A mia madre, la mia vera insegnante.

Se non puoi passare attraverso una montagna, giraci intorno;
se non puoi girarci intorno, passaci sopra;
se non puoi passarci sopra, siediti un attimo
e chiediti se raggiungere l'altro lato sia davvero così importante.

Se lo è comincia a scavare una galleria.

Indice

Capitolo 1

Presentazione delle reti neurali

1.1 Cenni storici ed origini

Le reti neurali artificiali, dall'inglese *Artificial Neural Network (ANN)*, e la conseguente computazione neurale, sono state costruite ispirandosi ai sistemi neurali biologici con l'obiettivo di simulare il comportamento di questi ultimi, riprodurne la struttura e le funzioni basilari.
Una definizione semplice e formale di tali strutture è fornita dall'inventore del primo neurocomputer, il Dr. Robert Hecht-Nielsen che le definì come:

«...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs»
(da Neural Network Primer: Parte uno, Maureen Caudill, 1989)

Tradotto in italiano
«... un sistema di calcolo costituito da una serie di semplici elementi di elaborazione, altamente interconnessi, che elaborano le informazioni attraverso il loro stato dinamico rispondendo agli input esterni.»

Le basi per lo studio di tali reti furono poste dallo psichiatra Warren McCulloch e del matematico Walter Pitts, i quali riuscirono a riprodurre una rete neurale utilizzando semplici circuiti elettrici collegati tra loro. Questa collaborazione portò alla luce l'analogia esistente tra le reti neurali e la macchina di Turing ed in tal modo si capì che qualsiasi operazione eseguita da una rete neurale poteva essere eseguita anche da un computer. Le reti che

furono prodotte risultavano essere automi a stati finiti in grado di realizzare la logica delle proposizioni e di formulare ipotesi sulla natura dei meccanismi cerebrali come un programma per computer. Il frutto di tale lavoro fu reso noto nel 1943 con la pubblicazione del libro "*A logical calculus of the ideas immanent in nervous activity*", nel quale fu schematizzato un combinatore lineare a soglia con dati binari multipli in entrata e un singolo dato binario in uscita.
McCulloch e Pitts riuscirono quindi a presentare il modello di neurone formale dimostrando che reti formate da tali neuroni riuscivano a computare funzioni della logica del primo ordine.
Un punto di svolta per lo studio delle reti neurali si ebbe successivamente alla pubblicazione del lavoro dello psicologo Donald Hebb, "*The Organization of Behavior*" nel 1949. Hebb trovò un'interconnessione tra la psicologia del comportamento umano e la fisiologia del sistema nervoso, offrendo così un grande contributo alla teoria sull'apprendimento associativo. Tale teoria risultò essere alla base dei metodi di apprendimento delle reti neurali e si fonda sulla nota legge di Hebb: ≪*Se un neurone A è abbastanza vicino ad un neurone B da contribuire ripetutamente e in maniera duratura alla sua eccitazione, allora ha luogo, in entrambi i neuroni, un processo di crescita o di cambiamento metabolico per cui l'efficacia di A nell'eccitare B viene accresciuta*≫.
Il decennio che va dagli anni cinquanta agli anni sessanta fu totalmente influenzato dalla legge di Hebb che vide che numerosi gruppi di ricerca condurre esperimenti e test sulle funzionalità del cervello fino a porre le basi per la nascita dell'intelligenza artificiale (AI).
Nel 1958 Frank Rosenblatt introdusse il primo schema di rete neurale che designò con il termine "***perceptron***", in italiano percettrone, allo scopo di fornire un'interpretazione dell'organizzazione generale dei sistemi biologici attraverso un modello mirato all'analisi di funzioni in forma matematica. Il percettrone fu il primo modello di apprendimento supervisionato, costituito da uno strato di ingresso ed uno di uscita in grado di discriminare gli ingressi in due insiemi linearmente separabili.
Il percettrone risultò essere un modello più efficace rispetto a quello binario di McCulloch e Pitts, poichè i suoi pesi sinaptici erano variabili e quindi in grado di apprendere.
Nonostante l'iniziale successo di tale modello e l'interesse mostrato dalla comunità scientifica, tale rete neurale non risultò abbastanza potente; le reti a due strati basate sui percettroni avevano limiti operativi, non riuscivano cioè a risolvere tutte le classi di problemi, in particolare quelli non caratterizzati dalla separabilità lineare delle soluzioni come ad esempio l'operatore *XOR*.
Solo negli anni ottanta, con il matematico Paul Werbos, si superarono i limiti

del percettrone di Rosenblatt. Werbos introdusse uno o più livelli intermedi all'interno delle reti neurali creando una classe chiamata "***Multi-Layers Perceptron***" ovvero percettrone multistrato, il cui metodo di addestramento principale era l'*error backpropagation*, l'algoritmo di retropropagazione dell'errore che permetteva la modifica sistematica dei pesi delle connessioni in modo da rendere la risposta della rete quanto più vicina a quella desiderata.
Tale algoritmo proposto nel 1986 da David E.Rumelhart, G. Hinton e R. J. Williams consentì di superare le problematiche legate al percettrone di Rosenblatt e permise di risolvere il problema della separabilità non lineare delle soluzioni, rendendo quindi possibile calcolare la funzione XOR e segnando il definitivo rilancio delle reti neurali.

1.2 Analogie con il sistema nervoso ed applicazioni

Come accennato nel paragrafo precedente, una rete neurale è un sistema computazionale costruito basandosi sui processi biologici naturali il cui obiettivo è la riproduzione delle attività tipiche del cervello umano, ad esempio la comprensione del linguaggio, la percezione di immagini, il riconoscimento di forme ecc.
In altre parole lo scopo di una rete neurale artificiale è l'emulazione del sistema nervoso animale in particolar modo di quello umano. Quest'ultimo presenta numerose caratteristiche che risultano essere idonee per la riproduzione di un sistema computazionale: esso è flessibile poiché si adatta ad ogni tipologia di situazione, è robusto, resistente, piccolo e dissipa poca energia.

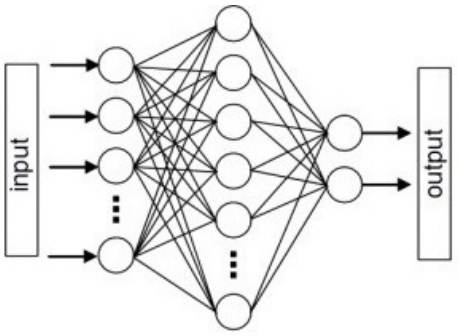

Figura 1.1: Esempio di rete neurale

Tale riproduzione non deve essere però intesa come atta alla costruzione di un cervello artificiale. Infatti le caratteristiche delle reti neurali biologiche, riprese dalla computazione neurale artificiale, sono in esigua minoranza: gli stessi neuroni artificiali sono solo un'approssimazione dei neuroni biologici e sono in grado di riprodurre solo tre dei circa centocinquanta processi che sono tipici dei neuroni del cervello umano.
Il sistema nervoso umano può essere pensato quindi come una grande struttura computazionale formata da milioni di unità fortemente interconnesse tra loro in modo parallelo riuscendo a trasformare continui input in output ragionevoli. Le reti neurali rappresentano una riproduzione significativa di tale struttura, in particolar modo degli algoritmi di apprendimento e di ottimizzazione basati su un modello *connessionistico* di calcolo: le operazioni responsabili dello scambio di informazioni avvengono per mezzo dell'interazione tra le unità elementari. Esse sono sistemi altamente paralleli: fornendo i dati del problema alle unità di input, la computazione si propaga in parallelo nella rete fino alle unità di output che producono il risultato.
II campo tipico di applicazione delle reti neurali è quello dell'individuazione di legami di **ingresso-uscita** all'interno di sistemi complessi in cui i dati a disposizione sono molto numerosi oppure contengono poca informazione e quindi non risulta chiaro quali relazioni esistano tra le diverse variabili che caratterizzano il problema. Si tratta di campi in cui l'analisi statistica di tutte le variabili risulta difficoltosa o dispendiosa in termini di calcolo.
Ecco un breve elenco di applicazioni note:

- elaborazione delle immagini (compressione e miglioramento di immagini in tempo reale);
- elaborazione del suono (compressione e miglioramento in tempo reale della voce e della musica, aiuti audio e protesi ecc...);
- rivelazione, verifica e identificazione di caratteri e oggetti (codici a barre, impronte digitali, firme, facce ecc...);
- riconoscimento della voce (elaborazione di parole dettate, identificazione di persone per via vocale, selezione telefonica ecc...);
- applicazioni basate su ingressi di origine sensoriale (ingresso tattile, visivo, acustico, olfattivo);
- archivi (musica, video, immagini ecc...);
- sintesi e miglioramento dati (fax, grafica, cancellazione di rumore, ecc...);
- robot autonomi.

