1 Introduzione

Nel capitolo precedente si é dimostrato come gli algoritmi numerici di ottimizzazione possono essere d'aiuto nella progettazione e quali sono i campi d'applicazione per i vari metodi, al variare del problema proposto.

Si é anche accennato al fatto che non é sempre agevole utilizzare questi approcci nell'ottimizzazione, soprattutto per il fatto che sono richieste numerose simulazioni del processo in esame, situazione che molte volte non é conveniente.

É quindi necessario ridurre il tempo di calcolo richiesto: due sono essenzialmente i settori dove é possibile intervenire:

- miglioramentento degli algoritmi;
- riduzione del numero delle simulazioni con l'introduzione di funzioni approssimanti.

Il primo punto é stato affrontato nella prima parte della tesi, dove sono stati studiati i vari algoritmi di ottimizzazione e sono stati introdotti nuovi operatori nell'algoritmo genetico, al fine di renderlo più veloce in convergenza.

Per intervenire sul secondo punto occorre introdurre le superfici di risposta, al fine di estrapolare il valore della funzione obiettivo da valori noti, riducendo cosí in maniera considerevole i tempi di calcolo totali.

La scelta del modello di superficie di risposta da utilizzare é caduta sulla *rete neurale* [**?**] che, come mostrato in [**?**] può essere considerato il miglior approccio per approssimare funzioni generiche.

2 La rete neurale

La rete neurale é un modello matematico che si basa sulla simulazione, estremamente semplificata, del comportamento del cervello.

Come tutti sappiamo il cervello é formato da un numero elevatissimo di cellule (neuroni), tutte collegate tra loro mediante terminazioni nervose (sinapsi). Le cellule interagiscono tra loro mediante impulsi eletrici che vengono rielaborati all'interno di esse e trasmessi ai neuroni vicini.

La rete neurale prende spunto da questo comportamento per riuscire a creare un modello della funzione che si vuole interpolare: dati alcuni modelli esatti, intesi come valori delle variabili e valore corrispondente della funzione da modellizare, la rete neurale li elabora in maniera da riuscire a trovare il valore della funzione per una serie di variabili diverse dalle precedenti. Come si vede l'applicazione più logica della rete neurale é come estrapolatore di funzione.

Per la rete neurale un neurone puó essere rappresentato nella seguente maniera:

Ciascun nodo (neurone) di una rete neurale puó essere visto come una funzione primitiva capace di trasformare i vari input in un ben definito output. Differenti modelli di rete neurale differiranno quindi tra loro per il tipo di funzione primitiva usata, per il tipo di interconnessioni tra nodo e nodo, e per l'algoritmo usato per trovare i pesi delle interconnessioni.

Una tipica rete neurale ha la struttura mostrata nella figura sotto:

Come si vede la rete neurale trova il valore della funzione Φ valutata nel punto (x, y, z). I nodi sono formati dalle funzioni primitive f_1, f_2, f_3, f_4, che si ricombinano per produrre il valore di Φ. La funzione viene chimata **funzione di rete**. Differenti valori dei pesi $\alpha_1, \alpha_2, , \alpha_n$ producono differenti valori per la funzione di rete.

É importante ribadire i tre elementi caratteristici di ogni rete:

- la struttura dei nodi,
- la topologia della rete,
- l'algoritmo di apprendimento usato per trovare i pesi della rete.

3 L'algoritmo backpropagation

Uno dei fattori di maggior importanza per la configurazione di una rete neurale é l'algoritmo di apprendimento; ora verrà mostrato il piú popolare di questi algoritmi: il **backpropagation**. Questo metodo é sicuramente il più diffuso ed é uno dei più importanti sia per la sua semplicità di implementazione, sia per la qualità dei risultati ottenibili.

L'algoritmo bakpropagation trova il minimo della funzione errore nello spazio dei pesi usando il metodo del gradiente; si tratta di ricercare la combinazione di pesi che minimizza la funzione errore; questa sarà considerata la soluzione del problema di apprendimento.

Molto importante é definire la funzione di rete, detta anche funzione d'attivazione. La più usata é sicuramente la funzione sigmoide $s_c : \mathbb{R} \Rightarrow (0, 1)$ definita nella seguente maniera:

$$s_c = \frac{1}{1 + e^{-cx}} \tag{1}$$

Il valore di c può essere scelto arbitrariamente, normalmente tra i valori $c = 1, c = 2, c = 3$.

