ANNO ACCADEMICO 2025/2026

Apprendimento Automatico

Teoria

Altair's Notes

UNIVERSITÀ DI TORINO

DIPARTIMENTO DI INFORMATICA

Capitolo 1 Introduzione Pagina 5

1.1 Che Cos'è l'Apprendimento Automatico? 5
Terminologia — 5 • Tasks — 7 • Spazio di Ipotesi — 7 • No Free Lunch Theorem — 9

1.2 Selezione del Modello e Valutazione 9
Errore Empirico e Overfitting — 9 • Metodi di Valutazione — 10 • Misure di Performance — 13 • Comparison Test — 17 • Decomposizione Bias-Varianza — 20

1.3 Modelli Lineari 21
Least Squares — 21 • Regolarizzazione — 25 • Generalizzare Modelli Lineari e Regressioni Logistiche — 26 • Classificazione Multi-Classe — 27 • Problemi di Sbilanciamento — 29

Capitolo 2 Alberi di Decisione, Reti Neurali e Classificatori Bayesiani Pagina 32

2.1 Alberi di Decisione 32
Basic Processing — 32 • Split Selection — 34 • Pruning — 36 • Continuous and Missing Values — 36 • Multivariate Decision Trees — 36

2.2 Reti Neurali 36

2.3 Classificatori Bayesiani 36

Premessa

Licenza

Formato utilizzato

Box di "Concetto sbagliato":

Concetto sbagliato 0.1: Testo del concetto sbagliato

Testo contente il concetto giusto.

Box di "Corollario":

Corollario 0.0.1 Nome del corollario

Testo del corollario. Per corollario si intende una definizione minore, legata a un'altra definizione.

Box di "Definizione":

Definizione 0.0.1: Nome delle definizione

Testo della definizione.

Box di "Domanda":

Domanda 0.1

Testo della domanda. Le domande sono spesso utilizzate per far riflettere sulle definizioni o sui concetti.

Box di "Esempio":

Esempio 0.0.1 (Nome dell'esempio)

Testo dell'esempio. Gli esempi sono tratti dalle slides del corso.

Box di "Note":

Note:-

Testo della nota. Le note sono spesso utilizzate per chiarire concetti o per dare informazioni aggiuntive.

Box di "Osservazioni":

Osservazioni 0.0.1

Testo delle osservazioni. Le osservazioni sono spesso utilizzate per chiarire concetti o per dare informazioni aggiuntive. A differenza delle note le osservazioni sono più specifiche.

1 Introduzione

1.1 Che Cos'è l'Apprendimento Automatico?

Definizione 1.1.1: Machine Learning

Un programma informatico apprende dall'esperienza E rispetto a una classe di task T e una performance P, se la sua performance nel task T, misurata da P, aumenta con l'esperienza E.

Sostanzialmente:

- L'esperienza viene data sotto forma di esempi "risolti" al computer.
- Un task (compito) da risolvere.
- Con un modo per valutare la risoluzione (performance).

1.1.1 Terminologia

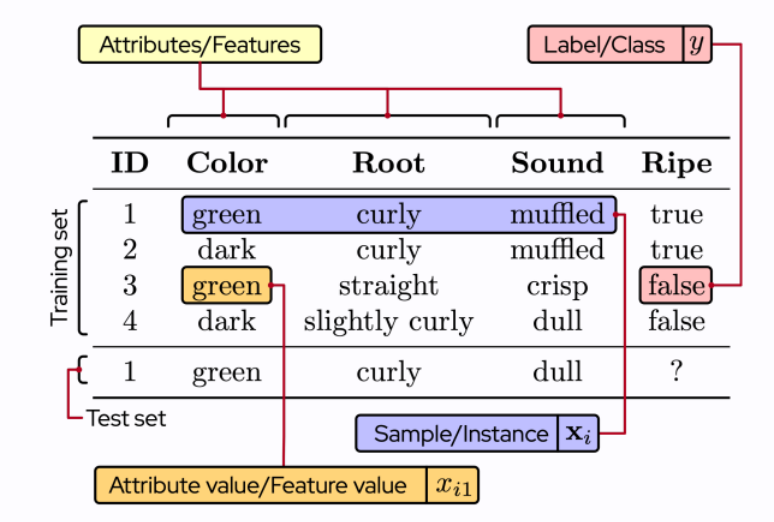


Figure 1.1: Terminologia.

- Attributi (Features): le colonne.
- Etichetta (Classe): elemento che indica come risolvere un task.
- Istanza (Sample): una riga.
- Valori: le celle.
- Set di Training: insieme su cui si va a dedurre una regola per classificare.
- Test Set: insieme per vedere quanto si sarà accurati su insiemi futuri.

Note:-

Si usa un set diverso per il training e il test perché se si usasse lo stesso il modello farebbe risultati elevati essendo addestrato su quello.

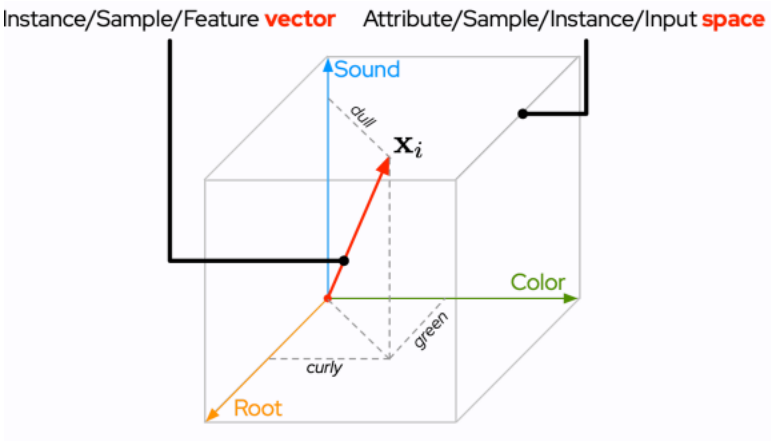


Figure 1.2: Terminologia 2.

Immaginando gli oggetti in un qualche campo euclideo:

- *Features vector:* ogni esempio corrisponde a un vettore.
- *Attribute space:* l'insieme di tutti gli esempi.

Symbol	Meaning
$\mathcal{X}$	instance space, in many cases $\mathcal{X} = \mathbb{R}^d$
$\mathcal{Y}$	label space,e.g., $\mathcal{Y} = \{0, 1\}$ for binary classification
$\mathbf{x}_i \in \mathcal{X}$	i-th instance/sample, $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]^\top$
d	dimensionality of the instance space
$y_i \in \mathcal{Y}$	label of the i-th instance, e.g., $y_i = 1$ if the watermelon is ripe, $y_i = 0$ otherwise
D	Dataset, $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$
n	number of instances in the dataset
h	model/hypothesis, a function $h : \mathcal{X} \mapsto \mathcal{Y}$ that maps instances to labels
$\mathfrak{L}$	Learning algorithm $\mathfrak{L} : \mathcal{P}(\mathcal{X} \times \mathcal{Y}) \to \mathcal{H}$, where $\mathcal{P}$ is the power set and $\mathcal{H}$ is the hypothesis space
$\mathbb{I}$	Indicator function, $\mathbb{I}(P)$ (sometime denoted as $\mathbb{I}[P]$) is 1 if P is true, 0 otherwise

Figure 1.3: Tabella di riferimento.

Definizione 1.1.2: Learning (Training)

Il learning è un processo in cui si usano algoritmi di apprendimento automatico per costruire dei modelli.

- I dati utilizzati in questo processo sono detti training data.
- Ogni istanza è un training example.
- L'insieme di tutti i training example è il training set.

Note:-

Un modello addestrato corrisponde a una serie di regole sui dati, quindi si chiama anche *ipotesi* e le regole sono i *fatti* (grounded-truth).

1.1.2 Tasks

Definizione 1.1.3: Tasks predittivi

Un task predittivo è focalizzato sul prevedere una variabile sulla base degli esempi. Si parte da problemi vecchi per trovare la soluzione a nuovi problemi.

I tasks predittivi possono essere:

- *Binari e Multi-classe:* di categorizzazione.
- *Regressivi:* con un target numerico.
- *Clustering:* un target sconosciuto.

Definizione 1.1.4: Tasks descrittivi

Un task descrittivo si concentra sul fornire regolarità nel dataset.

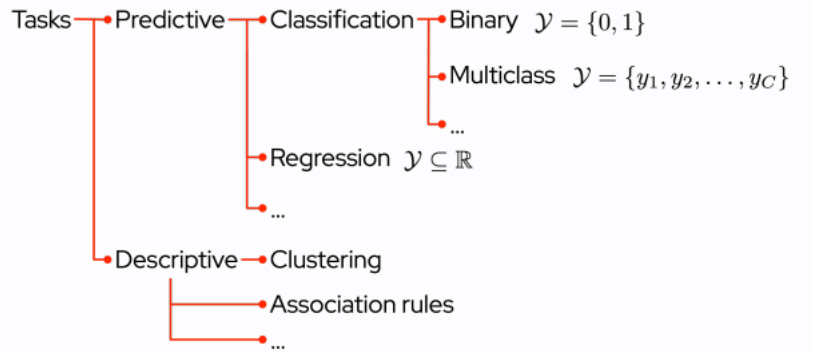


Figure 1.4: Vari tasks.

Assunzione:

- Si assume che i dati siano *indipendenti*.
- Si assume che i dati siano *identicamente distribuiti*.

1.1.3 Spazio di Ipotesi

Nei sistemi *assiomatici* il processo di derivare un teorema da assiomi è detto *deduzione*. Questo è un processo corretto se si assume che gli assiomi siano veri. L'*induzione* è il processo opposto ed è il principio su cui si basa tutto l'apprendimento automatico.

Note:-
ATTENZIONE: l'induzione come intesa in questo corso non è l'induzione matematica.