1.3 Il neurone biologico

Per capire a fondo la connessione tra le reti neurali artificiali e le reti neurali del sistema nervoso umano e le numerose analogie che ne derivano, è bene dare una breve descrizione del secondo sistema.
Il sistema nervoso è composto principalmente dai **neuroni**, le unità fondamentali di produzione di scambi e di segnali. Ogni neurone deve adempire a cinque funzioni principali:

- ricevere informazioni dall'ambiente esterno o interno, oppure da altri neuroni;
- integrare le informazioni ricevute e produrre un'adeguata risposta in forma di segnale;
- condurre il segnale al suo terminale di uscita;
- trasmettere il segnale ad altre cellule nervose, ghiandole o muscoli;
- coordinare le proprie attività metaboliche.

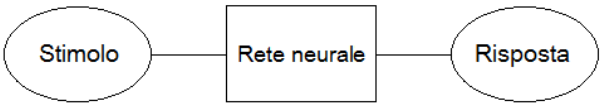

Figura 1.2: Stadi del sistema nervoso

Un neurone è formato da un corpo centrale identificato con il nome di *soma* all'interno del quale è presente il nucleo, e da molti prolungamenti citoplasmatici che si distinguono in:

- ***dendriti*** , organizzati con diramazioni ad albero che costituiscono il *ramo dendritico*;
- ***assone*** , la cui parte finale prende il nome di *bottone sinaptico*.

I dendriti ricevono segnali dai neuroni afferenti e li propagano verso il nucleo. Viaggiando lungo i dendriti, i segnali confluiscono al corpo del neurone che, comportandosi come un centro di integrazione, li interpreta e decide se produrre un potenziale d'azione ovvero il segnale elettrico di uscita del neurone. L'assone conduce il segnale verso altre cellule grazie alla presenza del bottone sinaptico alla sua estremità. La maggior parte dei terminali sinaptici contiene una sostanza chimica specifica, detta *neurotrasmettitore*, che viene liberata in risposta a un potenziale d'azione che percorre l'assone.
Il punto di connessione tra il terminale di un neurone ed il dendrite di un altro costituisce una struttura altamente specializzata che prende il nome di *sinapsi* e che risulta quindi essere la responsabile delle interazioni.

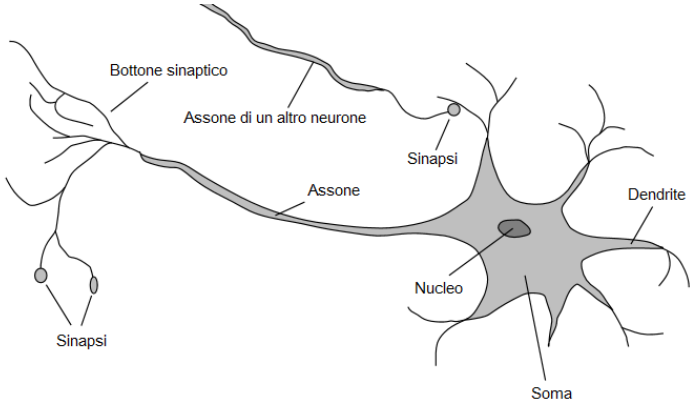

Figura 1.3: Neurone biologico

Nel sistema nervoso sono presenti circa 10^{11} cellule nervose connesse strettamente tra loro attraverso numerosi collegamenti che nel loro insieme vanno a formare la rete neurale. L'attività neurale è quindi l'attività di reti di neuroni che trasportano l'informazione ed ha come risultato finale l'emergere di sequenze significative di segnali.

1.4 Modello del neurone artificiale

La breve illustrazione del sistema neurale umano nel paragrafo precedente suggerisce uno schema per delineare l'organizzazione delle reti neurali artificiali: queste ultime sono formate da un elevato numero di unità computazionali, che possono essere equiparate ai neuroni umani, capaci di eseguire una

somma pesata. Tali unità sono collegate tra loro attraverso delle connessioni, così come le sinapsi collegano i neuroni nella rete umana.
Consideriamo una generica unità j costituita da n canali di ingresso $x_1, x_2, ..., x_n$. Gli input provenienti da strati precedenti o direttamente dall'esterno entrano nel neurone tramite tali canali.

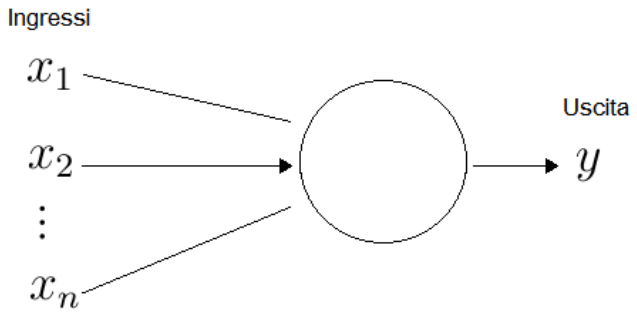

Figura 1.4: Canali di ingresso in un neurone

Sulle connessioni sono presenti dei *pesi sinaptici* w_i, numeri reali che denotano l'*efficacia sinaptica*, ovvero la forza della connessione. Se $w_i > 0$ il canale è detto *eccitatorio*, se $w_i < 0$ il canale è *inibitorio*.
I segnali in entrata, pesati dalle rispettive sinapsi, sono convogliati nel *soma* del neurone artificiale all'interno del quale vengono sommati producendo una combinazione lineare così definita:

$$\sum_{i=1}^{n} w_i x_i \tag{1.1}$$

La somma pesata degli ingressi viene indicata con la parola *net* ed il segnale con cui il neurone trasmette la sua attività all'esterno è calcolato applicando una *funzione di attivazione* φ che limita l'ampiezza dell'output; si assume per comodità che le ampiezze degli output appartengono all'intervallo $[0, 1]$ oppure $[-1, 1]$.