Molto importante ai fini dell'algoritmo é definire la derivata della sigmoide:

$$\frac{\mathrm{d}}{\mathrm{d}x} s(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = s(x)(1 - s(x)) \tag{2}$$

Si possono usare altri tipi di funzione di attivazione: l'unica condizione a cui devono sottostare é quella di essere differenziabili in ogni punto e di avere sempre la derivata non nulla; questo secondo aspetto é dovuto al fatto che il metodo di ricerca del minimo della funzione errore é il metodo del gradiente e quindi si capisce che avere una funzione attivazione che in qualche punto abbia derivata nulla può causare dei gravi problemi all'algoritmo.

Un grosso problema che si pone nella ricerca del minimo della funzione errore é che questa sarà topologicamente molto complessa con la sicura presenza di minimi relativi; é quindi molto importante riuscire a guidare la ricerca del minimo nella giusta direzione, evitando di cadere in minimi non assoluti.

3.1 Il problema dell'apprendimento

La rete neurale é un insieme di celle (funzioni attivazione) collegate tra loro (pesi) in modo da riuscire a definire una funzione.

Il problema dell'apprendimento consiste nel trovare la migliore combinazione dei valori dei pesi in modo che la funzione di rete Φ riesca ad approssimare la funzione data F nella migliore maniera possibile. La funzione F non verrà data esplicitamente ma implicitamente attraverso una serie di esempi.

Adesso considereremo l'approssimazione di una funzione $F : \mathbb{R}^n \Rightarrow \mathbb{R}^m$; la rete sarà quindi formata da n input (variabili) ed m output (valori della funzione F). Per riuscire a risolvere il problema viene dato un set d'addestramento $\{(x_1, t_1), , (x_p, t_p)\}$, consistente in p vettori ordinati di dimensione n e m, praticamente sono p punti della funzione F conosciuti ai quali viene dato il nome di set d'addestramento.

Quando il modello di input x_i del set d'addestramento é presentato alla rete, questo produrrà un valore della funzione di rete o_i, generalmente diverso dal valore t_i della funzione F. Facendo questo per tutti i p modelli d'allenamento sarà possibile trovare l'errore che dovrà essere minimizzato:

$$E = \frac{1}{2} \sum_{i=1}^{p} \parallel o_i - t_i \parallel^2 \tag{3}$$

Come gia' detto l'algoritmo backpropagation é usato per trovare il minimo della funzione errore (3). Il procedimento é molto semplice: la rete viene inizializzata dando valori random ai pesi, si valuta il gradiente dell'errore, rispetto ai pesi, si correggono i pesi, si itera questo procedimento fino a convergenza.

Come si vede l'elemento più importante é la ricerca dei valori del gradiente della funzione errore rispetto i pesi w ($\nabla E = (\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, , \frac{\partial E}{\partial w_l})$).

3.2 Caso della rete a piani

La rete neurale più utilizzata per l'approssimazione di funzioni é senz'altro quella a piani nascosti. Di seguito verrá mostrata la maniera di agire dell'algoritmo backprobagation per questo tipo di rete neurale.

La topologia della rete sarà la seguente: n input, m output, un piano interno formato da k nodi. Il peso tra l'input in posizione i e il nodo del piano nascosto in posizione j sarà $w_{ij}^{(1)}$, mentre il peso tra il nodo del piano nascosto i e l'output j sarà $w_{ij}^{(2)}$. Esiste un input aggiuntivo di valore 1 per date alla rete la possibilità di aggiungere una costante additiva.

La topologi della rete sarà di conseguenza la seguente:

Ricordando che la sigmoide é definita in (1) l'algoritmo di apprendimento backpropagation sarà costituito da quattro passi successivi:

- valutazione del valore approssimato della funzione,
- backpropagation dal piano finale,

- backpropagation dal piano interno,
- aggiornamento dei valori dei pesi.,

3.3 Calcolo del valore approssimato della funzione

Il vettore n-dimensionale degli input $\mathbf{o} = (o_1, , o_n)$ viene trasformato nel vettore $\mathbf{oi} = (o_1, , o_n, 1)$.

