Appunti di Intelligenza Artificiale

A.A. 2023/2024

Rosso Carlo

Indice

1 Introduzione **2**

1.1 Che cosa sono le Artificial Neural Networks (ANN) 2

1.2 Caratteristiche delle ANN 5

2 Fondamenti **5**

2.1 Il neurone artificiale 5

2.2 Rappresentazione 6

Codifica locale 7

Codifica distribuita 7

Codifica grezza 7

Campi recettivi sovrapposti 8

Preparazione 8

Normalizzazione 8

2.3 Apprendimento 9

2.4 Analisi vettoriale di un neurone artificiale 10

2.5 Apprendimento hebbiano 11

La regola di Hebb 11

La regola postsinaptica 12

La regola presinaptica 12

La regola della covarianza 12

3 Regola Delta **13**

3.1 Unità lineari 13

3.2 Unità non lineari 14

4 Back-propagation **15**

4.1 Algoritmo 15

4.2 Momentum 17

Parametri addattivi 18

4.3 Pesi sinaptici iniziali 18

4.4 Addizione di rumore 19

5 Quickdrop **19**

6 Funzioni di attivazione **20**

6.1 Funzione "a gradino" 20

6.2 Funzione lineare 21

6.3 Funzione sigmoide 21

6.4 Funzione a base radiale 22

6.5 Funzioni logaritmiche 23

1 Introduzione

I programmi tradizionali elaborano l'informazione in modo radicalmente diverso rispetto ai sistemi nervosi biologici. Un computer tradizionale è composto sostanzialmente da un'unità di calcolo (processore, CPU), che esegue in succesione un altissimo numero di operazioni; e da tre tipi di memoria: una che contiene le istruzioni necessaria a svolgere le operazioni, una temporanea da cui vengono letti i dati necessari e salvati i risultati dei calcoli effettuati e una permanente in cui questi dati rimangono registrati. L'architettura e i principi di funzionamento del computer seriale sono stati utilizzati dal cognitivismo come metafora della mente; questa scelta presenta due vantaggi:

1. **Formalismo**: la scienza dell'informazione fornisce un formalismo scientifico e universalmente accettato con cui è possibile descrivere il funzionamento della mente in modo univoco;

2. **Modelli**: rende possibile l'impiego di modelli della mente per la creazione di nuove tecnologie con caratteristiche di intelligenza artificiale.

Il sistema nervoso può essere paragonato a un'immensa società (Cervellopoli). Ciascun abitante conosce quasi tutti gli abitanti del poprio paese o quartiere e passa il proprio tempo a parlare con tutti loro. Alcuni abitanti possiedono anche relazioni con individui che vivono in zone più distanti e mantengono così la propria comunità continuamente aggiornata su quello che succede. La comunicazione è spesso caratterizzata da ripetizioni, rumore e interruzioni.
L'elaborazione nei sistemi nervosi avviene in *parallelo* (mentre nei calcolatori tradizionali avviene in successione). L'elaborazione nei sistemi nervosi è *distribuita* su molti elementi. Durante lo svolgimento dei compiti, si attivano molti neuroni, a volte organizzati in gruppetti locali, altre volte distribuiti a macchie in zone diverse nel cervello. I sistemi nervosi non devono essere programmati per svolgere un compito, bensì imparano autonomamente in base all'esperienza o con l'aiuto di un insegnante esterno. Si ritiene che l'apprendimento consista soprattutto nella modifica della "forza" delle connessioni attraverso cui i neuroni comunicano.

1.1 Che cosa sono le Artificial Neural Networks (ANN)

Le reti neurali artificiali sono dei sistemi di elaborazione dell'informazione, il cui funzionamento trae ispirazione dai sistemi nervosi biologici. Una rete neurale artificiale possiede molte semplici unità

di elaborazione variamente connesse tra di loro. Alcune unità ricevono informazioni dall'ambiente (input), altre emettono risposte nell'ambiente (output), altre ancora comunicano solamente con le unità all'interno della rete (hidden).

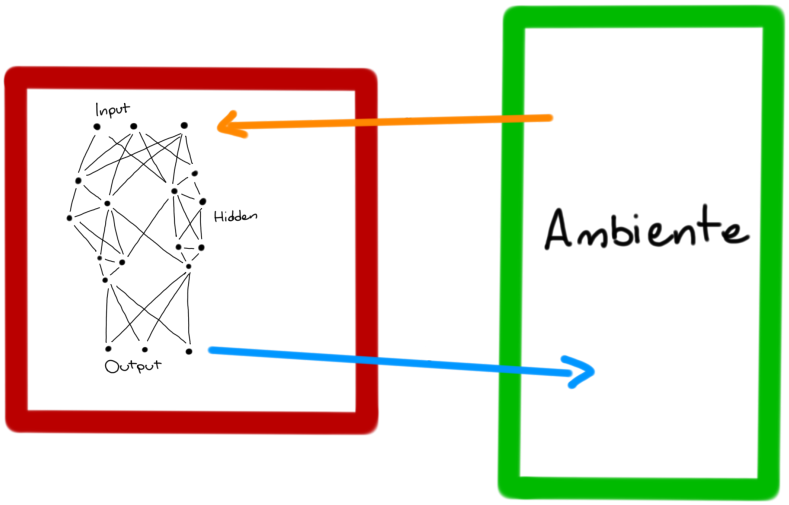


Figura 1: Schema di una rete neurale artificiale

Ciascuna unità simula il ruolo di un neurone (infatti sono anche chiamate neuroni o percettori (perceptrons)). Ciascuna unità diventa attiva se la quantità totale di segnale che riceve supera la propria soglia di attivazione. Ciascun punto di connessione agisce come un filtro che trasforma il messaggio ricevuto in un segnale inibitorio o eccitatorio, aumentandone o diminuendone l'intensità. Poiché il loro ruolo consiste in effetti nel pesare l'intensità dei segnali trasmessi, essi vengono definiti anche con il nome di pesi sinaptici o semplicemente pesi.