Osservazioni 1.1.1 Deduzione vs. Induzione

- La deduzione è valida: assumere le premesse vere garantisce che le conclusioni siano vere.
- L'induzione non è valida come forma di ragionamento: non garantisce che la conclusione sia vera anche se tutte le osservazioni sono corrette.

Definizione 1.1.5: Boolean Concept Learning

L'obiettivo è quello di apprendere una funzione booleana $h : \mathcal{X} \mapsto \{0, 1\}$

Note:-
Siamo nel campo del *symbolic concept learning*, studiato nel campo dellla *inductive logic programming*.

Definizione 1.1.6: Spazio di Ipotesi

Lo spazio di Ipotesi è l'insieme di tutte le possibili ipotesi che possono essere imparate da un algoritmo di apprendimento.

Note:-
Il Machine Learning è la ricerca attraverso lo spazio delle ipotesi per trovare l'insieme di tutte le ipotesi che sono consistenti con i training data e selezionare i migliori secondo un qualche criterio.

Corollario 1.1.1 Spazio delle Versioni

Sottoinsieme dello spazio delle ipotesi in cui tutte le ipotesi sono consistenti con il training data.

Definizione 1.1.7: Bias Induttivo

Il bias induttivo è un insieme di assunzioni che permette agli algoritmi di apprendimento di scegliere un'ipotesi dallo spazio delle versioni.

Osservazioni 1.1.2

- Ogni algoritmo ha un bias induttivo.
- L'unica altra possibilità sarebbe quella di scegliere a caso, ma è dumb as fuck come cosa.

Domanda 1.1

Possiamo adottare un'algoritmo con il miglior bias induttivo possibile?

Rasoio di Occam: tra ipotesi in competizione si sceglie la più semplice.

- Sarebbe troppo bello se fosse così.
- Ma cosa vuol dire "più semplice"?
  - Minor numero di parole/termini?
  - Più facile da interpretare?
  - Che faccia meno assunzioni sui dati?

1.1.4 No Free Lunch Theorem

Teorema 1.1.1 No Free Lunch Theorem

Nessun algoritmo è universalmente migliore di tutti gli altri quando la loro performance è la media su tutti i possibili problemi.

Denotiamo con:

- $P(h|X, \mathcal{L}_a)$: la probabilità che l'algoritmo $\mathcal{L}_a$ restituisca l'ipotesi h dato il training set X.
- f la funzione che si vuole apprendere.
- L'errore out-of-sample, media dell'errore sugli esempi non nel training set:

$$E_{ote}(\mathcal{L}_a \mid X, f) = \sum_h \sum_{x \in \mathcal{X}-X} P(x)\, \mathbb{I}(h(x) \neq f(x))\, P(h \mid X, \mathcal{L}_a)$$

Sommando tutte le possibili funzioni f_i otteniamo:

$$\sum_f E_{ote}(\mathcal{L}_a \mid X, f) = \sum_f \sum_h \sum_{x \in \mathcal{X}-X} P(x)\mathbb{I}(h(x) \neq f(x))P(h \mid X, \mathcal{L}_a)$$

$$= \sum_{x \in \mathcal{X}-X} P(x) \sum_h P(h \mid X, \mathcal{L}_a) \sum_f \mathbb{I}(h(x) \neq f(x))$$

$$= \sum_{x \in \mathcal{X}-X} P(x) \sum_h P(h \mid X, \mathcal{L}_a)\, \frac{1}{2} 2^{|\mathcal{X}|}$$

$$= \frac{1}{2} 2^{|\mathcal{X}|} \sum_{x \in \mathcal{X}-X} P(x) \sum_h P(h \mid X, \mathcal{L}_a)$$

$$= \frac{1}{2} 2^{|\mathcal{X}|} \sum_{x \in \mathcal{X}-X} P(x) \cdot 1$$

Note:-

Il No Free Lunch theorem si basa sull'assunzione che tutte le funzioni f sono ugualmente probabili. Ciò implica che la scelta dell'algoritmo di learning dovrebbe essere guidata dalle caratteristiche del problema.

1.2 Selezione del Modello e Valutazione

1.2.1 Errore Empirico e Overfitting

Domanda 1.2

Come misuriamo la bontà di un modello?

- *Error Rate:* si contano le predizioni sbagliate fatte, $E = \frac{a}{m}$.
- *Accuratezza:* accuracy $= 1 - E$.
- L'errore valutato sul training set è chiamato *empirical error* o *training error*.
- L'errore valutato su nuovi esempi al di fuori del training set è chiamato *generalization error* o *test error*.

Note:-

Lo scopo dell'apprendimento è quello di minimizzare il generalization error. Però non si può fare direttamente, per cui si tende a minimizzare l'errore di training.

Definizione 1.2.1: Overfitting

Quando un modello si adatti ai dati troppo bene si rischia che esso sia troppo legato ai dati del training.

Note:-

Il fenomeno opposto è l'underfitting.

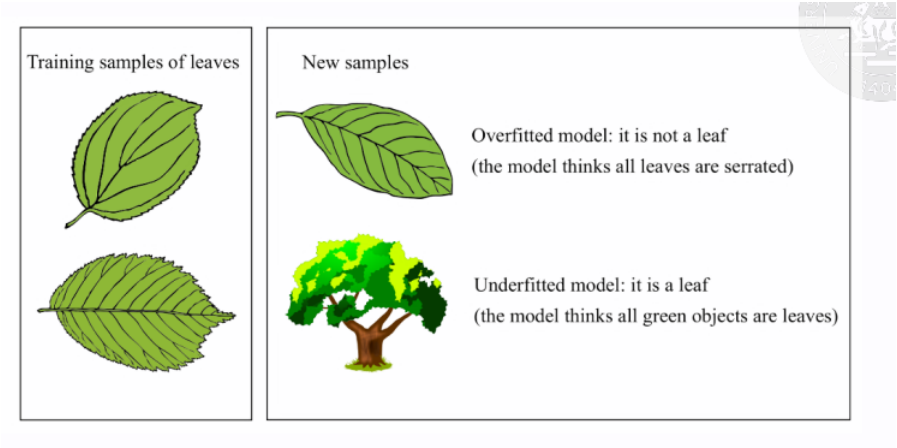


Figure 1.5: Overfitting e Underfitting.

Osservazioni 1.2.1

- L'underfitting si risolve facilmente usando modelli più complessi.
- L'overfitting richiede un bilancio tra complessità del modello e capacità di generalizzazione.
- Molti algoritmi di apprendimento automatico hanno meccanismi per prevenire l'overfitting come *tecniche di regolarizzazione* o *early stopping*.

1.2.2 Metodi di Valutazione

Quello che si vuole fare è misurare l'errore di generalizzazione su nuovi esempi.

Definizione 1.2.2: Hold-Out Validation

Dato l'insieme dei dati se ne tiene nascosta una parte all'algoritmo (test set).

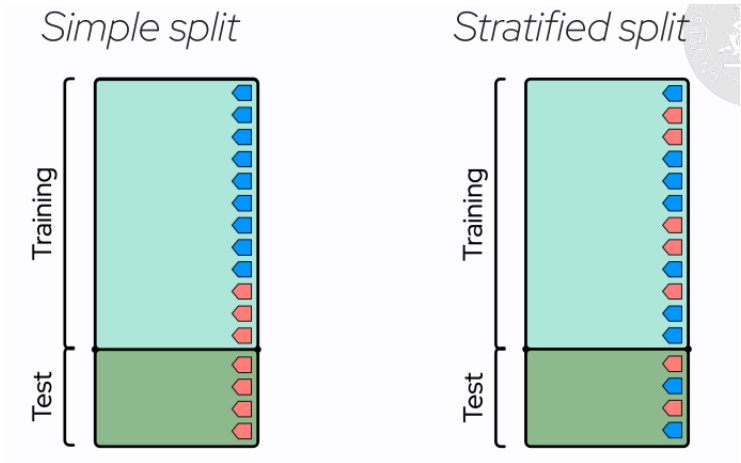


Figure 1.6: Validazione Hold-Out.

Gli split:

- Split semplice: si prendono i primi n dati come test set. I problemi sorgono quando i dati non sono ben distribuiti.
- Split stratificato: si mescolano i dati prima di estrarli.

Osservazioni 1.2.2

- Ripetere la validazione Hold-Out più volte produrra risultati diversi poiché lo split è randomico.
- Per ottenere una stima migliore dell'errore di generalizzazione possiamo ripetere più volte e fare una media.
- La media di n variabili indipendenti è una nuova variabile avente media:

$$\frac{1}{n}\sum_{i=1}^{n} E_i = \frac{1}{n}\sum_{i=1}^{n} \mu = \mu$$

- E varianza:

$$\frac{1}{n^2}\sum_{i=1}^{n} \sigma^2 = \frac{\sigma^2}{n}$$

Figure 1.7: Validazione Hold-Out con differenti split ratio.

Note:-

Se il dataset è abbastanza grande la validazione Hold-Out è semplice ed efficacie. Se il dataset è piccolo si ricorre alla *Cross-Validation*.

Definizione 1.2.3: Cross-Validation

Ripetizione dell'Hold-Out validation k volte, la prima volta si usa come test set $[0, \frac{1}{k}]$, la seconda $[\frac{1}{k}, \frac{2}{k}]$ fino all'ultima in cui si usa $[\frac{k-1}{k}, 1]$.