L'eccitazione $net_j^{(1)}$ del j-esimo nodo del piano interno é uguale a:

$$net_j^{(1)} = \sum_{i=1}^{n+1} w_{ij}^{(1)} oi_i \tag{4}$$

La funzione attivazione é la sigmoide e quindi il valore $o_j^{(1)}$ del nodo j-esimo del piano interno é:

$$o_j^{(1)} = s(net_j^{(1)}) \tag{5}$$

Similmente l'attivazione del piano di output $net_j^{(2)}$ é:

$$net_j^{(2)} = \sum_{i=1}^{k+1} w_{ij}^{(2)} o_j^{(1)} \tag{6}$$

e quindi l'output finale, valore della funzione F in posizione j, sarà:

$$o_j^{(2)} = s(net_j^{(2)}) \tag{7}$$

Iterando questo procedimento per tutti gli m output é possibile definire l'errore per l'addestramento del modello i nella seguente maniera:

$$E = \frac{1}{2} \sum_{i=1}^{m} (o_i^{(2)} - t_i)^2 \tag{8}$$

Ovviamente $E > 0$ e quindi bisognerà modificare il valore dei pesi. Per far ciò sono necessari i seguenti passi.

3.4 Backpropagation dal piano finale

Il passo successivo é trovare il valore del gradiente della funzione errore E rispetto i pesi W_2:

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = \frac{\partial E}{\partial o_j^{(2)}} \frac{\partial o_j^{(2)}}{\partial net_j^{(2)}} \frac{\partial net_j^{(2)}}{\partial w_{ij}^{(2)}} \tag{9}$$

dalla formula (4):

$$\frac{\partial net_j^{(2)}}{\partial w_{ij}^{(2)}} = o_i^{(1)} \tag{10}$$

dalla (5):

$$\frac{\partial o_j^{(2)}}{\partial net_j^{(2)}} = o_j^{(2)}(1 - o_j^{(2)}) \tag{11}$$

dalla (6):

$$\frac{\partial E}{\partial o_j^{(2)}} = (o_j^{(2)} - t_j) \tag{12}$$

Alla fine si trova quindi il gradiente:

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = (o_j^{(2)} - t_j) o_j^{(2)}(1 - o_j^{(2)}) o_i^{(1)} \tag{13}$$

Per comodità viene effettuata la seguente trasformazione:

$$\frac{\partial E}{\partial net_j^{(2)}} = (o_j^{(2)} - t_j) o_j^{(2)}(1 - o_j^{(2)}) = \delta_j^{(2)} \tag{14}$$

Allora:

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = o_i^{(1)} \delta_j^{(2)} \tag{15}$$

3.5 Backpropagation dal piano interno

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \frac{\partial E}{\partial o_j^{(1)}} \frac{\partial o_j^{(1)}}{\partial net_j^{(1)}} \frac{\partial net_j^{(1)}}{\partial w_{ij}^{(1)}} \tag{16}$$

Dalla formula (4):

$$\frac{\partial net_j^{(1)}}{\partial w_{ij}^{(1)}} = oi_i \tag{17}$$

dalla (5):

$$\frac{\partial o_j^{(1)}}{\partial net_j^{(1)}} = o_j^{(1)}(1 - o_j^{(1)}) \tag{18}$$

Per il primo termine le cose sono un pò più complesse, infatti:

$$\frac{\partial E}{\partial o_j^{(1)}} = \sum_{q=1}^{m} \frac{\partial E}{\partial net_q^{(2)}} \frac{\partial net_q^{(2)}}{\partial o_j^{(1)}} \tag{19}$$

Il primo termine della sommatoria ricordando la (14)

$$\frac{\partial E}{\partial net_q^{(2)}} = (o_q^{(2)} - t_q) o_q^{(2)} (1 - o_q^{(2)}) = \delta_q^{(2)} \tag{20}$$

mentre per il secondo termine:

$$\frac{\partial net_q^{(2)}}{\partial o_j^{(1)}} = \frac{\partial \sum_{r=1}^{k+1} w_{rq}^{(2)} o_r^{(1)}}{\partial o_j^{(1)}} = w_{jq}^{(2)} \tag{21}$$