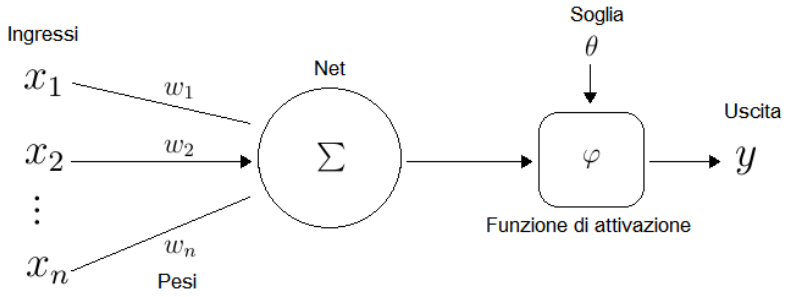

Figura 1.5: Modello neurone

Il modello neuronale include anche un valore *soglia* che ha l'effetto, a seconda della sua positività o negatività, di aumentare o diminuire il valore in ingresso alla funzione di attivazione.
L'output finale sarà allora:

$$y = \varphi(\sum_{i=1}^{n} w_i x_i) \tag{1.2}$$

E se indichiamo con θ il valore di soglia, la (1.2) diventerà:

$$y = \varphi(\sum_{i=1}^{n} w_i x_i - \theta) \tag{1.3}$$

Interpretando la soglia come il peso associato ad un ulteriore canale di ingresso x_0, e quindi $w_0 = \theta$, potremmo anche scrivere:

$$y = \varphi(\sum_{i=0}^{n} w_i x_i) \tag{1.4}$$

Il modello finale sarà

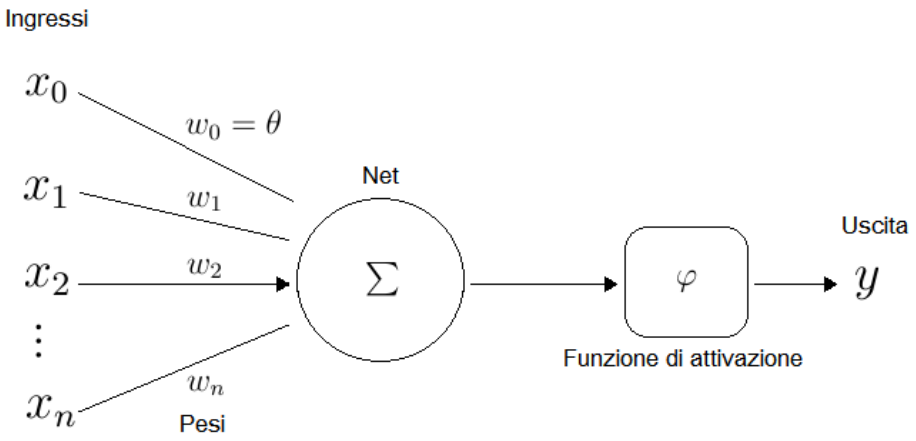

Figura 1.6: Modello neurone con soglia

L'effetto di un segnale x_i sul neurone è quindi uguale al prodotto $w_i \cdot x_i$ dove w_i è il peso attribuito alla sinapsi corrispondente ed il potenziale di attivazione è dato dalla somma algebrica dei prodotti di tutti i segnali di ingresso e dei valori dei pesi corrispondenti.
Schematizzando ed individuando il neurone artificiale come l'unità di calcolo fondamentale della rete neurale, gli elementi base che lo rappresentano sono:

- un insieme di connessioni;
- un sommatore;
- una funzione di attivazione;
- un valore di soglia.

1.5 Attivazione di un neurone e funzioni di attivazione

Nelle reti neurali si può avere una *attivazione asincrona* in cui i neuroni si attivano uno per volta ed un'attivazione sincrona in cui tutti i neuroni si attivano contemporaneamente. L'attivazione di un neurone dipende solo dall'informazione ricevuta ed è indipendente dalle operazioni svolte dagli altri

nodi. La funzione di attivazione determina il tipo di risposta che un neurone è in grado di emettere. Definisce, quindi, l'uscita di un neurone in funzione del livello di attivazione. L'uscita può essere un numero reale, un numero reale appartenente ad un intervallo, oppure un numero appartenente ad un insieme discreto.
Gli step per l'attivazione di un neurone sono:

1. i valori di input arrivano al neurone e vengono combinati con i relativi pesi;
2. i prodotti risultanti vengono sommati e tale somma viene confrontata con un valore di soglia che dipende dalla funzione di attivazione del neurone;
3. se la somma supera il valore di soglia, il neurone si attiva inviando output sulle connessioni pesate in uscita, altrimenti viene inibito.

Esistono varie tipologie di funzioni di attivazione, le principali sono: funzioni a soglia, funzioni lineari, funzioni lineari a tratti, funzione sigmoide.

1.5.1 Funzione a soglia

Imponendo $a = \sum_{i=0}^{n} w_i x_i$, il valore di uscita di un neurone assunto é:

$$y = \varphi(a) = \begin{cases} 1 & se\ a \geq 0 \\ 0 & se\ a < 0 \end{cases} \tag{1.5}$$

Se si tralascia il contributo dell'ingresso x_0 dal livello di attivazione e si ha quindi $a = \sum_{i=1}^{n} w_i x_i$, si avrà:

$$y = \varphi(a) = \begin{cases} 1 & se\ a \geq \theta \\ 0 & se\ a < \theta \end{cases} \tag{1.6}$$

I rispettivi grafici saranno

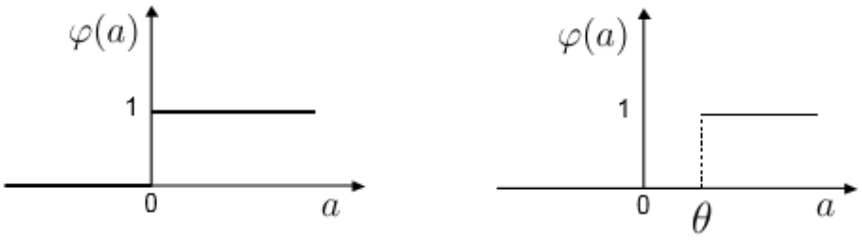

Figura 1.7: Funzione di attivazione a soglia

A volte è opportuno che la funzione di attivazione assuma valori tra –1 e +1 ed in questo caso la funzione a soglia che diviene la ben nota *funzione segno* è ridefinita così :

$$y = \varphi(a) = \begin{cases} 1 & se\ a > 0 \\ 0 & se\ a = 0 \\ -1 & se\ a < 0 \end{cases} \tag{1.7}$$

1.5.2 Funzione lineare e funzioni lineare a tratti

Se $a = \sum_{i=0}^{n} w_i x_i$, una funzione lineare è di questo tipo:

$$y = \varphi(a) = a \tag{1.8}$$

Un esempio di funzione di attivazione lineare a tratti è:

$$y = \varphi(a) = \begin{cases} 1 & se\ a \leq -0.5 \\ a + 0.5 & se\ -0.5 < a < 0.5 \\ 0 & se\ a \geq 0.5 \end{cases} \tag{1.9}$$

Rappresentandole si ha

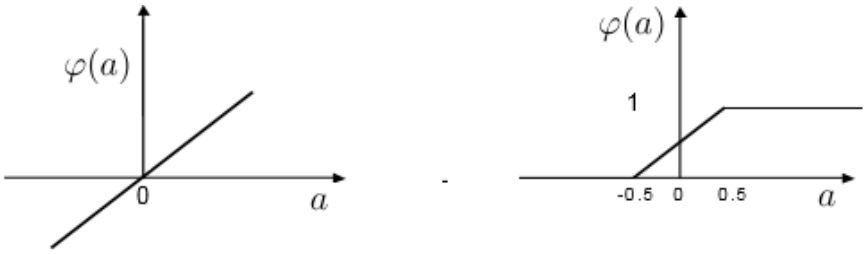

Figura 1.8: Funzione lineare(a sinistra), funzione lineare a tratti (a destra)

1.5.3 Funzione sigmoide

La funzione sigmoide appartenente alla famiglia di funzioni continue non lineari ed è tra le più utilizzate.
Tra queste funzioni riportiamo la *funzione logistica* così definita:

$$y = \varphi(a) = \frac{1}{1 + e^{-a}} \tag{1.10}$$

imponendo sempre $a = \sum_{i=0}^{n} w_i x_i$.

