Lezione 6: Classificazione

Davide Evangelista
davide.evangelista5@unibo.it

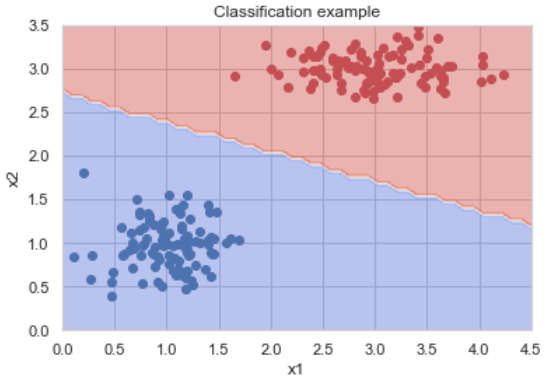
Università di Bologna

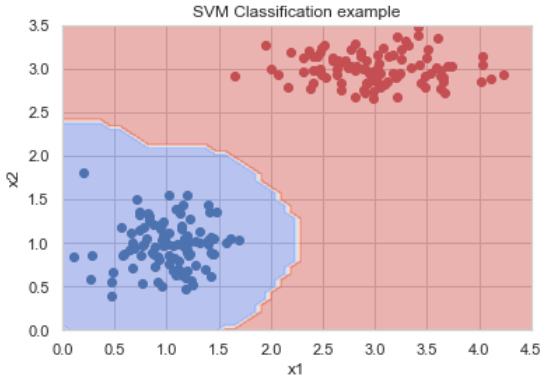
15 Aprile 2024

Metriche

Come valutare un modello?

Supponiamo di avere a disposizione un dataset $\mathcal{S}$, non linearmente separabile, e di addestrare su di esso diversi tipi di SVM, cambiando la tipologia di kernel e alcuni iperparametri.
Come scegliamo qual'è il migliore?





Metriche

Il modo più semplice (e intuitivo) è quello di definire una o più metriche, in grado di valutare l'abilità di un modello nel fare predizioni.

Definizione

Una metrica è una qualunque funzione $\rho : \mathbb{R}^m \times \mathbb{R}^m \longrightarrow \mathbb{R}$, *tale che:*

- $\rho(y, y') \geq 0$.
- $\rho(y, y') = \rho(y', y)$.
- $\rho(y, y') = 0 \iff y = y'$

Intuitivamente, una metrica rappresenta una distanza tra la l'output corretto e la predizione fatta da $h_\theta(x)$, ed è tanto più piccola quanto più il predittore è corretto.

MSE

Molte delle misure statistiche che avete visto (e vedrete) a lezione sono delle metriche.
Una delle più comuni, è il cosiddetto *Mean Squared Error (MSE)*, la quale, fissato un dataset $\mathcal{S} = \{x_i, y_i\}_{i=1}^{N}$, è data da

$$MSE(\mathcal{S}) = \frac{1}{N} \sum_{i=1}^{N} ||y_i - h_\theta(x_i)||_2^2 \tag{1}$$

Questa, sebbene molto utilizzata nella pratica, non è la più adatta per problemi di classificazione, poiché in questo caso la variabile di output y è di tipo Categorico, e l'MSE non sfrutta in maniera efficiente questa caratteristica.

Misclassification Error (ME)

Una misura molto più comune per problemi di classificazione è l'errore di Misclassificazione (ME), definito come il numero totale di punti mal classificati dal nostro predittore. Formalmente, definita la funzione

$$I(y \neq y') = \begin{cases} 1 & \text{se } y \neq y' \\ 0 & \text{se } y = y' \end{cases} \tag{2}$$

Si definisce l'errore di misclassificazione come

$$ME(\mathcal{S}) = \sum_{i=1}^{N} I(y_i \neq h_\theta(x_i)) \tag{3}$$

Misclassification Rate (MR)

In realtà, spesso, al posto dell'errore di misclassificazione, si utilizza la frequenza di misclassificazione (MR), definita come il numero medio di predizioni accurate che vengono fatte dal predittore. Formalmente

$$MR(\mathcal{S}) = \frac{1}{N} \sum_{i=1}^{N} I(y_i \neq h_\theta(x_i)) \tag{4}$$

Si osservi che, nel caso di classificazione binaria (in cui la variabile di output y assume valori in $\{0; 1\}$, la frequenza di misclassificazione e l'MSE sono equivalenti.
Moltiplicando per 100 la MR, si ottiene la cosiddetta percentuale di misclassificazione $M\% = 100 \cdot MR$, che dice quanti punti in percentuale sono stati mal classificati dal modello.

