Algoritmi Fondamentali del Machine Learning

Un'analisi dettagliata dei principali approcci computazionali

Prof. Massimo

27 dicembre 2025

Indice

1 Introduzione al Machine Learning

Il **Machine Learning** (ML) è una branca dell'intelligenza artificiale che consente ai computer di apprendere dai dati senza essere esplicitamente programmati per ogni situazione specifica. Gli algoritmi di ML identificano pattern nei dati e utilizzano questi pattern per fare previsioni o prendere decisioni.

1.1 Categorie principali

Il Machine Learning si suddivide tradizionalmente in tre categorie:

- **Apprendimento Supervisionato**: l'algoritmo apprende da dati etichettati
- **Apprendimento Non Supervisionato**: l'algoritmo scopre pattern in dati non etichettati
- **Apprendimento per Rinforzo**: l'algoritmo apprende attraverso interazioni e ricompense

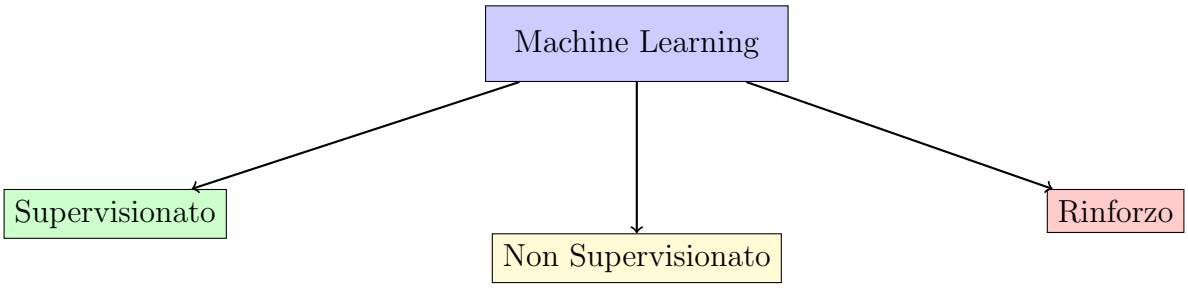

Figura 1: Categorie del Machine Learning

2 Regressione Lineare

2.1 Descrizione

La **regressione lineare** è uno degli algoritmi più importanti nel machine learning supervisionato. L'obiettivo è trovare una relazione lineare tra variabili indipendenti (features) e una variabile dipendente continua (target).

2.2 Formulazione matematica

Il modello di regressione lineare semplice è espresso come:

$$y = \beta_0 + \beta_1 x + \epsilon \tag{1}$$

dove:

- y è la variabile dipendente
- x è la variabile indipendente
- β_0 è l'intercetta
- β_1 è il coefficiente angolare
- ϵ è l'errore

Si tratta di una equazione di primo grado rappresentata sul piano cartesiano da una retta.

2.3 Funzione di costo

L'obiettivo è minimizzare l'errore quadratico medio (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2}$$

Dove

$$(y_i - \hat{y}_i)^2 \tag{3}$$

- n è il numero totale di osservazioni
- y_i è il valore reale della i-esima osservazione
- n è il numero totale di osservazioni
- $\hat{y}_i$ è il valore previsto dal modello per la stessa osservazione.
- $(y_i - \hat{y}_i)^2$ è l'errore di previsione, elevato al quadrato.

La sommatoria indica che questi errori quadratici vengono calcolati per tutte le osservazioni.

Il fattore 1/n serve a fare la media.