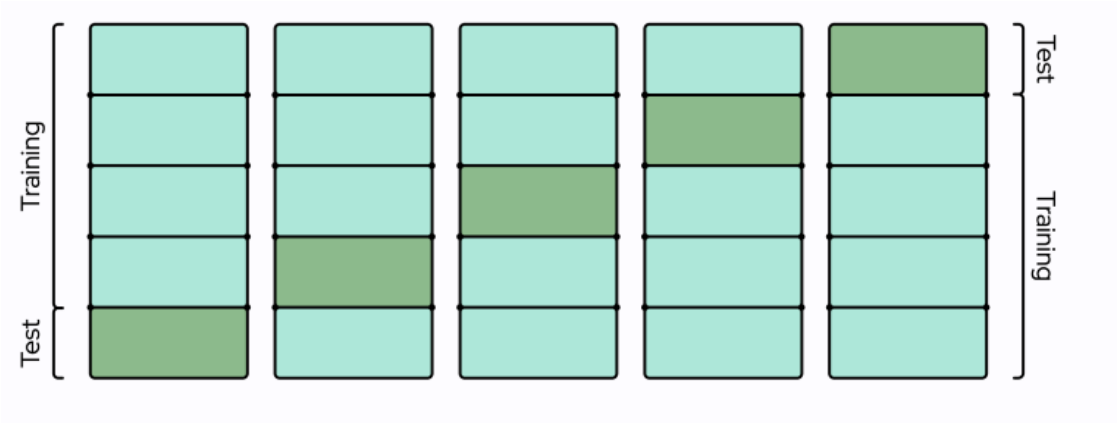


Figure 1.8: Cross-Validation.

Domanda 1.3

Perché usare la Cross-Validation invece di prendere la media degli Hold-Out ripetuti?

- Nella Cross-Validation ogni esempio è usato come test esattamente una volta, mentre ripetendo Hold-Out alcuni esempi possono essere usati più volte e alcuni mai usati.
- È più accurata, ma se il dataset è sufficientemente grande la differenza è trascurabile.

Definizione 1.2.4: Leave-One-Out

Ogni esempio è usato per testare esattamente una volra e tutti gli altri esempi sono usati per il training. L'errore di generalizzazione ha varianza molto bassa.

Note:-

È lo stimatore più preciso, ma è estremamente costoso dal punto di vista computazionale.

Definizione 1.2.5: Bootstrapping

Dato un dataset D con m esempi l'obiettivo è di stimare l'errore con l'intero dataset. L'idea è quella di creare multipli esempi di bootstrap dal dataset originale per rimpiazzare gli esempi che verranno usati per stimare l'errore.

Note:-

Gli esempi di bootstrap non sono disgiunti: diversi esempi saranno ripetuti e alcuni non compariranno.

Probabilità che un esempio non sarà selezionato: $P(sns) = (1 - \frac{1}{m})^m$

Osservazioni 1.2.3

- Consideriamo D' come bootstrap di D.
- Possiamo addestrare un modello su D' e valutare la performance su $D - D'$.
- È particolarmente utile su piccoli dataset o quando non c'è un modo decente di splittare in training e dataset.

- *Parametri:* gli oggetti modificati dall'algoritmo di apprendimento per adattarsi ai dati.

- *Hyperparametri:* parametri dell'algoritmo di apprendimento in sè e per sè.

Definizione 1.2.6: Parameter Tuning

Scegliere il miglior modello da un insieme di modelli candidati o scegliere i miglior Hyperparametri per un dato algoritmo.

Note:-

Bisogna prestare attenzione a non fare overfitting sul test set.

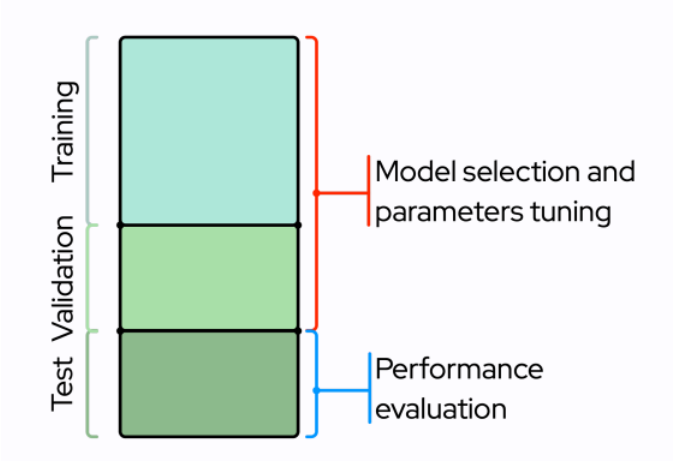


Figure 1.9: Tuning.

1.2.3 Misure di Performance

Misure di performance differenti riflettono le varie domande di task e producono diversi risultati. Scegliere la misura giusta è cruciale per una valutazione equa del modello.

Definizione 1.2.7: Mean Square Error (MSE)

È una misura per task di regressione. Se si conosce la distribuzione dei dati di D:

$$E(f;\mathcal{D}) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[(f(x)-y)^2\right] = \int_{(x,y)\sim\mathcal{D}} (f(x)-y)^2 p(x,y)\,dx\,dy$$

Però spesso non si conosce la distribuzione, per cui si stima MSE come:

$$E(f;D) = \frac{1}{m}\sum_{i=1}^{m}(f(x_i)-y_i)^2.$$

Definizione 1.2.8: Error Rate

Assumendo nota la conoscenza della distribuzione D:

$$E(f;D) = \mathbb{E}_{(\mathbf{x},y)\sim D}[\mathbb{I}(f(\mathbf{x})\neq y)] = \int_{(\mathbf{x},y)\sim D} \mathbb{I}(f(\mathbf{x})\neq y)p(\mathbf{x},y)dxdy$$

dove $\mathbb{I}$ è la funzione indicatore che restituisce 1 se la condizione è vera e 0 altrimenti.

Si può stimare come:

$$E(f;D) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}(f(\mathbf{x}_i) \neq y_i)$$

Note:-

L'accuratezza si calcola come complemento dell'error rate.

Definizione 1.2.9: Matrice di Confusione

Si tratta di una matrice 2x2 con esempi e predizioni (su righe o su colonne dipende da come gira all'autore).

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

with:

- TP: True Positive
- FP: False Positive
- TN: True Negative
- FN: False Negative

Figure 1.10: Matrice di confusione.

Altre misure:

- *Precision:* l'abilità di identificare i casi positivi.

$$\text{Precision} = \frac{TP}{TP+FP}$$

- *Recall:* l'abilità di evitare di etichettare in modo sbagliato.

$$\text{Recall} = \frac{TP}{TP+FN}$$

Osservazioni 1.2.4

- Precision e Recall sono spesso usate insieme perché complementari.
- Solitamente l'aumento di una implica la diminuzione dell'altra.

Definizione 1.2.10: P-R Plots

Assumendo di avere un classificatore che restituisce una misura di confidenza per ogni campione indicando quanto sia confidente che il campione appartenga alla classe positiva, s può variare un threshold per ottenere diversi valori di precision e recall.

Domanda 1.4

Come selezioniamo il miglior modello date poche curve P-R?

- Fissare una data precision e prendere il modello con la maggiore recall (e viceversa).
- Prendere il miglior *break-even point*: in cui precision e recall sono uguali.

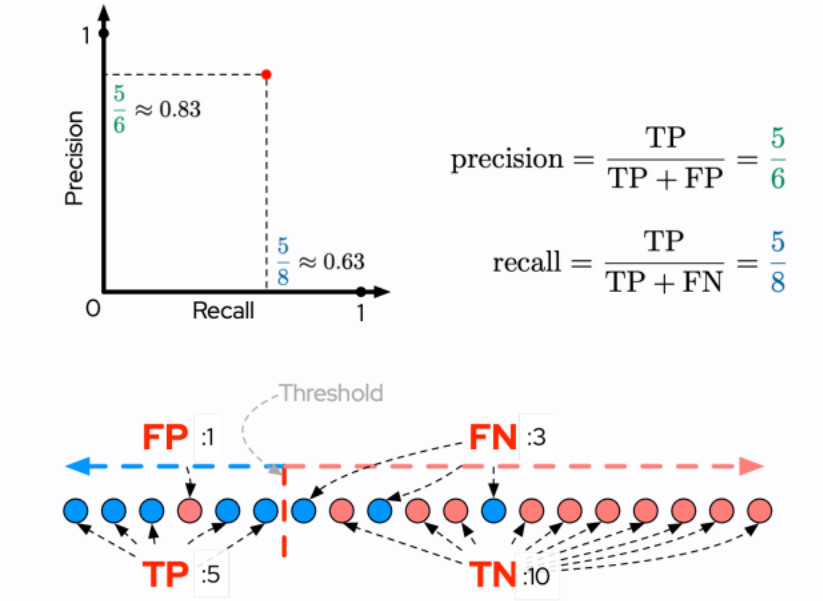


Figure 1.11: Esempio di P-R Plot.

Definizione 1.2.11: F1-Score

Media armonizzata di precision e recall:

$$F1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

Note:-

Si può generalizzare come F_β-score in cui si dà un peso diverso a precision e recall.

