(F94) Metodi statistici per l'apprendimento

L'algoritmo Nearest Neighbour

Docente: *Nicolò Cesa-Bianchi* versione 14 marzo 2016

Vediamo ora un esempio concreto di algoritmo di apprendimento per classificazione binaria che utilizza la regola seguente: "predici ogni $\boldsymbol{x}$ del training set con la sua etichetta e predici ogni altro $\boldsymbol{x}$ con l'etichetta del punto del training set ad esso più vicino". Consideriamo per il momento problemi di classificazione binaria con attributi numerici, cioè problemi tali che $\mathcal{X} = \mathbb{R}^d$ e $\mathcal{Y} = \{-1, +1\}$.

Dato un training set S contenente gli esempi $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)$, l'algoritmo *nearest neighbour* (NN) genera un classificatore $h_{\mathrm{NN}} : \mathbb{R}^d \to \{-1, +1\}$ definito come:

$$h_{\mathrm{NN}}(\boldsymbol{x}) = \text{etichetta } y_t \text{ del punto } \boldsymbol{x}_t \in S \text{ più vicino a } \boldsymbol{x}.$$

Se c'è più di un punto in S a distanza minima da $\boldsymbol{x}$, allora prediciamo con la maggioranza delle etichette di questi punti più vicini. Se c'è un uguale numero di punti più vicini ad $\boldsymbol{x}$ con etichette positive e negative, prediciamo un valore di default in $\{-1, +1\}$.

In particolare, $h_{\mathrm{NN}}(\boldsymbol{x}_t) = y_t$ per ogni esempio $(\boldsymbol{x}_t, y_t)$ del training set. La distanza fra $\boldsymbol{x}$ e $\boldsymbol{x}_t$, denotata con $\|\boldsymbol{x} - \boldsymbol{x}_t\|$, viene misurata con la formula della distanza Euclidea fra due vettori,

$$\|\boldsymbol{x} - \boldsymbol{x}_t\| = \sqrt{\sum_{i=1}^{d} (x_i - x_{i,t})^2} \ .$$

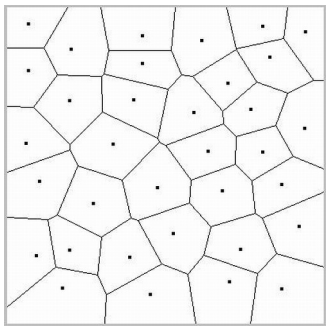

Figura 1: Diagramma di Voronoi per un insieme di punti sul piano.

Il classificatore generato da NN induce una partizione di $X = \mathbb{R}^n$ in *celle di Voronoi*, dove ogni istanza $\boldsymbol{x}_t$ del training set è il centro di una cella e il confine fra due celle è costituito dai punti $\boldsymbol{x}$ equidistanti dai due centri (si veda la Figura 1).

In generale, NN deve memorizzare l'intero training set, quindi l'algoritmo non è pratico per dataset di grandi dimensioni. Inoltre, dato un qualunque $\boldsymbol{x}$ di test, il calcolo dell'etichetta $h_{\mathrm{NN}}(\boldsymbol{x})$ è

computazionalmente oneroso in quanto richiede, in generale, il calcolo delle distanze fra $\boldsymbol{x}$ e ogni $\boldsymbol{x}_t$ del training set. Infine, si noti che l'algoritmo NN genera sempre un classificatore h_{NN} tale che $\widehat{\text{er}}(h_{\text{NN}}) = 0$, cosa non sorprendente dato che NN memorizza tutto il training set.

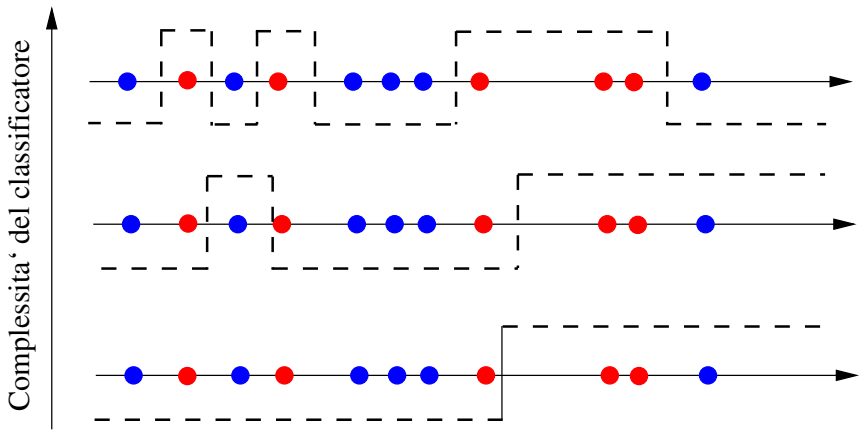

Figura 2: Forma del classificatore $h_{\text{k-NN}}$ per i valori $k = 1, 3, 5$ su un training set unidimensionale. Si noti come all'aumentare di k il classificatore diventi progressivamente più semplice. Qual è il training error in ciascuno dei tre casi?