2.4 Visualizzazione

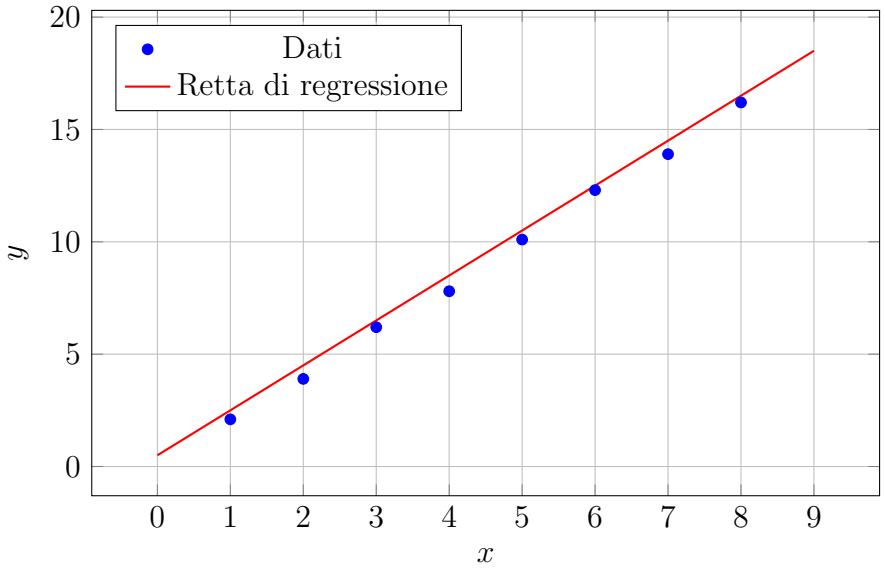

Figura 2: Esempio di regressione lineare

3 Regressione Logistica

3.1 Descrizione

La **regressione logistica** è utilizzata per problemi di classificazione binaria. Nonostante il nome, è un algoritmo di classificazione che predice la probabilità che un'istanza appartenga a una classe specifica.

3.2 Funzione Sigmoide

La regressione logistica utilizza la funzione sigmoide per mappare valori nell'intervallo [0,1]:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{4}$$

dove $z = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$

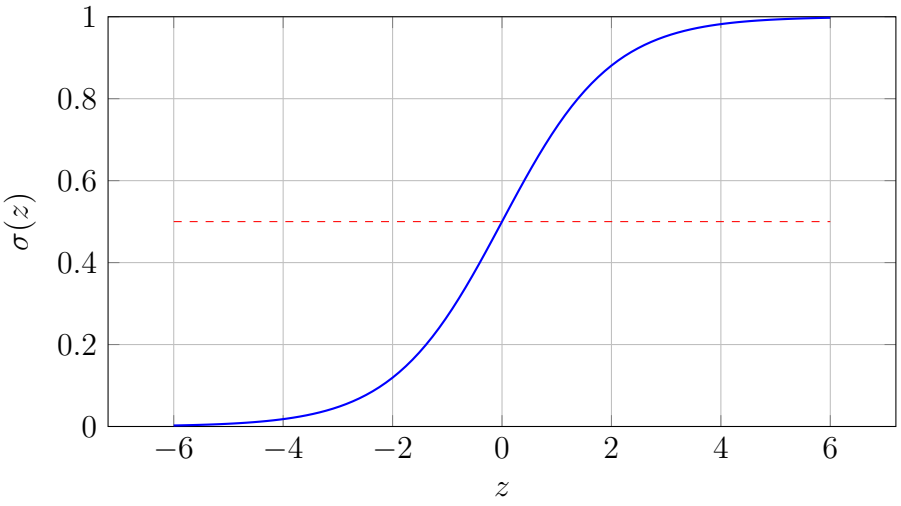

Figura 3: Funzione Sigmoide

3.3 Applicazioni

- Diagnosi medica (malattia presente/assente)
- Rilevamento spam
- Approvazione crediti
- Classificazione binaria di immagini

4 K-Nearest Neighbors (KNN)

4.1 Descrizione

L'**algoritmo K-Nearest Neighbors** è un metodo di apprendimento supervisionato non parametrico utilizzato sia per classificazione che per regressione. L'idea base è che istanze simili tendono ad avere etichette simili.

4.2 Principio di funzionamento

Per classificare un nuovo punto:

1. Calcola la distanza tra il nuovo punto e tutti i punti nel training set
2. Seleziona i K punti più vicini
3. Assegna la classe più frequente tra i K vicini (classificazione) o la media dei valori (regressione)

4.3 Metriche di distanza

Distanza Euclidea:

$$d(p, q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{5}$$

Distanza di Manhattan:

$$d(p, q) = \sum_{i=1}^{n} |p_i - q_i| \tag{6}$$

4.4 Visualizzazione

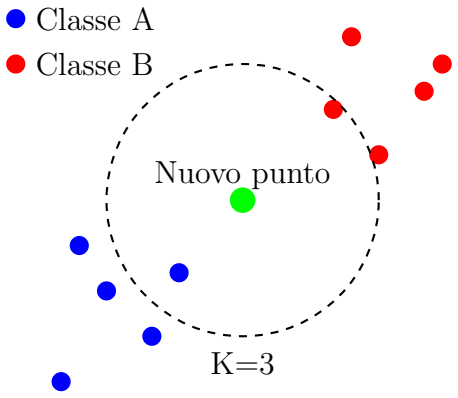

Figura 4: Classificazione KNN con K=3

4.5 Scelta del parametro K

- **K piccolo**: maggiore sensibilità al rumore, possibile overfitting
- **K grande**: decisioni più stabili ma possibile underfitting