Accuratezza

Strettamente collegata con la frequenza di misclassificazione, è la cosiddetta Accuratezza, che pur non essendo propriamente una metrica (l'accuratezza è tanto più alta quanto migliore è il modello), viene spesso usata per valutare il modello. L'accuratezza è definita come:

$$Acc(\mathcal{S}) = \frac{1}{N} \sum_{i=1}^{N} I(y_i = h_\theta(x_i) = 1 - MR(\mathcal{S}) \tag{5}$$

Implementazione

Supponendo di avere un modello già addestrato, chiamato `SVM`, calcolare la frequenza di misclassificazione con i seguenti comandi

```
y_pred = model.predict(df[['x1','x2']])
y_true = df['class']

MR = np.mean(y_pred != y_true)
```

Similmente, l'accuratezza può essere facilmente calcolata come

```
acc = 1 - MR
```

Matrice di Confusione

In alcune situazioni (ad esempio, in problemi di classificazione medici), non tutti gli errori hanno la stessa importanza. Nell'esempio di classificazione medica, sbagliare una predizione e diagnosticare ad un paziente una malattia che in realtà non ha è poco grave: basteranno altri semplici esami per identificare l'errore e risolvere il problema, il paziente si prenderà al massimo un po' di paura.
Invece, sbagliare una predizione dicendo ad un paziente malato che è sano, potrebbe avere delle conseguenze pericolose: il paziente non si curerà e potrà incorrere in problemi di vario genere.

Matrice di Confusione

Le metriche che abbiamo studiato fino ad adesso, però, non rendevano possibile questa distinzione.
Come è possibile risolvere questo problema?
Prima di tutto, fissiamo un po' di nomenclatura.

TP, FP, TN, FN

Identifichiamo quattro tipologie di predizioni per modelli binari (in cui l'output y assume valori 0 (Negativo) e 1 (Positivo)), due delle quali saranno predizioni corrette, mentre le altre due saranno errori.

- Veri Positivi (TP): Il modello predice 1 (Positivo), e l'output corretto è effettivamente 1 (Positivo).
- Falsi Positivi (FP): Il modello predice 1 (Positivo), ma l'output corretto è 0 (Negativo).
- Veri Negativi (TN): Il modello predice 0 (Negativo), e l'output corretto è effettivamente 0 (Negativo).
- Falsi Negativi (FN): Il modello predice 0 (Negativo), ma l'output corretto è 1 (Positivo).

TP, FP, TN, FN

Chiaramente, una predizione di tipo Vero Positivo o Vero Negativo (TP o TN) è una predizione corretta, quindi non ci crea alcun tipo di problema.
I Falsi Positivi e i Falsi Negativi, invece, rappresentano gli errori di predizione e possono avere valenza diversa. Nell'esempio predecente sui dati medici, i FN sono molto più pericolosi dei FP!
Queste definizioni di permettono di definire alcune metriche molto usate nella pratica e che permettono di distinguere la tipologia di errore compita dal modello.

Sensitività

Definiamo la **Sensitività (o TPR, True Positive Rate)** di un modello la grandezza

$$TPR = \frac{TP}{TP + FN} = 1 - FNR \tag{6}$$

Come la percentuale di Veri Positivi sul totale delle osservazioni Positive (calcolate come $TP + FN$). Il *contrario* della TPR è la **FNR, o False Negative Rate**, calcolata come $1 - TPR$, che rappresenta la percentuale di Falsi Negativi sul totale delle osservazioni Positive. Una TPR alta rappresenta un modello che, quando predice esito positivo, è molto probabilmente un Vero Positivo. La TPR non da alcuna informazione su come si comporta il modello con le osservazioni Negative.

