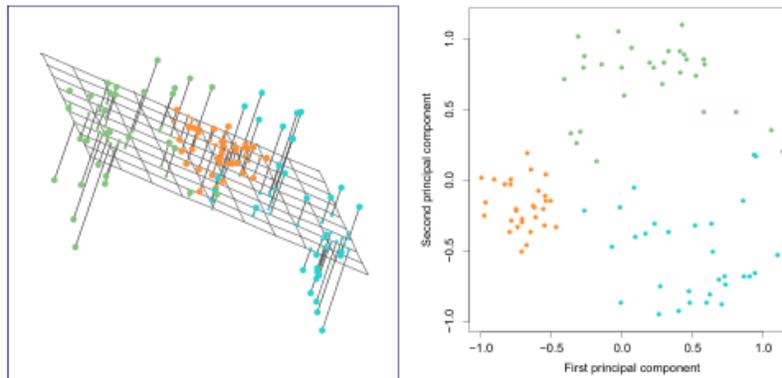


Figura 9.2: 90 osservazioni simulate in 3 dimensioni.

9.1.2 La Proporzioe della Varianza Spiegata

La *varianza totale* presente in un data set (assumendo che le variabili siano centrate ed abbiano media zero) è definita come:

$$\sum_{j=1}^p \text{Var}(X_j) = \sum_{j=1}^p \frac{1}{n} \sum_{i=1}^n x_{ij}^2 \quad (9.3)$$

La varianza spiegata dalla componente principale m-esima è:

$$\frac{1}{n} \sum_{i=1}^n z_{im}^2 = \frac{1}{n} \sum_{i=1}^n \left(\sum_{j=1}^p \phi_{jm} x_{ij} \right)^2 \quad (9.4)$$

In totale quindi si hanno un numero di di componenti principali pari al $\min(n-1, p)$; e la somma delle loro PVEs (*Proportion of Variance Explained*) è pari ad uno.

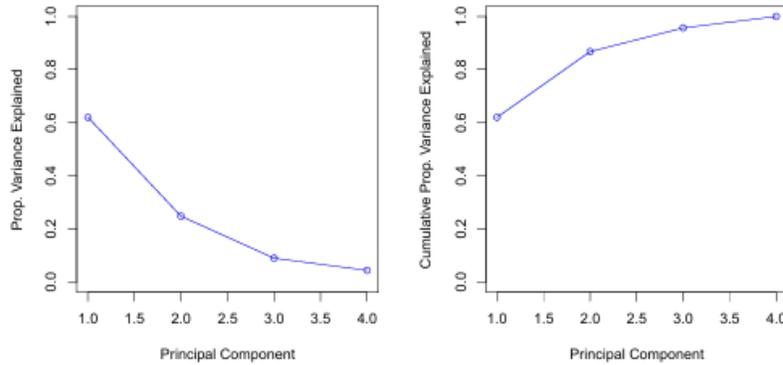


Figura 9.3: Sinistra: uno *scree plot* raffigurante la PVE delle quattro componenti principali del `USArrests` dataset. Destra: la proporzione cumulativa di varianza spiegata dalle quattro componenti principali nel `USArrests` dataset.

I risultati ottenuti applicando la PCA cambiano se le variabile vengono scalate individualmente. Generalmente è raccomandato scalare le variabili affinché abbiano deviazione standard unitaria.

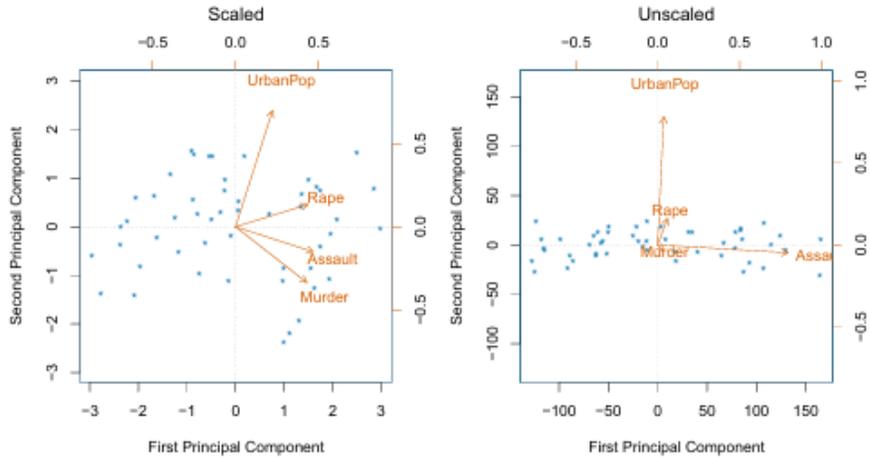


Figura 9.4: Confronto tra i grafici delle componenti principali per il dataset USArrests con variabili scalate e non.