- Tipicamente si usa cross-validation per determinare K ottimale

5 Decision Trees (Alberi Decisionali)

5.1 Descrizione

Gli **alberi decisionali** sono modelli predittivi che utilizzano una struttura ad albero per rappresentare decisioni e le loro possibili conseguenze. Ogni nodo interno rappresenta un test su un attributo, ogni ramo rappresenta l'esito del test, e ogni foglia rappresenta una classe o un valore.

5.2 Struttura

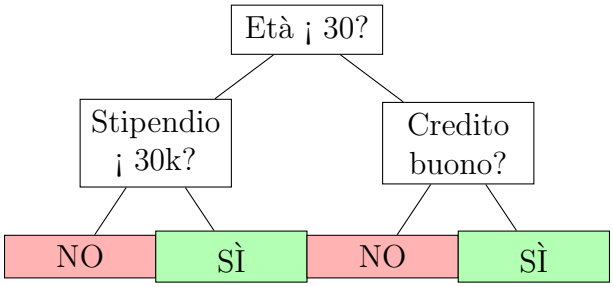

Figura 5: Esempio di albero decisionale per approvazione prestito

5.3 Algoritmo di costruzione

Algorithm 1 Costruzione Albero Decisionale (ID3)

Input: Dataset D, Attributi A
if tutti gli esempi hanno la stessa classe **then**
  return foglia con quella classe
else if A è vuoto **then**
  return foglia con classe più frequente
else
  Seleziona attributo $a \in A$ con massimo Information Gain
  Crea nodo decisionale per a
  for ogni valore v di a **do**
    D_v = sottoinsieme di D dove $a = v$
    Aggiungi sottoalbero costruito ricorsivamente su D_v e $A \setminus \{a\}$
  end for
end if

5.4 Vantaggi e svantaggi

Vantaggi:

- Facili da interpretare e visualizzare
- Richiedono poca preparazione dei dati
- Gestiscono dati numerici e categorici

Svantaggi:

- Tendenza all'overfitting
- Instabilità (piccole variazioni nei dati possono creare alberi molto diversi)
- Bias verso attributi con molti valori

6 Support Vector Machines (SVM)

6.1 Descrizione

Le **Support Vector Machines** sono algoritmi supervisionati utilizzati per classificazione e regressione. L'idea principale è trovare l'iperpiano ottimale che separa al meglio le classi, massimizzando il margine tra esse.

6.2 Concetto di margine

Il margine è la distanza tra l'iperpiano di separazione e i punti più vicini di ciascuna classe (support vectors).

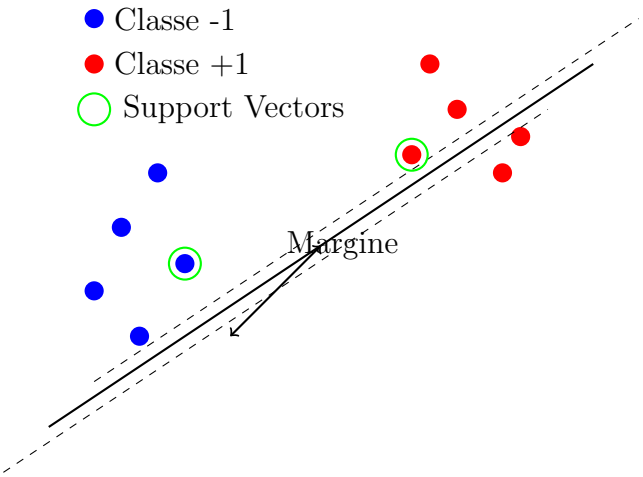

Figura 6: SVM lineare con margine massimo

6.3 Formulazione matematica

L'iperpiano è definito da:

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{7}$$

L'obiettivo è:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \tag{8}$$

soggetto a:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \forall i \tag{9}$$

7 K-Means Clustering

7.1 Descrizione

K-Means è un algoritmo di apprendimento non supervisionato utilizzato per raggruppare dati in K cluster. L'obiettivo è minimizzare la varianza intra-cluster e massimizzare la separazione tra cluster.

