Compendio di Teoria AI

Preprocessing, Imbalance, Benchmarking, Regressione e Allineamento

Basato sulle slide di Vincenzo Bonnici
Corso di Laurea Magistrale in Scienze Informatiche - UniPR

Anno Accademico 2025-2026

Indice

1 Introduzione e Workflow

Il preprocessamento dei dati è una fase critica nel ciclo di vita di un progetto di Machine Learning. La qualità dei dati influenza direttamente la qualità del modello appreso (*Garbage In, Garbage Out*).

1.1 Il Ciclo di Vita (Workflow)

Il processo di gestione dei dati non è lineare ma ciclico, composto dalle seguenti fasi principali:

1. **Get Data:** Acquisizione dei dati grezzi.
2. **Clean, Prepare & Manipulate Data:** Fase centrale di pulizia e trasformazione.
3. **Train Model:** Addestramento del modello.
4. **Test Data:** Verifica delle prestazioni.
5. **Improve:** Miglioramento iterativo.

1.2 Fasi Dettagliate di Pianificazione

- **Define:** Descrizione del problema, ambito (scope), fattibilità, assunzioni e vincoli.
- **Data Cleanup:** Gestione dati mancanti, imputazione, eliminazione (drop).
- **Analyze:** Analisi delle correlazioni, importanza delle feature (feature importance), tipologia del problema.
- **Prepare:** Trasformazione, normalizzazione.
- **Evaluate & Tune:** Validazione incrociata (Cross-validation), tuning dei parametri.

2 Data Preparation

La preparazione dei dati (Data Cleaning) ha lo scopo principale di ridurre il rumore e trattare le inconsistenze.

2.1 Problemi Comuni

- **Dati Incompleti:** Valori mancanti (es. attributi vuoti).
- **Dati Rumorosi (Noisy):** Errori, outlier o valori che deviano dall'atteso.
- **Dati Inconsistenti:** Discrepanze nei codici, nei nomi o nei formati.

2.2 Tecniche di Gestione dei Dati Mancanti

- **Ignorare la tupla:** Solitamente fatto quando manca l'etichetta di classe.
- **Riempimento manuale:** Laborioso e spesso non fattibile.
- **Costante globale:** Riempire con "Unknown" o $-\infty$ (può essere rischioso).
- **Misure di tendenza centrale:** Usare la media (mean) o la mediana dell'attributo.
- **Media condizionata:** Usare la media per la classe di appartenenza.
- **Valore più probabile:** Usare metodi inferenziali (es. regressione, Bayes o alberi decisionali).

3 Data Transformation e Normalizzazione

La normalizzazione è essenziale per evitare che attributi con range numerici maggiori dominino su quelli con range minori (specialmente in algoritmi basati sulla distanza).

3.1 Tecniche di Normalizzazione

- **Min-Max Normalization:**

$$v' = \frac{v - \min_A}{\max_A - \min_A}(\text{new\_max}_A - \text{new\_min}_A) + \text{new\_min}_A$$

  Scala i dati in un intervallo specifico, tipicamente $[0, 1]$.

- **Z-Score Normalization (Standardizzazione):**

$$v' = \frac{v - \mu_A}{\sigma_A}$$

  Dove μ_A è la media e σ_A è la deviazione standard. Utile quando i minimi e massimi sono sconosciuti o affetti da outlier.

- **Decimal Scaling:**

$$v' = \frac{v}{10^j}$$

  Dove j è il più piccolo intero tale che $\max(|v'|) < 1$.

4 Self-Organizing Maps (SOM)

Le SOM sono reti neurali non supervisionate utilizzate per la riduzione della dimensionalità e la visualizzazione dei dati, preservando la topologia dello spazio di input.

4.1 Concetto

Le SOM mappano input ad alta dimensionalità su una griglia (mappa) 2D di neuroni.