Essa assume questa forma

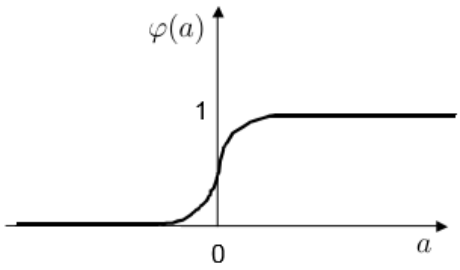

Figura 1.9: Funzione sigmoide

1.6 Architettura di una rete neurale

L'architettura di una rete neurale artificiale è caratterizzata da:

- numero di strati di sinapsi;
- numero di neuroni presenti nell'*input layer* ovvero lo strato di ingresso;
- numero di neuroni nell'*output layer*, lo strato di uscita.

Le reti neurali si suddividono principalmente in due grandi classi: le reti *feedforward* e le reti *feedback* o *ricorrenti*. Si ha inoltre un'altra tipologia di reti che sono le *reti completamente connesse*.

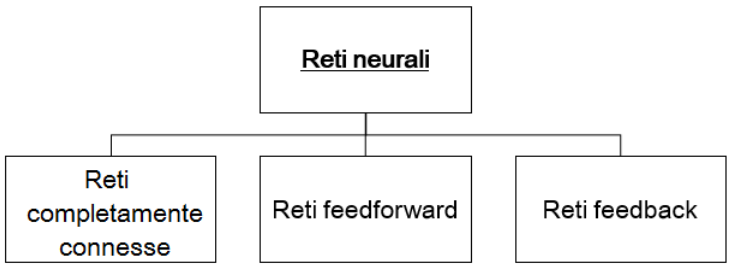

Figura 1.10: Suddivisione delle reti neurali

Reti completamente connesse

Nelle reti completamente connesse ogni neurone è connesso con tutti gli altri. Le connessioni tra i neuroni sono bidirezionali e possono essere rappresentate per mezzo di una matrice quadrata W di dimensione pari al numero di neuroni. Un suo generico elemento $w_{i,j}$ rappresenta il peso della connessione tra il neurone i ed il neurone j.

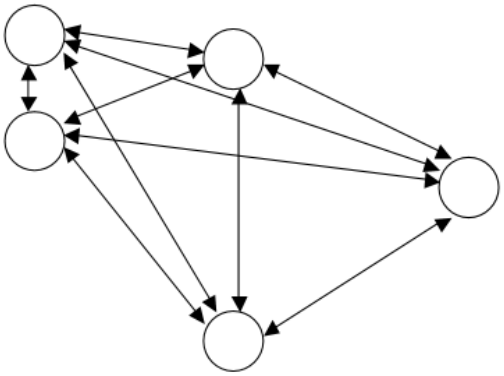

Figura 1.11: Rete completamente connessa

Reti feedforward a strati

In queste reti i segnali viaggiano dallo strato di ingresso verso lo strato di uscita e pertanto vengono anche chiamate **reti feedforward**. Nelle reti stratificate non esistono connessioni tra i neuroni all'interno di uno stesso strato, nè tra neuroni di strati non limitrofi. Ogni neurone in un generico strato è connesso con tutti quelli dello strato successivo ed i neuroni dello strato di ingresso hanno come unico compito quello di trasmettere i segnali ricevuti allo strato successivo, all'interno di essi non avviene alcuna computazione.
Le reti feedforward a strati si distinguono in base al numero di strati che presentano, numero che dipende dallo specifico problema che si intende risolvere.
- *Reti feedforward ad uno strato* : questa è una forma semplice di reti a strati. Il segnale nella rete si propaga in avanti senza cicli, non ci sono connessioni che tornano indietro e nemmeno connessioni trasversali nel layer di output.
- *Reti feedforward a più strati* : le reti a più strati sono anche dette *reti multilivello* , in inglese **Multi-Layer Perceptron, MLP**. Tra l'input layer e l'output layer sono presenti uno o più strati di neuroni nascosti, si parla quindi di *hidden layers* Nelle MLP non esistono connessioni nè tra neuroni di uno stesso strato nè tra neuroni di strati non limitrofi. Ogni strato ha connessioni entranti dal precedente strato e uscenti in quello successivo, quindi la propagazione del segnale avviene in avanti in modo aciclico e senza

connessioni trasversali.
Tali reti vengono utilizzate per superare problemi che possono sorgere nella discriminazione dei segnali: attraverso i neuroni nascosti si ottengono delle rappresentazioni interne dei segnali di input che consentono il riconoscimento di forme più complesse, facilitando il compito della rete. Nonostante tutto l'aggiunta di ulteriori strati nascosti non ottimizza le abilità di discriminazione della rete; questo è possibile solo se la funzione di attivazione è non lineare (una rete multistrato a neuroni lineari è sempre riconducibile ad una rete con due soli strati).

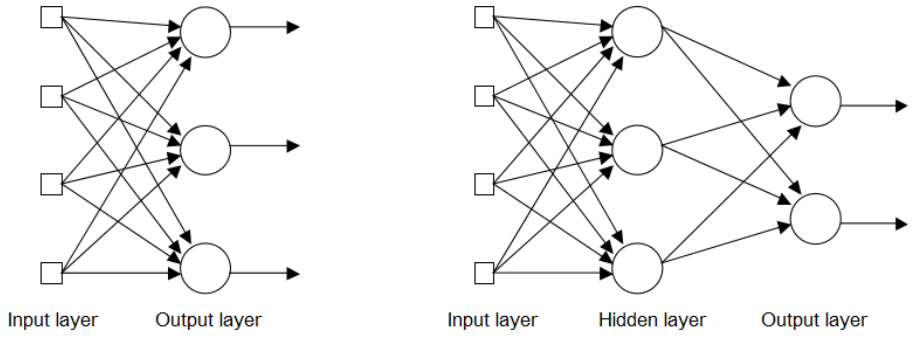

Figura 1.12: Rete ad uno strato (a sinistra), rete multilivello (a destra)

Reti ricorrenti
Una rete neurale ricorrente (RNN) si distingue dalle precedenti nel fatto che è ciclica. Il sistema è dinamico, le unità di output sono connesse con quelle intermedie e con quelle di input, si ha una *retroazione*. Dato un determinato stimolo la risposta della rete ricorrente non viene dettata soltanto dai caratteri strutturali della rete stessa, come si verifica nella rete feedforward, ma varia in funzione del precedente contesto in cui si è manifestato lo stimolo. L'output non è determinato solo dall'input, ma anche da una *cronologia* di input che fornisce una forma di memoria a breve termine. Queste reti sono adatte per dati strutturati temporalmente ed hanno anche la possibilità di avere un'attività in assenza di una stimolazione input poiché gli impulsi che viaggiano sulle vie ricorrenti possono essere sufficienti a mantenere il sistema in attività.

Capitolo 2

L'apprendimento delle reti neurali

2.1 Addestramento e generalizzazione

Per costruire una rete neurale efficiente un passo fondamentale è individuare un **insieme di apprendimento** ed un **algoritmo di apprendimento**.
Per insieme di apprendimento si intende una collezione di esempi chiamata **training set** dai quali la rete può attingere per raggiungere lo scopo per il quale è stata progettata; per algoritmo, invece, un procedimento che permetta di prelevare le informazioni dall'insieme di apprendimento e di fissare dei parametri che vengano poi modificati attraverso operazioni iterative interfaccianti con l'ambiente.
L'apprendimento avviene sempre grazie ad un certo numero di esempi prelevati dal mondo reale ed opera in due fasi distinte:

- Fase di apprendimento o addestramento nota come **learning** o **training**;

- Fase di generalizzazione nota come **recall**.

