"Per quello che abbiamo visto fino ad ora, questo argomento sembrerà marginale" cit.

con queste si riusciranno a ridurre i tempi di addestramento — oppure — riuscire ad addestrare la rete

a parità di tempo speso per l'addestramento si riesce ad elaborare più campioni del training set

4 DNN e Approssimazione di Funzioni

Nonostante il teorema di approssimazione universale garantisca che un MLP con uno strato nascosto sia sufficiente per approssimare bene funzioni continue, come verrà discusso nel seguito, è spesso utile avere a disposizione MLP con molti strati nascosti. Convenzionalmente, un MLP con più di due strati nascosti viene chiamato **rete neurale profonda** (**Deep Neural Network**, **DNN**). Normalmente, il numero di strati nascosti delle DNN usate nella pratica supera abbondantemente la decina.

Una DNN non si differenzia in modo sostanziale da un MLP con uno strato nascosto e l'unico problema da affrontare è come estendere l'algoritmo di backpropagation al caso di più di uno strato nascosto. Fortunatamente, l'algoritmo di backpropagation può essere facilmente esteso a più di uno strato nascosto perché il calcolo del relativo gradiente può essere facilmente generalizzato. Non verrà affrontato il calcolo del gradiente nel dettaglio ma, sfruttando la notazione vista per gli MLP con uno strato nascosto, non è difficile arrivare a dimostrare i risultati che qui verranno solo presentati.

Si consideri un MLP con $h \in \mathbb{N}_+$ strati nascosti ognuno dei quali è caratterizzato da un numero specifico di neuroni. Si ipotizzi inoltre che il MLP abbia il massimo numero di archi compatibile con la sua struttura a strati e che tutti i neuroni utilizzino la stessa funzione di attivazione $g : \mathbb{R} \to \mathbb{R}$. Se $(m_i)_{i=1}^h$ è il vettore di $\mathbb{N}_+^h$ che raccoglie il numero di neuroni di ogni strato nascosto, comprensivi di neuroni fissati a -1 per gestire i pesi di bias, allora la rete è completamente definita a meno del numero di valori di input e di output. Una rete di questo tipo viene normalmente chiamata $(n-1, m_1-1, m_2-1, \ldots, m_h-1, m_{h+1})-$MLP, dove $n \in \mathbb{N}_+$ è il numero di neuroni nello strato di input e $m_{h+1} \in \mathbb{N}_+$ è il numero di neuroni nello strato di output. Si noti che, per semplificare la trattazione, normalmente si definiscono $m_0 = n$ e $m_{h+1} = 1$. In più, una volta che è stata adottata questa notazione, si parla di $k-$esimo strato del MLP, con $0 \leq k \leq h+1$, senza specificare se lo strato è di input, di output o nascosto. In Figura 10 viene mostrata una DNN del tipo $(3, 4, 2, 4, 1)-$MLP e, quindi, caratterizzata da tre strati nascosti.