9.2 Clustering Methods

Per *clustering* intendiamo una serie di tecniche usate per trovare sottogruppi in un data set. I due metodi di clustering più conosciuti sono il K-means (dove sappiamo in quante partizioni dobbiamo dividere) ed il clustering hierarchical (in cui non sappiamo in anticipo quanti cluster vogliamo; otteniamo infatti una rappresentazione simil tree chiamata *dendrogramma* che permette di visualizzare tutti i clustering ottenuti per ogni possibile numero di sottogruppi da 1 ad n).

9.2.1 K-Means Clustering

È un metodo semplice per dividere un data set in K cluster distinti. Per effettuare questo metodo dobbiamo prima indicare il numero K di sottogruppi desiderati; successivamente l'algoritmo assegnerà ogni osservazione esattamente ad una tra i K clusters.

Se C_1, \dots, C_K denotano un set contenenti gli indici delle osservazioni in ogni cluster; sono verificate due proprietà:

1. $C_1 \cup C_2 \cup \dots \cup C_K = \{1, \dots, n\}$. Ovvero, ogni osservazione appartiene almeno ad uno dei K clusters.
2. $C_k \cap C_{k'} = \emptyset; \forall k \neq k'$. Ovvero, i clusters non devono essere sovrapposti (una osservazione non può appartenere più di un cluster).

Un buon clustering quindi è uno che ha la *within-cluster variation* (indica di quanto variano le osservazioni l'una dall'altra in ogni cluster) il più piccolo

possibile:

$$\min_{C_1, \dots, C_K} \left\{ \sum_{k=1}^K W(C_k) \right\} \quad (9.5)$$

Dove:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i, i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 \quad (9.6)$$

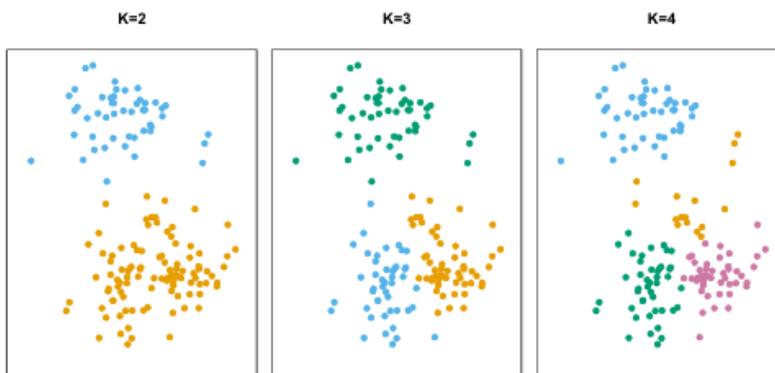


Figura 9.5: Data set simulato contenente 150 osservazioni in uno spazio bi-dimensionale. Ogni pannello mostra i risultati del K-means clustering, per diversi valori di K. I colori indicano i cluster a cui ogni osservazione è stata assegnata (questi sono casuali e potrebbero cambiare da un fit ad un altro).

Algoritmo:

1. Assegna casualmente un numero da 1 a K ad ogni osservazione (inizializzazione delle osservazioni di cluster).
2. Iterare fino a quando le assegnazioni di cluster non variano più
 - (a) per ogni K clusters calcola il *centroide*. Il k-esimo centroide del cluster è il vettore della media dei p predittori per le osservazioni nel k-esimo gruppo.
 - (b) Assegniamo ogni osservazione al cluster che ha centroide più vicino (la vicinanza è definita dalla distanza Euclidiana).

$$\frac{1}{|C_k|} \sum_{i, i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^p (x_{ij} - \bar{x}_{kj})^2 \quad (9.7)$$