Specificità

La **Specificità (o TNR, True Negative Rate)** è la grandezza equivalente alla Sensitività, ma nel caso negativo, ovvero

$$TNR = \frac{TN}{TN + FP} = 1 - FPR \tag{7}$$

Che conta la percentuale di Veri Negativi sul totale delle osservazioni Negative (calcolate come $TN + FP$. Il *contrario* della TNR è la **FPR, o False Positive Rate**, calcolata come $1 - TNR$, che rappresenta la percentuale di Falsi Positivi sul totale delle osservazioni Negative.
Una TNR alta rappresenta un modello che, quando predice esito Negativo, è molto probabilmente un Vero Negativo. La TNR non da alcuna informazione su come si comporta il modello con le osservazioni Positive.

Precisione

La **Precisione** di un modello è la grandezza

$$PPV = \frac{TP}{TP + FP} \tag{8}$$

che conta la percentuale di Veri Positivi sul totale delle osservazioni predette come Positive. Una PPV alta indica un modello che predice un numero di Falsi Positivi bassi rispetto al numero dei Veri Positivi.

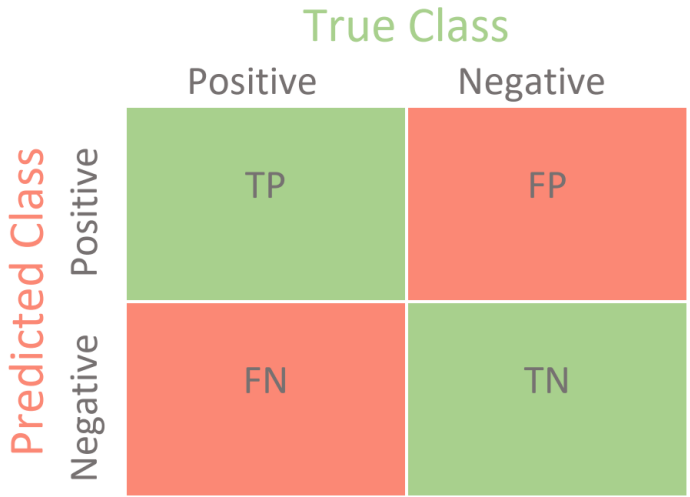
Matrice di Confusione

Come già osservato, la caratteristica negativa di queste metriche è il fatto che, sebbene esse riescano a distinguere le varie tipologie di errore, in genere una misura da sola non è sufficiente a spiegare la resa effettiva del modello, e bisogna confrontare tra loro varie grandezze per identificare la bontà della previsione.
Per risolvere questo problema, è spesso comodo visualizzare la Matrice di Confusione, che permette di visualizzare in un colpo d'occhio la resa del modello in termini di TP, FP, TN, FN.
La Matrice di Confusione viene implementata semplicemente con il comando

```
from sklearn import metrics
y_pred = model.predict(df[['x1','x2']])
y_true = df['class']

confusion_matrix = metrics.confusion_matrix(y_true, y_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =
    confusion_matrix, display_labels = [False, True])
```

Matrice di Confusione



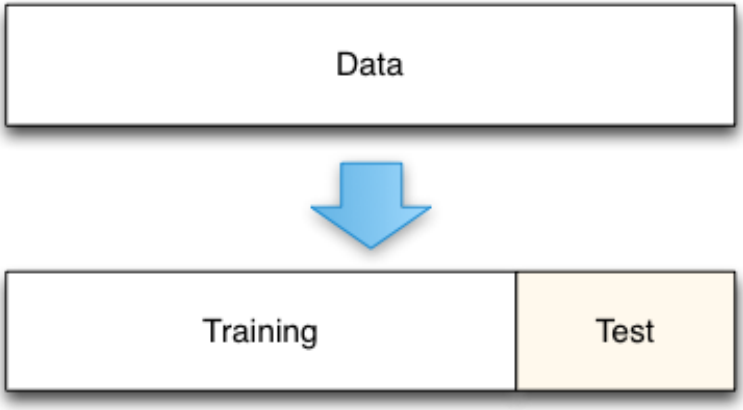
Training Set e Test Set

Valutare un modello

Abbiamo già visto che, nella pratica, un algoritmo di Machine Learning viene utilizzato per fare predizioni sui risultati di future osservazioni. Quindi, nel valutare quale modello sia il migliore su un determinato Dataset, vorremmo valutarne l'abilità su delle nuove osservazioni, non viste durante l'addestramento.