Si ricava quindi il valore del gradiente:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = o_j^{(1)} (1 - o_j^{(1)}) oi_i \sum_{q=1}^{m} w_{jq}^{(2)} \delta_q^{(2)} \tag{22}$$

Per semplicità:

$$\delta_j^{(1)} = o_j^{(1)} (1 - o_j^{(1)}) \sum_{q=1}^{m} w_{jq}^{(2)} \delta_q^{(2)} \tag{23}$$

Quindi:

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = oi_i \delta_j^{(1)} \tag{24}$$

3.6 Aggiornamento dei valori dei pesi

Dalle equazioni (16) (24) si è in grado di trovare il valore del gradiente della funzione errore rispetto i soli parametri liberi della rete: i pesi; adesso bisognerà aggiornarl in modo da trovare il minimo della funzione. La maniera più semplice per effettuare ciò é la seguente:

- $\Delta w_{ij}^{(2)} = \gamma o_i^{(2)} \delta_j^{(2)}$ per $i = 1, , k+1$ e $j = 1, , m$
- $\Delta w_{ij}^{(1)} = \gamma o_i^{(1)} \delta_j^{(2)}$ per $i = 1, , n+1$ e $j = 1, , k$

Questo modo di agire ha però il limite di richiedere troppe iterazioni per andare a convergenza, oltrettutto il buon risultato dell'addestramento dipende molto da come sono stati calcolati i valori iniziali dei pesi per l'inizializzazione della rete.

Per questo motivo sono stati creati molti altri tipi di algoritmi per la variazione dei pesi in dipendenza del valore del gradiente [**?**] [**?**].

Tra questi possiamo ricordare il metodo con l'uso dell'inerzia:

$$\Delta w_k^{(i)} = -\gamma \frac{\partial E}{\partial w_k} + \alpha \Delta w_k^{(i-1)} \tag{25}$$

Cioè la correzione del k-esimo peso alla iterazione i-esima dipende anche dalla correzione all'iterazione precedente; il vantaggio di ciò é evidente: in questo

modo si riesce a far saltare al peso dei micro avallamenti della funzione errore che, se non ci fosse l'inerzia causata dall'iterazione precedente, potrebbero causare dei seri problemi per la ricerca del minimo assoluto.

Il modo più interessante per la variazione dei pesi, sia come velocità di convergenza, che come qualità delle soluzioni ottenute, é il seguente [**?**]:

$$\Delta w_j^{(k)} = -\gamma_j^{(k)} \frac{\partial E^{(k)}}{\partial w_j} + \alpha \Delta w_j^{(k-1)} \tag{26}$$

$$\gamma_j^{(k)} = \begin{cases} min(\gamma_j^{(k-1)} * mol, \gamma_{max}) & \text{se } \frac{\partial E^{(k)}}{\partial w_j} \frac{\partial E^{(k-1)}}{\partial w_j} > 0 \\ \gamma_{min} & \text{altrimenti} \end{cases} \tag{27}$$

dove mol, α, γ_{max}, and γ_{min} sono parametri dipendenti dal problema.

Questo metodo deriva dall'Rprop [**?**]; infatti si nota che il modello adattativo é il medesimo. In questo caso però si osserva che se il gradiente inizia ad oscillare il valore di γ viene inizializzato ponendolo uguale a γ_{min}; in questo modo si perde un pò in velocità di convergenza, ma si guadagna certamente in robustezza.

4 Nuovi tipi di rete

La rete neurale vista precedentemente, ad uno o più piani interni, é la più semplice tipologia di rete neurale esistente: la convergenza verso il minimo assoluto della funzione errore é assicurata, ma questo richiede un gran numero di iterazioni e un numero di nodi molto ampio. Per questo motivo sono state sviluppate delle rete che si diversificano tra loro soprattutto per l'uso di diverse funzioni di attivazione.

La rete neurale che verrà esposta adesso viene chiamata **rete neurale con soglia dinamica** [**?**] [**?**], che ha dimostrato di avere prestazioni migliori delle reti neurali precedenti.