Per ottenere la media dei contributi di diverse matrici di confusione ci sono 2 modi:

- *Macro-averaging:* si computa la precision (P) e la recall (R) per ogni matrice di confusione.

$$\text{macro-}P = \frac{1}{n}\sum_{i=1}^{n} P_i$$

$$\text{macro-}R = \frac{1}{n}\sum_{i=1}^{n} R_i$$

$$\text{macro-}F_1 = \frac{2 \times \text{macro-}P \times \text{macro-}R}{\text{macro-}P + \text{macro-}R}$$

- *Micro-averaging:* si calcola la media degli elementi sulla matrice di confusione.

$$\text{micro-}P = \frac{\overline{TP}}{\overline{TP} + \overline{FP}}$$

$$\text{micro-}R = \frac{\overline{TP}}{\overline{TP} + \overline{FN}}$$

$$\text{micro-}F_1 = \frac{2 \times \text{micro-}P \times \text{micro-}R}{\text{micro-}P + \text{micro-}R}$$

Definizione 1.2.12: ROC Plot

Invece di usare precision e recall si fa un plot dei false positive rate (FPR) contro il true positive rate (TPR) per ottenere la Receiver Operating Characteristic curve (ROC):

$$FPR = \frac{FP}{FP + TN} = \frac{FP}{\#\text{ actual negatives}}$$

$$TPR = \frac{TP}{TP + FN} = \frac{TP}{\#\text{ actual positives}}$$

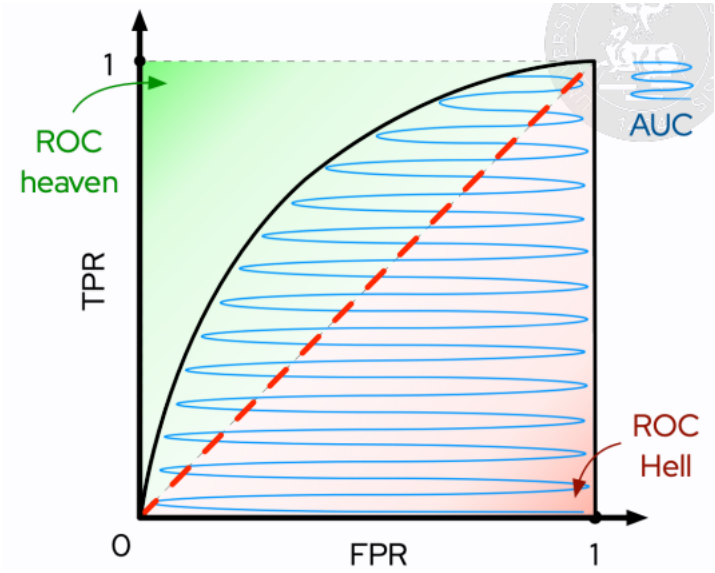


Figure 1.12: ROC Plot.

- *ROC Heaven:* classificatore perfetto, non sbaglia mai.
- *ROC Hell:* classificatore che sbaglia sempre.

Note:-

Avere un classificatore perfetto o uno che sbaglia sempre è la stessa cosa (basta prendere il contrario di cosa restituisce). Il punto peggiore è nel mezzo: tira sempre a caso.

Area sotto la curva (AUC):

$$1 - \text{AUC} = \ell_{\text{rank}} \triangle \frac{1}{m^+ m^-} \sum_{\mathbf{x}^+ \in D^+} \sum_{\mathbf{x}^- \in D^-} \mathbb{I}[f(\mathbf{x}^+) < f(\mathbf{x}^-)] + \frac{1}{2}\mathbb{I}[f(\mathbf{x}^+) = f(\mathbf{x}^-)]$$

Definizione 1.2.13: Cost-Sensitive Error Rate

Il cost-sensitive error rate si calcola come:

$$E(f; D; \text{cost}) = \frac{1}{m}\left(\sum_{(\mathbf{x},y)\in D^+} \mathbb{I}[f(\mathbf{x}) \neq y] \times \text{cost}_{01} + \sum_{(\mathbf{x},y)\in D^-} \mathbb{I}[f(\mathbf{x}) \neq y] \times \text{cost}_{10} \right)$$

Dove si dà un costo agli errori.

1.2.4 Comparison Test

Confrontare i risultati di diversi classificatori non è triviale:

- Desideriamo valutare una performance generalizzata, ma abbiamo accesso solo a un insieme di test finito.
- Le performance del modello variano su insiemi di test diversi anche se hanno la stessa dimensione.
- La randomicità di diversi algoritmi di apprendimento possono produrre modelli diversi quando addestrati diverse volte sugli stessi dati.

Definizione 1.2.14: Binomial Testing

Dato un modello f con un error rate sconosciuto ϵ si desidera testare se il modello performi meglio di un dato threshold $\epsilon_0 \in [0,1]$ avendo osservato il tasso di errore empirico $\hat{\epsilon}$ su un insieme di test di taglia m per il modello f.

Hypothesis Test:

- *Null Hypothesis:* il modello non è migliore di ϵ_0.

$$H_0 : \epsilon \geq \epsilon_0$$

- *Alternative Hypothesis:* il modello è migliore di ϵ_0.

$$H_1 : \epsilon < \epsilon_0$$

- Assumendo E come variabile casuale che conta il numero di errori commessi dal modello sull'insieme di test. Con H_0 abbiamo: $E \sim \text{Bin}(m, \epsilon_0)$.

Corollario 1.2.1 Decision Rule

Si rigetta l'ipotesi nulla H_0 a un livello α (con confidenza $1-\alpha$) se:

$$P(E \leq \hat{e} | \epsilon = \epsilon_0) < \alpha$$

Note:-

In altre parole: se è improbabile osservare poche errori concludiamo che il modello performi meglio di H_0.

Corollario 1.2.2 P-Value

Il P-Value di un test statistico è la probabilità di osservare una statistica almeno estrema quanto quella effettivamente osservata, assumendo che l'ipotesi nulla sia vera.

$$\text{p-value} = P(E \leq \hat{e} \mid \epsilon = \epsilon_0) = \sum_{k=0}^{\hat{e}} \binom{m}{k} \epsilon_0^k (1-\epsilon_0)^{m-k}$$

Corollario 1.2.3 Critical Value

Assumendo di voler ripetere un test più volte si può calcolare l'errore massimo per rigettare l'ipotesini nulla.

$$\bar{\epsilon} = \underset{\epsilon}{\text{maximize}} \quad \epsilon$$

$$\text{subject to} \quad \sum_{i=0}^{\lfloor \epsilon \times m \rfloor} \binom{m}{i} \epsilon^i (1-\epsilon)^{m-i} \leqslant \alpha$$

Definizione 1.2.15: Student's T-Test

Ripetendo Hold-Out validation o Cross-Validation si possono calcolare:

- Media:

$$\mu = \frac{1}{k}\sum_{i=1}^{k}\hat{\epsilon}_i$$

- Varianza:

$$\sigma^2 = \frac{1}{k}\sum_{i=1}^{k}(\hat{\epsilon}_i - \mu)^2$$

E così facendo si può calcolare la distribuzione:

$$\tau = \sqrt{k}\frac{\mu - \epsilon_0}{\sigma}$$

Osservazioni 1.2.5

- Il test di Student è usato per determinare se la media di due gruppi è significativamente diversa.
- Si può rigettare l'ipotesi nulla a un livello di significatività α se τ è al di fuori dell'intervallo $[-t_{\frac{\alpha}{2}}, t_{\frac{\alpha}{2}}]$.

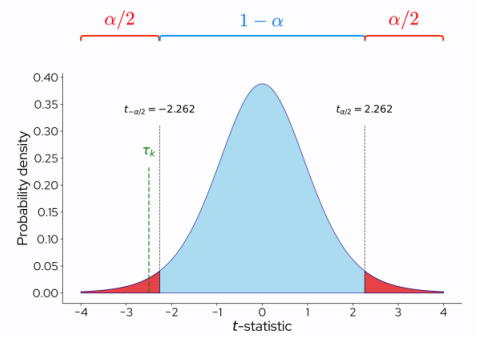


Figure 1.13: Esempio di t di Student.

Corollario 1.2.4 Cross-Validated Student's T-Test

Assumendo di avere due classificatori A e B denotando con $\epsilon_1^A, ..., \epsilon_k^A$ e $\epsilon_1^B, ..., \epsilon_k^B$ gli error rate ottenuti dalla Cross-Validation, se le loro performance sono le stesse allora i testing error rates dovrebbero essere circa gli stessi: $\epsilon_i^A \approx \epsilon_i^B$.

Un metodo più robusto è 5x2 Cross-Validated T-Test:

1. Eseguire un 2-fold Cross-Validation, con la differenza tra errori $\Delta_{1,1}$ e $\Delta_{1,2}$.
2. Ripetere il punto 1 quattro volte con mischiaggi randomici dei dati, in questo modo si ottenfono 5 paia differenti.
3. Per ognuna si va a calcolare la varianza delle differenze:

$$\sigma_i^2 = (\Delta_{i,1} - \bar{\Delta}_i)^2 + (\Delta_{i,2} - \bar{\Delta}_i)^2 \quad \text{where} \quad \bar{\Delta}_i = \frac{\Delta_{i,1} + \Delta_{i,2}}{2}$$

4. Si utilizza la differenza del primo fold[1] e la media delle stime delle 5 varianze:

$$\tau = \frac{\Delta_{1,1}}{\sqrt{\frac{1}{5}\sum_{i=1}^{5}\sigma_i^2}}$$

5. Sotto l'ipotesi nulla la statistica si distribuisce come una Student's T-Distribution con 5 gradi di libertà.

Definizione 1.2.16: McNemar's Test

Assumendo di avere due classificatori A e B e volendo comparare le loro performance tramite Hold-Out si possono addestrare sullo stesso training set e valutarli sullo stesso test set. L'idea è che sotto l'ipotesi nulla il numero di errori fatto da ogni classificatore dovrebbe essere simile.

$$T_{\chi^2} = \frac{(|e_{10} - e_{01}| - 1)^2}{e_{10} + e_{01}},$$

Decision Rule:

- Si rigetta l'ipotesi nulla a un livello di significatività α se:

$$\tau_{\chi^2} > \chi^2_{\alpha,1}$$

Note:-

Ma sia il test di Student che quello di McNemar sono pensati per confrontare due algoritmi su un singolo dataset, però spesso si vogliono confrontare più algoritmi su più dataset.

Definizione 1.2.17: Friedman's Test

Si tratta di un ranking test che permette di confrontare più algoritmi su più dataset. Se l'ipotesi nulla è rigettata si effettua un test post-hoc (solitamente Nemenyi's Test) per identificare specifiche paia di algoritmi con performance molto diverse.

