Elaborato finale

Rita Folisi

Indice

1 Introduzione

2 Apprendimento automatico e logica fuzzy

Nel seguente capitolo si passano in rassegna i concetti di apprendimento automatico e di logica fuzzy, illustrando come i due argomenti possano essere coniugati insieme. Nel paragrafo 2.1 ci si soffermerà sull'apprendimento supervisionato, di cui i metodi proposti in seguito faranno parte. Nel paragrafo 2.2 si riprenderanno i concetti relativi ai *fuzzy set*, fino ad illustrare le modalità di induzione della funzione di appartenenza.

2.1 Apprendimento supervisionato

Con apprendimento automatico si intende l'abilità di una macchina di apprendere in maniera autonoma, attraverso diversi algoritmi ed esempi di addestramento. Ciò che la macchina impara è una funzione, una struttura computazionale attraverso il cui apprendimento la macchina è in grado di rispondere a problemi di vario genere, quali la classificazione, la regressione e il clustering.

Da un punto di vista formale[**?**] si considerino una funzione *f*, rappresentante il compito al quale si è interessati, e una funzione *h*, che approssima al meglio *f*. Le funzioni f e h sono definite sul vettore $\mathrm{X} = x_1 ... x_n$ composto da n elementi. Viene assunto a priori che h viene selezionata da una classe di funzioni H, alla quale f può appartenere. All'interno della classe di funzioni, h viene scelta in base a Ξ, un insieme contenente m esempi di apprendimento. Una volta scelta, h viene implementata da una macchina, che avrà dunque input X e in output $h(\mathrm{X})$.

L'apprendimento automatico può essere in generale di due tipi: modalità supervisionata e non supervisionata, la cui trattazione non verrà tuttavia effettuata in sede. Nel caso di modalità superivisionata, si conoscono anticipatamente gli output di f per i valori di Ξ. Se il modello di apprendimento riesce a selezionare una funzione h che approssima al meglio f per i valori di Ξ, allora si può ipotizzare che h sia un'ottima approssimazione per f anche in generale. In altri termini, attraverso esempi di apprendimento costituiti da coppie di input e di output, il modello impara una funzione capace di fornire output corretti in relazione a nuovi input. Quando tale output è rappresentato da una variabile discreta si parla di classificazione, se invece è una variabile continua si parla di regressione. Il campo esaminato riguarda i problemi di classificazione e pertanto l'obiettivo dei metodi presentati in seguito sarà, dato un vettore di input, fornire la *label* corrispondente e corretta. Un esempio di problema di classificazione è il filtraggio antispam. D'altra parte, il vettore di input X può essere definito anche come *feature vector* e le componenti x_i come feature o attributi, i cui valori possono essere di tre tipologie: numeri reali, discreti o valori categorici. L'input viene spesso rappresentato in forma non ordinata affiancando il valore dell'attributo all'attributo stesso. Ad esempio, si può avere (forma: rettangolo, perimetro: 14, colore: rosso), dove forma, perimetro e colore sono attributi, mentre rettangolo, 14 e rosso sono i valori degli attributi.

Una volta addestrato il modello con gli esempi di apprendimento, occorre valutare le performance per comprendere se questo è in grado di eseguire correttamente il suo

compito. L'idea è quindi misurare quanto le predizioni eseguite dal modello siano vicine all'output teorico. I criteri più noti sono l'accuratezza e le funzioni di errore. L'accuratezza è il numero di predizioni corrette divise per il totale delle predizioni effettuate e viene espresso in percentuale. Tuttavia, l'accuratezza non viene indicata come buon valutatore, dal momento che non riassume bene la robustezza assunta dal modello. Si introducono pertanto una funzione di errore da minimizzare, come il *mean square error*, e una soglia sotto la quale il margine di errore è considerato accettabile. L'errore quadratico medio quantifica la differenza tra valore teorico e risposta predetta dal modello attraverso questa formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - h(x_i))^2$$

dove y_i è la label teorica, mentre $h(x_i)$ è la label predetta dal modello. Un'altra funzione d'errore utilizzata è RMSE ovvero la radice quadrata dell'errore quadratico medio:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - h(x_i))^2}$$

La valutazione, tuttavia, non verrà effettuata sugli esempi utilizzati per l'apprendimento, dal momento che l'obiettivo del modello è generalizzare con dati diversi da quelli osservati in precedenza. Una volta scelto il campo su cui è definito il compito che la macchina dovrà imparare, occorre distinguere una parte di dati dedita ad addestrare il modello e una parte di dati con cui valutarne la prestazione. In altre parole, dato un dataset, sarà opportuno distinguere tra *training set* e *testing set*, ognuno con compiti sensibilmente diversi. Il primo verrà utilizzato in fase di apprendimento, mentre il secondo verrà utilizzato esclusivamente per decretare le performance del modello. I due nuovi set possono presentare delle criticità tali da poter alterare l'intero processo di apprendimento e valutazione. Ad esempio, possono avere una dimensione molto elevata di features, comportando un aumento della complessità computazionale dell'algoritmo di apprendimento; per risolvere questo problema, si utilizzano algoritmi di selezione delle features, capaci di selezionare quelle più rilevanti per l'assegnazione dell'output corretto. Un altra criticità è causata dal rumore nei dati. Nel caso sia presente nei dati di training, il rumore di classe altera i valori di *f*, mentre il rumore di attributo altera i valori delle componenti dei vettori di input, compromettendo dunque la corretta associazione input-output. Inoltre, è bene osservare quanto il training set sia rappresentativo del campo in analisi: infatti, può accadere che questo presenti poca varietà e che il metodo dunque non sia in grado di generalizzare. Ad esempio, se nel training set sono presenti, sotto la label di "orologio", solo immagini di orologi rotondi e di color rosso, quando il modello si ritroverà a dover predire la label di un orologio quadrato e giallo probabilmente non darà la risposta corretta. Questo fenomeno prende il nome di *overfitting* ed è facilmente visibile quando il modello si adatta molto ai dati di training ma riporta performance sensibilmente inferiori con i dati di testing.