Da NN possiamo ottenere una famiglia di algoritmi k-NN per $k = 1, 3, 5, \ldots$ definiti nel modo seguente. Dato un training set $S = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)$, k-NN genera un classificatore $h_{\text{k-NN}}$ tale che $h_{\text{k-NN}}(\boldsymbol{x})$ è l'etichetta $y_t \in \{-1, +1\}$ che appare nella maggioranza dei k punti $\boldsymbol{x}_t \in S$ più vicini a $\boldsymbol{x}$.[1] Quindi, per valutare $h_{\text{k-NN}}(\boldsymbol{x})$, si compiono le seguenti operazioni:

1. trovo i k punti $\boldsymbol{x}_{t_1}, \ldots, \boldsymbol{x}_{t_k}$ del training set più vicini a $\boldsymbol{x}$ (potrei trovarne $k' \geq k$, si veda la nota a piè pagina). Siano $y_{t_1}, \ldots, y_{t_k}$ le etichette di questi punti;
2. se la maggioranza delle etichette $y_{t_1}, \ldots, y_{t_k}$ è pari a $+1$, allora $h_{\text{k-NN}}(\boldsymbol{x}) = +1$; se la maggioranza è pari a -1, allora $h_{\text{k-NN}}(\boldsymbol{x}) = -1$; se la maggioranza è pari predico un valore di default in $\{-1, +1\}$.

Si noti che, per ogni $k \geq 1$, $\boldsymbol{x}_t$ è sempre incluso nei k vicini di se stesso.

Oltre alla classificazione binaria, k-NN si presta bene a risolvere problemi di classificazione multiclasse (dove $\mathcal{Y}$ contiene più di due simboli) e di regressione (dove $\mathcal{Y} = \mathbb{R}$). Nel primo caso la predizione è —come nel caso binario— l'etichetta corrispondente alla maggioranza dei k punti più vicini. Nel secondo caso, la predizione è la media delle etichette dei k punti più vicini.

[1] Come nel caso 1-NN, potrebbero esserci più punti del training set alla stessa distanza da $\boldsymbol{x}$ per cui non è possibile avere esattamente k punti più vicini a $\boldsymbol{x}$ a meno di scegliere arbitrariamente fra questi punti equidistanti. In questo caso procediamo ordinando tutti i punti del training set in base alla loro distanza da $\boldsymbol{x}$ e poi prendiamo i k' punti più vicini, dove k' è il più piccolo intero maggiore o uguale a k tale che il $(k'+1)$-esimo punto nell'ordinamento ha distanza da $\boldsymbol{x}$ strettamente maggiore del k'-esimo.

È importante notare che, a differenza di 1-NN, in generale avremo che $\widehat{\text{er}}(h_{\text{k-NN}}) > 0$. Inoltre, in Figura 2 vediamo che, all'aumentare di k, i classificatori costruiti da k-NN su un dataset unidimensionale diventano sempre più semplici. In particolare, quando $k = m$, dove m è la taglia del training set, $h_{\text{k-NN}}$ diventa il classificatore costante che predice ogni punto $\boldsymbol{x}$ con la maggioranza di tutte le etichette del training set.

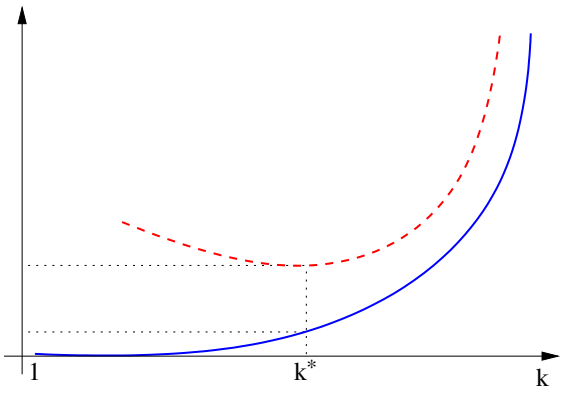

Figura 3: Andamento tipico del training error (blu) e test error (rosso tratteggiato) del classificatore prodotto da k-NN per valori crescenti del parametro k. Si noti che il minimo del test error viene raggiunto per un valore k^* che corrisponde ad un classificatore $h_{\text{k-NN}}$ con training error in generale maggiore di zero. Valori di $k < k^*$ causano overfitting (training error in discesa, test error in salita), mentre valori $k > k^*$ causano underfitting (training e test error vicini ma entrambi relativamente alti).

