(F94) Metodi statistici per l'apprendimento

Reti neuronali e deep learning

Docente: *Nicolò Cesa-Bianchi* versione 7 luglio 2016

Le reti neuronali sono una famiglia di predittori che, nella sua forma più semplice (detta rete neuronale *feedforward*), è ottenuta combinando semplici predittori della forma $g(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x})$. La funzione non lineare $\sigma : \mathbb{R} \to \mathbb{R}$ è detta funzione di attivazione e tipicamente è definita come

$$\sigma(z) = \frac{1 - e^{-z}}{1 + e^{-z}} \in [-1, +1] \ . \tag{1}$$

Una rete neuronale feedforward calcola una funzione $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^n$ rappresentata come un grafo diretto aciclico $G = (V, E)$. Un esempio particolare è quello mostrato in Figura 1, dove i nodi sono suddivisi in strati, c'è un solo nodo nello strato di output (*output layer*) e, ordinando gli strati da quello di input (*input layer*) a quello di output, ogni nodo non di input è connesso a tutti i nodi dello strato precedente. In generale, possiamo partizionare i nodi in tre sottoinsiemi: $V = V_{\text{in}} \cup V_{\text{hid}} \cup V_{\text{out}}$, dove V_{in} (con $|V_{\text{in}}| = d$) sono i nodi dello strato di input (ovvero, nodi che non ricevono nessun arco in ingresso), V_{out} (con $V_{\text{out}} = n$) sono i nodi nello strato di output (ovvero, nodi che non emettono nessun arco in uscita) e V_{hid} sono i nodi dello strato nascosto (*hidden layer*).

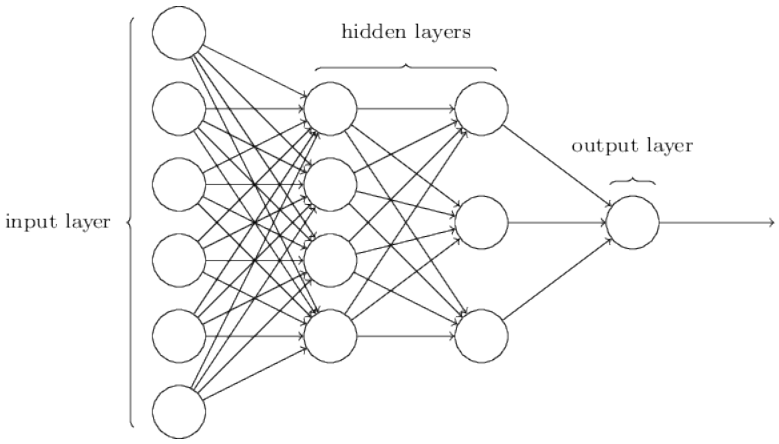

Figura 1: Un esempio di rete neuronale feedforward con output singolo e due strati nascosti in cui le connessioni sono complete fra ogni coppia di strati adiacenti (per esempio fra i due strati nascosti) e assenti fra ogni coppia di strati non adiacenti (per esempio fra strato di input e strato di output).

Ad ogni arco $(i, j) \in E$ è associato un parametro (detto peso) $w_{i,j} \in \mathbb{R}$; indichiamo con W la matrice $|V| \times |V|$ dei pesi, dove $w_{i,j} = 0$ se $(i, j) \notin E$. Il grafo G, la matrice dei pesi W e la funzione di attivazione σ definiscono la funzione $\boldsymbol{f} = \boldsymbol{f}_{G,W,\sigma}$ calcolata dalla rete. Per calcolare questa funzione, ad ogni nodo $i \in V$ è associato un valore $v_i \in \mathbb{R}$ nel modo spiegato in seguito.

Dato $j \in V \setminus V_{\text{in}}$, indichiamo con:

- $\boldsymbol{w}(j)$ il vettore le cui componenti sono i pesi $w_{i,j}$ per ogni $(i,j) \in E$,
- $\boldsymbol{v}(j)$ il vettore le cui componenti sono i valori v_i per ogni $(i,j) \in E$.