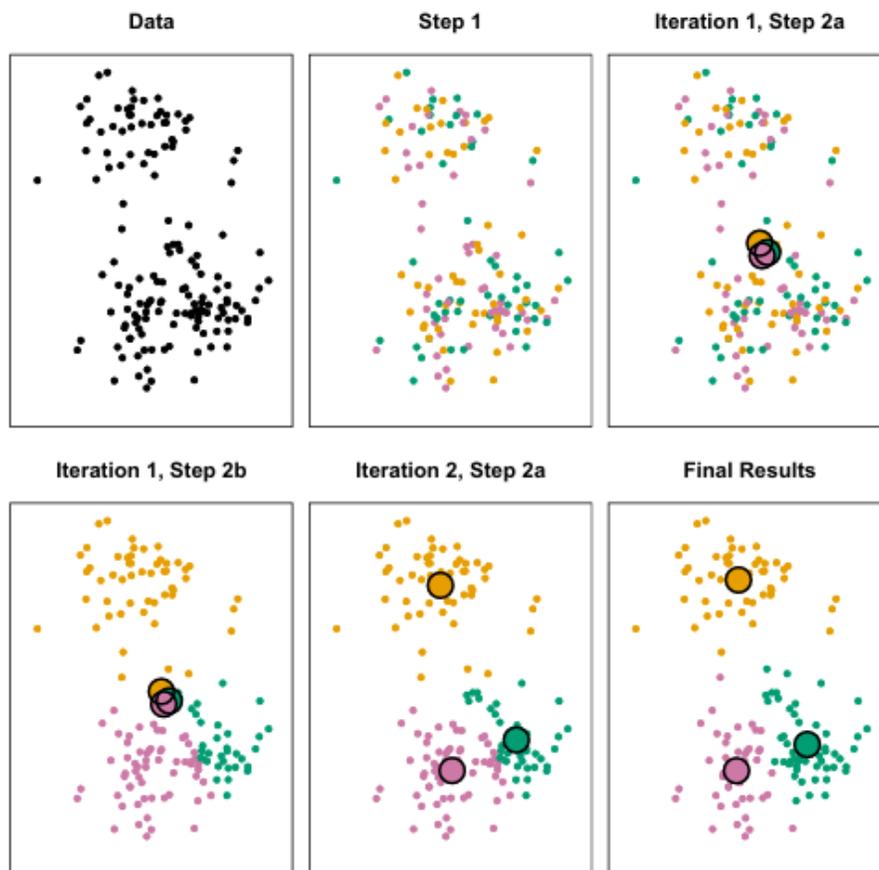


Figura 9.6: Progresso dell'algoritmo K-means (con $K=3$).

Dato che questo algoritmo trova un ottimo locale e non globale, quindi i risultati dipendono dalle condizioni (casuali) iniziali. Per questo motivo allora è importante runnare l'algoritmo diverse volte, partendo da condizioni iniziali diverse; per poi scegliere la migliore soluzione (che ha un objective value più basso).



Figura 9.7: Sei diverse volte in cui è stato runnato l’algoritmo per $K=3$. La migliore soluzione è stata ottenuta quattro volte, con un objective value segnato in rosso (di 235.8).

9.2.2 Hierarchical Clustering

Il metodo K-means ha un potenziale svantaggio che consiste nello specificare in partenza il numero di clusters. Tramite il cluster gerarchico non abbiamo bisogno di specificare K , inoltre presenta un ulteriore vantaggio, il dendrogramma (una rappresentazione simile a quella degli alberi per osservare i clusters).

Il cluster gerarchico che tratteremo è quello *bottom-up* (anche detto *agglomerante*). Iniziamo avendo il dataset simulato mostrato in Figura 9.8.

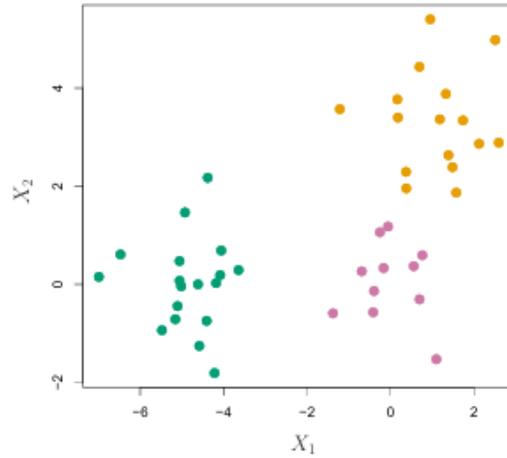


Figura 9.8: 45 osservazioni generate in uno spazio bi-dimensionale. Nella realtà ci sono 3 classi distinte (mostrate dai diversi colori) ma no tratteremo questo dataset come sconosciuto.