7.2 Algoritmo

Algorithm 2 K-Means Clustering

Scegli K centroidi iniziali casualmente
repeat
  Assignment step: Assegna ogni punto al centroide più vicino
  Update step: Ricalcola i centroidi come media dei punti assegnati
until i centroidi non cambiano significativamente

7.3 Funzione obiettivo

$$J = \sum_{i=1}^{K} \sum_{x \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \tag{10}$$

dove C_i è il cluster i e $\boldsymbol{\mu}_i$ è il centroide del cluster i.

7.4 Visualizzazione del processo

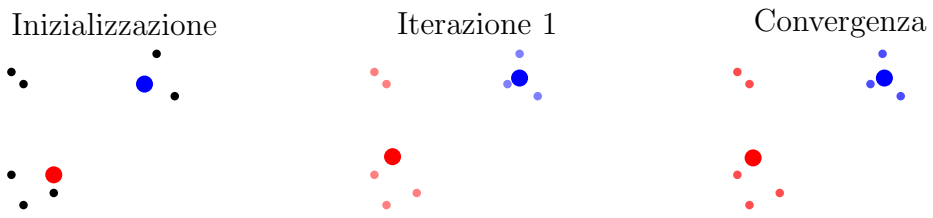

Figura 7: Processo iterativo di K-Means

7.5 Metodo del gomito (Elbow Method)

Per scegliere K ottimale, si grafica la varianza intra-cluster in funzione di K:

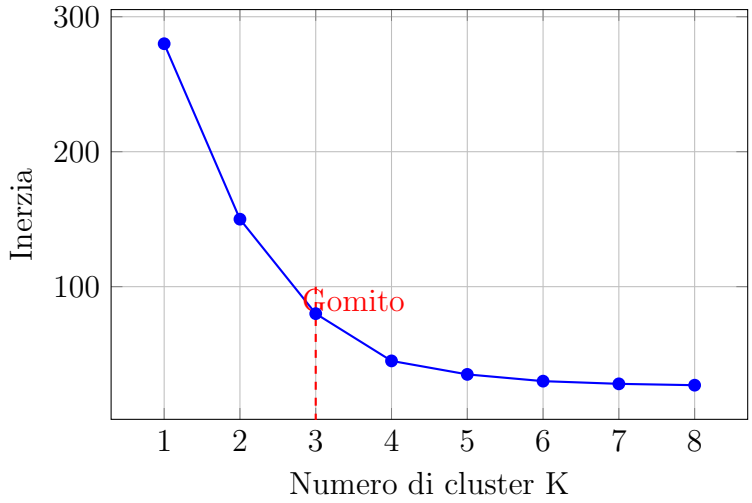

Figura 8: Metodo del gomito per selezione K

8 Reti Neurali Artificiali

8.1 Descrizione

Le **reti neurali artificiali** (ANN) sono modelli computazionali ispirati al funzionamento del cervello biologico. Sono composte da neuroni artificiali organizzati in layer che elaborano informazioni attraverso connessioni pesate.

8.2 Struttura di un neurone artificiale

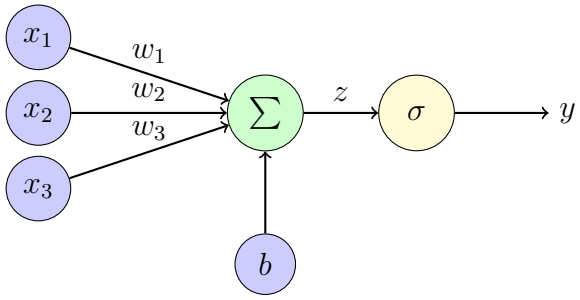

Figura 9: Struttura di un neurone artificiale

8.3 Funzioni di attivazione

Sigmoid:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{11}$$

Tanh:

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{12}$$

ReLU (Rectified Linear Unit):