Durante la fase di learning si cerca di far imparare alla rete tutte le informazioni contenute nel training set ottenendo un modello che verrà poi utilizzato nella fase di generalizzazione per analizzare nuovi ingressi.
L'architettura della rete neurale gioca un ruolo fondamentale per l'efficienza nella fase di apprendimento. Non bisogna però immaginare che con l'aumentare del numero di unità cresca il potere computazionale della rete. Infatti,

in tal caso, ci si troverebbe di fronte ad una situazione di *overfitting*, un adattamento forzato della rete dato che risulterebbe avere un numero eccessivo di parametri rispetto al numero di esempi forniti. Questa situazione tende a far diminuire la capacità della rete di generalizzare su nuovi esempi originando una sorta di principio di indeterminazione dell'apprendimento che può essere affrontato con la limitazione del numero degli ingressi.
Per **generalizzazione** di una rete si intende la sua capacità di fornire le risposte appropriate a pattern di input che non sono mai stati incontrati. Essa risulta essere un punto di forza delle reti neurali e dipende dal numero di unità nascoste, dai pesi sinaptici e dal numero di training record.
Una rete neurale "generalizza bene" se la mappa input/output che essa genera è corretta per esempi di test mai presentati in fase di training (si assume che gli esempi di test siano tratti dalla stessa popolazione usata per generare il training set). In questi termini essa produce mappature input/output corrette anche se l'input è lievemente differente dagli esempi usati in fase di training; se però la rete viene addestrata con troppi esempi, si rischia che quest'ultima memorizzi il training set trovandosi nella condizione di *overtraining*.
Per avere una struttura ottimale della rete dal punto di vista della capacità di generalizzazione una guida adatta è il **Principio di Occam**, che suggerisce molto sulla complessità delle reti e sulla loro adeguatezza e che in relazione a queste ultime, può essere enunciato nel seguente modo:

"Date due reti che soddisfano l'insieme di apprendimento, la rete di minore complessità è quella che si comporta meglio su esempi non visti, cioè ha la migliore capacità di generalizzazione."

2.2 Paradigmi di apprendimento

Le reti neurali si ispirano al tratto caratteristico del sistema nervoso, ovvero la capacità di acquisire esperienza da esempi del mondo reale: per questo, oltre che di apprendimento, si parla di *addestramento* delle reti neurali attraverso dei *paradigmi di apprendimento*.
I paradigmi di apprendimento si suddividono in:

- *apprendimento* ***supervisionato***
- *apprendimento* ***non supervisionato***

Apprendimento supervisionato

Tale paradigma presuppone un *training set* cioè un set di esempi nel quale sono presenti coppie del tipo (x_k, y_{dk}) dove la prima variabile indica il k-esimo ingresso e la seconda la k-esima uscita desiderata. Con y_k si indica l'uscita reale e la si confronta con l'uscita desiderata: l'obiettivo è modificare i pesi affinché si minimizzi la differenza tra le due uscite. Il training set iniziale viene proposto ripetutamente finché $y_k \approx y_{dk}$, ovvero l'uscita reale sia il più simile possibile a quella desiderata, il tutto modificando i pesi in base alla legge di apprendimento scelta.

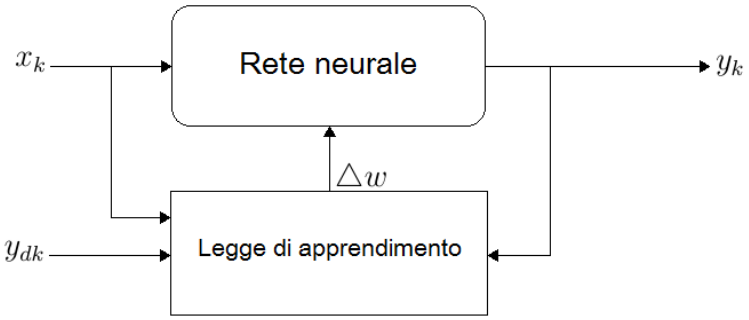

Figura 2.1: Apprendimento supervisionato

Apprendimento non supervisionato

Nell'apprendimento non supervisionato la rete modifica i pesi autonomamente, si auto-organizza. Viene fornito solo il training set senza precisare le uscite.

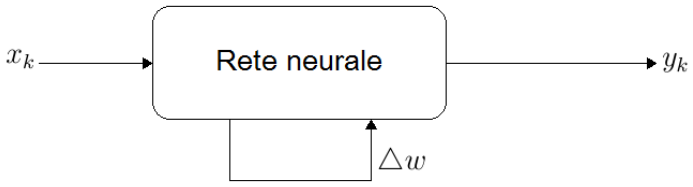

Figura 2.2: Apprendimento non supervisionato

2.3 Processi di apprendimento

L'addestramento è un processo ad hoc dipendente dallo specifico problema trattato. Riportiamo alcuni processi comunemente usati.

2.3.1 Delta rule

Con regola delta, o *delta rule*, si intende un apprendimento con correzione dell'errore. Siano:
$\overline{x}_k = [x_1, x_2, ..., x_n]^T$ il vettore k-esimo degli ingressi,
$\overline{W}_k = [w_1, w_2, ..., w_n]^T$ il vettore dei pesi al k-esimo ingresso,
e siano y_k ed y_{dk} rispettivamente l'uscita ottenuta e l'uscita desiderata. Definiamo l'*errore* come:

$$\delta_k = y_k - y_{dk} \tag{2.1}$$

Allora se consideriamo la variazione del generico vettore dei pesi $\overline{W}_k$ si ha che:

$$\triangle \overline{W}_k = \eta \overline{x}_k \delta_k \tag{2.2}$$

Il numero η che compare nell'equazione (2.2) è compreso tra 0 ed 1, prende il nome di **learning rate** ed indica la velocità di apprendimento del neurone. Questa regola modifica in maniera proporzionale solo i pesi delle connessioni che hanno contribuito all'errore.

L'algoritmo è il seguente:

1. se $y_k = y_{dk}$ nessuna modifica dei pesi
2. se $y_k \neq y_{dk}$ allora $\triangle \overline{W}_k = \eta \overline{x}_k \delta_k$.

2.3.2 Apprendimento con correzione di errore o metodo discesa del gradiente

Se indichiamo con y_k una risposta generata da un segnale di stimolo x in un tempo t ed indichiamo con y_{dk} la risposta desiderata, avremo un *segnale di errore* che può essere così espresso:

$$\delta = y_{dk} - y_k \tag{2.3}$$

Tale segnale dà il via ad un meccanismo di controllo che va ad applicare una sequenza di modifiche ai pesi sinaptici del neurone interessato al fine di avvicinare la risposta ottenuta a quella desiderata.

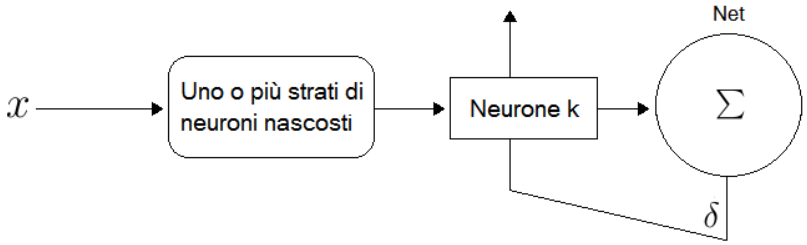

Figura 2.3: Apprendimento con correzione

Questo processo di ricerca dei pesi migliori si basa sulla scelta di pesi che minimizzano una funzione errore $E(\overline{W})$, costruita al variare dei pesi stessi. Per la scelta di tali pesi si sfruttano le informazioni fornite dal gradiente locale della funzione errore costruita.
Se con $\nabla_w E(\overline{W})$ indichiamo il gradiente della funzione, allora:

$$\Delta W \propto \nabla_w E(\overline{W}) \tag{2.4}$$

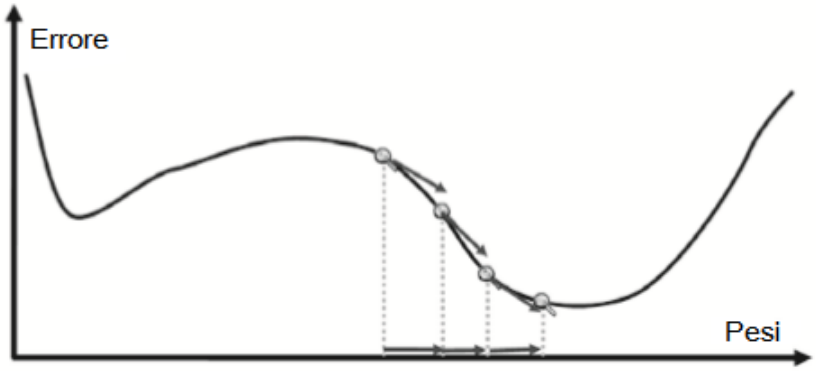

Figura 2.4: Metodo della discesa del gradiente

La ricerca risulta quindi essere guidata in modo proporzionale e dato che:

$$\frac{dE(\overline{w})}{d\overline{w}} = \nabla_w E(\overline{w}) = (\frac{\partial E(\overline{w})}{\partial w_1}, \frac{\partial E(\overline{w})}{\partial w_2}, ..., \frac{\partial E(\overline{w})}{\partial w_n})^T \tag{2.5}$$

si avrà:

$$\triangle \overline{W_k} = \eta \frac{dE(\overline{w})}{d(\overline{w})} \tag{2.6}$$

Se consideriamo la k-esima uscita y_k e la rispettiva uscita desiderata y_{dk} ed indichiamo con g la funzione di attivazione:

$$\begin{cases} y_k = g(net_k) \\ \delta_k = y_k - y_{dk} \end{cases}$$

(2.7)

Si definisce l'**errore quadratico medio**:

$$E_k = \frac{1}{2}(y_k - y_{dk})^2$$

(2.8)

Ma allora se consideriamo l'i-esimo peso del k-esimo input si avrà che:

$$\triangle w_i = \eta \frac{\partial E_k}{dw_i}$$

(2.9)

e sviluppando:

$$\frac{\partial E_k}{dw_i} = \frac{\partial}{\partial w_i}[\frac{1}{2}(y_k - y_{dk})^2] = (y_k - y_{dk})\frac{\partial (y_k - y_{dk})}{\partial w_i} = (y_k - y_{dk})\frac{\partial y_k}{\partial w_i} =$$

$$= (y_k - y_{dk})\frac{\partial g(net_k)}{\partial w_i} = (y_k - y_{dk})\frac{\partial g(net_k)}{\partial net_k}\frac{\partial net_k}{\partial w_i} = \delta_k g'(net_k)x_i$$

(2.10)

Quindi il peso w_i sarà modificato:

$$\triangle w_i = \eta \frac{\partial E_k}{dw_i} = \eta \delta_k g'(net_k)x_i$$

(2.11)

2.3.3 Apprendimento di Hebbian

Come nella delta rule, definiamo:
$\overline{x}_k = [x_1, x_2, ..., x_n]^T$ il vettore k-esimo degli ingressi,
$\overline{W}_k = [w_1, w_2, ..., w_n]^T$ il vettore dei pesi al k-esimo ingresso,
e siano y_k ed y_{dk} rispettivamente l'uscita ottenuta e l'uscita desiderata.
Nell'apprendimento di tipo hebbiano viene modificato ogni valore del peso di una connessione in questo modo:

$$\triangle \overline{W}_k = \eta \overline{x}_k y_{dk} \tag{2.12}$$

con η learning rate.

I passi dell'algoritmo sono i seguenti:

1. se $y_k = y_{dk}$ nessuna modifica pesi
2. se $y_k > y_{dk}$ allora $\triangle \overline{W}_k = -\eta \overline{x}_k y_{dk}$
3. se $y_k < y_{dk}$ allora $\triangle \overline{W}_k = \eta \overline{x}_k y_{dk}$

2.3.4 Apprendimento competitivo

Tale apprendimento si basa su una vera a propria competizione tra i neuroni di uscita di una rete neurale per attivarsi in seguito ad uno stimolo. In un certo momento t può attivarsi un solo neurone che viene denominato **winners-takes-all**.
In tale processo è importare avere quindi un meccanismo che permetta ai neuroni di competere ed un valore che indichi il limite della "forza" di ciascun neurone.
Nella forma più semplice la rete neurale ha un solo strato di neuroni di uscita completamente connessi ai nodi di input e che possono essere connessi tra loro.

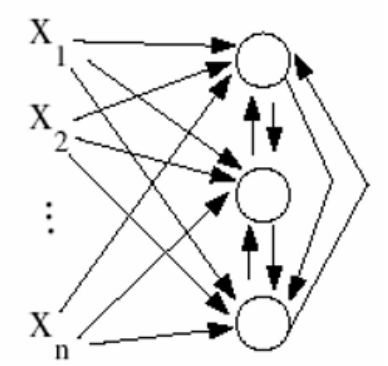

Figura 2.5: Struttura rete con apprendimento competitivo

Il neurone k che si attiva è quello con input netto ν_k più alto per un dato input x, con ν_k combinazione lineare di tutti gli input. Il suo segnale di output y_k sarà 1 mentre quello degli altri neuroni rimasti inattivi è 0.

$$y_k = \begin{cases} 1 & se\ \nu_k > \nu j \quad \forall j,\ j \neq k; \\ 0 & altrimenti \end{cases} \tag{2.13}$$

Ora se indichiamo con $w_{k,j}$ i pesi tra il neurone k e l'input j, supponendo che ogni neurone abbia un ammontare di peso sinaptico fisso $\sum_j w_{kj} = 1, \forall k$, il neurone che vince apprende spostando i pesi dagli input inattivi agli input attivi utilizzando tale regola:

$$\triangle w_{kj} = \begin{cases} \eta(x_j - w_{kj}) & se\ il\ neurone\ k\ vince; \\ 0 & altrimenti \end{cases} \tag{2.14}$$

2.4 Attività di apprendimento

La scelta del paradigma di apprendimento, e quindi la metodologia di addestramento di una rete neurale, avviene in base all'attività che la rete deve svolgere ovvero il problema che deve risolvere. Le reti neurali possono essere utilizzate come *classificatori* in grado cioè di riconoscere oggetti appartenenti a diverse categorie o possono essere responsabili dell'approssimazione di funzioni. Data una funzione nota per punti, di cui non si conosce la forma analitica, il compito della rete neurale è quello di approssimarla nel modo migliore possibile calcolandone il valore anche in punti diversi da quelli proposti in ingresso.

2.4.1 Associazione di pattern e memoria associativa

Per memoria associativa intendiamo un criterio di memorizzazione e recupero di informazioni attraverso l'associazione. Completamente ispirata al metodo di memorizzazione del cervello umano, consente il recupero dell'informazione sulle basi di una conoscenza parziale del suo contenuto senza conoscerne la locazione di memoria.
La memoria associativa si articola in due fasi:

- fase di memorizzazione: la rete viene addestrata per fare in modo che dei *pattern*, vettori di numeri reali, vengano memorizzati ed associati;
- fase di richiamo: si richiama dalla rete un pattern memorizzato a seguito della presentazione di una versione parziale o distorta di un pattern chiave.

Le *associazioni* possono essere di due forme:
- <u>*autoassociazioni*</u>: generalmente utilizzate nell'apprendimento non supervisionato, la rete neurale memorizza un insieme di pattern che vengono ripetutamente presentati. Tra essi ne viene presentato in particolare uno che risulta essere distorto o in forma parziale. L'obiettivo è che la rete riconosca il pattern e ne richiami la versione completa/originale precedentemente memorizzata.
- <u>*eteroassociazioni*</u>: ogni pattern è associato ad un altro pattern; se x_k è un generico pattern chiave associato al pattern memorizzato y_k, allora x_k opera in modo tale da richiamare y_k.