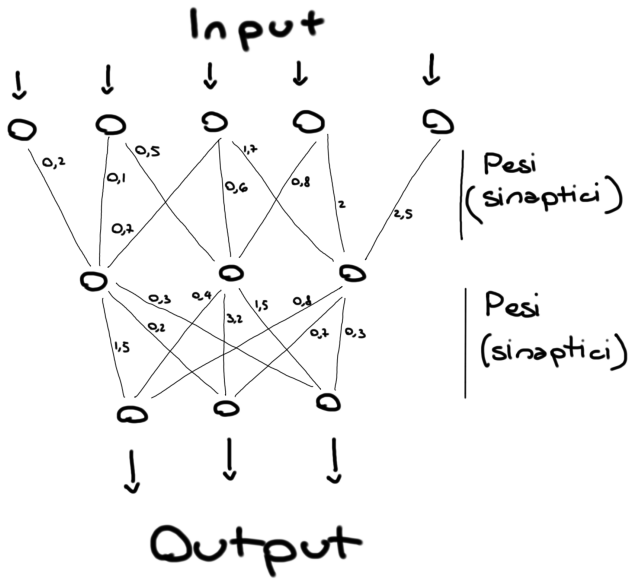


Figura 2: Schema di una rete neurale artificiale

Formalmente, il segnale di risposta emesso da un nodo n, è uguale a

$$n_i = \phi(\sum_{j \in \text{input}} w_{ij} \cdot n_j - \theta_i) \tag{1}$$

Dove:

- n_j è l'output del nodo j;

- w_{ij} è il peso della connessione tra i e j;

- θ_i è la soglia di attivazione del nodo i;

- ϕ è una generica funzione di attivazione (ne studieremo alcune all'interno del corso).

Quando uno stimolo viene applicato ai neuroni d'ingresso della rete neurale i segnali viaggiano in parallelo lungo le connessioni attraverso i nodi interni fino ai nodi di uscita la cui attivazione rappresenta la risposta della rete neurale. La configurazione delle connessioni e i valori delle sinapsi artificiali determinano in gran parte il comportamento e la risposta della rete. Per questo motivo si dice che le sinapsi rappresentano la conoscenza o memoria a lungo termine della rete neurale. Nelle reti neurali la memoria è totalmente distribuita e non richiede l'utilizzo di componenti aggiuntivi poichè è una proprietà intrinseca del sistema stesso.
La regola di apprendimento di un modello neurale indica semplicemente le condizioni locali e le

modalità in cui le sinapsi si modificano a prescindere dal tipo di compito per cui la rete verrà utilizzata.

1.2 Caratteristiche delle ANN

- **Robustezza**: una ANN è resistente al rumore, ovvero è in grado di continuare a dare una risposta corretta anche se alcune delle sue connessioni vengono eliminate o se viene aggiunto del rumore al segnale di ingresso;

- **Flessibilità**: una ANN può essere impiegata per un grande numero di finalità diverse: non ha bisongo di conoscere le proprietà del dominio specifico di applicazione perché le apprende in base all'esperienza. Questa caratteristica rappresenta un vantaggio perché permette di affrontare molti problemi di cui non sono note le soluzioni analitiche. Tuttavia, si rischia di rinunciare a cercare di comprendere a fondo la natura di un problema e di rifugiarsi in una soluzione neurale che non aumenta la nostra conoscenza;

- **Generalizzazione**: una ANN allenata su un numero limitato di esempi è in grado di produrre una risposta adeguata a dei pattern di ingresso che non ha mai visto in precedenza, ma che presentano qualche somiglianza con gli esempi presentati durante l'allenamento. In effetti, una ANN estrae caratteristiche invarianti dei pattern di ingresso piuttosto che memorizzare ciascun singolo pattern;

- **Recupero in base al contenuto**: le ANN sono in grado di recuperare le proprie memorie in base al contenuto partendo da dati incompleti, simili o corrotti da rumore.

2 Fondamenti

2.1 Il neurone artificiale

La descrizione degli elementi fondamentali dei circuiti neurali artificiali che segue si riferisce a proprietà generali dei modelli presentati nei capitoli successivi.
Un neurone artificiale è caratterizza da un insieme di sinapsi che corrispondono ai terminali di altri neuroni, da una soglia e da una funzione di attivazione. L'input netto o potenziale di attivazione

A_i di un neurone i è la somma algebrica dei prodotti fra tutti i segnali di ingresso x_j e i valori dei pesi corrispondenti w_{ij}:

$$A_i = \sum_{j=1}^{n} w_{ij} x_j$$

A cui solitamente si sottrae il valore di soglia θ_i del neurone:

$$A_i = \sum_{j=1}^{n} w_{ij} x_j - \theta_i$$

La risposta del neurone y_i viene calcolata sottoponendo il potenziale di attivazione (A_i) così ottenuto all'azione di una funzione di attivazione $\phi(A)$:

$$y_i = \phi(A_i) = \phi(\sum_{j}^{N} w_{ij} x_j - \theta_i)$$

Nella maggior parte dei modelli il peso w_{ij} può assumere valori positivi o negativi continui (non interi, ovvero con la virgola) ed è il valore che si modifica durante la fase di apprendimento. Ora risulta vantaggioso analizzare il sistema in notazione vettoriale. Dato che il potenziale di attivazione di un neurone è una funzione lineare di segnali di ingresso, il potenziale di attivazione di un intero strato di neuroni $A^T = \{A_1, A_2, \ldots, A_n\}$ è riscrivibile come:

$$A = W \cdot x$$

Dove:

- x^T è il vettore dei segnali d'ingresso e possono essere sia valori di input esterno (dall'ambiente) che valori di attivazione di uno strato inferiore di unità;
- $W = \{w_{ij}\}$ è la matrice dei pesi sinaptici, con w_{ij} che rappresenta il peso della connessione tra l'unità i e l'unità j;

2.2 Rappresentazione

Riporto alcune riflessioni sulle rappresentazione dei dati di input.
In generale le codifiche bipolari si rivelano vantaggiose rispetto alle codifiche binarie. Infatti, permette di trattare dati incompleti in modo neutro, al posto di usare -1 o 1, il dato mancante può

essere rappresentato con 0, che non ha alcun effetto sulle operazioni di somma e prodotto (quindi la tangente iperbolica è una funzione di attivazione più adatta rispetto alla sigmoide).

Codifica locale

Nella codifica locale, ogni unità di input rappresenta un determinato oggetto. Si tratta di una rappresentazione molto semplice, ma ha diversi svantaggi:

1. richiede un alto numero di unità di input: ovvero n unità per rappresentare n oggetti;
2. richiede di conoscere il numero di oggetti da rappresentare in anticipo;
3. è molto fragile alla perdita di un'unità di input, che corrisponde alla perdita dell'oggetto che rappresenta.

Codifica distribuita

Nella codifica distribuita, gli oggetti sono rappresentati da un codice binario di n nodi che rappresentano 2^n oggetti. Anche questa rappresentazione ha alcuni svantaggi:

1. mancanza di flessibilità: non è possibile rappresentare nuovi oggetti senza modificare la struttura della rete;
2. fragilità: la perdita di un nodo di input può portare alla perdita di un'informazione significativa.

Codifica grezza

Una soluzione proposta codifica le caratteristiche degli oggetti. Per cui ciascun oggetto è descrivibile da un insieme di caratteristiche (e.g. peso, colore, altezza, ecc.). Ciascuna unità di input codifica la presenza o il valore di una certa caratteristica. Questa rappresentazione ha diversi vantaggi:

1. flessibilità: è possibile rappresentare nuovi oggetti senza modificare la struttura della rete;
2. robustezza: la perdita di un'unità di input, comporta la perdita di una caratteristica dell'oggetto, ma sono comunque presenti altre caratteristiche che permettono di identificare l'oggetto.

Le unità di input possono essere rappresentate come uno spazio multi-dimensionale con tante dimensioni quante sono le unità. Oggetti con caratteristiche simili saranno rappresentati da punti vicini nello spazio multi-dimensionale. Per cui questa codifica facilita la classificazione e generalizzazione della rete neurale.

Campi recettivi sovrapposti

C'è un altro modo per ottenere una codifica grezza: si tassella l'input (e.g. l'immagine) in zone di dimensioni uguali e parzialmente sovrapposte. In questo caso, l'accuratezza della rappresentazione dipende dalla dimensione delle zone e dal numero di zone che coprono l'input. Approfondiamo le implicazioni che hanno le dimensioni e il numero di zone sulle informazioni che possono essere estratte:

- **campi recettivi piccoli e non sovrapposti**: sono usate per categorizzare le figure;
- **campi recettivi grandi e parzialmente sovrapposti**: sono usate per distinguere le singole figure. Infatti, larghi campi recettivi fanno sì che piccole caratteristiche possano essere codificate su un numero maggiore di nodi, permettendo una maggior discriminazione dei dettagli.

Preparazione

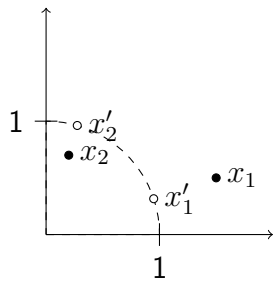
Prendendo spunto dalla biologia, risulta utile preparare i dati prima di essere dati in pasto alla rete neurale. Per esempio, si possono filtrare i dati, oppure si esegue la derivata sulle immagini per evidenziare i bordi e annullare i colori, senza perdere informazioni utili allo scopo del modello. In alternativa, si può usare una cascata di nodi con campi recettivi localmente ristretti in modo che l'ampiezza della finestra attraverso la quale ogni nodo osserva l'immagine diventa più grande man mano che si procede verso strati di elaborazione sucessiva.

Normalizzazione

Un'altra operazione che può essere eseguita sui dati è la normalizzazione. Questa operazione è utile per rendere i dati indipendenti dalla scala:

$$x' = \frac{x}{||x||} = \frac{x}{\sqrt{\sum_{i=1}^{n} x_i^2}}$$

In questo modo la lunghezza del vettore ottenuto sarà sempre 1.



2.3 Apprendimento

La risposta di una rete neurale è determinata dai valori sinaptici delle connessioni fra i nodi. Le reti neurali artificiali apprendono modificando gradulamente i propri valori sinaptici attraverso la presentazione ripetuta di una serie di esempi. Si distinguono tre tipi di apprendimento:

- **Apprendimento supervisionato**: la modifica dei valori sinaptici avviene impiegando una misura di errore tra la risposta fornita dalla rete neurale e la risposta desiderata per ogni vettore di input;

- **Apprendimento per rinforzo**: L'apprendimento supervisionato include anche una gamma di algoritmi che richiedono solamente una misura di bontà della risposta della rete neurale, piuttosto che la specificazione della risposta esatta per ogni pattern di input. Questi algoritmi sono denominati algoritmi di apprendimento per rinforzo;

- **Apprendimento non supervisionato**: sono anche chiamati apprendimento per auto-organizzazione. In questo caso, la rete neurale deve apprendere a riconoscere regolarità nei dati di input, in quanto la risosta desiderata è sconosciuta e deve essere individuata dalla rete medesima. Molti algoritmi di apprendimento senza supervisore derivano da una precisa formulazione dell'informazione che deve essere estratta dall'ambiente e richiedono dettagliate assunzioni sulla struttura dei pattern di input.