Il layout di queso nuovo tipo di rete neurale é il seguente:

Dal grafico si nota la presenza nei piani interni di due tipologia di funzioni attivazione: quella con soglia statica (QSF) e quella con soglia dinamica (QSF estesa).

$$QSF : f(net_i, \theta_i) = \frac{1}{1 + exp(net_i^2 - \theta_i^2)} \tag{28}$$

$$QSFestesa : f(net_i, \bar{\Theta}_i) = \frac{1}{1 + exp(net_i^2 - (g(\bar{\Theta}_i, \bar{X}))^2)} \tag{29}$$

Dove $net_i = \bar{W}_i \cdot \bar{X} = w_{i,0} + \sum_{j=1}^{n} w_{i,j} x_j$.

I vettori $\bar{W}_i = (w_0, ..., w_n)$ e $\bar{X} = (1, x_1, ..., x_n)$ sono rispettivamente i pesi e il vettore degli input mentre $\bar{\Theta}_i = (\theta_{i,0}, ..., \theta_{i,n})$ e $g(\bar{\Theta}_i, \bar{X})) = \theta_{i,0} + \sum_{j=1}^{n} \theta_{i,j} x_j$ sono chiamate funzioni di soglia.

Come si vede da (29), questa nuova tipologia di funzione di attivazione migliora la QSF (4), infatti ora la soglia non é più dipendente dai valori in

ingresso al nodo, ma dipende da questi; questo vuol dire che in questa maniera si riesce ad avere stati di transizione da un nodo ad un altro molto variabili, situazione che certamente aiuta la rete ad adattarsi in maniera migliore alla funzione che deve approssimare.

L'uso della QSF estesa viene fatta solamente sull'ultimo piano, e non su tutti. É stato usato questo compromesso perché la soglia mobile, più performante nei confronti di quella statica, ha bisogno di un numero di parametri maggiore: infatti c'é la presenza di tutte le connessioni con il piano precedente; questo causa un aumento notevole dei calcoli, infatti bisogna trovare il gradiente della funzione errore anche per le connessioni $\theta_{i,j}$; l'aumento di velocità di convergenza verrebbe quindi quasi totalmente annullato dalla maggiore quantità di mole di calcoli. Si é quindi pensato di mettere la soglia dinamica proprio nell'ultimo piano, quello degli output, in modo da effettuare meno calcoli; notare che la soglia dinamica é stata situata nel piano più importante per il conseguimento di buoni risultati nell'approssiamazione.

5 Calcolo del gradiente mediante le reti neurali

Come già ricordato, uno dei problemi più importanti nell'uso degli algoritmi che usano il gradiente della funzione obiettivo, stà proprio nel calcolo del gradiente, che molte volte può diventare troppo oneroso in termini di tempo se la funzione obiettivo é definita a molte variabili. Per questo motivo é stata implementata una metodologia, basata sulla rete neurale, che può essere d'aiuto nel calcolo del gradiente.

Normalmente il calcolo della derivata parziale avviene mediante differenze finite:

$$\frac{\partial y}{\partial x_i} = \frac{y(x_1, \ldots, x_i + h, \ldots, x_n) - y(x_1, \ldots, x_i - h, \ldots, x_n)}{2h} \tag{30}$$

In questo caso sono necessarie due chiamate al solver per il calcolo approssimato della derivata parziale. Però se la rete neurale é una approssimazione della funzione y si possono calcolare le derivate parziali *analitiche* della stessa:

$$\frac{\partial y}{\partial x_i} = 2net^{(2)} y(y-1) \sum_{j=1}^{k+1} w_j^{(2)} q_{i,j} + 2\theta^{(2)} y(1-y) \sum_{j=1}^{k+1} z_j^{(2)} q_{i,j} \tag{31}$$

dove

$$q_{i,j} = 2net_j^{(1)} w_{i,j}^{(1)} o_j^{(1)} (o_j^{(1)} - 1) \tag{32}$$

Come si vede se si usasse questo approccio durante l'ottimizzazione, si potrebbe avere un grosso risparmio in termini di tempo, non essendo più necessario effettuare le chiamate al solver per il calcolo delle derivate parziali, e quindi del gradiente. Infatti si potrebbe costruire il modello della funzione mediante interpolazione con rete neurale, e in seguito andare a calcolare mediante

la (31) il gradiente; a questo punto si potrebbe effettuare una iterazione dell'algoritmo di ottimizzazione, trovando di conseguenza una nuova soluzione, che andrà ad aumentare il data-base della rete neurale, e quindi l'accuratezza nell'estrapolazione.