In pratica:

- Assumendo che si vogliano comparare gli algoritmi A, B, C sui datasets D_1, D_2, D_3.
- Si inizia con un Hold-Out o una Cross-Validation per ottenere gli error rates di ogni algoritmo su ogni dataset.
- Si fa un ranking degli algoritmi per ogni dataset.

Error rates

Dataset	Algorithm A	Algorithm B	Algorithm C
D_1	0.2	0.3	0.4
D_2	0.1	0.2	0.2
D_3	0.05	0.1	0.2
D_4	0.3	0.4	0.5

Ranks

Dataset	Algorithm A	Algorithm B	Algorithm C
D_1	1	2	3
D_2	1	2.5	2.5
D_3	1	2	3
D_4	1	2	3
Averages	*1*	*2.125*	*2.875*

Figure 1.14: Esempio di Friedman's Test.

- Usiamo k per definire il numero di algoritmi, N per il numero dei datasets e r_i la media dell'algoritmo i.

[1] Per ammissione del professore nemmeno lui sa il perché.

- Per calcolare la statistica del test di Friedman iniziamo dalla formula:

$$\tau_{\chi^2} = \frac{12N}{k(k+1)} \left(\sum_{i=1}^{k} r_i^2 - \frac{k(k+1)^2}{4} \right)$$

- Se ci fermassimo qua ci servirebbermo molti algoritmi (circa 100-150), per cui questa statistica è corretta da *Iman* e *Davenport*:

$$\tau_F = \frac{(N-1)\tau_{\chi^2}}{N(k-1) - \tau_{\chi^2}}$$

Corollario 1.2.5 Nemenyi Post-Hoc Test

Si cofrontano le performance dei classificatori guardando le differenze tra i loro ranks. La differenza è considerata significativamente differente se è maggiore di CD:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$

Dove:

- k è il numero di algoritmi.
- N è il numero di datasets.
- q_α è il critical value della distribuzione di Student a livello di significatività α.

1.2.5 Decomposizione Bias-Varianza

Definizione 1.2.18: Decomposizione Bias-Varianza

Si tratta di un modo per capire meglio il motivo per cui un algoritmo produce un certo errore. Si tratta di 3 componenti:

- Bias: l'errore dovuto ad assunzioni sbagliate.
- Varianza: ci si sta adattando troppo al training set.
- Errori irriducibili: dovuti ai rumori nei dati.

In pratica:

- Consideriamo un problema di regressione con x come campione di test con etichetta y_D nel dataset D, con $f(x; D)$ la predizione del modello f addestrato su D.
- Il *Valore atteso* è:

$$\bar{f} = \mathbb{E}_D[f(\mathbf{x}; D)]$$

- La *Varianza* è:

$$var = \mathbb{E}_D \left[(f(\mathbf{x}; D) - \bar{f})^2 \right]$$

- Il *Rumore* è:

$$\varepsilon^2 = \mathbb{E}_D \left[(y_D - y)^2 \right]$$

- Il *Bias* è:

$$bias^2 = (\bar{f} - y)^2$$

Il valore atteso di MSE può essere scritto come:

$$
\begin{aligned}
E(f;D) &= \mathbb{E}_D[(f(\mathbf{x};D) - y_D)^2] \\
&= \mathbb{E}_D[(f(\mathbf{x};D) - \overbrace{\bar{f} + \bar{f}}^{a} - \overbrace{y_D}^{b})^2] = E_D[(a+b)^2] \\
&= \mathbb{E}_D[(f(\mathbf{x};D) - \bar{f})^2] + \mathbb{E}_D[(\bar{f} - y_D)^2] + \cancel{2\mathbb{E}_D[(f(\mathbf{x};D) - \bar{f})(\bar{f} - y_D)]} \\
&\quad \text{since } y_D \perp f_D \\
&= \mathbb{E}_D[(f(\mathbf{x};D) - \bar{f})^2] + \mathbb{E}_D[(\bar{f} - y_D)^2] \\
&= var + \mathbb{E}_D[(\bar{f} - y + y - y_D)^2] \\
&= var + \cancel{\mathbb{E}_D}[(\bar{f} - y)^2] + \mathbb{E}_D[(y - y_D)^2] + \cancel{2\mathbb{E}_D[(\bar{f} - y)(y - y_D)]} \\
&\quad \text{since } \mathbb{E}_D[y_D - y] = 0 \ (\Rightarrow E_D[y_D] = y) \\
&= var + bias^2 + \varepsilon^2
\end{aligned}
$$

Osservazioni 1.2.6

- Il bias misura la capacità dell'algoritmo di adattarsi.
- La varianza misura l'impatto delle fluttuazioni nei dati sull'apprendimento.
- Il rumore rappresenta l'errore atteso per ogni algoritmo di apprendimento per un dato task.

Note:-

Per ottenere una performance eccellente un modello ha bisogno di un piccolo bias e di una piccola varianza.

Corollario 1.2.6 Bias-Varianza Dilemma

Trovare un trade-off tra bias e varianza, poiché generalmente un algoritmo con basso bias ha alta varianza e viceversa.

1.3 Modelli Lineari

1.3.1 Least Squares

Definizione 1.3.1: Modello Lineare

Assumiamo $x = (x_1, ..., x_d)^\top$ come campione descritto da d variabili. Un modello lineare mira ad apprendere una funzione che consente di vare predizioni con una combinazione lineare delle variabili in input:

$$f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + w_{d+1} = \mathbf{w}^\top \mathbf{x} + w_{d+1}$$

In cui $\mathbf{w} = (w_1, w_2, \ldots, w_d)^\top$ è il vettore peso e w_{d+1} è il termine di bias.

Note:-

Il modello è determinato quando w e $\mathbf{w}_{d+1}$ sono appresi dai dati.

Definizione 1.3.2: Regressione Lineare

Dato un dataset $D = (x_1, y_1), ..., (x_n, y_n)$ dove $x_i = (x_{i1}, ..., x_{id})$ e $y_i \in \mathbb{R}$, la regressione lineare punta a imparare un modello lineare che possa predirre accuratamente delle vere etichette. Si cerca di ridurre la funzione:

$$\text{MSE}(\mathbf{w}, w_{d+1}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\mathbf{w}^\top \mathbf{x}_i + w_{d+1}))^2$$

Note:-

La spiegazione di questo argomento è diversa da quella del libro.

Definizione 1.3.3: Mean Square

Si ha un dataset con un certo numero di punti e le loro coordinate. Si vuole trovare la retta che passi per tutti i punti.

Note:-

È molto comoda la rappresentazione matriciale.

Per esempio, il sistema:

$$\begin{cases} w_1 \cdot (-1) + w_2 = -1.52 \\ w_1 \cdot (-0.8) + w_2 = -1.21 \\ w_1 \cdot (-0.6) + w_2 = -0.67 \\ \dots \end{cases}$$

Può essere riscritto come *matrice*:

$$\begin{bmatrix} -1 \\ -0.8 \\ -0.6 \\ \dots \end{bmatrix} [w_1] + w_2 = \begin{bmatrix} -1.52 \\ -1.21 \\ -0.67 \\ \dots \end{bmatrix} = \mathbf{X}\mathbf{w} + w_2$$

Corollario 1.3.1 Coordinate Omogenee

Incorporiamo il termine di bias b nel vettore peso (che diventa $w^i = (w_1, ..., w_d, w_{d+1})$ e aggiungiamo una colonna di 1 nella matrice X:

$$\underbrace{\begin{bmatrix} -1 & 1 \\ -0.8 & 1 \\ -0.6 & 1 \\ \dots & 1 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}}_{\mathbf{w}} = \underbrace{\begin{bmatrix} -1.52 \\ -1.21 \\ -0.67 \\ \dots \end{bmatrix}}_{\mathbf{y}} \equiv \mathbf{X}\mathbf{w} = \mathbf{y}$$

Esempio 1.3.1

Dato l'esempio:

$$\begin{aligned} x_1 &= (1, 2), \quad y_1 = 0.8 \\ x_2 &= (3, 4), \quad y_2 = 0.4 \\ x_3 &= (1, 4), \quad y_3 = 0.7 \\ x_4 &= (2, 2), \quad y_4 = 0.9 \end{aligned}$$

Quali sono la matrice X e il vettore y?
Nel caso ideale X è quadrata e l'equazione può essere semplicemente risolta moltiplicando entrambi i lati

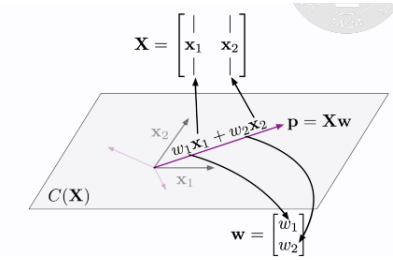
con l'inversa di X.

$$\begin{gathered}\mathbf{Xw} = \mathbf{y} \\ \Downarrow \\ \mathbf{X}^{-1}\mathbf{Xw} = \mathbf{X}^{-1}\mathbf{y} \\ \Downarrow \\ \mathbf{w} = \mathbf{X}^{-1}\mathbf{y}\end{gathered}$$

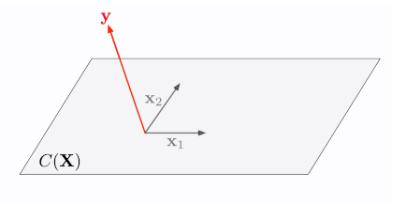
Sfortunatamente questo è vero solo nel caso triviale. X non è solitamente invertibile ed è necessario un approccio più generale.

Quindi, utilizzando un approccio geometrico:

- Iniziamo notando che variando w ci stiamo muovendo nello spazio delle colonne di X.

