(F94) Metodi statistici per l'apprendimento

Reti neuronali e deep learning

Docente: *Nicolò Cesa-Bianchi* versione 6 giugno 2018

Le reti neuronali sono una famiglia di predittori che, nella loro forma più semplice (detta rete neuronale *feedforward*), sono ottenuti 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*).

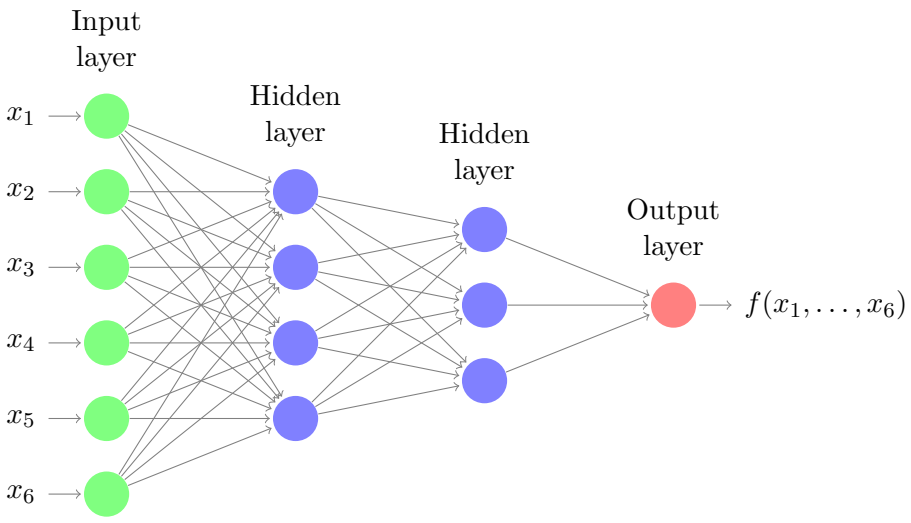

Figura 1: Un esempio di rete neuronale feedforward con output singolo e due strati nascosti in cui le connessioni sono complete fra ogni coppia di strati adiacenti (per esempio fra i due strati nascosti) e assenti fra ogni coppia di strati non adiacenti (per esempio fra strato di input e strato di output).

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 $\operatorname{sgn}\big(f_{G,W,\sigma}(\boldsymbol{x})\big)$. 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 $\boldsymbol{x} \in \{-1,+1\}^d$ 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, $g_i(\boldsymbol{x}) = \operatorname{sgn}\big(\boldsymbol{x}^\top \boldsymbol{x}_i - d + 1\big) = 1$ 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 $\operatorname{sgn}\big(\boldsymbol{w}(i)^\top \widetilde{\boldsymbol{x}}\big) = g_i(\boldsymbol{x})$. 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 un singolo strato nascosto di dimensione almeno pari al numero massimo di istanze positive per le funzioni f da apprendere. Questo numero è tipicamente esponenenziale nella dimensione d. Il risultato seguente (che non dimostreremo) stabilisce che se vogliamo una rete in grado di apprendere tutte le funzioni da $\{-1,+1\}^d$ a $\{-1,+1\}$, allora tale rete deve avere un numero complessivo di nodi 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 $|V| = \Omega(2^{d/3})$. Un risultato analogo vale quando* sgn *è rimpiazzata dalla funzione sigmoidale (1).*

Le reti feedforward con un singolo strato nascosto non solo implementano qualsiasi funzione booleana, ma sono anche in grado di approssimare qualsiasi funzione $f : [-1,+1]^d \to [-1,+1]$ che sia 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 $f : [-1,+1]^d \to [-1,+1]$ Lipschitziana esistono G (con un singolo strato nascosto) e W tali che $\big|f_{G,W,\sigma}(\boldsymbol{x}) - f(\boldsymbol{x})\big| \leq \varepsilon$ per ogni $\boldsymbol{x} \in [-1,+1]^d$. Come nel caso delle funzioni booleane, il numero di nodi nello strato nascosto necessario per rappresentare la funzione f con un grado di approssimazione fissato $\varepsilon > 0$ può essere esponenziale in d.