L'algoritmo diventa quindi il seguente:

Punto 1) Partenza delle iterazioni da un punto iniziale $\mathbf{X}_1$;
Punto 2) calcolo di n configurazioni attorno a $\mathbf{X}_1$;
Punto 3) creazione di un data-base con le configurazioni calcolate fino ad adesso;
Punto 4) allenamento della rete neurale sul data-base;
Punto 5) calcolo di $\nabla f(\mathbf{X}_i)$ con la formula (31);
Punto 6) effettuazione di una iterazione con l'algoritmo di ottimizzazione;
Punto 7) calcolo della configurazione trovata nel punto 6;
Punto 8) se non si é raggiunto il massimo ritornare al punto 3

5.0.1 Dimostrazione

Ora vedremo come si possa ricavare l'equazione (31):

La funzione attivazione é (29):

$$s(x,y) = \frac{1}{1+e^{x^2-y^2}} \tag{33}$$

allora

$$\frac{\partial s(x,y)}{\partial x} = 2x[s(x,y)-1]s(x,y) \tag{34}$$

$$\frac{\partial s(x,y)}{\partial y} = 2y[1-s(x,y)]s(x,y) \tag{35}$$

Per il calcolo di y :

$$net_i^{(1)} = \sum_{j=1}^{n+1} w_{j,i}^{(1)} x_j \tag{36}$$

$$o_i^{(1)} = s(net_i^{(1)}, \theta_i^{(1)}) \tag{37}$$

$$net^{(2)} = \sum_{j=1}^{k+1} w_j^{(2)} o_j^{(1)} \tag{38}$$

$$\theta^{(2)} = \sum_{j=1}^{k+1} z_j^{(2)} o_j^{(1)} \tag{39}$$

$$y = s(net^{(2)}, \theta^{(2)}) \tag{40}$$

$\frac{\partial y}{\partial x_i}$ può essere calcolata :

$$\frac{\partial y}{\partial x_i} = \frac{\partial[\frac{1}{1+e^{net^{(2)^2} - \theta^{(2)^2}}}]}{\partial x_i} \tag{41}$$

$$\frac{\partial y}{\partial x_i} = y(y-1)[2net^{(2)}\frac{\partial net^{(2)}}{\partial x_i} - 2\theta^{(2)}\frac{\partial \theta^{(2)}}{\partial x_i}] \tag{42}$$

con (34) (35) (40) :

$$\frac{\partial s(net^{(2)}, \theta^{(2)})}{\partial net^{(2)}} = 2net^{(2)}[y(y-1)] \tag{43}$$

$$\frac{\partial s(net^{(2)}, \theta^{(2)})}{\partial \theta^{(2)}} = 2\theta^{(2)}[y(1-y)] \tag{44}$$

con (38) :

$$\frac{\partial net^{(2)}}{\partial x_i} = \sum_{j=1}^{k+1} w_j^{(2)} \frac{\partial o_j^{(1)}}{\partial x_i} \tag{45}$$

$$\frac{\partial o_j^{(1)}}{\partial x_i} = \frac{\partial s(net_j^{(1)}, \theta_j^{(1)})}{\partial net_j^{(1)}} \frac{\partial net_j^{(1)}}{\partial x_i} \tag{46}$$

con (34) :

$$\frac{\partial s(net_j^{(1)}, \theta_j^{(1)})}{\partial net_j^{(1)}} = 2net_j^{(1)}[o_j^{(1)}(o_j^{(1)} - 1)] \tag{47}$$

$$\frac{\partial net_j^{(1)}}{\partial x_i} = w_{i,j}^{(1)} \tag{48}$$

$$\frac{\partial \theta^{(2)}}{\partial x_i} = \sum_{j=1}^{k+1} z_j^{(2)} \frac{\partial o_j^{(1)}}{\partial x_i} \tag{49}$$

e quindi si può scrivere:

$$\frac{\partial y}{\partial x_i} = 2net^{(2)}y(y-1)\sum_{j=1}^{k+1} w_j^{(2)} q_{i,j} + 2\theta^{(2)}y(1-y)\sum_{j=1}^{k+1} z_j^{(2)} q_{i,j} \tag{50}$$