Nel grafico a sinistra della Figura 9.9 ogni foglia del dendrogramma rappresenta un'osservazione, man mano che ci muoviamo sopra alcune foglie vengono fuse in rami. Questo succede alle osservazioni simili tra di loro. Prima accade una fusione e maggiormente simili sono i gruppi di osservazioni tra di loro. Quindi le osservazioni che si fondono dopo possono essere molto differenti tra di loro. L'altezza della fusione misurata sull'asse verticale, quindi, indica quando due osservazioni sono diverse.

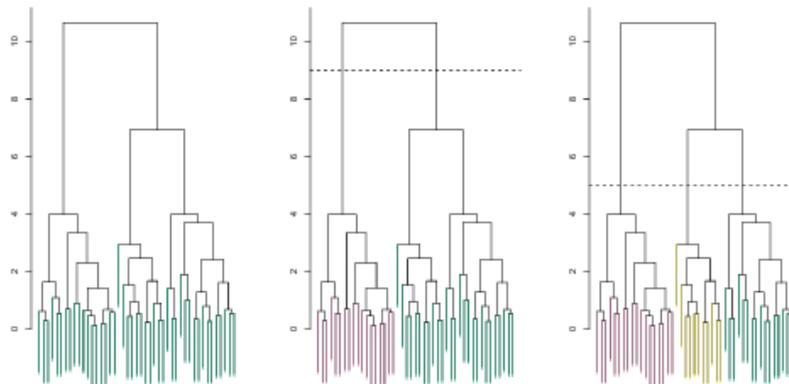


Figura 9.9: Dendrogrammi ottenuti con un completo linkage utilizzando la distanza Euclidiana (Sinistra), linkage tagliato ad un'altezza pari a 9 (Centro), linkage tagliato ad un'altezza pari a 5 (Destra).

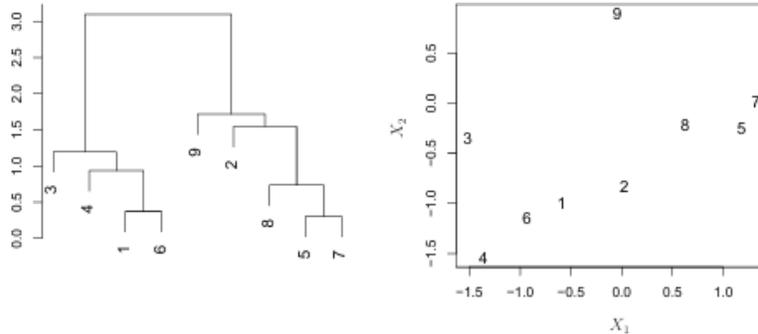


Figura 9.10: Un'illustrazione che aiuta a capire come interpretare un dendrogramma con 9 osservazioni in uno spazio bi-dimensionale.

Notiamo dalla Figura 9.10 come le osservazioni 5 e 7 sono simili perché sono fuse al punto più basso del dendrogramma. Altre osservazioni simili sono la 1 e la 6.

Matematicamente sono possibili 2^{n-1} riordinazioni del dendrogramma, con n numero di foglie. Quindi, non possiamo trarre conclusioni riguardo la similarità di due osservazioni basandoci sulla loro prossimità lungo l'asse orizzontale (ad esempio la 9 e la 2) ma dobbiamo basarci sulla distanza dell'asse verticale in cui vengono fusi i rami.

Possiamo ottenere ogni numero di clusters: 1 se non facciamo tagli, n se tagliamo ad altezza 0 (ogni osservazione coincide con il suo cluster).

Nella pratica si osserva il dendrogramma e si decide ad occhio un numero di clusters, basandoci sull'altezza delle fusioni e sul numero di clusters desiderati.

