1

Apprendimento Automatico

1.1 Introduzione

Il problema di *apprendere* dai dati può essere formalizzato in termini matematici definendo tre componenti principali [Vapnik99]:

1. Un generatore di vettori casuali x, estratti a caso da una data distribuzione $P(x)$, non nota.

2. Un **supervisore** che restituisce un vettore di output y per ogni vettore di input x, seguendo una data distribuzione condizionata $P(y|x)$, non nota.

3. Un algoritmo di apprendimento, capace di implementare un insieme di funzioni $f(x, w)$ con $w \in \Lambda$.

Il problema dell'apprendimento automatico consiste nel determinare una funzione dell'insieme $f(x, w), w \in \Lambda$ che predice la risposta del supervisore nel miglior modo possibile. La scelta della funzione si basa su un campione di dati (il *training set*) che consiste di N osservazioni casuali, indipendenti e identicamente distribuiti, secondo la distribuzione $P(x, y) = P(x)P(y|x)$:

$$(x_1, y_1), \ldots, (x_N, y_N). \tag{1.1}$$

Per poter scegliere la miglior approssimazione possibile della risposta del supervisore, si deve misurare la **distanza** $L(y, f(x, w))$ (in inglese, la funzione $L()$ viene chiamata funzione di *loss* o *discrepancy*) tra la risposta y del supervisore per un dato vettore di input x, e la risposta $f(x, w)$ restituita dall'algoritmo di apprendimento.

Per avere una definizione più precisa, si consideri il valore atteso di questa distanza, data dalla *funzione di rischio*

$$R(w) = \int L(y, f(x, w))\, dP(x, y). \tag{1.2}$$

L'obbiettivo dell'algoritmo di apprendimento è quello di trovare la funzione $f(x, w^*)$ che minimizza la funzione di rischio $R(w)$, considerando che la distribuzione congiunta $P(x, y)$ non è nota e la sola informazione che si possiede è data dal training set (1.1).

Questa formulazione del problema di apprendimento automatico è molto generale e comprende i tre seguenti problemi particolari.

Pattern Recognition (Problema di Classificazione). Il supervisore restituisce solo due possibili valori $y \in \{0, 1\}$ e la classe di funzioni $f(x, w), w \in \Lambda$ comprende solo le funzioni che possono restituire solo due valori: 0 oppure 1. Si definisce la funzione di loss seguente:

$$L(y, f(x, w)) = \begin{cases} 0 & \text{se } y = f(x, w) \\ 1 & \text{se } y \neq f(x, w). \end{cases} \tag{1.3}$$

Con questa funzione di loss, la (1.2) calcola la probabilità di avere un errore di classificazione: la risposta data dal supervisore e dalla funzione $f(x, w)$ sono diverse. Quindi, in questo caso, il problema è di trovare la funzione che minimizza la probabilità di avere un errore di classificazione quando la distribuzione $P(x, y)$ non è nota e sono solo disponibili gli N campioni di dati (1.1).

Stima della funzione di regressione. Il supervisore restituisce un numero reale y, e la classe di funzioni $f(x, w), w \in \Lambda$ è dato dall'insieme di funzioni reali che contiene la *funzione di regressione*

$$f(x, w^*) = \int y \, dP(y|x). \tag{1.4}$$

Si dimostra che se $f(x, w) \in L_2$, allora la funzione di regressione è quella che minimizza (1.2) con la seguente funzione di loss

$$L(y, f(x, w)) = (y - f(x, w))^2. \tag{1.5}$$

Quindi, il problema di stima dei parametri di regressione è il problema di minimizzare la funzione (1.2) con la funzione di rischio (1.5) quando la distribuzione $P(x, y)$ non è nota e sono solo disponibili gli N campioni di dati (1.1).

Stime della funzione di densità: Si consideri il problema di trovare la funzione di densità a partire dall'insieme di funzioni di densità $p(x, w), w \in \Lambda$. In questo caso, consideriamo la funzione di loss

$$L(p(x, w)) = -\log p(x, w). \tag{1.6}$$

Si dimostra, che la funzione di densità cercata minimizza la funzione (1.2) con la funzione di rischio (1.6). Quindi, anche in questo caso, per il problema

di stimare la funzione di densità, si deve minimizzare la funzione di rischio con la condizione che la misura di probabilità $P(x)$ non è nota, ma sono dati N campioni indipendenti e identicamente distribuiti

$$x_1, \ldots, x_N.$$

Si osservi, che solitamente nella stima delle densità di probabilità si *massimizza* la likelihood: invece di minimizzare la (1.6), si massimizza $-L(p(x, w)) = \log p(x, w)$, ma a meno del segno, il problema è lo stesso.