Training e Test Splitting

Questo si ottiene, in genere, dividendo il dataset $\mathcal{S}$ in due parti: il Training Set $\mathcal{TR}$ (usato per addestrare il modello) e il Test Set $\mathcal{TE}$ (usato per valutare il modello).
In generale, $\mathcal{TR}$ contiene circa il 70% delle osservazioni di $\mathcal{S}$, mentre $\mathcal{TE}$ il restante 30%.

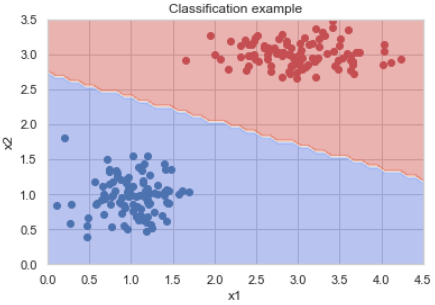


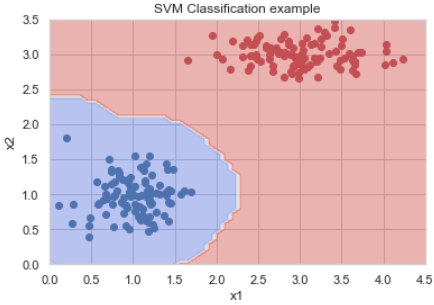
Configurare un Modello

Come già visto nel caso di SVM, ogni modello di Machine Learning è composto da due serie di parametri: una parte, contenuta nel vettore θ che definisce il predittore $h_\theta(x)$, viene imparato in automatico durante l'addestramento. L'altra parte, solitamente detta **iperparametri**, contiene tutte le informazioni che devono essere impostate a mano dall'utente e che definiscono lo specifico algoritmo (nel caso di SVM, gli iperparametri sono il Costo, il tipo di Kernel, il grado del polinomio nel caso di Kernel polinomiale, il termine γ nel caso di Kernel radiale).
Trovare gli iperparametri più adatti per il compito che stiamo cercando di svolgere è una delle operazioni più lunghe e dispendiose del lavoro del Data Scientist.

Flessibilità

Nello scegliere i valori corretti per gli iperparametri, bisogna essere sempre guidati dal concetto di Flessibilità del modello. La Flessibilità descrive l'abilità di un modello di adattarsi perfettamente ai dati.

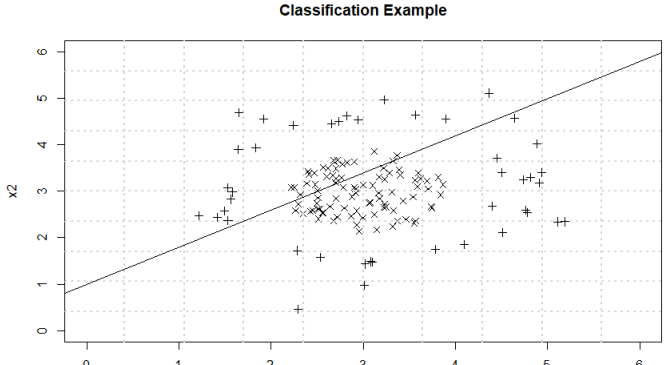




Qua, il modello di sinistra (che ammette curve di separazione solo lineari) è molto meno flessibile del modello di destra (che ammette curve di separazioni molto più complesse).

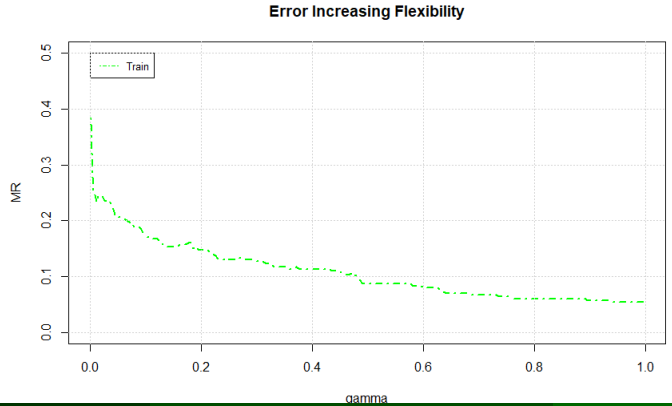
Underfit

Cosa succede se un modello è troppo poco flessibile?
La curva di separazione non riesce a distinguere in maniera adeguata le due classi, e l'errore di predizione è molto alto (anche in fase di addestramento). Il modello non riesce ad imparare nulla.
In una situazione del genere, si dice che il modello **Underfitta**.