Il valore $\boldsymbol{f}(\boldsymbol{x}) = (f_1, \ldots, f_n)$ viene calcolato nel modo seguente:

1. ogni nodo $i \in V_{\text{in}}$ assume valore $v_i = x_i$,
2. ogni nodo $j \in V \setminus V_{\text{in}}$ assume valore $v_j = \sigma\big(\boldsymbol{w}(j)^\top \boldsymbol{v}(j)\big)$,
3. $f_k = v_k$, dove k è il k-esimo nodo di output.

Fissati d (dimensionalità dell'input) e n (dimensionalità dell'output), introduciamo la classe $\mathcal{F}_{G,\sigma}$ dei predittori $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^n$ tali che $\boldsymbol{f} = \boldsymbol{f}_{G,W,\sigma}$ per una qualche matrice dei pesi W, ovvero predittori rappresentabili da una rete neuronale feedforward con grafo G, matrice dei pesi W e funzione di attivazione σ.

Chiaramente, possiamo utilizzare una rete neuronale con un singolo nodo di output come classificatore binario assegnando a un'instanza $\boldsymbol{x}$ l'etichetta $\text{sgn}\big(f_{G,W,\sigma}(\boldsymbol{x})\big)$. Ci chiediamo allora quali classificatori binari $f : \mathbb{R}^d \to \{-1,+1\}$ può realizzare una rete neuronale. Dimostriamo ora un risultato piuttosto sorprendente: quando le istanze sono binarie ($\boldsymbol{x} \in \{-1,+1\}^d$) e la funzione di attivazione σ è la funzione segno sgn, una singola rete con un unico strato nascosto è in grado di implementare qualsiasi classificatore $f : \{-1,+1\}^d \to \{-1,+1\}$. Si noti che la restrizione a istanze binarie non è vincolante in pratica, in quanto ogni coordinata reale x_i è comunque rappresentata in un computer da una stringa di bit di lunghezza fissa.

Teorema 1 *Per ogni d, esiste una rete neuronale $G = (V,E)$ con $d+1$ nodi di input, uno strato di nodi nascosti e un nodo di output, tale the $\mathcal{F}_{G,\text{sgn}}$ contiene tutte le funzioni della forma $f : \{-1,+1\}^d \to \{-1,+1\}$.*

Dimostrazione. Fissiamo una rete con $2^d + 1$ nodi nascosti e connettività completa dello strato di input con lo strato nascosto e dello strato nascosto con lo strato di output. Numeriamo i nodi come segue: 0 è il nodo di output, $1, \ldots, 2^d + 1$ sono i $2^d + 1$ nodi nascosti e i seguenti sono i $d+1$ nodi di input. Dato una qualunque funzione $f : \{-1,+1\}^d \to \{-1,+1\}$ dimostriamo che esiste una matrice dei pesi W tale che $f_{G,W,\text{sgn}} = f$. Siano $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, con $N \leq 2^d$, tutte e sole le istanze $\boldsymbol{x}_i \in \{-1,+1\}^d$ tali che $f(\boldsymbol{x}_i) = 1$. Si osservi che per ogni $\boldsymbol{x} \in \{-1,+1\}$ e per ogni $i = 1, \ldots, N$,

$$\begin{array}{ll} \boldsymbol{x}^\top \boldsymbol{x}_i \leq d-2 & \text{se } \boldsymbol{x} \neq \boldsymbol{x}_i \\ \boldsymbol{x}^\top \boldsymbol{x}_i = d & \text{altrimenti.} \end{array}$$