dove

$$q_{i,j} = 2net_j^{(1)} w_{i,j}^{(1)} o_j^{(1)}(o_j^{(1)} - 1) \tag{51}$$

6 Conclusioni

L'uso della rete neurale assieme agli algoritmi di ottimizzazione per la riduzione dell'onerosità di calcolo é una strategia ben conosciuta [**?**] [**?**] [**?**]. In tutti questi casi si utilizza semplicemente il valore della funzione obiettivo estrapolata per mezzo della rete neurale, con il vantaggio di richiedere meno chiamate al solver, per ottenere il vero valore della funzione; in tutti i casi si nota come i risultati siano buoni, pur risparmiando un buon numero di simulazioni.

In [**?**] é stato invece dimostrato come la rete neurale può essere un valido aiuto anche nell'utilizzo di un algoritmo basato sul metodo del gradiente, grazie alla possibilità di calcolare in maniera analitica il gradiente della funzione obiettivo (31). I vantaggi di questo modo d'agire sono molteplici: innanzitutto c'é la possibilità di migliorare una soluzione localmente senza dover effettare una molteplicità di simulazioni esatte; questo modo d'agire é molto interessante in sequenza ad una ottimizzazione basata su un metodo stocastico, che pur essendo molto robusto, soffre in termini di accuratezza. Il secondo punto positivo nell'utilizzo della rete neurale con questa modalità stà sulla creazione di una superficie di risposta locale, cioè che non va ad approssimare la funzione su tutto il campo di esistenza, ma solo nei dintorni della soluzione di partenza, in questo modo l'accuratezza dell'approssimazione aumenta di molto, portando risultati migliori.

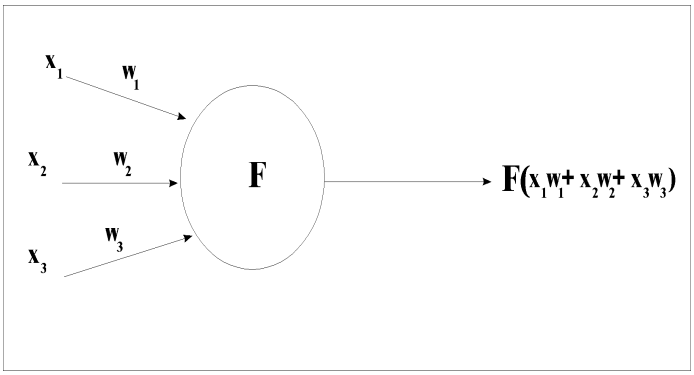

Figura 1: Esempio di neurone di rete neurale.

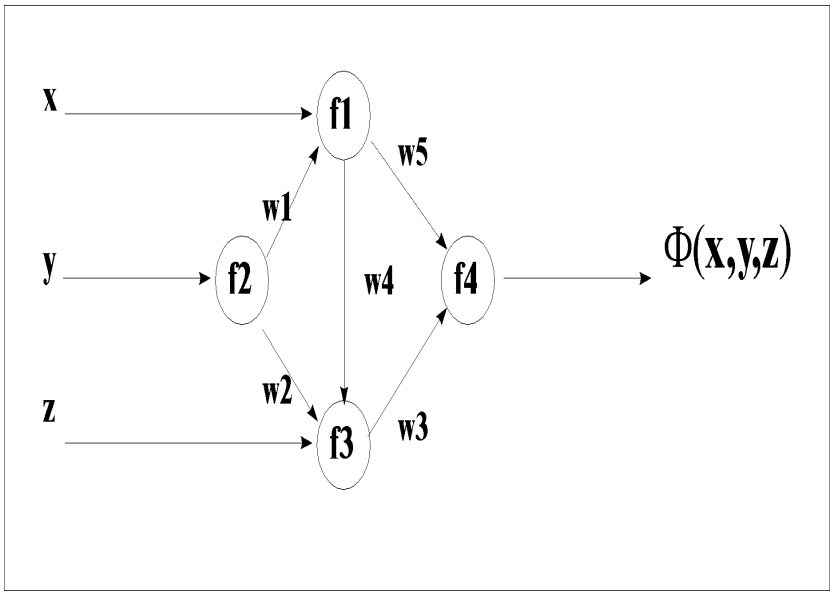

Figura 2: Esempio struttura di una rete neurale.