In Figura 3 è riportato l'andamento tipico di training error e test error dei classificatori $h_{\text{k-NN}}$ per valori crescenti di k su un dataset affetto da "rumore". In generale, diciamo che un dataset è affetto da rumore quando un dato $\boldsymbol{x}$ può comparire (nel training set o nel test set) a volte con un'etichetta e a volte con un'altra. La presenza del rumore può avere due cause (anche concomitanti):

1. L'etichetta è assegnata da una persona che esamina il dato ed esprime un giudizio soggettivo. Pensiamo al caso di documenti che trattano di più argomenti e di conseguenza posso essere categorizzati in modo diverso da persone diverse.

2. L'istanza $\boldsymbol{x}$ non contiene abbastanza informazioni. Per esempio, supponiamo che $\boldsymbol{x}$ rappresenti delle misure atmosferiche (temperatura, pressione, umidità) e $y \in \{-1, +1\}$ indichi se il giorno successivo piove o meno. È molto probabile che queste misure non siano sufficienti a determinare l'etichetta, quindi a parità di valori $\boldsymbol{x}$ rilevati potrebbe seguire sia un giorno di pioggia che un giorno di sole.

In questi casi possiamo parlare di etichette "tipiche" rispetto a etichette "anomale". Nell'esempio dei documenti, l'etichetta di un documento è tipica se corrisponde alla categoria che la maggior parte delle persone assegnerebbe a quel documento ed è anomala se vale il viceversa. Nel caso delle previsioni metereologiche, un'etichetta tipica è la condizione atmosferica (pioggia o sole) che storicamente è stata riscontrata il giorno successivo a una rilevazione di dati valori. Un buon algoritmo

di apprendimento dovrebbe scegliere un classificatore (o regressore) che minimizza il training error sui punti con etichette tipiche, ignorando quelli con etichette anomale. Dato che a priori non è noto quali siano i punti anomali, si può procedere vincolando l'algoritmo di apprendimento a scegliere classificatori o regressori di complessità sufficientemente bassa, in modo da forzarlo a trascurare esattamente i punti anomali del training set. Questo è esattamente ciò che fa k-NN all'aumentare di k. Infatti, in Figura 2 notiamo che all'aumentare di k la complessità (espressa in questo caso dal numero di cambi di segno) del classificatore descresce.

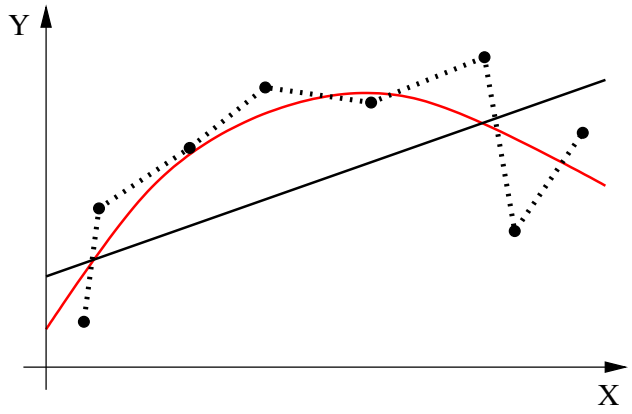

Figura 4: Controllo dell'overfitting in un problema di regressione unidimensionale. Il training set è formato dagli otto punti $(x_t, y_t) \in \mathbb{R}^2$ indicati in nero in figura. Le curve rappresentano predittori della forma $f : \mathbb{R} \to \mathbb{R}$ per il problema di regressione consistente nel predire il valore della coordinata y associata a punti x nel test set (non mostrati in figura). La curva in colore rappresenta il predittore che minimizza il test error. Le altre curve rappresentano predittori di complessità inferiore (linea retta, underfitting) o superiore (linee tratteggiate, overfitting) alla complessità del predittore ottimo.

Il fenomeno per il quale un algoritmo di apprendimento tende a generare predittori con basso training error e alto test error prende il nome di **overfitting**. Il controllo dell'overfitting è uno dei temi chiave nell'apprendimento automatico. Nel caso di problemi di regressione, un semplice esempio di overfitting è fornito dalla Figura 4.