2.4.2 Riconoscimento di pattern

Tale processo si articola in due fasi: la prima fase consiste in una sessione di addestramento in cui alla rete vengono presentati ripetutamente un insieme di pattern specificando per ognuno la classe di appartenenza; nella seconda fase alla rete viene presentato un pattern mai mostrato precedentemente, ma appartenente ad una categoria di pattern che la rete ha memorizzato. La rete neurale sarà in grado di riconoscerne la categoria e classificare il pattern grazie ai dati precedentemente acquisiti.
Il sistema è diviso in tre spazi:

- lo *spazio dei dati* costituito da tutti i pattern, un generico pattern x identifica un punto dimensionale di tale spazio;
- lo *spazio delle caratteristiche* all'interno del quale sono presenti dei punti y;
- lo *spazio decisionale* multidimensionale suddiviso in regioni determinate dalla rete nella fase di addestramento, ognuna delle quali è associata ad una classe.

.

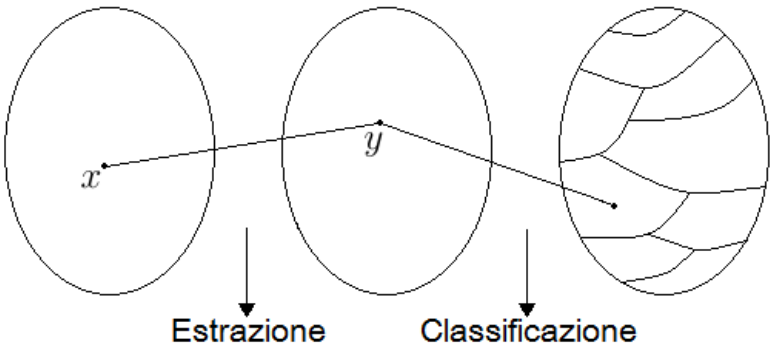

Figura 2.6: Classificazione di pattern

Il processo di riconoscimento inizia con l'estrazione delle *features* ovvero le caratteristiche che un generico pattern nello spazio dei dati deve rispettare. L'*estrazione* è una trasformazione che collega il pattern x con un punto y

nello spazio delle caratteristiche. Infine un'ultima trasformazione detta *classificazione* mappa dal punto y in una regione dello spazio decisionale.

2.4.3 Approssimazione di funzioni

Si prenda in esame una rappresentazione non lineare input/output descritta da $y = f(x)$ dove x è l'input e y è l'output e si consideri un insieme di esempi (x_i, y_i) per $i = 1, 2, ..., n$. L'incognita è la funzione f ed il problema consiste nel creare una rete capace di generare una rappresentazione F quanto più vicina alla funzione f. In senso euclidiano bisogna soddisfare la condizione

$$||F(x) - f(x) < \varepsilon|| \qquad \forall x$$

dove ε è un numero positivo piccolo, che risulterà essere sempre più piccolo con l'accrescere delle dimensioni e dei parametri liberi della rete.

2.5 Un esempio di apprendimento supervisionato: il percettrone elementare

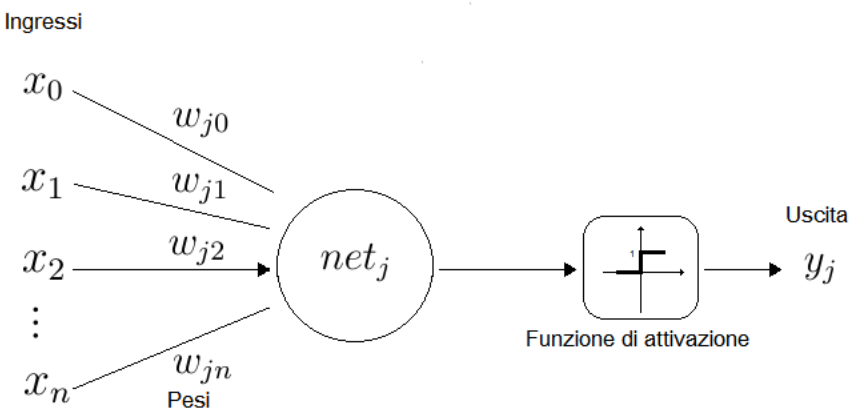

Figura 2.7: Il percettrone

Proposto da Rosenblatt, il percettrone è la forma più semplice di rete neurale e coincide con la struttura del neurone formale. Esso è costituito da un singolo neurone in cui confluiscono più ingressi i cui pesi sinaptici sono modificabili.

La funzione di attivazione è una funzione a gradino detta *funzione di Heavside*:

$$g(net_j) = \begin{cases} 1 & se\ net_j \geq 0 \\ 0 & se\ net_j < 0 \end{cases} \tag{2.15}$$

Il percettrone divide lo spazio di ingresso in due regione mediante l'iperpiano in $\Re^n$ di equazione:

$$w_0 + \sum_{i=1}^{n} w_i x_i = 0 \tag{2.16}$$

ed è quindi in grado di risolvere solo problemi *linearmente separabili*.
Supponiamo di voler classificare in due classi distinte C_1 e C_2, degli oggetti rappresentati mediante punti nel piano e di sfruttare $n = 2$ ingressi.
Se le due classi sono linearmente separabili, l'iperpiano è una retta che funge da separatore lineare tra le due classi. Un oggetto sarà quindi classificato in base alla parte di semipiano in cui il suo punto di rappresentazione cadrà.
Nel piano (x_1, x_2) degli ingressi della rete le classi C_1 e C_2 sono rappresentate da due semipiani separati dalla retta $x_2 = -\frac{w_1}{w_2}x_1 + \frac{w_0}{w_2}$ e possiamo classificare gli stimoli in questo modo:

$$\begin{cases} x \in C1 & se\ y = 1 \\ x \in C2 & se\ y = 0 \end{cases} \tag{2.17}$$

L'apprendimento utilizzato nel perceptron è di tipo supervisionato e la learning rule attraverso i quali si modificano i singoli pesi in modo appropriato

è la delta rule, definita in forma generale come:

$$\overline{W}_{k+1} = \overline{W}_k + \triangle\overline{W}_k \tag{2.18}$$

dove $\overline{W}_k = [w_1, w_2, ..., w_n]^T$ è il vettore dei pesi al k-esimo ingresso.

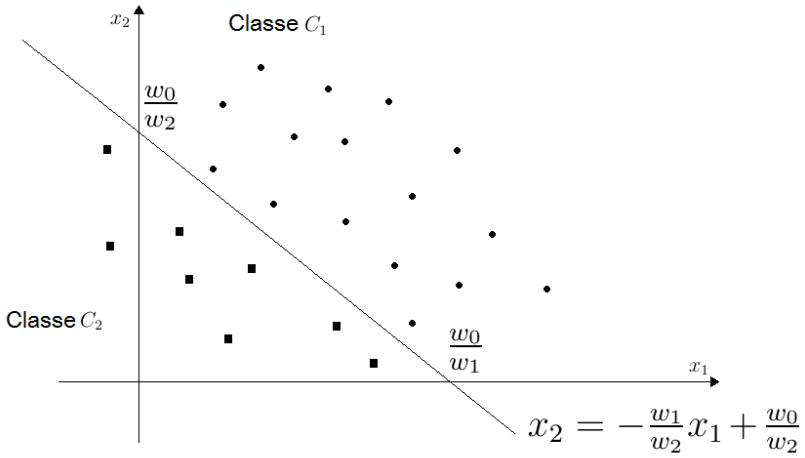

Figura 2.8: Separazione lineare dello spazio di ingresso di un Perceptron

I passi da eseguire sono:

1. si inizializzano i pesi con valori casuali;
2. si presenta un ingresso x_k ed il valore di uscita desiderato y_{dk};
3. si calcola la risposta y_k e si aggiornano i pesi attraverso la delta rule;
4. si ripete il ciclo dal passo 2 finché non si ottiene una risposta soddisfacente.