Quindi, $g_i(\boldsymbol{x}) = \text{sgn}\big(\boldsymbol{x}^\top \boldsymbol{x}_i - d + 1\big) = 1$ se e solo se $\boldsymbol{x} = \boldsymbol{x}_i$. Ogni funzione g_i può essere implementata dal nodo nascosto i nel modo seguente: si trasforma ogni istanza $\boldsymbol{x} = (x_1, \ldots, x_d)$ nell'istanza $\widetilde{\boldsymbol{x}} = (\boldsymbol{x}, 1) = (x_1, \ldots, x_d, 1)$ —questa è la tecnica standard che abbiamo usato anche per i classificatori lineari— e si assegnano i pesi $\boldsymbol{w}(i) = (\boldsymbol{x}_i, -d+1)$. Questo implica che $\text{sgn}\big(\boldsymbol{w}(i)^\top \widetilde{\boldsymbol{x}}\big) = g_i(\boldsymbol{x})$. Poi

assegnamo $\boldsymbol{w}(N+1) = (0, \ldots, 0, 1)$ in modo che $\operatorname{sgn}\big(\boldsymbol{w}(i)^\top \widetilde{\boldsymbol{x}}\big) = 1$ per ogni $\boldsymbol{x}$ (i rimanenti $\boldsymbol{w}(i)$ per $i = N+2, \ldots, 2^d+1$ possono essere assegnati arbitrariamente). Infine, f può essere rappresentata come $f(\boldsymbol{x}) = g_1(\boldsymbol{x}) \vee \cdots \vee g_N(\boldsymbol{x})$, il che corrisponde ad assegnare i pesi

$$\boldsymbol{w}(0) = \big(\underbrace{1, \ldots, 1}_{N \text{ volte}}, N-1, \underbrace{0, \ldots, 0}_{2^d - N \text{ volte}}\big)$$

dove 0 è il nodo di output. □

La dimostrazione del teorema precedente utilizza una rete con uno strato nascosto di dimensione almeno pari al numero di istanze positive per la funzione f da simulare. Questo numero è tipicamente esponenenziale nella dimensione d. Il risultato seguente (che non dimostreremo) stabilisce che esistono delle funzioni f che richiedono un numero di nodi nello strato nascosto esponenziale in d.

Teorema 2 *Per ogni intero d sia $s(d)$ il più piccolo intero tale che esiste $G = (V, E)$ con $|V| = s(d)$ per cui $\mathcal{F}_{G,\mathrm{sgn}}$ contiene tutte le funzioni della forma $f : \{-1,+1\}^d \to \{-1,+1\}$. Allora $|V| = \Omega(d^{2/3})$. Un risultato analogo vale quando* sgn *è rimpiazzata dalla funzione sigmoidale (1).*

Le reti feedforward non solo implementano qualsiasi funzione booleana $f : \{-1,+1\}^d \to \{-1,+1\}$ ma sono anche in grado di approssimare qualsiasi funzione continua $f : [-1,+1]^d \to [-1,+1]^d$ e Lipschitziana (ovvero tale che esiste $L < \infty$ che soddisfa $\big|f(\boldsymbol{x}) - f(\boldsymbol{x}')\big| \leq L \|\boldsymbol{x} - \boldsymbol{x}'\|$ per ogni $\boldsymbol{x}, \boldsymbol{x}' \in [-1,+1]^d$) quando la funzione di attivazione utilizzata è la sigmoidale (1). Ovvero, per ogni $\varepsilon > 0$ e per ogni f esistono G, W tali che $\big|f_{G,W,\sigma}(\boldsymbol{x}) - f(\boldsymbol{x})\big| \leq \varepsilon$ per ogni $\boldsymbol{x} \in [-1,+1]$. Però, come nel caso delle funzioni booleane, anche in questo caso esistono delle funzioni f che sono approssimabili da reti neuronali feedforward con un unico strato nascosto e funzione di attivazione sigmoidale soltanto quando il numero di nodi nello strato nascosto è esponenziale in d.

Sulla base di questi risultati, ci chiediamo allora se aggiungendo strati nascosti alla rete (*deep neural networks*) possiamo implementare/approssimare funzioni usando un numero minore di nodi rispetto a reti che utilizzano un unico strato nascosto. Questo è un tema attivo di ricerca e i risultati fino ad oggi disponibili indicano che le deep neural networks sono esponenzialmente più espressive delle reti che utilizzano un numero basso di strati nascosti, anche se molti dettagli sono ancora da investigare.