1.2 Il problema dell'apprendimento automatico

I tre problemi precedenti possono essere visti come tre casi particolari del problema di apprendimento automatico seguente.

Funzione di rischio (teorico). Sia definita una misura di probabilità $P(z)$ su uno spazio Z. Si consideri l'insieme di funzioni $Q(z, w), w \in \Lambda$. L'obiettivo è di minimizzare la funzione di rischio

$$R(w) = \int Q(z, w)\, dP(z), \qquad w \in \Lambda \tag{1.7}$$

nel caso in cui la misura di probabilità $P(z)$ non è nota, ma sono dati N campioni indipendenti e identicamente distribuiti

$$z_1, \ldots, z_N.$$

In altri termini, si risolve il seguente problema di ottimizzazione

$$w^* = \arg\min_{w \in \Lambda} R(w) = \arg\min_{w \in \Lambda} \int Q(z, w)\, dP(z). \tag{1.8}$$

Nei tre casi visti nella sezione precedente, le variabili z erano date in termini delle coppie (x, y), e la funzione $Q(z, w)$ era una delle tre funzioni di loss: (1.3), (1.5) o (1.6).

Funzione di rischio empirico. Nella pratica, poiché la distribuzione $P(z)$ non è nota, invece di usare la funzione di rischio (1.7), si usa la cosiddetta funzione di rischio empirica

$$\hat{R}(w) = \frac{1}{N} \sum_{i=1}^{N} Q(z_i, w) \tag{1.9}$$

in cui si calcola la media dei valori della funzione nei punti del training set. In questo caso, il problema di ottimizzazione che si deve risolvere diventa:

$$\hat{w} = \arg\min_{w \in \Lambda} \hat{R}(w) = \arg\min_{w \in \Lambda} \frac{1}{N} \sum_{i=1}^{N} Q(z_i, w). \tag{1.10}$$

L'idea è quella di *approssimare* la funzione $Q(z, w^*)$ che minimizza (1.7) con la funzione $Q(z, \hat{w})$ che minimizza la (1.9). Questo viene chiamato l'*empirical risk minimization induction principle (ERM principle).*

Da un punto di vista teorico, siamo interessati a stimare quanto la funzione $Q(z, \hat{w})$ che minimizza il rischio empirico sia una buona approssimazione della funzione $Q(z, w^*)$ che minimizza il rischio teorico, in cui, teoricamente, si considerano tutti i punti della distribuzione $P(y|x)$. Ovvero, si vuole analizzare sotto quali condizioni $Q(z, \hat{w})$ è una buona approssimazione di $Q(z, w^*)$ quando $N \to \infty$. Per una trattazione di questo interessante argomento si rimanda, per esempio, a [testo].

Esempio: Regressione lineare e minimi quadrati. Nel caso della stima dei parametri per una regressione lineare, si ha un vettore di $p + 1$ dimensioni

$$z = (x, y) = (x_1, \ldots, x_p, y)$$

e si usa la funzione di rischio (1.5). I parametri della regressione lineare si trovano risolvendo il problema:

$$\hat{w} = \arg\min_{w \in \Lambda} \frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i, w))^2. \tag{1.11}$$

che corrisponde proprio al metodo dei "minimi quadrati".

1.3 Regressione Lineare

La regressione lineare rimane uno degli strumenti più usati per l'apprendimento automatico basato su metodi di statistica.

Sia dato un vettore $x \in \mathbb{R}^p$, e si consideri come classe di funzioni $f(x, w)$, la seguente funzione lineare

$$y = f(x, w) = w_0 + \sum_{i=j}^{p} x_j w_j. \tag{1.12}$$

Il termine w_0 è l'*intercetta*, chiamato anche *bias*. Per semplificare la notazione, si include nel vettore x il termine 1, in modo da includere il termine w_0 nel vettore dei parametri w. In questo modo, possiamo scrivere la nostra funzione lineare come (in cui x è un vettore colonna)

$$y = f(x, w) = x^T w. \tag{1.13}$$

Si noti che nel caso in cui l'output invece di essere uno scalare sia una funzione di q coefficienti, i parametri w sono dati da una matrice di $(p+1) \times q$ coefficienti.

Se calcoliamo il gradiente della (1.13), otteniamo $f'(x, w) = w$, ovvero esattamente il vettore w dei parametri del nostro modello. Essendo w il gradiente, esso punta nella direzione di massima crescita per la nostra funzione lineare.