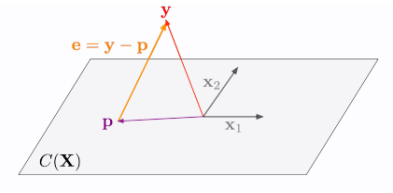


- Dato che il sistema di equazioni non è risolvibile si capisce che nessuna combinazione di w_i consente di ottenere y, per cui y non appartiene al piano C(X).

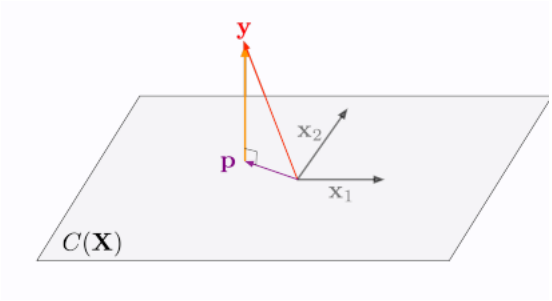


- Il miglior modo è quello di trovare il vettore w che minimizza la lunghezza del vettore errore $e = y - Xw$. In altre parole vogliamo trovare:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{e}\| = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{e}\|^2 = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{Xw}\|^2$$



- Come scegliamo il miglior punto sul piano $C(X)$ per approssimare y?
- Il miglior punto è la proiezione ortogonale di y su $C(X)$. Quel punto corrisponde a $p = Xw$ dove w è il vettore peso ottimale.



- Per imporre l'ortogonalità al piano si impone che sia ortogonale a tutti i vettori del piano:

$$\mathbf{x}_1 \perp \mathbf{e} \Rightarrow \mathbf{x}_1^\top \mathbf{e} = 0$$
$$\mathbf{x}_2 \perp \mathbf{e} \Rightarrow \mathbf{x}_2^\top \mathbf{e} = 0$$
$$\dots$$

- Che può essere scritto come $X^\top e = 0$.
- La soluzione si può trovare risolvendo:

$$\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$$
$$\mathbf{X}^\top\mathbf{y} - \mathbf{X}^\top\mathbf{X}\mathbf{w} = \mathbf{0}$$
$$\mathbf{X}^\top\mathbf{X}\mathbf{w} = \mathbf{X}^\top\mathbf{y}$$
$$\mathbf{w} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

Note:-

Se i dati sono pochi o rumorosi la matrice $X^\top X$ può essere malcondizionata rendendo difficile l'inversione. In tali casi si effettuano tecniche di regolarizzazione per stabilizzare la soluzione.

Un'altra possibilità per derivare la soluzione di Least Squares è l'analisi:

- Possiamo riscrivere il problema come:

$$\begin{aligned}\underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 &= \underset{\mathbf{w}}{\operatorname{argmin}} (\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \underset{\mathbf{w}}{\operatorname{argmin}} \left(\mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\mathbf{w} + \mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w}\right)\end{aligned}$$

- Differenziando rispetto a w e impostando la derivata a 0 otteniamo:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{w}} \left(\mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\mathbf{w} + \mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w}\right) &= 0 \\ -2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\mathbf{w} &= 0 \\ \mathbf{X}^\top\mathbf{X}\mathbf{w} &= \mathbf{X}^\top\mathbf{y} \\ \mathbf{w} &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}\end{aligned}$$

- Dove:

$$\frac{\partial}{\partial \mathbf{w}} \left[\mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w}\right] = 2\mathbf{X}^\top\mathbf{X}\mathbf{w}$$

- E:

$$\frac{\partial}{\partial \mathbf{w}} \left[\mathbf{y}^\top\mathbf{X}\mathbf{w}\right] = \mathbf{X}^\top\mathbf{y}$$

1.3.2 Regolarizzazione

Il metodo Least Squares è uno strumento potente, ma non è immune all'overfitting.

Definizione 1.3.4: Regolarizzazione

La regolarizzazione è un metodo generale per evitare l'overfitting applicando vincoli aggiuntivi al vettore peso. Un approccio comune consiste nell'assicurarsi che i pesi sono piccoli in magnitudo (shrinkage).

La versione regolarizzata del Least Squares è:

$$\arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|_p^p$$

Note:-

Dove λ è un Hyperparametro che controlla la quantità di regolarizzazione e p è la norma usata per misurare la dimensione dei pesi.

Corollario 1.3.2 Ridge Regression

Nel caso di norma L_2 la versione regolarizzata di Least Squares ha una soluzione in forma chiusa:

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$$

Dove I è la matrice identità.

Osservazioni 1.3.1

- Una matrice è invertibile se e solo se tutti i suoi autovalori sono non-zero.
- Aggiungere λ ci assicura che tutti gli autovalori sono reali e non-negativi.

Corollario 1.3.3 Lasso Regression

Nel caso di norma L_1:

$$\|\mathbf{w}\|_1 = \sum_i |w_i|$$

Il risultato è che alcuni pesi sono ridotti e altri settati a 0.

Note:-

Lasso Regression favorisce soluzioni sparse.

Domanda 1.5

Perché i metodi di regolarizzazione funzionano?

- Se si assume che X è affetta da un errore D allora: (X + D)w = Xw + Dw.
- Per cui minimizzare la norma del vettore peso minimizza gli effetti dell'errore su X.
- Limitando i pesi introduciamo un bias verso modelli più semplici riducendo la varianza:
  - Una grande riduzione nella varianza compensa un piccolo incremento nel bias, riducendo gli errori.
  - Possiamo utilizzare il rasosio di Occam, assumendo che modelli più semplici generalizzano meglio.

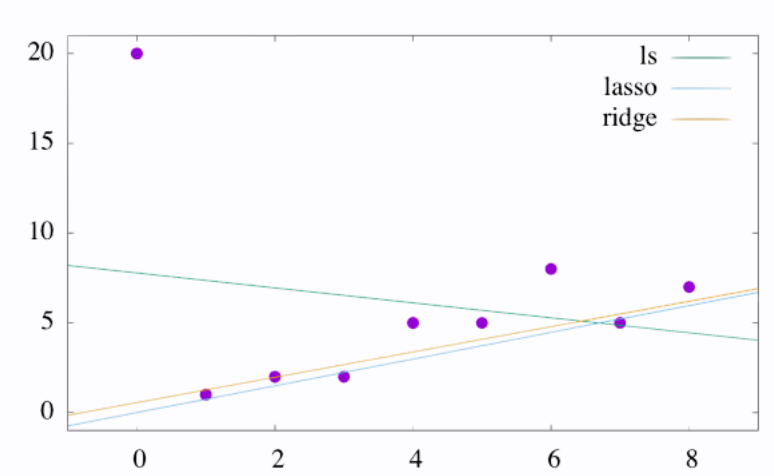


Figure 1.15: Confronti tra regressioni.

1.3.3 Generalizzare Modelli Lineari e Regressioni Logistiche

Definizione 1.3.5: Generalized Linear Model (GLM)

GLM estende i modelli lineari per permettere diversi diversi tipi di risposta. Si usa una link function per collegare il predittore lineare al valore atteso:

$$y = g^{-1}(\mathbf{w}^\top \mathbf{x} + b)$$

Dove g è la link function.

Note:-

Per semplificare utilizziamo la variabile y invece del suo valore atteso.

Domanda 1.6

Come possiamo risolvere dei problemi di classificazione utilizzando i modelli lineari?

- Possiamo cercare una funzione monotona differenziabile g che collega l'output della regressione alle etichette di classificazione.
- Nello specifico vogliamo convertire l'output del modello lineare in etichette binarie (0/1). Si potrebbe utilizzare una funzione:
$$y = f(z) = \begin{cases} 0, & \text{if } z < 0 \\ 0.5, & \text{if } z = 0 \\ 1 & \text{if } z > 0 \end{cases}$$
- Tuttavia questa funzione non è differenziabile.
- Per cui si usa la *funzione logistica*:
$$y = f(z) = \frac{1}{1 + e^{-z}}.$$
- E la sua inversa:
$$z = f^{-1}(y) = \ln \frac{y}{1 - y}$$
- Usando questa funzione e la sua inversa si può modellare la probabilità che un certo input appartenga a una particolare classe. La probabilità della classe positiva:
$$y' = p(y = 1|\mathbf{x}) = g^{-1}(\mathbf{w}^\top \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x} + b)}}$$

- Applicando la funzione di link a entrambi i membri otteniamo:

$$g(y') = \ln \frac{y'}{1-y'} = \ln \frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})} = g(g^{-1}(\mathbf{w}^\top \mathbf{x} + b)) = \mathbf{w}^\top \mathbf{x} + b$$

- Per cui:

$$\mathbf{w}^\top \mathbf{x} + b = \ln \frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})}$$

Note:-

Si può fare lo stesso partendo dai log-odds.

Definizione 1.3.6: Maximum Likelihood Estimation (MLE)

L'idea è quella di trovare i parametri w e b che massimizzano la likelihood dei dati osservati. Si utilizza una distribuzione di Bernoulli[a]:

$$p(y_i|\mathbf{x}_i) = p(y_i = 1|\mathbf{x}_i)^{y_i} \cdot p(y_i = 0|\mathbf{x}_i)^{1-y_i}$$

[a] Usata negli esperimenti binari.