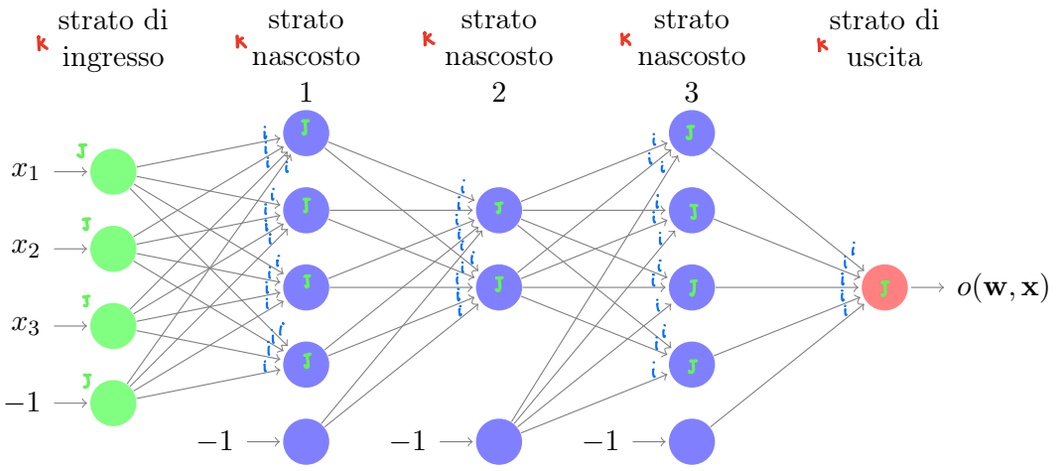

Figura 10: Esempio di DNN del tipo $(3, 4, 2, 4, 1)-$MLP → con questa dicitura vanno considerati il num. max di archi tra i vari strati

Si consideri un $(n-1, m_1-1, m_2-1, \ldots, m_h-1, 1)-$MLP caratterizzato da una funzione di attivazione $g : \mathbb{R} \to \mathbb{R}$ utilizzata in tutte le unità di McCulloch-Pitts. L'obiettivo è usare il MLP come un approssimatore di una funzione $f : \mathbb{R}^n \to \mathbb{R}$ di cui si conoscano i valori per alcuni vettori. Come nel caso del MLP con uno strato nascosto, alcuni dei vettori in cui la funzione è nota vengono raccolti in un training set, mentre i rimanenti vengono raccolti

Nel momento in cui andiamo a guardare il teorema di approssimazione universale, ci rendiamo conto che ci basta una rete con uno strato nascosto per poter approssimare una f.ne continua; esiste una variante del teorema che ci permette di dire che, se la f.ne non è continua sul dominio compatto, allora ci bastano 2 strati nascosti; se con 2 strati nascosti riusciamo ad approssimare bene una qualsiasi f.ne, anche non continua sul dominio, fine...

"introduciamo cose superflue che forse però ci potrebbero servire" cit.

1. Preattivazione: $y = g(x \cdot w)$ la preattivazione è lo scalare risultato da $w \cdot x$ prima di aver applicato la f.ne di attivazione (lui lo segna spesso con a.)
2. o_k è l'output dello strato k, anziché dire y_k
3. Numerari gli strati è comodo strato 0 = input, strato 1 = primo strato nascosto, ..., strato $h+1$ = livello uscita

in un test set. Come di consueto, per semplificare la notazione, senza comunque perdere di generalità, si ipotizza che tutti i vettori del training set, del test set e i vettori posti in ingresso agli strati nascosti abbiano l'ultima componente fissata a -1.

In queste ipotesi è possibile raggruppare i vettori dei pesi della rete in un $3-$**tensore** $\mathbf{W}$ le cui componenti valgono

$$w_{k,j,i} \in \mathbb{R} \tag{49}$$

dove

essendo i pesi entranti nello strato, non entrano in $k=0$

1. L'indice $1 \leq k \leq h+1$ permette di fare riferimento ai pesi in ingresso al $k-$esimo strato; = quale strato stiamo guardando
2. L'indice $1 \leq j \leq m_k - 1$ permette di fare riferimento agli $m_k - 1$ vettori dei pesi in ingresso al $k-$esimo strato; e = quale neurone dello strato
3. L'indice $1 \leq i \leq m_{k-1}$ permette di fare riferimento agli m_{k-1} pesi che formano le componenti di uno degli $m_k - 1$ vettori dei pesi in ingresso al k-esimo strato. = peso che stiamo considerando di tutti pesi che ci sono nel vettore che collega lo strato precedente col neurone che stiamo considerando

Utilizzando il $3-$tensore $\mathbf{W}$ ed introducendo il vettore dei pesi del MLP $\mathbf{w}$ come fatto nel caso dei MLP con uno strato nascosto si possono esprimere le parti che concorrono alla regola di aggiornamento dei pesi che minimizza l'errore quadratico

$$e(\mathbf{w}, \mathbf{x}) = \frac{1}{2}(f(\mathbf{x}) - o(\mathbf{w}, \mathbf{x}))^2 \tag{50}$$

compiuto nell'approssimare il valore della funzione f per uno dei vettori del training set $\mathbf{x}$ con il valore $o(\mathbf{w}, \mathbf{x})$. Si noti che la dipendenza di o e di altre quantità da $\mathbf{w}$ e $\mathbf{x}$ non verrà più esplicitata per semplificare ulteriormente la notazione adottata.