Algoritmo:

1. Inizia con n osservazioni ed una misura (come la distanza Euclidiana) di tutte le $\binom{n}{2} = n(n-1)/2$ dissomiglianze di coppia. Tratta ogni osservazione come proprio cluster.
2. per $i = n, n-1, \dots, 2$:
 - (a) Esamina tutte le coppie di dissomiglianza inter-cluster tra gli i clusters ed identifica la coppia di cluster che è più somigliante. Unisce questi due cluster. La dissomiglianza tra questi due cluster indica l'altezza nel dendrogramma del punto di fusione.
 - (b) Calcola la nuova coppia di dissomiglianza tra gli $i-1$ cluster rimasti.

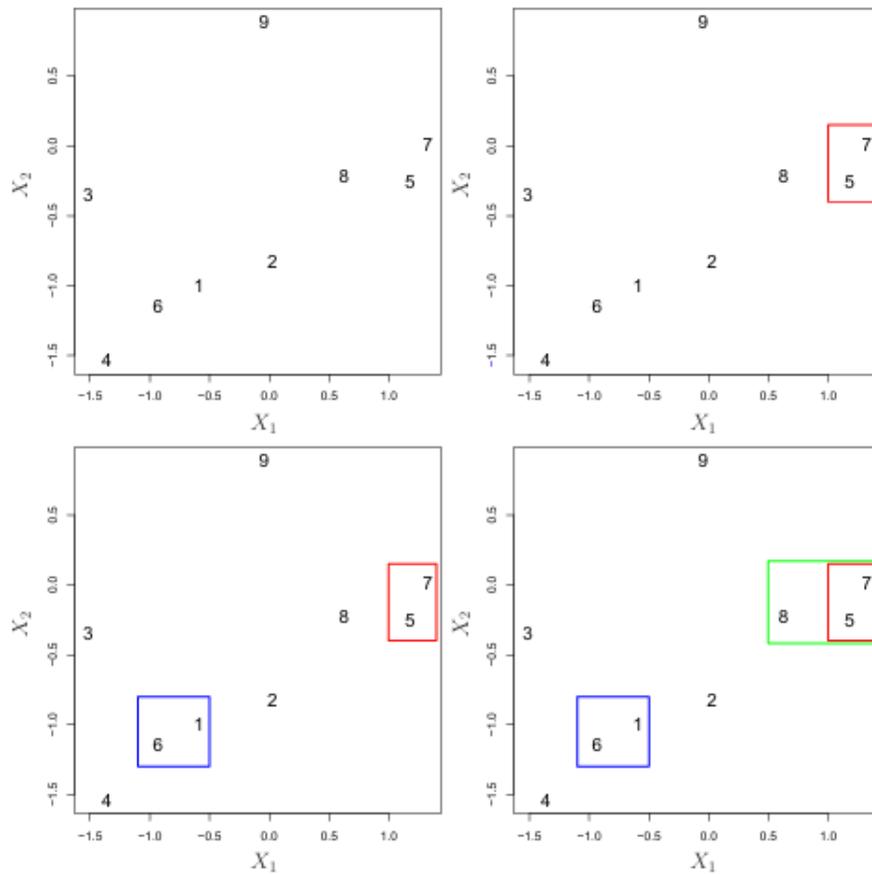


Figura 9.11: Illustrazione dei primi passi dell'algoritmo di clustering gerarchico per i dati della figura 9.10.

Se supponiamo di avere osservazioni corrispondenti ad un gruppo di persone maschi e femmine, suddivisi tra Americani, Giapponesi e Francesi. Una migliore divisione in due gruppi potrebbe essere quella di genere; mentre la miglior divisione in 3 gruppi potrebbe essere quella per nazionalità.

In questo caso in cui la migliore suddivisione in tre gruppi non dipende da quella in due e dividendo un di quei gruppi. Quindi non può essere rappresentate bene da un cluster gerarchico.

Il *linkage* definisce le dissomiglianze tra due gruppi di osservazioni. Quattro tipi sono descritti dalla Tabella 9.2.

Linkage	Description
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observation in cluster B, and record the largest of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observation in cluster B, and record the smallest of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observation in cluster B, and record the average of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable inversions.

Tabella 9.2: Sommario dei quattro tipi più comunemente usati di *linkage* nel cluster gerarchico.

I dendogrammi sono dimendenti dal tipo di linkage usato.