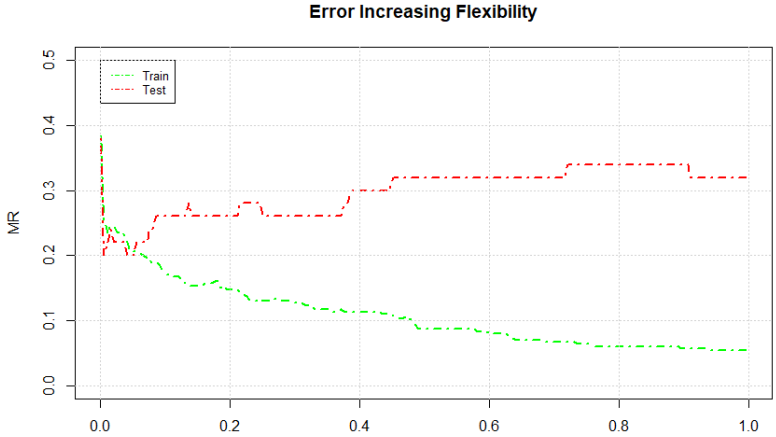
Overfit

Si potrebbe pensare che la scelta migliore sia quella di scegliere sempre il modello più flessibile in assoluto, in modo da essere sicuri di ridurre al minimo l'Underfit. Questa speranza è anche rafforzata dal fatto che, guardando un grafico dell'andamento dell'errore (misurato con MR) al crescere della flessibilità, si ottiene il seguente risultato:



Overfit

Purtroppo, però, come abbiamo detto, il modello va valutato rispetto alla sua performance sul Test Set, non sul Training Set. Facendo la curva dell'errore al variare della flessibilità sul Test Set otteniamo:



Overfit

Quindi, sebbene l'errore sul Training Set sia sempre più basso quanto più il modello è flessibile, l'errore sul Test Set inizialmente decresce, per poi raggiungere un punto di minimo oltre il quale inizia a salire. Quel punto corrisponde al momento in cui il modello è **troppo** flessibile, talmente tanto da imparare a memoria il dataset di input, e questo inficia sulla qualità della predizione su nuove osservazioni. In una situazione del genere, si dice che il modello **Overfitta**.

Overfit

L'Overfit è la ragione per cui scegliere un modello infinitamente flessibile non è necessariamente una buona idea. Il modello migliore è quello che corrisponde al punto di minimo della curva sul Test Set.
ATTENZIONE: il Test Set deve essere usato per la sola valutazione. Non è quindi possibile conoscere a priori il Test Set, e quindi non è possibile utilizzarlo per scegliere gli iperparametri che corrispondano al punto di minimo.

Hyperparameter Tuning

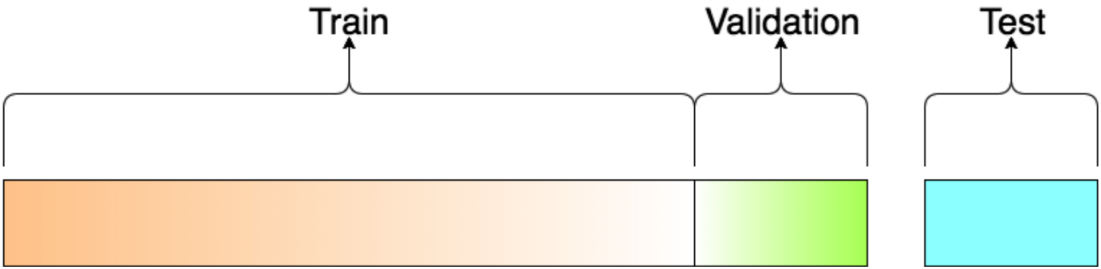
Hyperparameter Tuning

- Abbiamo visto come la scelta degli **iperparametri** di un modello sia cruciale, in quanto la diversa flessibilità del modello può dare luogo a **underfit** o **overfit**.
- La fase in cui gli iperparametri vengono selezionati, viene chiamata **Hyperparameter Tuning**.
- **IMPORTANTE:** Ricordarsi di NON utilizzare mai il test set durante la fase di Tuning (i.e. per identificare l'underfit o overfit del modello), per non falsare le statistiche della previsione!
- Vediamo due possibili tecniche comunemente utilizzate in questa fase: il Validation Set Approach e il k-fold Cross-Validation (k-CV).