Dato un insieme di campioni di punti i.i.d. $z_i = (x_i, y_i), i = 1, \ldots, N$, il problema di trovare i parametri w in modo tale che il modello lineare dia la migliore approssimazione possibile della distribuzione $P(y|x)$, viene risolto usando la funzione di loss (1.5), e si vuole minimizzare il rischio empirico

$$\hat{R}(w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - x_i^T w)^2 = (y - Xw)^T(y - Xw), \tag{1.14}$$

in cui X è una matrice di dimensione $N \times (p+1)$, in cui ogni riga corrisponde ad un vettore di input e y è il vettore degli N output del training set.

Per trovare il valore di w che minimizza la (1.14), si deriva la (1.14) rispetto w e si pone il risultato uguale a zero, ottenendo la cosiddetta **equazione normale**

$$X^T(y - Xw) = 0 \tag{1.15}$$

Se la matrice X non è singolare, e quindi è invertibile, allora l'unica soluzione dell'equazione normale è data da

$$\hat{w} = (X^T X)^{-1} X^T y. \tag{1.16}$$

Nell'ottica di un sistema di apprendimento automatico, un algoritmo in grado di calcolare $\hat{w}$ viene detto "imparare" la funzione $Q(z, \hat{w}) = x^T\hat{w}$ che può essere usata per calcolare il valore $\hat{y}$ per qualsiasi punto non appartenente al training set iniziale: $\hat{y} = x^T\hat{w}$.

Esempio. (TO-DO)

1.4 Regressione Logistica

In alcuni problemi di classificazione risulta conveniente usare come classi di funzioni $f(x, w)$ su cui fare apprendimento, delle funzioni che portano alla cosiddetta regressione logistica.

In pratica, il modello di regressione logistica nasce dal desiderio di rappresentare la probabilità a posteri di appartenere a k classi come funzioni lineari in x (problema di classificazione con K classi), mantenendo il vincolo che questi valori di probabilità siano compresi in $[0, 1]$ e la loro somma sia pari a 1. Il modello di regressione logistica viene scritto a partire dalle

seguenti relazioni

$$\begin{aligned}
\log \frac{Pr(y=1|x)}{Pr(y=K|x)} &= x^T w_1 \\
\log \frac{Pr(y=2|x)}{Pr(y=K|x)} &= x^T w_2 \\
&\vdots \\
\log \frac{Pr(y=k-1|x)}{Pr(y=K|x)} &= x^T w_{k-1}
\end{aligned}$$

Il modello viene specificato tramite $k-1$ logaritmi di rapporti. Si dimostra con dei semplici calcoli che

$$\begin{aligned}
Pr(y=k|x) &= \frac{e^{x^T w_k}}{1+\sum_{l=1}^{K-1} e^{x^T w_l}}, \quad k=1,\ldots,K-1, \\
Pr(y=K|x) &= \frac{1}{1+\sum_{l=1}^{K-1} e^{x^T w_l}}.
\end{aligned}$$

Si noti che ogni probabilità $Pr(y=k|x)$ dipende dall'intero insieme dei parametri $w=(w_1,\ldots,w_{K-1})$, ovvero w è una matrice di dimensione $(p+1)\times(K-1)$.

Nel caso della regressione logistica, dobbiamo quindi stimare i parametri delle $K-1$ densità di probabilità $p_k(x,w)$, e possiamo scegliere come funzione di loss la (1.6). Vediamo nei prossimi paragrafi cosa succede nel caso più semplice in cui si hanno solo due classi, ovvero in cui si usa una regressione logistica per un problema di classificazione binario.

Nel caso in cui si hanno due classi possiamo considerare il caso $y_i=1$ quando l'i-esimo campione appartiene alla prima classe, e il caso $y_i=0$ quando l'i-esimo campione appartiene alla seconda classe. Abbiamo quindi $p_1(x,w)=p(x,w)$ e $p_2(x,w)=1-p(x,w)$, ovvero

$$\begin{aligned}
p_1(x,w) &= \frac{e^{x^T w}}{1+e^{x^T w}} \\
p_2(x,w) &= \frac{1}{1+e^{x^T w}}.
\end{aligned}$$

In questo caso, i parametri sono completamenti descritti da un singolo vettore di parametri w.