$$ReLU(z) = \max(0, z) \tag{13}$$

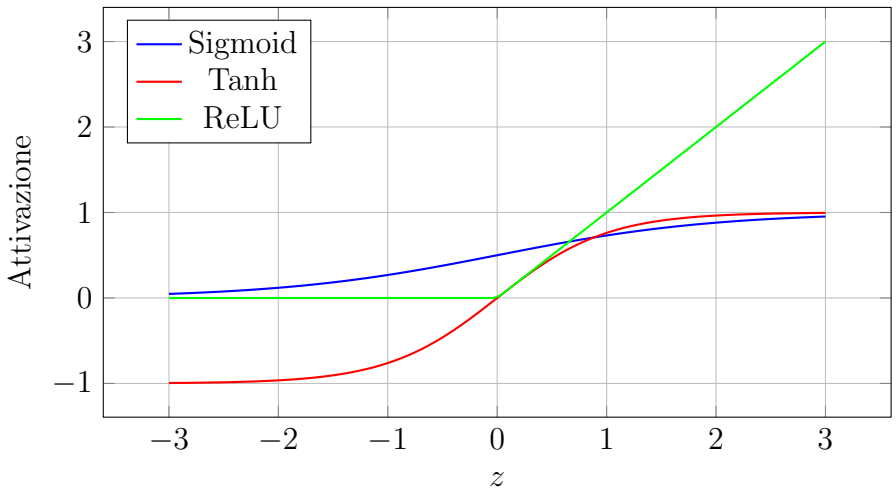

Figura 10: Funzioni di attivazione comuni

8.4 Architettura Multi-Layer Perceptron

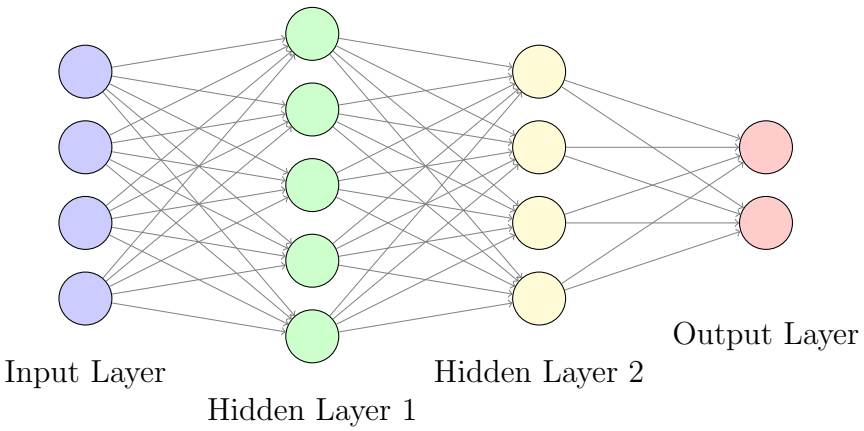

Figura 11: Architettura Multi-Layer Perceptron

8.5 Backpropagation

L'algoritmo di **backpropagation** aggiorna i pesi della rete propagando l'errore all'indietro:

1. **Forward pass**: calcola l'output per un input dato
2. **Calcola errore**: confronta output con target
3. **Backward pass**: propaga errore verso i layer precedenti
4. **Aggiorna pesi**: usando gradient descent

$$w_{ij}^{(new)} = w_{ij}^{(old)} - \eta \frac{\partial E}{\partial w_{ij}} \tag{14}$$

dove η è il learning rate ed E è la funzione di errore.

9 Random Forest

9.1 Descrizione

Random Forest è un algoritmo ensemble che costruisce multipli alberi decisionali durante il training e produce come output la classe che è la moda delle classi (classificazione) o la media delle predizioni (regressione) dei singoli alberi.

9.2 Principio di funzionamento

1. **Bootstrap Aggregating (Bagging)**: crea N campioni casuali con reinserimento dal dataset originale
2. **Feature Randomness**: per ogni split, considera solo un sottoinsieme casuale di features
3. **Costruzione alberi**: costruisce un albero decisionale completo per ogni campione
4. **Aggregazione**: combina le predizioni di tutti gli alberi