- **Competizione:** Per ogni input, i neuroni competono per essere attivati.
- **Vincitore (Winner):** Il neurone con il vettore dei pesi più simile al vettore di input viene eletto vincitore (Best Matching Unit - BMU).
- **Cooperazione:** Il vincitore definisce un vicinato spaziale di neuroni eccitati.
- **Adattamento:** Il vincitore e i suoi vicini aggiustano i loro pesi per assomigliare di più all'input.

4.2 Algoritmo di Apprendimento

Sia x^n il vettore di input e w_k il vettore dei pesi del neurone k.

4.2.1 1. Selezione del Vincitore

Si calcola la distanza (solitamente Euclidea) tra l'input e tutti i neuroni. Il neurone i che minimizza la distanza è il vincitore.

4.2.2 2. Aggiornamento (Adattamento)

I pesi vengono aggiornati secondo la regola:

$$w_k(t+1) = w_k(t) + \eta(t) \cdot \eta_{ik}(t) \cdot (x^n - w_k(t)) \tag{1}$$

Dove:

- $\eta(t)$ è il **tasso di apprendimento** (learning rate), che decresce nel tempo.
- $\eta_{ik}(t)$ è la **funzione di vicinato** (neighborhood kernel), che dipende dalla distanza reticolare tra il neurone k e il vincitore i.

La funzione di vicinato spesso assume la forma di una Gaussiana che si restringe nel tempo.

4.3 Decadimento dei Parametri

Per garantire la convergenza della mappa (fase di *fine-tuning*), il tasso di apprendimento $\eta(t)$ deve tendere a zero per $t \to \infty$. Una formula tipica è il decadimento esponenziale:

$$\eta(t) = \eta_0 e^{-\frac{t}{\tau_1}}$$

Anche l'ampiezza del vicinato diminuisce nel tempo, rendendo l'adattamento sempre più locale.

5 Gestione dello Sbilanciamento tra le Classi

Il problema dello sbilanciamento tra le classi (*Class Imbalance*) si verifica quando la distribuzione degli esempi tra le classi non è uniforme. In un contesto binario, distinguiamo:

- **Classe di Maggioranza (Negative):** La classe con il maggior numero di campioni.
- **Classe di Minoranza (Positive):** La classe rara, spesso quella di maggior interesse.

Esempi tipici includono il rilevamento di frodi bancarie o diagnosi mediche di malattie rare.

5.1 Il Paradosso dell'Accuratezza

Gli algoritmi standard mirano a massimizzare l'accuratezza globale. In presenza di forte sbilanciamento (es. 99:1), un classificatore che assegna sempre la classe di maggioranza otterrà un'accuratezza del 99%, fallendo però nell'identificare la classe rara.

6 Metriche di Valutazione per Dataset Sbilanciati

L'accuratezza non è affidabile. È necessario utilizzare la Matrice di Confusione.

6.1 Matrice di Confusione

		Classe Predetta	
		Positivo	Negativo
Classe Reale	Positivo	TP (True Positive)	FN (False Negative)
	Negativo	FP (False Positive)	TN (True Negative)

6.2 Metriche Fondamentali

- **Recall (Sensitivity):** $TP/(TP + FN)$. Capacità di trovare i positivi.
- **Precision:** $TP/(TP + FP)$. Affidabilità delle predizioni positive.
- **Specificity:** $TN/(TN + FP)$. Capacità di trovare i negativi.
- **F1-Score:** $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$. Media armonica.
- **G-Mean:** $\sqrt{\text{Sensitivity} \cdot \text{Specificity}}$. Equilibrio tra le classi.

6.3 Curve ROC e AUC

L'Area Under the Curve (AUC) riassume la performance della curva ROC (Sensitivity vs 1-Specificity). Un valore di 0.5 è casuale, 1.0 è perfetto.

7 Strategie di Risoluzione

7.1 Livello Dati (Resampling)

Queste tecniche bilanciano il dataset prima del training.

7.1.1 Undersampling (Sottocampionamento)

Riduce la classe di maggioranza.