Approfondiamo ora le caratteristiche comuni a tutti i tipi di apprendimento:

1. **Stato iniziale**: i pesi iniziali sono assegnati in modo casuale entro un piccolo campo di variazione (e.g. $[-0.1, 0.1]$), oppure sono messi a zero;

2. **Iterazione**: l'apprendimento consiste nella presentazione ripetuta di una serie di vettori, detti anche pattern d'addestramento;

3. **Nuove conoscenze**: Gli algoritmi di apprendimento riguardano il calcolo di Δw_{ij}, la modifica dei pesi rispetto al valore corrente:

$$w_{ij}^t = w_{ij}^{t-1} + \Delta w_{ij}^t$$

4. **Tasso di apprendimento**: la velocità di apprendimento è regolata da una costante η:

$$w_{ij}^t = w_{ij}^{t-1} + \eta \Delta w_{ij}^t, \quad 0 < \eta < 1$$

5. **Test**: quando l'apprendimento è completato, i pesi sono "congelati" e si studia la risposta della rete neurale su dei pattern di test, che non sono stati usati durante l'apprendimento. Questo non è vero per gli algoritmi che operano all'interno della Teoria della Risonanza Adattiva.

2.4 Analisi vettoriale di un neurone artificiale

Per semplificare l'analisi consideriamo il caso di un singolo neurone i, tale che:

$$n_i = \phi(\sum_{j=1}^{n} w_{ij} x_j - \theta_i) = w \cdot x$$

La risposta dell'unità n_i è una misura della somiglianza tra il vettore di input ed il peso sinaptico. Come abbiamo già visto la norma (o lunghezza) di un vettore x è definita come:

$$||x|| = \sqrt{\sum_{j=1}^{n} x_j^2}$$

Il coseno dell'angolo θ tra due vettori x e w è definito come:

$$\cos(\theta) = \frac{w \cdot x}{||w|| \cdot ||x||}, 0 \leq \theta \leq \pi$$

Quindi, il prodotto interno $n_i = w \cdot x$ è

$$n_i = ||w|| \cdot ||x|| \cdot \cos(\theta)$$

Questo significa che muovendo nello spazio i due vettori mantenendo costante la loro lunghezza, il loro prodotto interno sarà proporzionale al coseno dell'angolo θ tra i due vettori. Ovvero, minore è l'angolo e maggiore è il prodotto interno. In particolare il prodotto interno sarà massimo quando i due vettori sono allineati, vale 0 quando sono ortogonali e sarà minimo quando sono opposti (e $\max = -\min$, al variare dell'angolo).
Infine, è importante notare che in una rete neurale è possibile giudicare quale unità possieda il vettore sinaptico più simile al vettore di input solo se i vettori sinaptici sono normalizzati.

2.5 Apprendimento hebbiano

La regola di modifica sinaptica di Hebb costituisce le fondamenta su cui si basano o derivano tutti gli algoritmi di apprendimento. Approfondiamo l'uso delle regole hebbiane in una rete etero-associativa feedforward con un singolo strato di snapsi e con unità binarie. Consideriamo un algoritmo di apprendimento supervisionato.

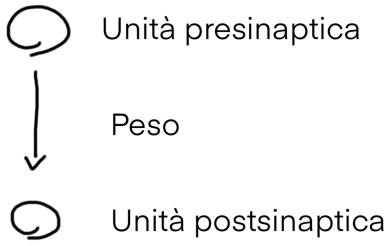


Figura 3: Rete neurale etero-associativa feedforward con un singolo strato di sinapsi e con unità binarie.

La regola di Hebb

Se due neuroni collegati tra di loro sono contemporaneamente attivi, il loro peso viene incrementato. La regola di Hebb prevede solamente l'incremento dei pesi, per cui la rete non è in grado di apprendere associazioni che presentano elementi in comune, ma che richiedono risposte differenti. Dunque la regola di Hebb permette di apprendere solamente pattern di input che sono ortogonali tra loro.

La regola postsinaptica

Questa regola è anche chiamata regola di Stent-Singer. Il peso viene incrementato ogni volta che l'unità postsinaptica e l'unità presinaptica sono entrambe attive; inoltre viene diminuito ogni volta che l'unità postsinaptica è attiva, ma quella presinaptica è inattiva. Questa regola migliora la generalità, tuttavia, non fallisce l'apprendimento di pattern di input parzialmente sovrapposti sullo stesso output, perché tende a creare troppe sinapsi inibitorie.

La regola presinaptica

Il peso viene aumentato quando sia l'unità presinaptica che l'unità postsinaptica sono attive; il peso viene diminuito quando l'unità presinaptica è attiva, ma l'unità postsinaptica è inattiva. Questa regola permette di generalizzare molto meglio rispetto alle regole precedenti.

La regola della covarianza

Viene anche chiamata regola di Hopfield. Quando l'unità presinaptica e l'unità postsinaptica hanno lo stesso stato, il peso viene incrementato; quando hanno stati opposti, il peso viene diminuito.

Unità presinaptica	+	+	−	−
Unità postsinaptica	+	−	+	−
Hebb	+			
Postsinaptica	+		−	
Presinaptica	+	−		
Covarianza	+	−	−	+

Tabella 1: Riassunto del funzionamento delle regole hebbiane.

Concludendo, le regole di Hebb possiedono parecchie limitazioni nel tipo di associazioni che sono in grado di apprendere, poiché sono spesso soggette a fenomeni di interferenza quando gli input d'ingresso non sono linearmente indipendenti.