9.3 Visualizzazione

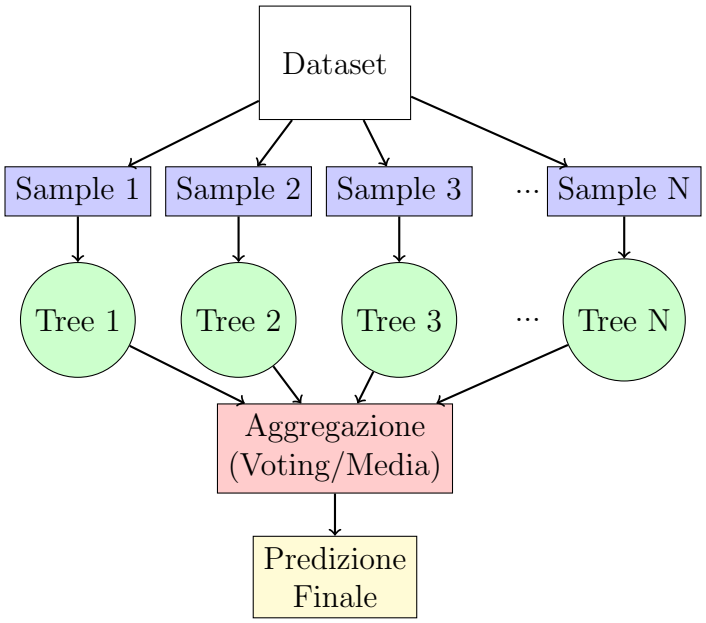

Figura 12: Architettura Random Forest

9.4 Parametri principali

- **n_estimators**: numero di alberi nella foresta
- **max_features**: numero massimo di features considerate per ogni split
- **max_depth**: profondità massima degli alberi
- **min_samples_split**: numero minimo di campioni richiesti per uno split

9.5 Vantaggi

- Riduce overfitting rispetto a singoli alberi decisionali
- Robusto al rumore e agli outliers
- Fornisce stime di feature importance
- Gestisce bene grandi dataset con molte features
- Non richiede normalizzazione dei dati

9.6 Feature Importance

Random Forest può calcolare l'importanza delle features basandosi sulla riduzione media dell'impurità:

$$Importance(f) = \frac{1}{N} \sum_{t=1}^{N} \sum_{n \in splits(t,f)} \Delta impurity(n) \tag{15}$$

10 Gradient Boosting

10.1 Descrizione

Gradient Boosting è una tecnica ensemble che costruisce modelli in modo sequenziale, dove ogni nuovo modello cerca di correggere gli errori dei modelli precedenti. È particolarmente efficace per problemi di classificazione e regressione.

10.2 Principio di funzionamento

Algorithm 3 Gradient Boosting

Inizializza modello $F_0(x)$ con una costante
for $m = 1$ to M **do**
  Calcola i residui: $r_i = y_i - F_{m-1}(x_i)$
  Addestra un modello h_m sui residui
  Aggiorna: $F_m(x) = F_{m-1}(x) + \nu \cdot h_m(x)$
end for
return $F_M(x)$

dove ν è il learning rate.

10.3 Visualizzazione del processo

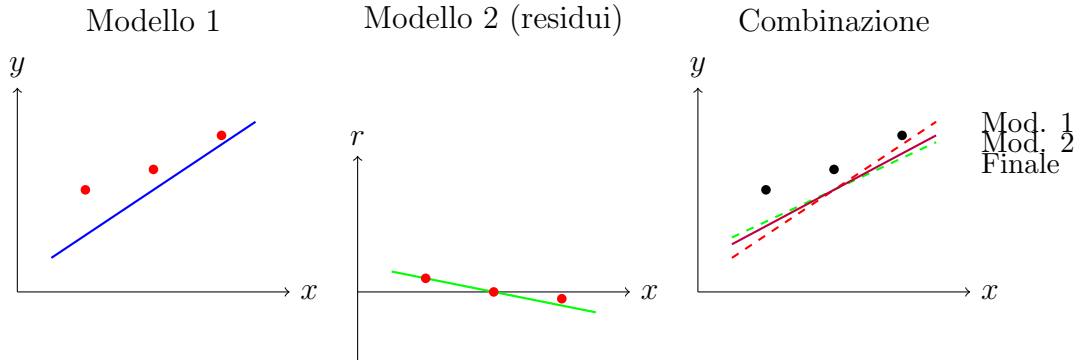

Figura 13: Processo iterativo di Gradient Boosting

10.4 Varianti popolari

- **XGBoost**: ottimizzazione e regolarizzazione avanzate
- **LightGBM**: efficiente per grandi dataset
- **CatBoost**: gestione nativa di variabili categoriche

10.5 Parametri di regolarizzazione

- **Learning rate** (ν): controlla il contributo di ogni albero

- **Number of estimators**: numero di alberi da costruire
- **Max depth**: profondità massima di ogni albero
- **Subsample**: frazione di campioni usati per ogni albero

11 Naive Bayes

11.1 Descrizione

Naive Bayes è un classificatore probabilistico basato sul teorema di Bayes con l'assunzione "naive" (ingenua) che le features siano indipendenti tra loro dato la classe.