- Per trovare i parametri ottimali possiamo lavorare con la log-likelihood. Assumendo $z_i = w^\top x_i + b$:

$$\begin{aligned}
\ln p(y_i|\mathbf{x}_i) &= y_i \ln p(y_i = 1|\mathbf{x}_i) + (1 - y_i) \ln p(y_i = 0|\mathbf{x}_i) \\
&= y_i \ln \frac{e^{z_i}}{1 + e^{z_i}} + (1 - y_i) \ln \frac{1}{1 + e^{z_i}} \\
&= y_i \left(z_i - \ln(1 + e^{z_i})\right) - (1 - y_i) \ln(1 + e^{z_i}) \\
&= y_i z_i - y_i \ln(1 + e^{z_i}) - \ln(1 + e^{z_i}) + y_i \ln(1 + e^{z_i}) \\
&= y_i z_i - \ln(1 + e^{z_i})
\end{aligned}$$

- Si vuole minimizzare la funzione costo:

$$\ell(\boldsymbol{\beta}; \mathbf{x}_i', y_i) = -\ln p(y_i|\mathbf{x}_i') = \ln(1 + e^{\boldsymbol{\beta}^\top \mathbf{x}_i'}) - y_i(\boldsymbol{\beta}^\top \mathbf{x}_i')$$

- Per un dataset con n osservazioni indipendenti la likelihood totale è il prodotto delle likelihood individuali:

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \ell(\boldsymbol{\beta}; \mathbf{x}_i', y_i) = \sum_{i=1}^{n} \left(\ln(1 + e^{\boldsymbol{\beta}^\top \mathbf{x}_i'}) - y_i(\boldsymbol{\beta}^\top \mathbf{x}_i')\right)$$

- Per minimizzare la funzione di loss:

$$\boldsymbol{\beta}^\star = \arg\min_{\boldsymbol{\beta}} \ell(\boldsymbol{\beta})$$

1.3.4 Classificazione Multi-Classe

Definizione 1.3.7: Metodi di Trasformazione del Problema

Questi metodi puntano a ridurre il problema della multiclassificazione in una serie di problemi di classificazione binaria.

Idea principale:

- Si divide il problema in una serie di piccoli task di classificazione binaria.
- Per ogni task si crea un nuovo dataset e si addestra un semplice classificatore binario.
- Dopo di ché si combinano i risultati per fare la predizione finale.

Ci sono diverse alternative a questo approccio basico:

- One-vs-One (OVO[2]).
- One-vs-Rest (OvR).
- Many-vs-Many (MvM).
- Error-Correcting Output Codes (ECOC).

Definizione 1.3.8: Matrice di Codifica

Una matrice di codifica è una matrice che definisce la relazione tra le classi originali ei classificatori binari.

	h_1	h_2	h_3	h_4
C_1	1	1	−1	−1
C_2	1	−1	1	−1
C_3	−1	1	−1	1
C_4	−1	−1	1	1

Osservazioni 1.3.2

- Ogni colonna corrisponde a un classificatore binario che distingue tra due gruppi di classi, le entries nella matrice indicano quali classi devono essere considerate positive o negative per ciascun classificatore.
- La riga nella matrice di codifica che è più simile al vettore di predizione indica la *classe predetta*.

Definizione 1.3.9: One-vs-One

Assumiamo di avere k classi, addestriamo un classificatore binario per ogni paia di classi. Il risultato sono $\frac{k(k-1)}{2}$ classificatori.

Esempio 1.3.2 (One-vs-One)

	h_{12}	h_{13}	h_{14}	h_{23}	h_{24}	h_{34}
C_1	1	1	1	0	0	0
C_2	−1	0	0	1	1	0
C_3	0	−1	0	−1	0	1
C_4	0	0	−1	0	−1	−1

Definizione 1.3.10: One-vs-Rest

Addestriamo k classificatori binari, ognuno addestrato per distinguere una classe da tutte le altre.

Esempio 1.3.3 (One-vs-Rest)

	h_1	h_2	h_3	h_4
C_1	1	−1	−1	−1
C_2	−1	1	−1	−1
C_3	−1	−1	1	−1
C_4	−1	−1	−1	1

[2] OwO

Definizione 1.3.11: Many-vs-Many

Ogni classificatore binario è addestrato su un sottoinsieme delle classi dove alcune classi sono trattate come positive e altre come negative.

Esempio 1.3.4 (Many-vs-Many)

	h_1	h_2	h_3	h_4	h_5	h_6
C_1	1	1	−1	−1	−1	−1
C_2	1	−1	−1	1	1	1
C_3	1	1	1	−1	1	1
C_4	−1	−1	1	−1	1	−1

Note:-

Sia OvO che OvR sono casi speciali di MvM.

Definizione 1.3.12: Error-Correcting Output Codes

La matrice di codifica è realizzata in modo tale che le righe sono ben separate dalla distanza di Hamming. Questa separazione permette di correggere errori fatti da predizioni individuali.

Osservazioni 1.3.3

- Trovare ECOC ottimale per un dato numero di classi è NP-Hard.
- Utilizzare una matrice di codifica randomica può essere una buona euristica se il codice è abbastanza lunga.
- Quando k è piccolo l'efficacia è limitata.

OvO vs OvR:

- OvO:
  - Ogni classificatore è addestrato solo un sottoinsieme dei dati, quindi ha efficacia limitata se i dati sono scarsi.
  - Può essere costosa quando ci sono tante classi.
- OvR:
  - Produce sottoproblemi binari molto sbilanciati, anche se il dataset originale era bilanciato.
  - Usa l'intero dataset per ogni classificatore binario, ciò aumenta l'efficienza.

1.3.5 Problemi di Sbilanciamento

Assumiamo di avere un dataset molto sbilanciato dove la classe di maggioranza ha 998 istanze e la classe di minoranza ne ha solo 2. Un classificatore può ottenere 99.8% di accuratezza predicendo per tutte le istanze quelle di maggioranza, però ciò fallisce nell'identificare la classe di minoranza che spesso è la *classe di interesse*.

Definizione 1.3.13: Threshold Update

Assumendo di usare un classificatore lineare $y = \mathbf{w}^\top \mathbf{x} + b$ e che le classi con $y > 0.5$ vengono classificate positivamente, se abbiamo m^+ istanze positive e m^- istanze negative possiamo utilizzare un likelihood ratio:

$$\frac{y}{1-y} > \frac{m^+}{m^-} \iff y > \frac{m^+}{m^+ + m^-}$$

Corollario 1.3.4 Oversampling

I metodi di oversampling mirano a bilanciare la distribuzione delle classi aumentando il numero di istanze nella classe di minoranza.

Osservazioni 1.3.4

- Il metodo più semplice consiste nel duplicare casualmente gli esempi nella classe di minoranza.
- Può portare all'overfitting.
- *SMOTE* (Synthetic Minority Oversampling Technique): si prendono esempi a coppie e si interpolano a vicenda prendendo i punti nel mezzo.
- Ciò introduce un bias.

Corollario 1.3.5 Undersampling

Si riducono gli esempi delle classi di maggioranza.

Osservazioni 1.3.5

- Si rischia di perdere informazione (problematico per piccoli datasets).
- *EasyEnsemble:* si inducono più classificatori e si prende il voto di maggioranza.

2

Alberi di Decisione, Reti Neurali e Classificatori Bayesiani

2.1 Alberi di Decisione

2.1.1 Basic Processing

Processo di base:

- Ogni domanda posta nel processo di decisione è un test su una feature.
- Alla fine del percorso si raggiunge una conclusione.
- Ogni test raggiunge o un nodo foglia o un nodo interno.
- Ogni percorso corrisponde a una sequenza di test (predicato congiuntivo).
- *Goal:* produrre un albero che può generalizzare su campioni sconosciuti[1].

Algorithm 4.1 Decision Tree Learning.

Input: Training set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$;
Feature set $A = \{a_1, a_2, \ldots, a_d\}$. {ID , color, root, sound, texture, umbilicus, surface}

Process: Function TreeGenerate(D, A)

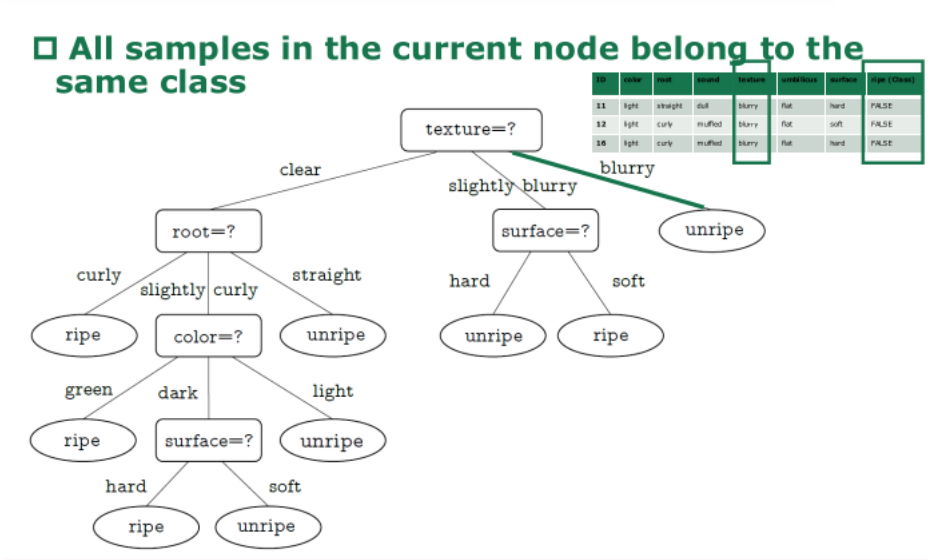
```
1: Generate node i;
2: if All samples in D belong to the same class C then
3:     Mark node i as a class C leaf node; return
4: end if
5: if A = ∅ OR all samples in D take the same value on A then
6:     Mark node i as a leaf node, and its class label is the majority class in D; return
7: end if
8: Select the optimal splitting feature a* from A;
9: for each value a*^v in a* do
10:    Generate a branch for node i; Let D_v be the subset of samples taking value a*^v on a*;
11:    if D_v is empty then
12:        Mark this child node as a leaf node, and label it with the majority class in D; return
13:    else
14:        Use TreeGenerate(D_v, A\{a*}) as the child node.
15:    end if
16: end for
```

Output: A decision tree with root node i.

(1) All samples in the current node belong to the same class.

Figure 2.1: Algoritmo per l'apprendimento basato su decision tree.