3 Regola Delta

La regola delta è applicabile a unità di output dotate di una funzione di attivazione continua e differenziabile. Questa caratteristica permette di descrivere le prestazioni di una rete neurale tramite una funzione continua E_w che misura l'errore della rete.

3.1 Unità lineari

Consideriamo una rete neurale di tipo feedforward con unità di output ad attivazione lineare:

$$y_i = \sum_{j=1}^{n} w_j x_j \tag{2}$$

L'obiettivo è ottenere una matrice di pesi sinaptici W, per cui ogni esempio restituisce in output il valore corretto, ovvero ciascun esempio viene classificato correttamente:

$$y_i^\mu = t_i^\mu \quad \forall i, \mu$$

dove t_i^μ è l'output corretto (t sta per *target*) per l'input μ-esimo sul neurone i-esimo. La funzione di errore è definita come:

$$E_W = \frac{1}{2} \sum_\mu \sum_i (t_i^\mu - y_i^\mu)^2 \tag{3}$$

Ovvero, viene calcolata la distanza (lo scarto quadratico medio) tra l'output desiderato e quello effettivo per ogni esempio e per ogni neurone di output. L'obiettivo è minimizzare la funzione di errore E, infatti nel caso in cui tutti gli esempi vengano classificati correttamente, l'errore sarà nullo ($E_W = 0$). Dunque dobbiamo capire in quale direzione muovere i pesi per ridurre l'errore. La derivata della funzione di errore rispetto ai pesi ci fornisce la pendenza del grafico, ovvero dato un punto dello spazio dei pesi (individuato dai pesi sinaptici correnti) ci indica quali pesi aumentare e quali diminuire per ridurre il valore dell'errore. In particolare:

$$\Delta w_{ij} = -\frac{\partial E_W}{\partial w_{ij}} \tag{4}$$

ci indica in quale direzione muovere il peso w_{ij} per ridurre l'errore, indica a tutti gli effetti la pendenza del grafico dell'errore rispetto al peso e quindi in quale direzione si muove una pallina che rotola lungo il grafico. Calcoliamo dunque le derivate che ci interessano:

$$\Delta w_{ij} = \sum_{\mu} (t_i^{\mu} - y_i^{\mu}) x_j^{\mu} \tag{5}$$

Interpretiamo la formula: per ciascun esempio μ calcoliamo la differenza tra il valore desiderato e quello effettivo, moltiplichiamo il risultato per il valore dell'input j-esimo e sommiamo il tutto per ciascun esempio. Notiamo che se ci fosse solo un esempio, dopo il cambio di peso l'errore sarebbe nullo, infatti si va a modificare il peso in modo tale che l'output sia uguale al target. Sommando le variazioni per ogni esempio, si ottiene la variazione totale dei pesi, per avvicinare l'output effettivo a quello desiderato, non è detto che esso sia raggiunto, ma ci si avvicina.

3.2 Unità non lineari

Allo stesso modo possiamo calcolare la variazione dei pesi per unità di output dotate di una funzione di attivazione non lineare. In particolare, data una funzione di attivazione ϕ e un output y_i calcolato come:

$$y_i = \phi(\sum_{j=1}^{n} w_{ij} x_j) \tag{6}$$

La variazione dei pesi è calcolata come:

$$\Delta w_{ij} = \sum_{\mu} (t_i^{\mu} - y_i^{\mu}) \phi'(\sum_{j} w_{ij} x_j^{\mu}) x_j^{\mu} \tag{7}$$

Dove $\phi'(\sum_j w_{ij} x_j^{\mu})$ è la derivata della funzione di attivazione rispetto all'input. Poiché la formula diventa lunga e complessa, poniamo $A_i^{\mu} = \sum_j w_{ij} x_j^{\mu}$, per cui $y_i^{\mu} = \phi(A_i^{\mu})$, ottenendo:

$$\Delta w_{ij} = \sum_{\mu} (t_i^{\mu} - y_i^{\mu}) \phi'(A_i^{\mu}) x_j^{\mu} \tag{8}$$

Dunque per ogni esempio μ la modifica dei pesi sinaptici diventa:

$$\Delta w_{ij} = \eta (t_i - y_i) \phi'(A_i) x_j \tag{9}$$

4 Back-propagation

Il metodo di propagazione all'indietro dell'errore è applicabile a reti neurali con un numero qualsiasi di strati di connessioni e con architetture molto diverse. Proprio come la regola delta, modifica i pesi sinaptici in base alla discrepanza tra la risposta fornita dalla rete e la risposta corretta.

4.1 Algoritmo

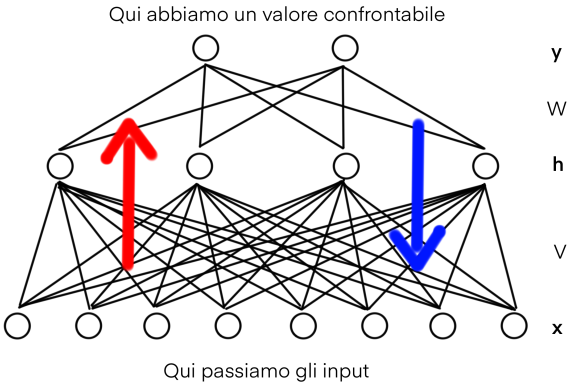


Figura 4: Algoritmo di back-propagation, la freccia rossa indica il flusso del calcolo della previsione, mentre la freccia blu indica il flusso del calcolo dell'errore.