11.2 Teorema di Bayes

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)} \tag{16}$$

dove:

- $P(C|X)$ è la probabilità a posteriori della classe C dato X
- $P(X|C)$ è la likelihood
- $P(C)$ è la probabilità a priori della classe
- $P(X)$ è la probabilità a priori dei dati

11.3 Assunzione di indipendenza

Con l'assunzione naive:

$$P(X|C) = P(x_1|C) \cdot P(x_2|C) \cdot ... \cdot P(x_n|C) = \prod_{i=1}^{n} P(x_i|C) \tag{17}$$

11.4 Classificazione

La classe predetta è:

$$\hat{y} = \arg\max_c P(C = c) \prod_{i=1}^{n} P(x_i|C = c) \tag{18}$$

11.5 Esempio: Classificazione Spam

Parola	**P(parola—spam)**	**P(parola—non spam)**
"gratis"	0.8	0.1
"offerta"	0.7	0.2
"urgente"	0.6	0.15
"ciao"	0.2	0.7

Tabella 1: Probabilità condizionate per classificazione spam

Per classificare "Offerta gratis urgente":

$$P(\text{spam}|\text{messaggio}) \propto P(\text{spam}) \cdot P(\text{offerta}|\text{spam}) \cdot P(\text{gratis}|\text{spam}) \cdot P(\text{urgente}|\text{spam})$$

11.6 Varianti

- **Gaussian Naive Bayes**: assume distribuzione gaussiana per features continue
- **Multinomial Naive Bayes**: per features discrete (es. conteggio parole)
- **Bernoulli Naive Bayes**: per features binarie

11.7 Vantaggi e limitazioni

Vantaggi:

- Semplice e veloce
- Funziona bene con pochi dati di training
- Efficace per classificazione di testo

Limitazioni:

- Assunzione di indipendenza raramente vera in pratica
- Sensibile a features irrilevanti
- Problemi con combinazioni di features mai viste nel training

12 Principal Component Analysis (PCA)

12.1 Descrizione

PCA è una tecnica di riduzione della dimensionalità non supervisionata che trasforma i dati in un nuovo sistema di coordinate dove le direzioni di massima varianza diventano i nuovi assi (componenti principali).

12.2 Obiettivi

- Ridurre la dimensionalità mantenendo la massima informazione
- Eliminare correlazioni tra features
- Visualizzare dati ad alta dimensionalità
- Ridurre overfitting e complessità computazionale

12.3 Algoritmo

Algorithm 4 Principal Component Analysis

Input: Matrice dati X di dimensione $n \times d$
Standardizza i dati: $X_{std} = \frac{X-\mu}{\sigma}$
Calcola matrice di covarianza: $C = \frac{1}{n} X_{std}^T X_{std}$
Calcola autovalori λ_i e autovettori v_i di C
Ordina autovettori per autovalori decrescenti
Seleziona i primi k autovettori: $W = [v_1, v_2, ..., v_k]$
Proietta i dati: $X_{pca} = X_{std} \cdot W$
return X_{pca}

12.4 Visualizzazione geometrica

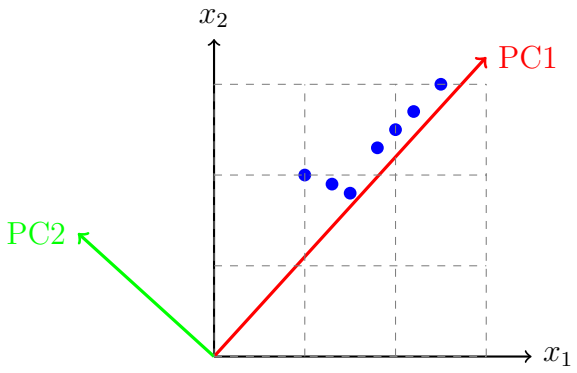

Figura 14: Identificazione delle componenti principali

12.5 Varianza spiegata

La varianza spiegata dalla componente i-esima:

$$\text{Varianza spiegata}_i = \frac{\lambda_i}{\sum_{j=1}^{d} \lambda_j} \tag{19}$$