Abbiamo quindi visto che un singolo strato nascosto è sufficiente per implementare qualsiasi funzione booleana o approssimare qualsiasi funzione Lipschitziana, a patto di essere disposti a utilizzare un numero esponenziale di nodi. Il Teorema 2 ci dice che se vogliamo poter imparare un numero elevato di funzioni con una rete G, allora G deve avere un numero esponenziale di nodi comunque esso venga scelto. L'esperienza sembra però indicare che una rete con un grande numero di strati nascosti ciascuno contenente relativamente pochi nodi (*deep neural network*) è in grado di rappresentare più funzioni rispetto ad una rete con pochi strati nascosti ciascuno contenente tanti nodi. In altre parole, aggiungere a G uno strato nascosto con pochi nodi aumenta $\mathcal{F}_{G,\sigma}$ tanto quanto aumenterebbe aggiungendo un gran numero di nodi ad uno strato esistente. I dettagli precisi di questo fenomeno sono ancora un tema attivo di ricerca.

In una rete neuronale profonda, il grafo G e la funzione di attivazione σ dovrebbero essere scelti in modo tale che $\mathcal{F}_{G,\sigma}$ contenga funzioni che siano utili rispetto alla tipologia di dati che si vogliono apprendere (immagini, testi, eccetera). Per esempio, le reti neuronali convolutive sono una famiglia di reti progettate specificatamente per risolvere problemi di classificazioni di immagini rappresentate come matrici di pixel. In generale, comunque, la questione di quale coppia G, σ sia la più adeguata a risolvere un dato problema di apprendimento è anch'esso un tema attivo di ricerca.

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.

La spiegazione del perché il gradiente stocastico applicato alle reti neuronali possa produrre predittori a basso rischio —nonostante i minimi locali— resta 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

$$\frac{\partial \ell_t(W)}{\partial w_{i,0}} = \frac{\partial \ell(v_0, y_t)}{\partial w_{i,0}} = \frac{d\ell(v_0, y_t)}{dv_0}\frac{dv_0}{ds_0}\frac{\partial s_0}{\partial w_{i,0}} = \ell'(v_0)\frac{d\sigma(s_0)}{ds_0}\frac{\partial s_0}{\partial w_{i,0}} \ .$$

La derivata $\ell'(v_0) = \frac{d\ell(v_0, y_t)}{dv_0}$ 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

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

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

$$\frac{\partial \ell_t(W)}{\partial w_{i,j}} = \underbrace{\frac{d\ell(v_0,y_t)}{ds_0}}_{\ell'(v_0)\sigma'(s_0)} \underbrace{\frac{\partial s_0}{\partial v_j}}_{w_{j,0}} \underbrace{\frac{dv_j}{ds_j}}_{\sigma'(s_j)} \underbrace{\frac{\partial s_j}{\partial w_{i,j}}}_{v_i} = \ell'(v_0)\,\sigma'(s_0) w_{j,0}\,\sigma'(s_j) v_i \ .$$

A partire dai nodi i del terzo strato nascosto, il calcolo di $\frac{\partial \ell_t(W)}{\partial s_i} = \frac{\partial \ell(v_0,y_t)}{\partial s_i}$ è un po' più complesso in quanto v_0 dipende da s_i attraverso tutti i nodi del secondo strato nascosto che sono connessi a i. Ovvero, $v_0 = g\big(s_{j_1},\ldots,s_{j_r}\big)$, dove g è la funzione che calcola l'output v_0 a partire dai valori s_{j_k} dei nodi $j_1,\ldots,j_r$ del secondo strato nascosto connessi a i e $s_{j_k} = h_k(s_i)$ dove h_k è la funzione che descrive come s_{j_k} dipende da s_i. La regola per la derivazione delle funzioni multidimensionali composte ci dice che se $v_0 = g\big(h_1(s_i),\ldots,h_r(s_i)\big)$ allora

$$\frac{dg}{ds_i} = \sum_{j=1}^{r} \frac{\partial g}{\partial h_j}\frac{dh_j}{ds_i}$$

Nel nostro caso abbiamo quindi

$$\frac{\partial \ell(v_0,y_t)}{\partial s_i} = \sum_{j\,:\,(i,j)\in E} \frac{\partial \ell(v_0,y_t)}{\partial s_j}\frac{\partial s_j}{\partial s_i} = \sum_{j\,:\,(i,j)\in E} \frac{\partial \ell(v_0,y_t)}{\partial s_j}\frac{\partial s_j}{\partial v_i}\frac{dv_i}{s_i} = \sum_{j\,:\,(i,j)\in E} \frac{\partial \ell(v_0,y_t)}{\partial s_j} w_{i,j}\,\sigma'(s_i) \ .$$

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.