Consideriamo una rete feedfoward a due strati di connessioni con unità di input x_k, unità interne h_j e unità di output y_i. Indichiamo con t_i^μ il valore della risposta correttà per ogni unità di output i sull'input μ.
Nell'esempio in questione, dunque abbiamo due strati di connessioni che devono essere aggiornati. Spieghiamo i passi dell'algoritmo:

1. **Input**: abbiamo in input un pattern d'ingresso x;
2. **Forward pass 1**: calcoliamo l'attivazione di tutte le unità interne $h_j = \phi(\sum_k v_{jk} x_k)$, dove v_{jk} è il peso della connessione tra l'unità in input k e l'unità interna j;
3. **Forward pass 2**: calcoliamo l'attivazione di tutte le unità di output $y_i = \phi(\sum_j w_{ij} h_j)$, dove w_{ij} è il peso della connessione tra l'unità interna j e l'unità di output i;

4. **Calcolo dell'errore 1**: ora vogliamo minimizzare l'errore, quindi confrontiamo il risultato ottenuto con il risultato corretto. Calcoliamo la correzione dei pesi sinaptici come

$$\Delta w_{ij} = \eta(t_i - y_i)\phi'(A_i)h_j,$$

dove η è il tasso di apprendimento, t_i è il valore corretto dell'unità di output i, y_i è il valore ottenuto dall'unità di output i, $\phi'(A_i)$ è la derivata della funzione di attivazione dell'unità di output i e h_j è l'attivazione dell'unità interna j. Infatti, abbiamo applicato la regola delta;

5. **Calcolo dell'errore 2**: calcoliamo l'errore dell'unità interna j come:

$$\Delta v_{jk} = -\eta \frac{\partial E}{\partial v_{jk}} \tag{10}$$

ovvero, calcoliamo l'errore dell'unità interna j come la somma degli errori delle unità di output i moltiplicati per i pesi delle connessioni tra l'unità interna j e l'unità di output i. Calcoliamo le derivate:

$$\Delta v_{jk} = \eta \sum_i (t_i - y_i)\phi'(A_i)w_{ij}\phi'(A_j)x_k \tag{11}$$

Dunque, l'algoritmo di back-propagation si svolge in due fasi: nella prima fase sono calcolati i valori di attivazione di ciascuno strato in ordine. In questo modo viene calcolata la previsione della rete. Nella seconda fase, invece, sono modificati i pesi per ridurre l'errore. Poiché le unità di output forniscono un risultato direttamente confrontabile con il risultato corretto, viene applicata la regola delta. In questo modo, viene calcolato l'errore commesso da ciacuna unità di output. Questo errore, passa allo strato inferiore seguendo i pesi sinaptici di ciascuna connessione, in questo modo viene calcolato l'errore per ciascuna unità sottostante. Dunque sono modificati i pesi sinaptici e di nuovo viene calcolato l'errore per lo strato inferiore. Fino ad arrivare allo strato di input. Riporto la formula per il calcolo della correzione dei pesi sinaptici per le unità di output:

$$\Delta w_{ij} = \eta(t_i - y_i)\phi'(A_i)h_j$$

E la formula per il calcolo dell'errore per le unità interne:

$$\Delta v_{jk} = \eta \sum_i (t_i - y_i)\phi'(A_i)w_{ij}\phi'(A_j)x_k$$

Notiamo la similitudine tra le due formule: se consideriamo l'errore come $E = (t_i - y_i)\phi'(A_i)$, per le unità di output, e l'errore come $E = \sum_i (t_i - y_i)\phi'(A_i)w_{ij}\phi'(A_j)$, per le unità interne, possiamo scrivere le formule in modo più compatto:

$$\Delta w_{ij} = \eta E_j x_j$$

Dove w_{ij} è il peso della connessione tra l'unità j e l'unità i, E_j è l'errore dell'unità j e x_j è l'attivazione dell'unità j. Comprendiamo ora la formula per il calcolo dell'errore per le unità interne:

$$E = \sum_i (t_i - y_i)\phi'(A_i)w_{ij}\phi'(A_j)$$

abbiamo $(t_i - y_i)$ che rappresenta l'errore dell'unità di output i, questo valore viene moltiplicato per la derivata della funzione di attivazione, in questo modo otteniamo l'errore dell'unità di output i. A questo errore moltiplichiamo il peso della connessione tra l'unità interna j e l'unità di output i. Infatti, il peso della connessione indica quanto l'unità interna j contribuisce al valore dell'attivazione dell'unità di output i e dunque quanto influisce sull'errore dell'unità di output i. A questo punto, ripetiamo il calcolo per tutte le unità di output e sommiamo i risultati, così otteniamo l'errore dell'unità interna j. Moltiplichiamo l'errore così ottenuto per la derivata del valore di attivazione, che rappresenta l'intensità del segnale; ovvero riduciamo l'errore se il segnale in input era molto basso e lo aumentiamo se il segnale era molto forte. Infine, applichiamo la regola delta per calcolare la correzione dei pesi sinaptici.

4.2 Momentum

Un metodo per incrementare la velocità di apprendimento riguarda l'adozione del momentum, ovvero viene aggiunta l'inerzia allo spostamento sulla superficie dell'errore, questo effetto si ottiene aggiungendo al calcolo della modifica di ciascuna connessione una frazione della modifica ottenuta all'istante precedente.

$$\Delta w_{ij}^t = \eta E_i x_j + \alpha \Delta w_{ij}^{t-1} \tag{12}$$

Dove α è la costante di momentum che contraolla la quantità di inerzia fornita alla modifica sinaptica. Tipicamente, α è fissato intorno a 0.8.

La presenza del momentum favorisce l'avanzamento su superfici piatte o semi-piatte e riduce le oscillazioni permettendo così l'adozione di valori più elevati per il tasso di apprendimento (η).