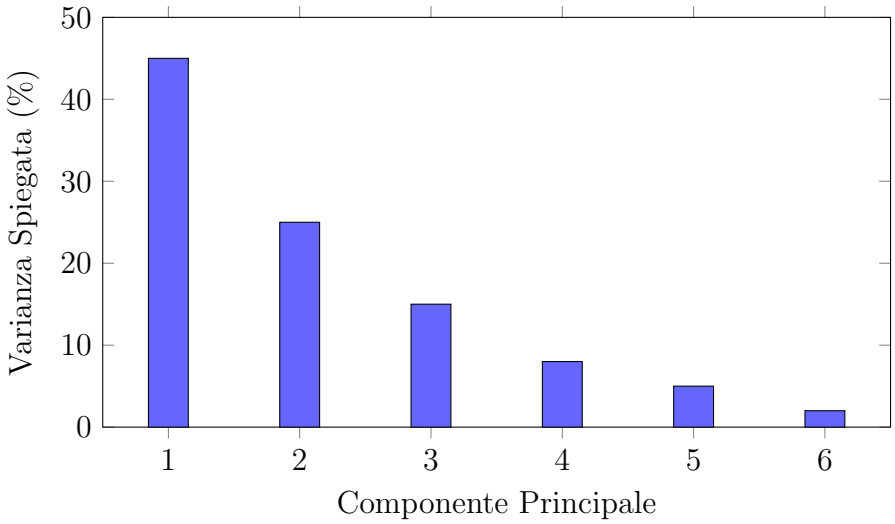

Figura 15: Varianza spiegata per componente

12.6 Scelta del numero di componenti

Criteri comuni:

- **Soglia di varianza**: mantenere componenti che spiegano almeno l'80-95% della varianza totale
- **Criterio di Kaiser**: mantenere componenti con $\lambda_i > 1$
- **Scree plot**: cercare il "gomito" nel grafico degli autovalori

12.7 Applicazioni

- Preprocessing per altri algoritmi ML
- Compressione di immagini
- Visualizzazione di dati multidimensionali
- Rilevamento di anomalie
- Analisi di dati genomici

13 Conclusioni e Confronto

13.1 Riepilogo degli algoritmi

Algoritmo	**Tipo**	**Complessità**	**Casi d'uso**
Regressione Lineare	Supervisionato	Bassa	Predizione valori continui
Regressione Logistica	Supervisionato	Bassa	Classificazione binaria
KNN	Supervisionato	Alta (predizione)	Classificazione, raccomandazioni
Decision Trees	Supervisionato	Media	Classificazione interpretabile
SVM	Supervisionato	Alta	Classificazione complessa
K-Means	Non supervisionato	Media	Segmentazione clienti
Neural Networks	Supervisionato	Molto alta	Visione, NLP, complessi
Random Forest	Supervisionato	Alta	Classificazione robusta
Gradient Boosting	Supervisionato	Alta	Competizioni, alta accuratezza
Naive Bayes	Supervisionato	Bassa	Classificazione testo
PCA	Non supervisionato	Media	Riduzione dimensionalità

Tabella 2: Confronto degli algoritmi di Machine Learning

13.2 Criteri di scelta

- **Dimensione del dataset**: algoritmi semplici per dataset piccoli, ensemble o deep learning per grandi dataset
- **Interpretabilità**: alberi decisionali e regressione lineare sono più interpretabili
- **Accuratezza richiesta**: ensemble e reti neurali per massima accuratezza
- **Velocità**: KNN e Naive Bayes sono veloci in training ma potrebbero essere lenti in predizione
- **Tipo di problema**: classificazione, regressione, clustering, o riduzione dimensionalità

13.3 Tendenze future

- **Deep Learning**: reti sempre più profonde e specializzate
- **AutoML**: automazione della selezione e ottimizzazione di modelli
- **Federated Learning**: apprendimento distribuito preservando la privacy

- **Explainable AI**: maggiore focus sull'interpretabilità dei modelli
- **Transfer Learning**: riutilizzo di modelli pre-addestrati

13.4 Considerazioni pratiche

Per un progetto di ML efficace:

1. **Comprendere il problema**: definire obiettivi chiari
2. **Esplorare i dati**: analisi esplorativa approfondita
3. **Feature engineering**: creare features significative
4. **Selezione modello**: provare diversi algoritmi
5. **Validazione**: usare cross-validation
6. **Tuning**: ottimizzare iperparametri
7. **Valutazione**: metriche appropriate al contesto
8. **Deployment**: monitorare performance in produzione

Bibliografia e Risorse

- Bishop, C.M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Hastie, T., Tibshirani, R., Friedman, J. (2009). *The Elements of Statistical Learning*. Springer.
- Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press.
- Murphy, K.P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.
- Scikit-learn Documentation: `https://scikit-learn.org`