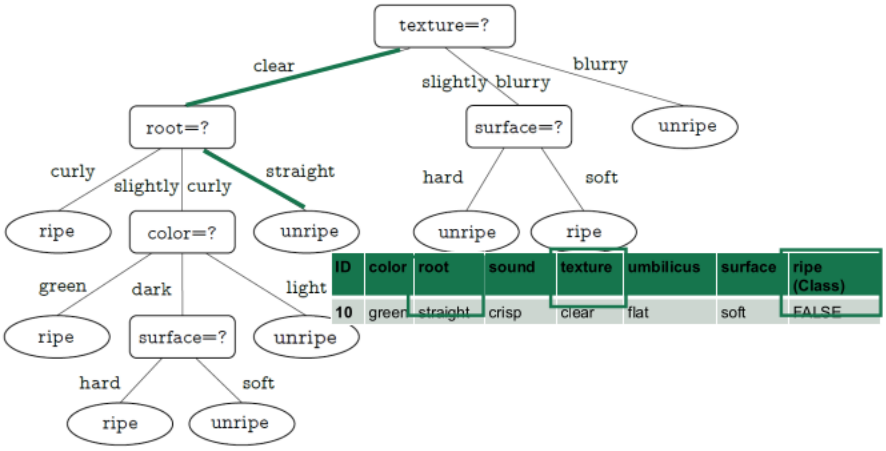
[1] Il "salto induttivo", per Peter Flach.



Algorithm 4.1 Decision Tree Learning.

Input: Training set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$;
Feature set $A = \{a_1, a_2, \ldots, a_d\}$. {ID , color, root, sound, texture, umbilicus, surface}
Process: Function TreeGenerate(D, A)
1: Generate node i;
2: **if** All samples in D belong to the same class C **then**
3: Mark node i as a class C leaf node; **return**
4: **end if**
5: **if** $A = \varnothing$ **OR** all samples in D take the same value on A **then**
6: Mark node i as a leaf node, and its class label is the majority class in D; **return**
7: **end if**
8: Select the optimal splitting feature a_* from A;
9: **for** each value a_*^v in a_* **do**
10: Generate a branch for node i; Let D_v be the subset of samples taking value a_*^v on a_*;
11: **if** D_v is empty **then**
12: Mark this child node as a leaf node, and label it with the majority class in D; **return**
13: **else**
14: Use TreeGenerate(D_v, $A \backslash \{a_*\}$) as the child node.
15: **end if**
16: **end for**
Output: A decision tree with root node i.

(2) The current feature set is empty, or all samples have the same feature values.



Algorithm 4.1 Decision Tree Learning.

Input: Training set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$;
Feature set $A = \{a_1, a_2, \ldots, a_d\}$. {ID , color, root, sound, texture, umbilicus, surface}
Process: Function TreeGenerate(D, A)
1: Generate node i;
2: **if** All samples in D belong to the same class C **then**
3: Mark node i as a class C leaf node; **return**
4: **end if**
5: **if** $A = \varnothing$ **OR** all samples in D take the same value on A **then**
6: Mark node i as a leaf node, and its class label is the majority class in D; **return**
7: **end if**
8: Select the optimal splitting feature a_* from A;
9: **for** each value a_*^v in a_* **do**
10: Generate a branch for node i; Let D_v be the subset of samples taking value a_*^v on a_*;
11: **if** D_v is empty **then**
12: Mark this child node as a leaf node, and label it with the majority class in D; **return**
13: **else**
14: Use TreeGenerate(D_v, $A \backslash \{a_*\}$) as the child node.
15: **end if**
16: **end for**
Output: A decision tree with root node i.

(3) There is no sample in the current node.

Figure 2.2

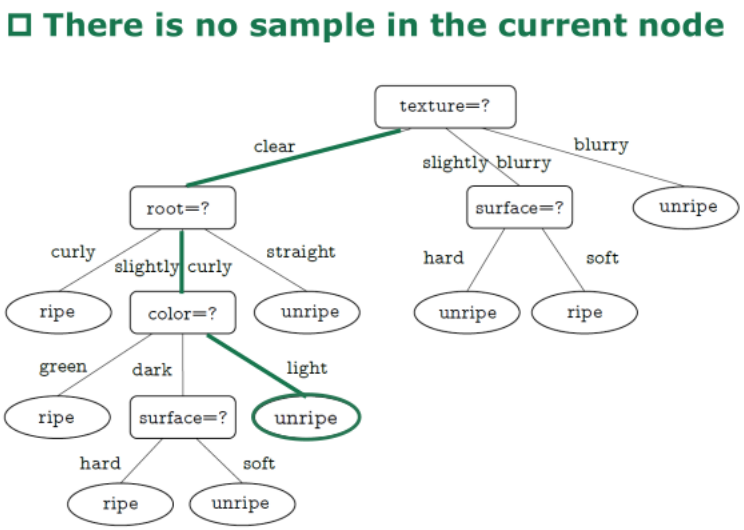


Figure 2.3

2.1.2 Split Selection

Definizione 2.1.1: Split Selection

Il cuore dell'algoritmo di apprendimento di un decision tree è la scelta delle features su cui fare splitting.

Guidelines per scegliere la features su cui fare splitting:

- *Minority Class:* $minp^+, p^-$, corrisponde al confronto tra la probabilità della classe positiva con quella della classe negativa e scegliere la minore.
- *Gini Index:* $2p^+p^-$, è l'errore atteso se gli esempi sono etichettati a caso (con una carta probabilità p^+ la classe positiva e p^- la classe negativa.
- *Entropy:* $-p^+log_2(p^+) - p^-log_2(p^-)$ è la quantità di informazione attesa (in bits) contenuta in un messaggio.
- *Gain Ratio:*

Gini Index:

- Assumiamo che gli esempi nella popolazione sono positivi con probabilità p e negativi con $1 - p$.

- Gli esempi sono etichettati casualmente e indipendentemente.
- Quando estraiamo un esempio ci sarà p probabilità di estrarlo positivo e $1-p$ probabilità di estrarlo negativo.
- Probabilità totale di errore: $p(1-p)+(1-p)p = 2p(1-p)$.

Definizione 2.1.2: Entropia

Viene utilizzata per quantificare il numero di unità (bits) richieste per rappresentare l'informazione. Misura la quantità di informazione associata a un evento.

Note:-

Se un esperimento a n possibili risultati (equiprobabili) il numero di bits richiesti per rappresentare qualsiasi risultato è b:

$$b = [log_2 n]$$

Esempio 2.1.1

- Supponiamo di lanciare un dado a 6 facce.
- Se vogliamo rappresentare e comunicare un risultato abbiamo bisogno di n bits:
$$b = [log_2 6] = [2.58] = 3$$
- Se invece comunichiamo solo la parità del risultato:
$$b_1 = [log_2 2] = 1$$
- Con solo la parità si ha incertezza sul risultato effettivo. Ci sono 3 possibili risultati per pari e 3 per dispari.
- Abbiamo bisogno di:
$$b_2 = [log_2 3] = [1.58] = 2$$
- b_2 rappresenta la quantità di confusione rimanente dopo aver inviato la parità:
$$b_2 = b - b_1$$
- L'informazione guadagnata dal primo messaggio è: $b_1 = b - b_2$

Corollario 2.1.1 Informazione

Se un evento E ha probabilità p di avvenire l'informazione guadagnata quando lo osserviamo è:

$$I(E) = log_2(\frac{1}{p}) = -log_2 p$$

Note:-

Se un evento è probabile (p tende a 1) si guadagnerà poca informazione dall'evento, se è improbabile (p tende a 0) si guadagnerà molta informazione.

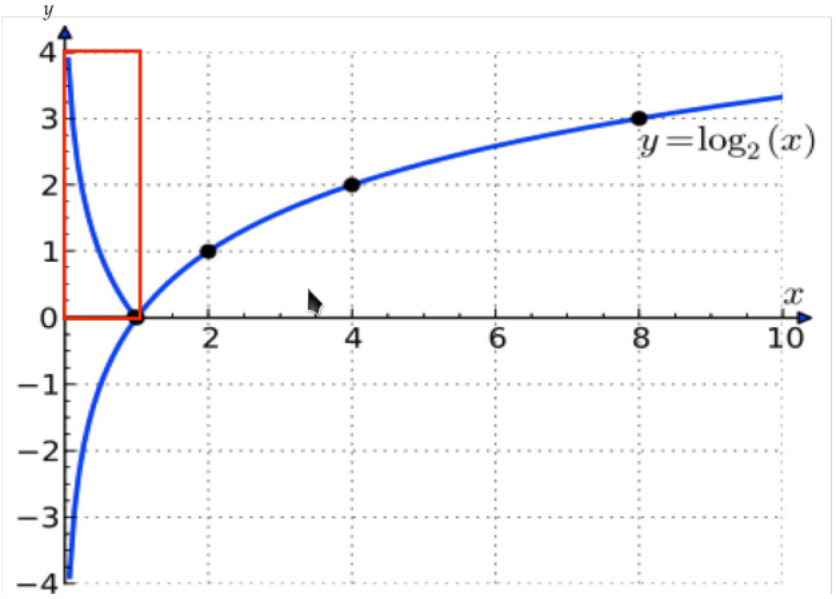


Figure 2.4: Curva della quantità di informazioni misurata.

Definizione 2.1.3: Informazione Attesa

Se un esperimento a n risultati $(E_1, ..., E_n)$ ognuno con probabilità $p_1, ..., p_n$ allora la quantità media di informazione guadagnata è:

$$H(E) = \sum_{i=1}^{n} p_i \cdot \log_2\left(\frac{1}{p_i}\right) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$$

2.1.3 Pruning

2.1.4 Continuous and Missing Values

2.1.5 Multivariate Decision Trees

2.2 Reti Neurali

2.3 Classificatori Bayesiani