Per esprimere in modo succinto la regola di aggiornamento dei pesi per l'addestramento supervisionato di una DNN, si introducono i vettori $\mathbf{o}_k \in \mathbb{R}^{m_k}$, con $0 \leq k \leq h+1$, che raccolgono i valori delle uscite degli m_k neuroni del $k-$esimo strato. Questi valori delle uscite permettono di definire la **preattivazione** del neurone $1 \leq j \leq m_k - 1$ dello strato $1 \leq k \leq h+1$ come

parte lineare che entrerà nella f.ne g (PREATTIVAZIONE) — vettore dei pesi entranti nel nodo j — uscita del nodo precedente

$$a_{k,j} = \mathbf{w}_{k,j} \cdot \mathbf{o}_{k-1} \tag{51}$$

Si noti che questo valore non è altro che l'argomento della funzione di attivazione del $j-$esimo neurone del $k-$esimo strato e, quindi, è un valore che viene comunque calcolato quando si calcola l'uscita della rete.

Fissato un coefficiente di apprendimento $\alpha \in \mathbb{R}_+$, è possibile esprimere la regola di aggiornamento dei pesi per un fissato campione del training set $(\mathbf{x}, f(\mathbf{x}))$ come segue

simile a quello che abbiamo visto fino ad ora

$$w_{k,j,i} \leftarrow w_{k,j,i} + \alpha\, \delta_{k,j}\, o_{k-1,i} \tag{52}$$

dipende dal neurone j dello strato k

dove il termine $\delta_{k,j}$ viene definito in modo induttivo partendo da $\delta_{h+1,1}$ come segue

questo non lo si ottiene con fatica

* strato di uscita — unico neurone presente — preattivazione del primo e ultimo strato — $g(v_k \cdot y)$

$$\delta_{h+1,1} = g'(\mathbf{w}_{h+1,1} \cdot \mathbf{o}_h)(f(\mathbf{x}) - o_{h+1,1}) = g'(a_{h+1,1})(f(\mathbf{x}) - o_{h+1,1}) \tag{53}$$

e

**

$$\delta_{k,j} = g'(\mathbf{w}_{k,j} \cdot \mathbf{o}_{k-1}) \sum_{s=1}^{m_{k+1}} w_{k+1,s,j}\, \delta_{k+1,s} = g'(a_{k,j}) \sum_{s=1}^{m_{k+1}} w_{k+1,s,j}\, \delta_{k+1,s} \tag{54}$$

Quindi, l'aggiornamento dei pesi di una DNN durante l'addestramento supervisionato una volta che sia stato fissato un campione $(\mathbf{x}, f(\mathbf{x}))$ può essere svolto utilizzando una fase di forward propagation seguita da una fase di backward propagation in modo simile a quanto

Struttura simile al MLP con un singolo strato nascosto

* Prima formula di aggiornamento: $v_j \leftarrow v_j + \alpha (f(x) - g(v \cdot y))\, g'(v_j \cdot y)\, y_i$ — coeff. di apprendimento — output di rete — derivata dell'output di rete

** Si calcola come la g' applicata alla preattivazione dello strato k, neurone j, poi viene fuori una combinazione lineare che qua non c'è perché qui c'è un solo neurone d'uscita, e quindi rimane un termine solo; se abbiamo più neuroni d'uscita, perché ogni strato a più neuroni nello strato successivo dobbiamo cercare di capire cosa succede nello strato successivo, e se svolgiamo il calcolo, cosa viene fuori? Una somma che va da 1 ai n° di neuroni dello strato successivo per il peso che peschiamo dallo strato successivo al variare di s per j fissato per δ dello strato succ. per s fissato.

fatto per un MLP con un solo strato nascosto. In questo caso, però, conviene memorizzare i valori calcolati nella fase di forward propagation, valori di output e preattivazioni, per usarli poi nella fase di backward propagation.