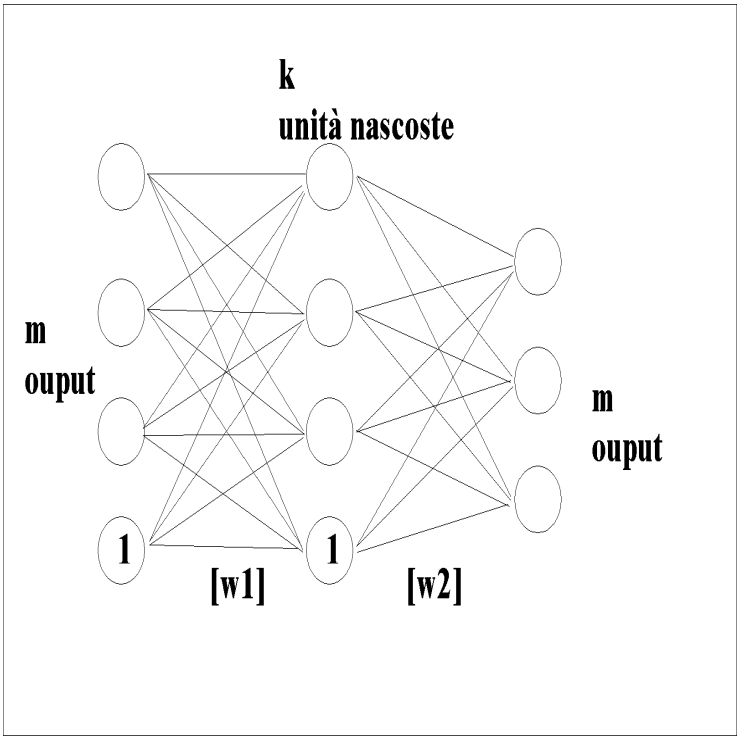

Figura 3: Caso di rete neurale ad un piano nascosto.

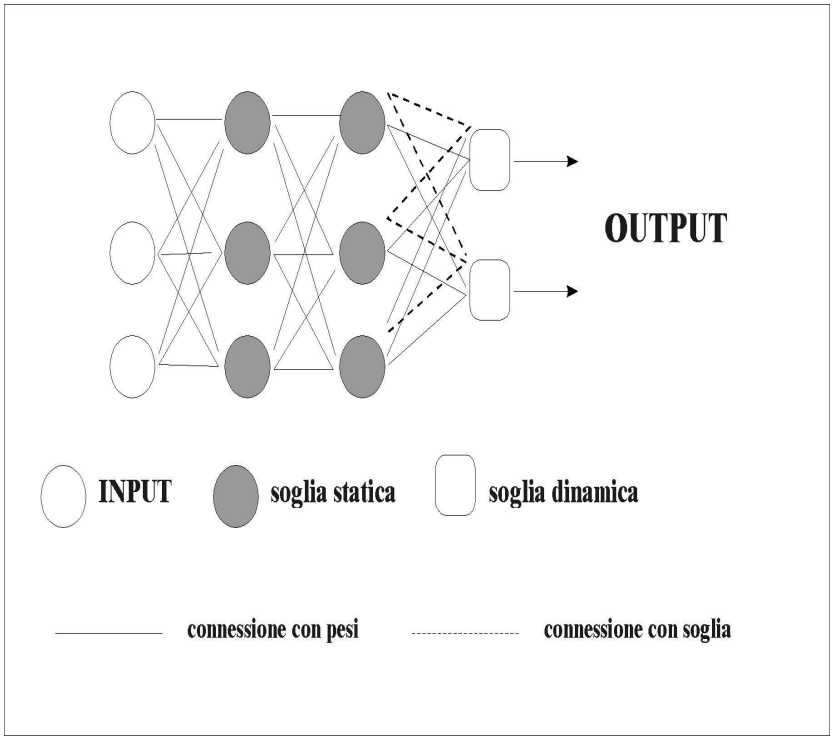

Figura 4: Rete neurale a due piani nascosti con soglia.