- **Random:** Elimina campioni a caso.
- **NearMiss:** Seleziona campioni di maggioranza basandosi sulla distanza dai campioni di minoranza (es. quelli più vicini).
- **Tomek Links:** Rimuove campioni di maggioranza che sono "troppo vicini" ai confini con la minoranza per pulire i bordi.

7.1.2 Oversampling (Sovracampionamento)

Aumenta la classe di minoranza.

- **Random:** Duplica campioni esistenti (rischio overfitting).
- **SMOTE (Synthetic Minority Over-sampling Technique):** Crea nuovi esempi sintetici interpolando tra un campione positivo e i suoi vicini (KNN).

$$x_{new} = x_i + \lambda \cdot (x_{zi} - x_i)$$

- **ADASYN:** Simile a SMOTE, ma si concentra sugli esempi più difficili da imparare.

7.2 Livello Algoritmo (Cost-Sensitive)

Modifica l'algoritmo introducendo una penalità maggiore per gli errori sulla classe rara. Si definisce una **Matrice dei Costi** dove il costo di un Falso Negativo è molto più alto di un Falso Positivo:

$$C(FN) \gg C(FP)$$

La classificazione minimizza il **Rischio di Bayes**:

$$L(x, i) = \sum_j P(j|x)C(i, j)$$

7.3 Metodi Ibridi (Ensemble)

Combinano resampling e algoritmi di insieme.

- **Bagging + Undersampling:** Addestra molti classificatori su sottocampioni bilanciati.
- **Boosting (SMOTEBoost, RUSBoost):** Integra il campionamento in algoritmi come AdaBoost.

8 Benchmarking

Il Benchmarking è il processo di analisi comparativa utilizzato per giudicare la qualità di un sistema in relazione ad altri.

8.1 Tipologie di Dataset

Un benchmark è costituito da uno o più dataset:

- **Reali:** Acquisiti tramite misurazione di eventi accaduti. Rappresentano fedelmente la realtà ma possono contenere errori di misurazione.
- **Sintetici (in silico):** Costruiti artificialmente seguendo regole specifiche (es. grafi random). Utili per testare scenari non osservati, ma rischiano di essere distanti dalla realtà.

8.2 Generazione di Grafi Sintetici

Per testare algoritmi su reti complesse, si utilizzano modelli generativi con proprietà topologiche note.

8.2.1 Modelli Principali

- **Erdos-Renyi (Random Graph):** Dato un numero di nodi N, ogni coppia di nodi è connessa con probabilità p.
  - *Proprietà:* Distribuzione dei gradi binomiale (Poisson per N grande). Clustering coefficient basso ($C \approx p$). Path length basso ($L \approx \ln N$).
- **Watts-Strogatz (Small World):** Parte da un reticolo regolare e "ri-cabla" (rewires) gli archi con probabilità p.
  - *Caratteristica:* Alto coefficiente di clustering (come i reticoli) e bassa lunghezza media del percorso (come i grafi random). Rappresenta reti sociali dove "tutti si conoscono in pochi passaggi".
- **Barabasi-Albert (Scale-Free):** Utilizza il meccanismo dell'*attaccamento preferenziale* ("rich get richer"). I nuovi nodi si collegano con maggiore probabilità ai nodi che hanno già un grado elevato.
  - *Proprietà:* Distribuzione dei gradi a legge di potenza ($P(k) \sim k^{-\gamma}$). Presenza di **Hub** (nodi molto connessi).

9 Regressione e Analisi di Serie Temporali

9.1 Regressione Lineare

A differenza della classificazione (etichette discrete), la regressione mira a predire un valore numerico continuo y.

9.1.1 Regressione Lineare Semplice

Modella la relazione tra una variabile indipendente X e una dipendente Y come una retta:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

L'obiettivo è stimare i coefficienti β minimizzando la somma dei quadrati degli errori (Metodo dei Minimi Quadrati - OLS).