In particolare, l'algoritmo di addestramento supervisionato può essere riassunto secondo le cinque parti principali descritte nel seguito:

1. Si propaga il vettore di input $\mathbf{x}$ attraverso la rete andando a calcolare i vettori $\mathbf{o}_k$ in uscita dagli strati intermedi e dallo strato di output e, quindi, con $1 \leq k \leq h+1$;
2. Durante la propagazione del vettore di input $\mathbf{x}$ attraverso la rete, si calcolano le preattivazioni;
3. Si calcola il valore $\delta_{h,1}$ usando l'errore effettivamente compiuto $f(\mathbf{x}) - o_{h+1,1}$ e la preattivazione $a_{h+1,1}$ dell'unico neurone di uscita;
4. Si calcola il valore $\delta_{k,j}$ sfruttando la preattivazione del $k-$esimo strato e i valori $\delta_{k+1,s}$ dello strato successivo, con $1 \leq s \leq m_{k+1}$; e
5. Per il $k-$esimo strato, partendo dall'uscita e tornando verso l'ingresso, si aggiornano i pesi applicando la regola di aggiornamento dei pesi.

Si noti che l'algoritmo di backpropagation visto per gli MLP con un solo strato nascosto è un caso particolare dell'algoritmo presentato e, quindi, si riserva il termine di algoritmo di backpropagation proprio a questa versione più generale. Infatti, anche questa versione più generale è strutturata in una fase di forward propagation, descritta dai punti 1 e 2 precedenti, e da una fase di backward propagation descritta dai punti 3, 4 e 5 precedenti. In più, si noti che le formule viste nel caso di un solo strato nascosto non sono altro che casi particolari delle formule presentate in questo caso più generale.

5 CNN e Approssimazione di Funzioni

Uno dei problemi principali che è necessario affrontare per utilizzare le reti neurali per risolvere problemi particolarmente complessi è quello di ridurre i tempi di addestramento. Infatti, la riduzione dei tempi di addestramento è l'unico modo perseguibile per aumentare il numero di campioni utilizzati durante l'addestramento e quindi potenzialmente aumentare le capacità delle reti di affrontare problemi complessi.

Da questo punto di vista, le DNN sembrano particolarmente svantaggiate perché ogni aumento del numero degli strati implica un corrispettivo aumento del numero di pesi da addestrare e, quindi, un corrispettivo aumento dei tempi di addestramento. In più, il teorema di approssimazione universale dice che un singolo strato nascosto è sufficiente per approssimare bene quanto si vuole una data funzione continua. Quindi, l'utilizzo di DNN sembra ulteriormente penalizzato perché, pur aumentando i tempi di addestramento, non sono in grado di fornire migliori capacità di approssimazione.

Per rendere le DNN utili nella risoluzione di problemi complessi, si preferisce quindi utilizzarle in una configurazione che permetta di ridurre il numero dei pesi da addestrare consentendo comunque di approssimare bene quanto si vuole una data funzione. In particolare, è possibile decidere di fissare il valore di alcuni pesi a zero utilizzando il numero di strati nascosti superiori a due per rendere comunque in grado la rete di approssimare bene la funzione su cui si sta lavorando. Quindi, utilizzando come gradi di libertà il numero degli strati nascosti e il numero di neuroni per ogni strato, è possibile fissare molti pesi a zero in modo da escluderli dall'addestramento con l'effetto quindi di ridurre i tempi di addestramento in modo significativo.