Parametri addattivi

Il tasso di apprendimento e la costante di momentum possono essere adattati dinamicamente durante il processo di apprendimento, in modo da ottenere una velocità di apprendimento ottimale, dunque la pallina si muove più velocemente o più lentamente a seconda della superficie su cui si trova e del percorso effettuato. Non solo anche l'accellerazione può essere adattata dinamicamente.

Non solo, si può aumentare il tasso di apprendimento per i pattern che sono sotto-rappresentati, ovvero per quei pattern che sono pochi in numero nel set di apprendimento, ma sono ben più frequenti nel mondo reale.

Oppure, si utilizza un tasso di apprendimento inversamente proporzionale al numero di connessioni per ciascun nodo. Ovvero, maggiore è il numero di connessioni, minore è il tasso di apprendimento. Questo perché, se un nodo ha molti input, allora il suo contributo all'errore è minore rispetto a un nodo con pochi input.

4.3 Pesi sinaptici iniziali

I valori iniziali dei pesi sinaptici sono importanti in quanto determinano i punto di partenza della rete neurale sulla superficie di errore. La backpropagation richiede che i pesi sinaptici assumano valori diversi tra loro, altrimenti tutti i pesi sinaptici si muoverebbero nella stessa direzione e con la stessa velocità. Date le proprietà delle funzioni di attivazione, considerando per esempio la sigmoide, oppure la tangente iperbolica, la derivata ha valore massimo in 0. Dunque se i pesi sinaptici sono inizializzati vicino allo 0, per esempio nell'intervallo $[-0.5, 0.5]$, allora la derivata della funzione di attivazione ha valore elevato e dunque la rete neurale apprende più velocemente. Si consiglia di limitare il valore dei pesi sinaptici entro $\pm 1/\sqrt{k_i}$, dove k_i è il numero di unità di input dell'unità i-esima.

Questa procedura stabilisce limiti diversi per i vari nodi della rete e assicura che la somma pesata degli input per ciascun nodo sia intorno a 0. Per cui, la funzione sigmoide e la tangente iperbolica hanno derivata massima.

4.4 Addizione di rumore

Un modo semplice per cercare di uscire dai minimi locali consiste nell'applicare un po' di rumore durante l'apprendimento della rete. Ovvero, si fanno variare i valori dei pesi sinaptici aggiungendo dei piccoli valori estratti da una distribuzione uniforme di numeri casuali intorno allo zero. Metaforicamente, è come introdurre dei piccoli terremoti sulla superficie dell'errore, per cercare di forzare la pallina a uscire dai minimi locali. Se l'errore continua a calare allora il rumore viene ridotto, altrimenti viene aumentato. Fondamentalmente si costringe la pallina a muoversi in modo casuale sulla superficie dell'errore, se l'errore è elevato, ma la pallina è bloccata in un minimo locale o in una superficie piatta.

5 Quickdrop

Fahlman ha proposto un algoritmo di discesa del gradiente completamente locale (per cui la modifica di una connessione richiede solo informazione direttamente disponibile a livello dell'unità presinaptica e di quella postsinaptica). Il metodo è basato sull'assunzione che la discesa della superficie di errore di un singolo peso sinaptico è relativamente indipendente dalle modifiche che vengono apportate agli altri pesi sinaptici. Questa assunzione è in realtà un'approssimazione, tuttavia l'algoritmo che si ottiene è molto molto rapido, per cui, nel momento in cui ci si ritrova in una stagnazione, è sufficiente far ripartire l'algoritmo da un punto diverso.
Il metodo consiste nel modificare ciascun peso sinaptico in base al confronto in due tempi successivi della variazione dell'errore ottenuta grazie a quel peso: se la variazione è nella stessa direzione della variazione precedente, allora il peso verrà modificato nella stessa direzione del cambiamento effettuato in precedenza, altriemnto esso verrà modificato nella direzione opposta. Per ogni peso della rete, la regola di modifica sinaptica è la seguente:

$$\Delta w^t = \frac{S^t}{S^{t-1} - S^t} \Delta w^{t-1} \tag{13}$$

Dove S esprime la variazione della funzione di errore E_W su tutti i pattern di addestramento per quel peso sinaptico:

$$S = \frac{\partial E_W}{\partial w} \tag{14}$$

Per cui, considerato una rete come quella della back-propagation (vedi Figure 4), la variazione dell'errore per un peso sinaptico, (S) è:

$$S_{ij} = -\sum_{\mu} E_i^{\mu} x_j^{\mu} \tag{15}$$

Dove S_{ij} è la variazione dell'errore per il peso sinaptico che connette l'unità j all'unità i, E_i^{μ} è l'errore dell'unità i per il pattern μ e x_j^{μ} è l'uscita dell'unità j per il pattern μ.
Dato che quando S assume la stessa direzione e valore per due epoche consecutive, il cambiamento dei pesi sinaptici cresce all'infinito, viene applicato un limite λ: $\Delta w^t = \lambda \Delta w^{t-1}$ (di solito $\lambda = 1.75$). Infine, viene addizionata una frazione della discesa di gradiente per evitare il "congelamento" dei pesi.

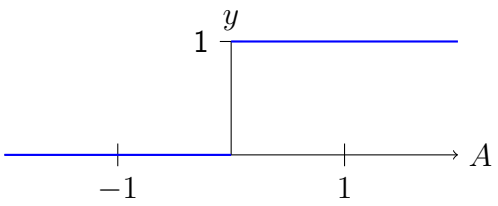
6 Funzioni di attivazione

La funzione di attivazione determina il tipo di risposta che un neurone può emettere. Di seguito sono elencati alcuni esempi di funzioni di attivazione.