Occupiamoci ora del problema di addestrare una rete neuronale feedforward con funzione di attivazione sigmoidale e $n = 1$ (un singolo nodo di output). Le reti feedforward vengono tipicamente addestrate usando l'algoritmo di discesa del gradiente stocastico,

$$w_{i,j} \leftarrow w_{i,j} - \eta_t \frac{\partial \ell_{Z_t}(W)}{\partial w_{i,j}}$$

dove Z_t è l'indice di un esempio estratto a caso dal training set e $\ell_t(W) = \ell\big(f_{G,W,\sigma}(\boldsymbol{x}_t), y_t\big)$ è una funzione di perdita tale che $\ell(\cdot, y)$ è convessa (per esempio, la funzione di perdita quadratica o la funzione di perdita logistica). Spesso, per accelerare la convergenza, viene anche usata la versione *mini-batched* del gradiente stocastico,

$$w_{i,j} \leftarrow w_{i,j} - \eta_t \frac{1}{|S_t|} \sum_{s \in S_t} \frac{\partial \ell_s(W)}{\partial w_{i,j}}$$

dove S_t è un sottoinsieme di cardinalità fissata estratto a caso dal training set.

È importante osservare che $\ell_t(W)$ è una funzione non convessa dei pesi W anche quando $\ell(\cdot, y)$ è convessa per ogni y. Questo ha due conseguenze importanti:

1. Il problema di determinare i pesi W che minimizzano (anche in modo approssimato) l'errore sul training set è considerato computazionalmente intrattabile.
2. Il metodo della discesa del gradiente converge tipicamente a un minimo locale del training error.

In pratica, si osserva che i minimi locali del training error corrispondono in generale a funzioni con basso rischio. Questo potrebbe significare che molti dei minimi locali sono vicini al minimo globale (che in assenza di overfitting corrisponde al modello di rischio minimo). La ragione precisa delle buone prestazioni del gradiente stocastico sulle reti neuronali resta però un problema di ricerca aperto.

L'applicazione della discesa del gradiente per l'addestramento di una rete neuronale feedforward prende il nome di algoritmo di retropropagazione dell'errore. Questo algoritmo utilizza la regola della derivazione di funzioni composte,

$$\frac{df(g(x))}{dx} = \frac{df(g)}{dg}\frac{dg(x)}{dx} \ . \tag{2}$$

Per comodità, supponiamo che 0 sia l'indice del nodo di output. Quindi $f_{G,W,\sigma}(\boldsymbol{x}_t) = v_0 = \sigma(s_0)$, dove $s_0 = \boldsymbol{w}(0)^\top \boldsymbol{v}(0)$. Per un qualunque i nel primo strato nascosto, ovvero tale che $(i,0) \in E$, applicando due volte la regola (2) abbiamo

$$\frac{\partial \ell_t(W)}{\partial w_{i,0}} = \frac{\partial \ell(v_0, y_t)}{\partial w_{i,0}} = \frac{\partial \ell(v_0, y_t)}{\partial v_0}\frac{\partial v_0}{\partial s_0}\frac{\partial s_0}{\partial w_{i,0}} = \ell'(v_0)\frac{d\sigma(s_0)}{ds_0}\frac{\partial s_0}{\partial w_{i,0}} \ .$$

La derivata $\ell'(v_0) = \frac{\partial \ell(v_0, y_t)}{\partial v_0}$ è calcolabile direttamente a partire dalla definizione della funzione di perdita. Per esempio,

$$\ell(x,y) = (x-y)^2 \qquad \text{implica} \qquad \frac{\partial \ell(x, y_t)}{\partial x} = 2(x-y) \ .$$