Un approccio tipico per decidere quali pesi fissare a zero è notare che spesso i problemi affrontati mediante le reti neurali sono contraddistinti da **feature** (o **caratteristiche**) locali. Quindi, il valore di output dipende principalmente da quello che succede in un **intorno** dei singoli neuroni di input e dipende poco da quello che succede lontano ad ogni singolo neurone di input. Se è possibile ipotizzare che l'output dipenda unicamente da feature locali dell'input, allora è possibile fissare a zero i pesi che collegano l'ingresso di un neurone con i neuroni ad esso lontani. Se una DNN viene strutturata mediante l'estrazione di feature locali dell'input, allora ogni livello della rete ha il compito di aggregare le feature estratte dal livello precedente fino a produrre l'output. La Figura 11 mostra un esempio di DNN in cui ogni neurone di uno strato è collegato mediante pesi non nulli a solo pochi neuroni dello strato successivo. In questo caso, il numero di pesi da trovare mediante addestramento si è ridotto a 37.

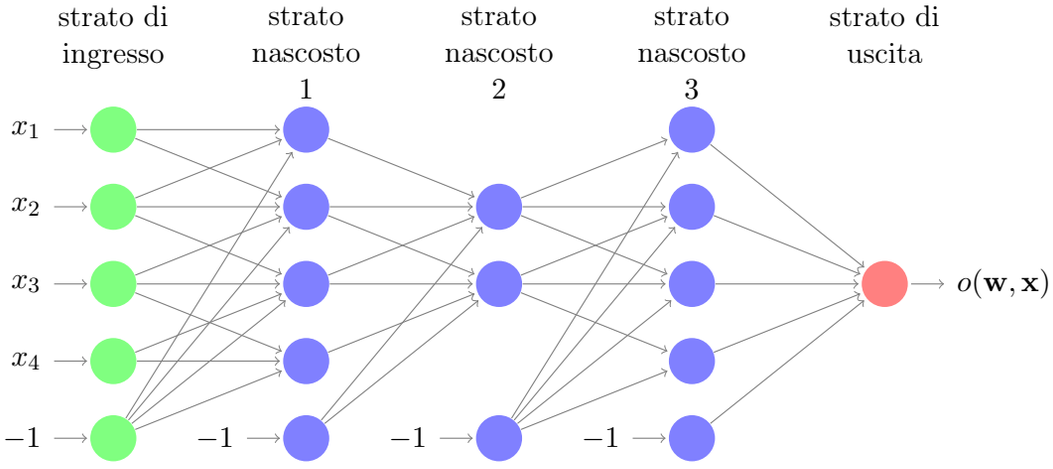

Figura 11: Esempio di DNN che lavora su feature locali

Questo approccio alla progettazione di DNN può essere spinto all'estremo ipotizzando che le elaborazioni svolte in ogni intorno siano sempre le stesse. Quindi, ipotizzando che i pesi che associano i neuroni di un intorno per produrre l'input dello strato successivo siano sempre gli stessi, indipendentemente dall'intorno considerato. Se una rete può essere strutturata in questo modo, allora il tempo di addestramento viene ulteriormente abbattuto perché l'addestramento ha come unico scopo quello di trovare gli unici pesi applicati a tutti gli intorni. Una rete neurale di questo tipo viene detta **convolutiva** (o **convoluzionale** da **Convolutional Neural Network**, **CNN**) e l'unico vettore dei pesi che viene cercato per ogni livello viene detto **nucleo** (o **kernel**) di **convoluzione**. Se si ipotizza che la rete in Figura 11 sia una CNN, allora ogni strato è caratterizzato unicamente da un nucleo di convoluzione formato da pochi pesi, che sono i presi che collegano uno strato con il successivo più il peso di bias. Naturalmente, strati diversi possono avere nuclei di convoluzione diversi e non è richiesto che tutti gli strati lavorino in modo convoluzionale. Infatti, normalmente una CNN ha almeno uno strato in cui il numero di archi entranti è massimo, come succede nello strato di uscita della rete in Figura 11. Facendo riferimento alla DNN mostrata in Figura 11, se la rete è in realtà una CNN, il numero di pesi da trovare mediante addestramento si è ridotto a 16.

Una rete convoluzionale chiama i pesi che si ripetono su tutta la rete maschera di convoluzione (n. di pesi nell'intorno + il peso di bias), che è molto più piccola di tutta la rete, e quindi molto più veloce da addestrare.