6.1 Funzione "a gradino"

$$\phi(A) = \begin{cases} 1 & \text{se } A \geq \theta \\ 0 & \text{se } altrimenti \end{cases}$$

Dove θ è la soglia del neurone. Ponendo $\theta = 0$ si ottiene:



Alternativamente, l'output può essere bipolare:

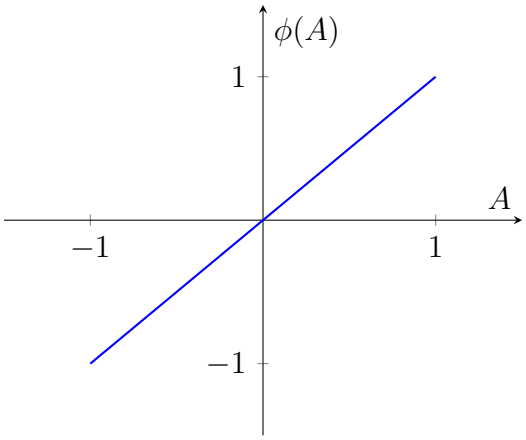
$$\phi(A) = \begin{cases} 1 & \text{se } A \geq \theta \\ -1 & \text{se } altrimenti \end{cases}$$

6.2 Funzione lineare

Maggiori informazioni sono trasmesse se si utilizza una funzione continua lineare:

$$\phi(A) = kA$$

dove k è una costante. Le funzioni continue permettono al neurone di trasmettere una gradazione di segnali di vaira intensità che può essere opportunamente sfruttata dai neuroni riceventi.

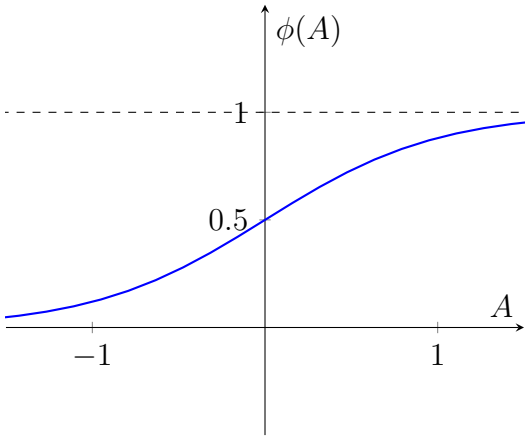


6.3 Funzione sigmoide

In alcune situazione risulta utile forzare l'output del neurone ad assumere valori compresi in un intervallo, per esempio $[0, 1]$ o $[-1, 1]$. Infatti, in questo modo si può interpretare l'output come una probabilità o come una misura di confidenza. La funzione sigmoide o logistica è definita come:

$$\phi(A) = \frac{1}{1 + e^{-kA}}$$

Dove k è la costante che controlla l'inclinazione della curva; per $k \to \infty$ la funzione si comporta come una funzione a gradino. Le rette $y = 0$ e $y = 1$ sono asintoti orizzontali della funzione sigmoide.



Una funzione simile è la tangente iperbolica ($\tanh(kA)$) che ha le rette $y = -1$ e $y = 1$ come asintoti orizzontali.
Nella maggior parte dei modelli tutti i neuroni eccetto i neuroni di input (recettori) utilizzano la medesima funzione di attivazione.

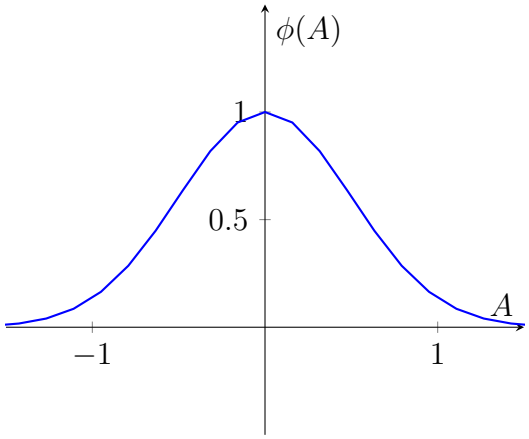
6.4 Funzione a base radiale

Una funzione a base radiale genera un output non-negativo per tutti i pattern ed è simmetrica rispetto al baricentro di un gruppo di input. Le funzioni più comunemente usaet sono delle variazioni della funzione gaussiana:

$$\phi(A) = \exp(-kA^2)$$

La cui derivata prima è esprimibile in funzione dell'output generato:

$$\frac{d\phi(A)}{dA} = -2kA\phi(A)$$



Le funzioni a base radiale sono particolarmente utili nel caso in i pattern di input si distribuiscano in gruppetti nello spazio; in questo caso durante l'apprendimento ciascuna unità centra la propria risposta sul baricentro di un gruppetto di pattern.

Moody e Darken hanno proposto di utilizzare la seguente funzione gaussiana:

$$\phi(A)_i = \frac{\exp(\frac{(A-\mu_i)^2}{2\sigma_i^2})}{\sum_k \exp(\frac{(A-\mu_k)^2}{2\sigma_k^2})}$$

Dove μ_i e σ_i sono rispettivamente la media e la varianza della funzione dell'unità i e il denominatore è la sommatoria dell'output di tutti i nodi k dello stesso strato. Per cui il vettore di attivazione dell'intero strato è normalizzato.

6.5 Funzioni logaritmiche

Per reti che possiedono un elevato numero di unità presinaptiche, è utile impiegare funzioni che non sono sottoposte a saturazione (cioè se diventa difficile cambiare i pesi sinaptici). In questi caso è utile utilizzare una funzione logaritmica:

$$\phi(A) = \begin{cases} \log(1+A) & \text{per } A \geq 0 \\ \log(1-A) & \text{per } A < 0 \end{cases}$$

Che ha derivata:

$$\phi'(A) = \begin{cases} \frac{1}{1+A} & \text{per } A \geq 0 \\ \frac{1}{1-A} & \text{per } A < 0 \end{cases}$$