

L'algoritmo Nearest Neighbour

Vediamo ora un esempio concreto di algoritmo di apprendimento per classificazione binaria che utilizza la regola seguente: “predici ogni \mathbf{x} del training set con la sua etichetta e predici ogni altro \mathbf{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 $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$, l'algoritmo *nearest neighbour* (NN) genera un classificatore $h_{\text{NN}} : \mathbb{R}^d \rightarrow \{-1, +1\}$ definito come:

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

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

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

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

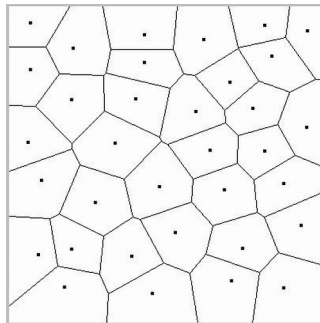


Figura 1: Diagramma di Voronoi per un insieme di punti sul piano.

Il classificatore generato da NN induce una partizione di $X = \mathbb{R}^n$ in *celle di Voronoi*, dove ogni istanza \mathbf{x}_t del training set è il centro di una cella e il confine fra due celle è costituito dai punti \mathbf{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 \mathbf{x} di test, il calcolo dell'etichetta $h_{\text{NN}}(\mathbf{x})$ è

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

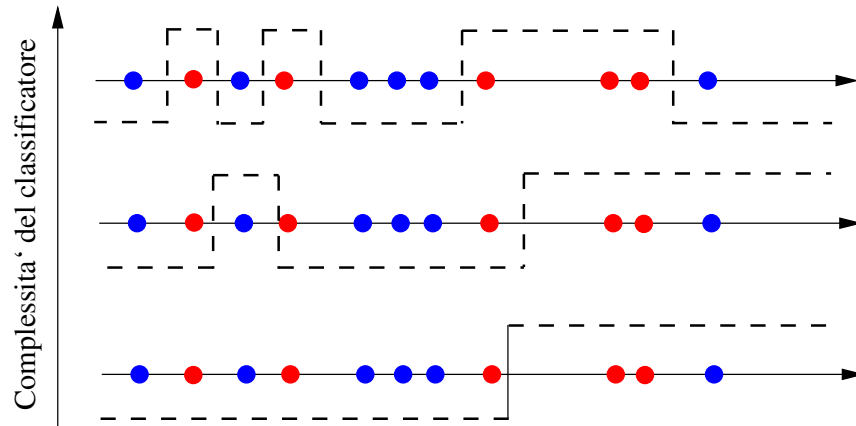


Figura 2: Forma del classificatore $h_{k\text{-NN}}$ per i valori $k = 1, 3, 5$ su un training set unidimensionale. Si noti come all’aumentare di k il classificatore diventi progressivamente più semplice. Qual è il training error in ciascuno dei tre casi?

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

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

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

Oltre alla classificazione binaria, $k\text{-NN}$ 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.

¹Come nel caso 1-NN, potrebbero esserci più punti del training set alla stessa distanza da \mathbf{x} per cui non è possibile avere esattamente k punti più vicini a \mathbf{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 \mathbf{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 \mathbf{x} strettamente maggiore del k' -esimo.

È importante notare che, a differenza di 1-NN, in generale avremo che $\hat{\text{er}}(h_{k\text{-NN}}) > 0$. 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, $h_{k\text{-NN}}$ diventa il classificatore costante che predice ogni punto \mathbf{x} con la maggioranza di tutte le etichette del training set.

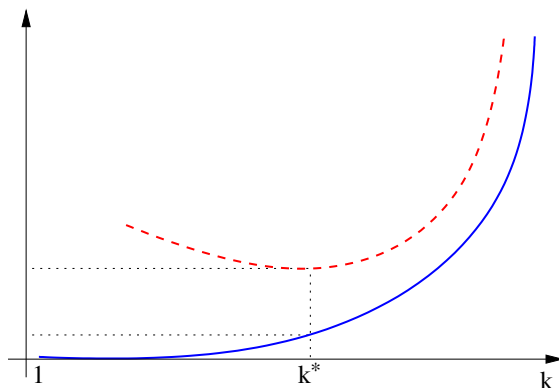


Figura 3: Andamento tipico del training error (blu) e test error (rosso tratteggiato) del classificatore prodotto da k -NN per valori crescenti del parametro k . Si noti che il minimo del test error viene raggiunto per un valore k^* che corrisponde ad un classificatore $h_{k\text{-NN}}$ con training 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 $h_{k\text{-NN}}$ per valori crescenti di k su un dataset affetto da rumore. L'overfitting si verifica quando il classificatore utilizzato è troppo complesso rispetto alla cardinalità del training set.