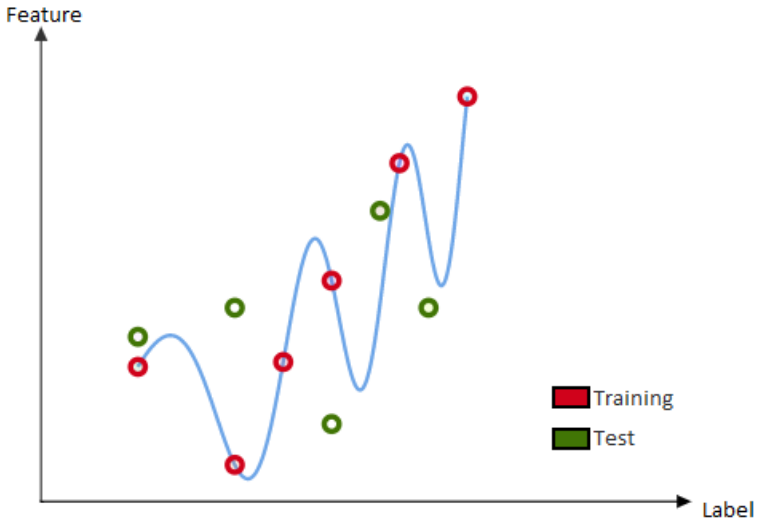

Figura 1: Illustrazione grafica di overfitting

Graficamente, si avrà la situazione presente in Figura 1, ovvero la funzione in blu h(x), appresa dal modello, si adatterà perfettamente ai dati di training (segnati in rosso), ma non riuscirà a ricoprire i dati di testing. In altri termini, se il modello allenato viene eseguito sui dati di training, il modello produrrà un numero sensibilmente minore di errori, rispetto a quando viene eseguito sui dati di testing.

Un buon modo per ridurre gli effetti dell'overfitting consiste nell'introdurre il *validation set*, cioè un sottinsieme del training set, atto a fornire una panoramica di come il modello stia lavorando, senza però fare valutazioni più precise come accade con il test set. In questo modo è già possibile vedere, durante la fase di addestramento, se il modello sappia generalizzare. L'overfitting, infatti, può avvenire anche per una configurazione errata degli iperparametri del modello, cioè parametri che non vengono appresi durante il processo di apprendimento ma che vengono definiti a priori. Il validation set permette di vedere se con i parametri impostati prima del training si stanno ottenendo buoni risultati e, in caso contrario, di modificarli senza dover attendere i risultati sul test set. Si introduce quindi il *tuning dei parametri*, ovvero una tecnica di ottimizzazione degli iperparametri. Viene definito il *range* degli iperparametri, il modello viene addestrato con tutte le combinazioni possibili di iperparametri e viene decretato il *validation score*, che generalmente coincide con l'accuratezza. Alla fine, si selezionano i parametri che massimizzano il validation score e vengono assegnati al modello.

Spesso, per poter eseguire tuning dei parametri e addestramento, si ricorre alla *k-fold cross validation*. Più in generale, la cross validation è una tecnica utilizzata per la suddivisione del dataset in training set, validation set e testing set. Si definisce il numero

fold k e si divide il dataset Ξ in k sottinsiemi mutualmente esclusivi e di ugual dimensione: $\Xi_1 ... \Xi_k$. In seguito, si reitera l'algoritmo di apprendimento k volte considerando ad ogni iterazione il sottinsieme Ξ_k come training set e i rimanenti come validation o testing set. Il motivo per cui si utilizza questa tecnica e non si fa un solo semplice *split* del dataset è da ricercarsi nell'aleatorietà della suddivisione. Infatti, può succedere che uno split casuale inserisca nel training set osservazioni non generali, comportando dunque il già sopracitato problema dell'overfitting. In questo modo, alternando di volta in volta il sottinsieme di training, validation e testing, il modello risulterà più generalizzato e dunque robusto. La valutazione finale viene effettuata facendo la media delle valutazioni parziali dei sottinsiemi. In Figura 2 è presente una visualizzazione grafica della cross validation, con suddivisione in training set e validation set.

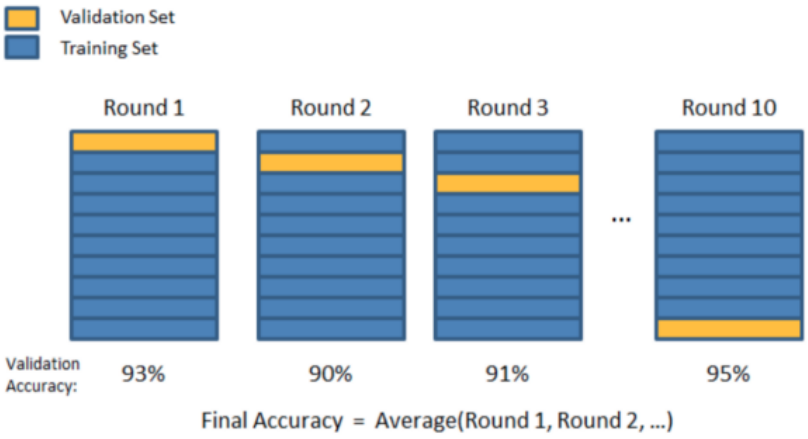

Figura 2: Cross validation