Nel caso di due classi, la funzione di rischio empirico diventa (ovvero

$p(x, w)$ è una distribuzione multinomiale)

$$\begin{aligned}\hat{R}(w) &= -\sum_{i=1}^{N} \left(y_i \log p(x_i, w) + (1 - y_i) \log(1 - p(x_i, w))\right) \\ &= -\sum_{i=1}^{N} \left(y_i \log \frac{e^{x_i^T w}}{1 + e^{x_i^T w}} + (1 - y_i) \log \frac{1}{1 + e^{x_i^T w}}\right) \\ &= -\sum_{i=1}^{N} \left(y_i x_i^T w - \log(1 + e^{x^T w})\right).\end{aligned}$$

Per trovare il valore $\hat{w}$ che dia il minimo della funzione precedente, come per la regressione lineare, calcoliamo la derivata prima della funzione precedente e la poniamo uguale a zero, ottenendo le $p + 1$ equazioni non lineari in w

$$\frac{\partial \hat{R}(w)}{\partial w} = -\sum_{i=1}^{N} x_i \left(y_i - \frac{e^{x_i^T w}}{1 + e^{x_i^T w}}\right) = 0. \tag{1.17}$$

Si noti che poiché la prima componente del vettore x è uguale a 1, la prima equazione sarà

$$\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} \frac{e^{x_i^T w}}{1 + e^{x_i^T w}},$$

imponendo che il numero atteso di campioni della prima classe è pari al numero di campioni osservati che appartengono alla prima classe.

Per risolvere le equazioni non lineari precedenti possiamo usare l'algoritmo di Newton Raphson, che ha bisogno delle derivate seconde, ovvero della matrice hessiana

$$\frac{\partial^2 \hat{R}(w)}{\partial w \partial w^T} = \sum_{i=1}^{N} x_i \, x_i^T \, \frac{e^{x_i^T w}}{(1 + e^{x_i^T w})^2}. \tag{1.18}$$

L'algoritmo di Newton Raphson calcola il valore otteniamo w in maniera iterativa, utilizzando la seguente regola di aggiornamento

$$w^{j+1} = w^j - \left(\frac{\partial^2 \hat{R}(w)}{\partial w \partial w^T}\right)^{-1} \frac{\partial \hat{R}(w)}{\partial w}, \tag{1.19}$$

in cui tutte le derivate sono valutate in w^j. Anche se non si ha alcuna dimostrazione di convergenza, nella pratica di solito si parte con $w = 0$.

Se riscriviamo tutto in notazione matriciale possiamo ottenere delle regole di aggiornamento più semplici da leggere e interpretare. Sia y il vettore dei valori di output y_i, sia X la matrice data dagli N vettori di input di dimensione $p + 1$, sia p il vettore delle probabilità calcolate per l'i-esimo campione di dati $p(x_i, w^j)$, e sia W la matrice $N \times N$ di pesi con l'i-esimo elemento diagonale pari a $p(x_i, w^j)(1 - p(x_i, w^j))$. Allora possiamo scrivere

$$\frac{\partial \hat{R}(w)}{\partial w} = -X^T(y - p) \tag{1.20}$$

$$\frac{\partial^2 \hat{R}(w)}{\partial w \partial w^T} = X^T W X. \tag{1.21}$$

Il passo di Newton Raphson diventa quindi

$$\begin{aligned} w^{j+1} &= w^j + (X^T W X)^{-1} X^T (y - p) && (1.22)\\ &= (X^T W X)^{-1} X^T W (X w^j + W^{-1}(y - p)) && (1.23)\\ &= (X^T W X)^{-1} X^T W z, && (1.24) \end{aligned}$$

in cui, per semplificare la notazione, abbiamo prima moltiplicato w^j per $(X^T W X)^{-1}(X^T W X)$ e poi abbiamo moltiplicato $(y - p)$ per $W^{-1}W$. In pratica abbiamo riformulato il passo di Newton Raphson come un passo di minimi quadrati pesati, con risposta

$$z = X w^j + W^{-1}(y - p). \tag{1.25}$$

Queste equazioni vengono aggiornate ad ogni iterazione j, in quanto ad ogni iterazione cambia la matrice p, la matrice W e il vettore z. Questo algoritmo viene chiamato *iteratively reweighted least square (IRLS)*, in quanto ad ogni iterazione risolve il seguente problema ai minimi quadrati pesato

$$w^{j+1} = \arg\min_{w \in \Lambda} (z - Xw)^T W (z - Xw). \tag{1.26}$$

Poiché stiamo minimizzando una funzione che ha come minimo 0, possiamo terminare le iterazione del metodo quando il rischio empirico è minore di una certa soglia, oppure dopo un numero massimo di iterazioni. Si osservi che nel calcolo di w^{j+1}, si potrebbero avere delle matrici non singolari: in questo caso si può usare la matrice pseudo inversa.

1.5 Support Vector Machine

(TO-DO)

1.6 Reti Neurali

(TO-DO)