Validation Set

- Come abbiamo detto, la scelta ottimale degli iperparametri è quella che corrisponde al punto di minimo della curva dell'errore rispetto al Test Set, ma questo deve essere trovato senza sfruttare il Test Set (che va lasciato libero per la sola fase di valutazione). Come possiamo quindi formulare una strategia in grado di indirizzarci nella scelta degli iperparametri?
- La scelta più comune è quella di dividere nuovamente il training set in due parti, la prima (più grande) continuerà a chiamarsi Training Set, e verrà usata per addestrare il modello e quindi calcolare in maniera automatica i parametri ottimali mentre il secondo, detto **Validation Set**, viene usato per avere una stima della curva dell'errore sul Test Set, e quindi per trovare gli iperparametri ottimali.

Divisione del Dataset



A visualisation of the splits

k-fold Cross-Validation (I)

- L'approccio di utilizzare il Validation Set ha un grande potenziale, ma richiede di avere una quantità sufficiente di dati da poterli dividere in 3 parti, facendo sì che ciascuna delle tre sia sufficientemente grande da essere statisticamente rilevante.
- Quando ciò non è possibile, si utilizza l'approccio noto come k-fold Cross-Validation, o k-CV.

k-fold Cross-Validation (II))

1. Fissata una metrica (per esempio MSE), che quantifichi le perfornance del modello, dividere il training set $\mathcal{S}$ in k sottoinsiemi di ugual dimensione, e indicarli con $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\}$.
2. Per ogni indice $i = 1, \ldots, k$, ripetere:
   1. Rimuovere il sottoinsieme $\mathcal{S}_i$ dal training set, e addestrare il modello su tutti gli altri.
   2. Misurare l'errore compiuto su $\mathcal{S}_i$ (di fatti, trattandolo come se fosse il test set), e salvarlo come MSE_i.
3. Definito:

$$\overline{MSE} = \frac{1}{k} \sum_{i=1}^{k} MSE_i,$$

   si ha che $\overline{MSE}$ è uno stimatore dell'MSE misurato con l'approccio validation set, ma utilizzando almeno una volta tutti i dati per addestrare il modello.
4. Aggiustare gli iperparametri minimizzando $\overline{MSE}$.

LOOCV

- Una scelta importante nell'utilizzo di k-CV è il valore del numero k di insiemi in cui $\mathcal{S}$ viene suddiviso.
- Intuitivamente vorremmo k più grande possibile, in maniera tale che in ogni step il training set sia il più grande possibile.
- Viceversa, però, il metodo k-CV richiede di eseguire k volte l'addestramento, quindi se questo è dispendioso a livello di tempo, vorremmo ridurre k in maniera da ottimizzare le tempistiche.
- C'è quindi un **trade-off** sulla scelta di k, il cui valore ottimale dipenderà dall'esperimento di riferimento.
- Il caso limite, in cui $k = N$, ovvero il training set viene diviso in N sottinsiemi di dimensione 1, viene chiamato **Leave-One-Out Cross-Validation (LOOCV)**, ed è molto utilizzato quando si ha a disposizione molta potenza di calcolo.

No Free Lunch Theorem

- Per concludere, osserviamo come il corretto bilanciamento della flessibilità del modello, in maniera da evitare sia Overfit che Underfit, dipende strettamente dalla complessità dei dati che abbiamo a disposizione. Dati più complessi (come immagini) avranno bisogno di un modello più flessibile (e.g. reti neurali) per evitare l'Underfit, mentre dati più semplici avrano bisogno di un modello poco flessibile, per evitare l'Overfit.
- In poche parole, **non esiste** un modello assolutamente migliore di un altro, ma la bontà di un modello dipende strettamente dal tipo di dati con cui deve confrontarsi. Questo sta alla base del cosiddetto **No Free Lunch Theorem**.