In modo simile, la derivata $\frac{d\sigma(s_0)}{ds_0} = \sigma'(s_0)$ è calcolabile direttamente a partire dalla funzione di attivazione. Per esempio,

$$\sigma(z) = \frac{1-e^{-z}}{1+e^{-z}} \qquad \text{implica} \qquad \sigma'(z) = \frac{(1-\sigma(z))^2}{2} \ .$$

Infine, ricordando che $s_0 = \boldsymbol{w}(0)^\top \boldsymbol{v}(0)$, abbiamo

$$\frac{\partial s_0}{\partial w_{i,0}} = v_i = \sigma\big(\boldsymbol{w}(i)^\top \boldsymbol{v}(i)\big) \ .$$

Usando $s_i = \boldsymbol{w}(i)^\top \boldsymbol{v}(i)$ possiamo quindi scrivere

$$\frac{\partial \ell_t(W)}{\partial w_{i,0}} = \ell'(v_0)\,\sigma'(s_0)\,v_i \ .$$

Calcolamo ora $\frac{\partial \ell_t(W)}{\partial w_{i,j}}$ per i nodi i nel secondo strato nascosto. Si noti che

$$v_0 = \sigma\big(\boldsymbol{w}(0)^\top \boldsymbol{v}(0)\big) = \sigma\left(\sum_{j\,:\,(j,0)\in E} w_{j,0}\,\sigma\big(\boldsymbol{w}(j)^\top \boldsymbol{v}(j)\big)\right) = \sigma\left(\sum_{j\,:\,(j,0)\in E} w_{j,0}\,\sigma\left(\sum_{i\,:\,(i,j)\in E} w_{i,j} v_i\right)\right) \ .$$

Quindi, per ogni nodo i nel secondo strato nascosto,

$$\frac{\partial \ell_t(W)}{\partial w_{i,j}} = \underbrace{\frac{\partial \ell(v_0, y_t)}{\partial s_0}}_{\ell'(v_0)\sigma'(s_0)} \underbrace{\frac{\partial s_0}{\partial v_j}}_{w_{j,0}} \underbrace{\frac{\partial v_j}{\partial s_j}}_{\sigma'(s_j)} \underbrace{\frac{\partial s_j}{\partial w_{i,j}}}_{v_i} = \ell'(v_0)\,\sigma'(s_0) w_{j,0}\,\sigma'(s_j) v_i \ .$$

Possiamo riscrivere quanto fatto in modo più compatto sfruttando il fatto che per ogni nodo i,

$$\frac{\partial \ell_t(W)}{\partial s_i} = \sum_{j\,:\,(i,j)\in E} \frac{\partial \ell_t(W)}{\partial s_j} \frac{\partial s_j}{\partial v_i} \frac{\partial v_i}{\partial s_i} = \sum_{j\,:\,(i,j)\in E} \frac{\partial \ell_t(W)}{\partial s_j} w_{i,j}\,\sigma'(s_i) \ .$$

Quindi, introducendo la definizione ricorsiva

$$\delta_i = \frac{\partial \ell_t(W)}{\partial s_i} = \begin{cases} \ell'(v_0)\,\sigma'(s_0) & \text{se } i = 0 \\ \sigma'(s_i) \displaystyle\sum_{j\,:\,(i,j)\in E} \delta_j\, w_{i,j} & \text{altrimenti} \end{cases}$$

possiamo infine scrivere la derivata parziale per un qualunque $(i,j) \in E$ come

$$\frac{\partial \ell_t(W)}{\partial w_{i,j}} = \frac{\partial \ell_t(W)}{\partial s_j} \frac{\partial s_j}{\partial w_{i,j}} = \delta_j\, v_i \ .$$

Addestrare una deep network usando l'algoritmo di retropropagazione dell'errore non è immediato. Per facilitare una convergenza rapida ad un buon minimo locale si utilizzano un insieme di euristiche (per esempio, inizializzazione casuale dei pesi in W). Il corretto utilizzo di queste euristiche non è ancora compreso a fondo e deve quandi essere appreso dall'utente attraverso l'esperienza.