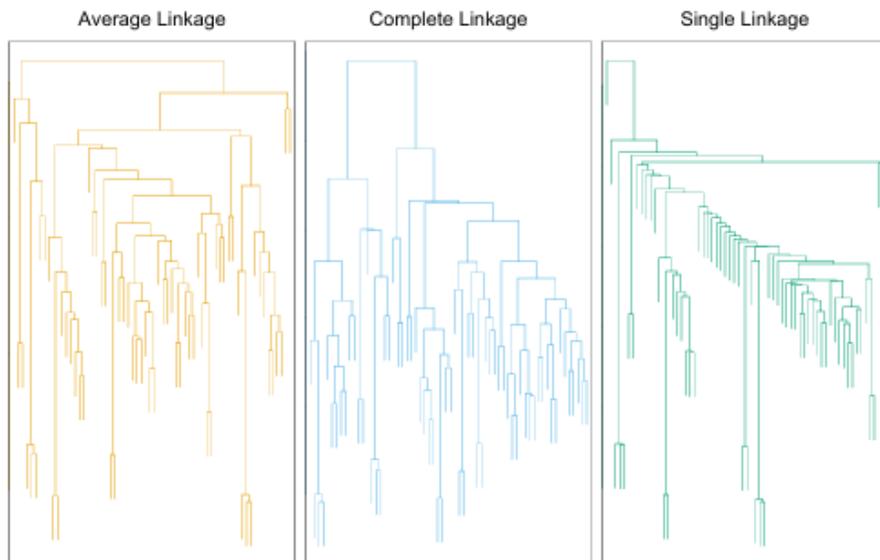


Figura 9.12: Average, Complete e Single Linkage applicato ad un dataset di esempio. Average e Complete linkage ottengono dei cluster più bilanciati.

Ad esempio, *correlation-based distance* considera due osservazioni simili se le loro features sono altamente correlate, anche se i valori osservati risultano

distanti in termini di distanza Euclidea.

Questo è un utilizzo non usuale della correlazione, che normalmente è calcolata tra variabili.

La Figura 9.13 illustra le differenze tra la distanza Euclidea e quella basata sulla correlazione.

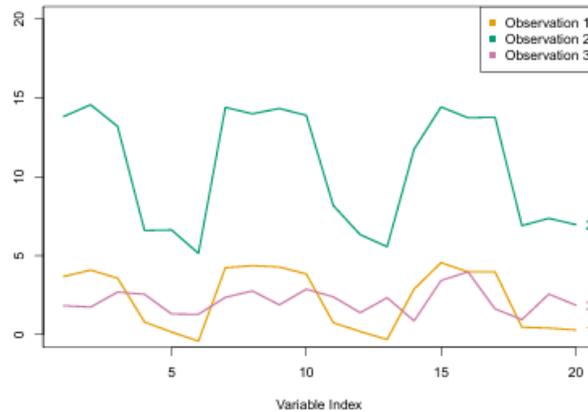


Figura 9.13: Sono mostrate 3 osservazioni con misurazioni su 20 diverse variabili. Osservazioni 1 e 3 hanno valori simili per ogni variabile, quindi hanno una piccola distanza Euclidea. Però non sono correlati, quindi hanno una maggiore correlation-based distance. Al contrario, le osservazioni 1 e 2 hanno una grande distanza Euclidea tra di loro ma sono fortemente correlati.

La distanza basata sulla correlazione si focalizza sulla forma dei profili delle osservazioni invece che delle loro magnitudini.

Il Clustering può essere un tool utile per la data analysis nell'unsuervised learning. Comunque si presentano alcuni problemi:

- Le osservazioni dovrebbero essere standardizzate (scalate per avere deviazione standard 1)?
- In caso di clustering gerarchico: Quale misura di dissomiglianza deve essere usata? Quale tipo di linkage? A che altezza dobbiamo tagliare il dendrogramma per ottenere i clusters?
- In caso di K-means clustering: Quanti cluster dobbiamo cercare nei dati?

Ognuna di queste decisioni può avere un forte impatto sui risultati ottenuti. Nella pratica proviamo diverse scelte e consideriamo quella con la soluzione più utile o interpretabile. Utilizzando questi metodi non esiste una singola risposta giusta, ma ogni soluzione che espone degli aspetti importanti del data dovrebbe essere considerata.