9.1.2 Regressione Lineare Multipla

Estensione a più variabili indipendenti. In forma matriciale:

$$Y = X\beta + \epsilon$$

La soluzione per il vettore dei coefficienti β è data da:

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

9.1.3 Metriche di Valutazione

- **RSS (Residual Sum of Squares):** $\sum(y_i - \hat{y}_i)^2$. Misura la varianza non spiegata.
- R^2 **(Coefficiente di determinazione):**

  $$R^2 = 1 - \frac{RSS}{TSS}$$

  Indica la proporzione di varianza dei dati spiegata dal modello. $R^2 = 1$ indica un fit perfetto.

9.2 Serie Temporali

Una serie temporale è una sequenza di dati ordinati nel tempo. L'obiettivo è predire il valore futuro basandosi su quelli passati.

9.2.1 Stazionarietà

Una serie è stazionaria se le sue proprietà statistiche (media, varianza) sono costanti nel tempo. La maggior parte dei modelli (es. ARIMA) richiede che la serie sia resa stazionaria (es. tramite differenziazione) prima dell'analisi.

9.2.2 Modelli Predittivi

- **AR (AutoRegressive):** Il valore corrente dipende linearmente dai suoi valori passati.
- **MA (Moving Average):** Il valore corrente dipende dagli errori di previsione passati.
- **RNN (Recurrent Neural Networks):** Reti neurali progettate per dati sequenziali. Introducono uno **Stato Nascosto** $h(t)$ che funge da memoria:

  $$h(t) = f(Ux(t) + Wh(t-1))$$

  L'output $y(t)$ dipende dallo stato nascosto corrente.

10 Programmazione Dinamica e Allineamento Sequenze

La Programmazione Dinamica (DP) è una tecnica per risolvere problemi complessi scomponendoli in sottoproblemi sovrapposti e memorizzando i risultati (*memoization*). È fondamentale nella bioinformatica per l'allineamento di sequenze (DNA, Proteine).

10.1 Dynamic Time Warping (DTW)

Algoritmo per misurare la similarità tra due serie temporali che possono variare in velocità. Permette un allineamento "elastico" (warping) non lineare, accoppiando un punto di una serie con più punti dell'altra per minimizzare la distanza cumulativa.

10.2 Allineamento di Sequenze

L'obiettivo è identificare regioni di similarità che indicano relazioni funzionali, strutturali o evolutive.

10.2.1 Edit Distance (Distanza di Levenshtein)

Il numero minimo di operazioni per trasformare una stringa nell'altra. Le operazioni ammesse sono:

- Inserimento (Gap)
- Cancellazione (Gap)
- Sostituzione (Match/Mismatch)

10.2.2 Allineamento Globale (Needleman-Wunsch)

Allinea le sequenze per tutta la loro lunghezza. Si costruisce una matrice M dove $M_{i,j}$ è il punteggio ottimale per allineare i primi i caratteri di A con i primi j di B. La relazione di ricorrenza considera il massimo tra:

- Match/Mismatch: $M_{i-1,j-1} + S(A_i, B_j)$
- Gap in A: $M_{i-1,j} + \text{gap\_penalty}$
- Gap in B: $M_{i,j-1} + \text{gap\_penalty}$

10.2.3 Allineamento Locale (Smith-Waterman)

Cerca la migliore sottosequenza allineata (utile se le sequenze sono molto diverse ma condividono un motivo locale). Differenza principale con Needleman-Wunsch: i punteggi negativi nella matrice vengono posti a **zero**. Questo permette di "riavviare" l'allineamento in qualsiasi punto.

10.2.4 Allineamento Multiplo (MSA)

Per allineare più di due sequenze (NP-Hard). Si usano metodi euristici come l'**Allineamento Progressivo** (es. Clustal): 1. Calcolo distanze a coppie. 2. Costruzione di un albero guida. 3. Allineamento delle sequenze seguendo i nodi dell'albero, dal più simile al meno simile.