Con riferimento alla figura 2.8 si osservi che modificando i pesi w_1, w_2, w_0 si modifica la posizione e la pendenza della retta di separazione tra le due classi. Il processo termina quando la retta separa correttamente le due classi.

Capitolo 3

Reti di Hopfield

3.1 Presentazione ed utilizzo

Le reti di Hopfield furono presentate nel 1982 dal fisico John Hopfield nel suo articolo *"Neural networks and physical systems with emergent collective computational abilities"*. A seguito di numerose osservazioni sulle proprietà dinamiche ed elettrochimiche dei neuroni e delle loro interconnessioni, Hopfield cercò di fare emergere gli schemi di funzionamento che erano dietro ai comportamenti biologici elementari. Il modello matematico realizzato da Hopfield dimostrò che un gran numero di semplici elementi di elaborazione possedevano capacità computazionali dal *comportamento collettivo* e che tali capacità emergevano spontaneamente come conseguenza di interazioni tra le unità.
Le proprietà collettive di tale modello contraddistinguono tutt'oggi le reti di Hopfield dalle restanti e danno vita ad una memoria associativa per il riconoscimento di rappresentazioni alterate ed il recupero di informazioni mancanti. Per capire cosa si intende per memoria associativa, un esempio è la capacità del nostro cervello di riconoscere un'immagine anche quando questa non è esatta, risulta deformata o distorta o di riuscire a leggere parole con lettere mancanti e quindi non complete.
Le reti di Hopfield sono principalmente utilizzate:

- nella realizzazione di memorie associative;
- nella risoluzione di problemi d'ottimizzazione combinatoriale;
- nei problemi di classificazione.

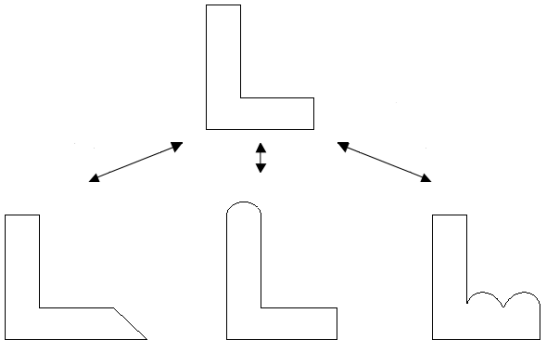

Figura 3.1: Rete di Hopfield utilizzata come memoria associativa: l'immagine deve essere sempre riconosciuta anche se leggermente distorta.

3.2 Architettura ed aggiornamento

La topologia delle reti di Hopfield è completamente diversa da quella delle reti a strati. Ogni unità è collegata a tutte le altre attraverso connessioni simmetriche, in termini di pesi sinaptici si ha $w_{i,j} = w_{j,i}$, $\forall i, j$. Sono quindi reti completamente connesse, in cui non si hanno autoconnession, $w_{i,i} = 0$, la cui matrice dei pesi sinaptici è simmetrica e sono ricorrenti.

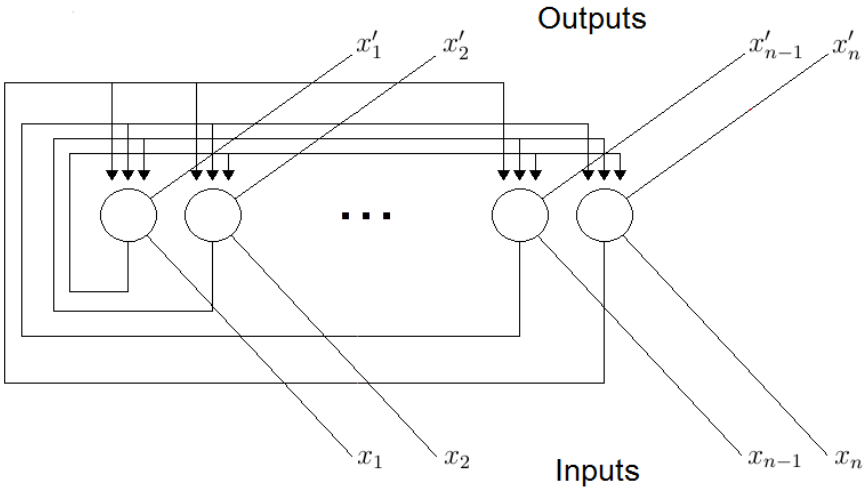

Figura 3.2: Esempio di una rete di Hopfield

I valori di input di ogni unità sono binari e non c'è distinzione tra unità di input ed unità di output in quanto la rete riceve i valori in entrata in numero pari alle unità che presenta, per poi farli fuoriuscire dalle stesse unità una volta elaborati. In questo modello, l'uscita di ogni neurone è collegata all'ingresso di tutti gli altri neuroni, formando così una topologia di rete a grafo completo unidirezionale.
Nella fase di apprendimento la rete memorizza un certo numero di informazioni dette *stati stabili* mentre nella fase operativa modifica il proprio stato sulla base dei pesi colleganti i vari neuroni: la rete itera fino a convergere su dei valori di output che rappresentano il pattern più simile all'input memorizzato dalla rete.

Il meccanismo di aggiornamento dei neuroni può essere diviso in:

- **aggiornamento asincrono** in cui si aggiorna un neurone alla volta;
- **aggiornamento sincrono** in cui tutti i neuroni sono aggiornati nello stesso istante;
- **aggiornamento continuo** in cui tutti i neuroni si aggiornano continuamente.

Abbiamo visto che il principio alla base della memorizzazione delle reti neurali è che l'informazione sia contenuta nei coefficienti di connessione. Nel modello di Hopfield si persegue tale obiettivo sfruttando la legge di Hebb, esposta nel capitolo 1 e che riprendiamo a grandi linee: se un'unità j manda un input ad un'altra unità i e se entrambe sono attive, il peso corrispondente $w_{i,j}$ deve essere rafforzato.
Supponiamo di voler memorizzare il set di stati v^s con $s = 1, ..., n$, utilizzando la notazione in cui $0 \Longleftrightarrow -1$ e $1 \Longleftrightarrow 1$.
I pesi delle connessioni sono fissati attraverso un apprendimento non iterativo *fixed point learning*, seguendo la seguente formula:

$$w_{i,j} = \sum_s v_i^s v_j^s \qquad i \neq j, \;\; i, j = 1, ..., n \tag{3.1}$$

dove $w_{i,j}$ è il peso della connessione dal nodo i al nodo j e v_i^s che può assumere i valori $+1$ o -1, è l'i-esimo elemento del set di stati.
Esempio: In forma vettoriale possiamo scrivere:

$$\overline{w} = \sum_s \overline{x_s} \cdot \overline{x_s}^T - 1 \tag{3.2}$$

Supponiamo di voler memorizzare il vettore $[+1, -1, +1, +1]$. La matrice dei pesi verrà calcolata seguendo questo procedimento

$$\begin{bmatrix} +1 \\ -1 \\ +1 \\ +1 \end{bmatrix} [+1, -1, +1, +1] - \begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & +1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & +1 & +1 \\ -1 & 0 & -1 & -1 \\ +1 & -1 & 0 & +1 \\ +1 & -1 & +1 & 0 \end{bmatrix} \tag{3.3}$$

3.3 Cenni sulla stabilità

La rete di Hopfield può essere vista come un sistema dinamico non lineare che evolve verso una configurazione stabile e la stabilità del sistema è strettamente legata ad una *funzione energia*. Prima di introdurre tale funzione, è bene dare qualche definizione in merito ai sistemi dinamici ed al concetto di stabilità vista in senso di Lyapuanov.
Il modello matematico per descrivere le dinamiche di un sistema non lineare è pensato in termini di **variabili di stato**. Tali variabili rendono possibile prevedere l'evoluzione del sistema a partire da un certo istante temporale utilizzando i valori che sono associati ad esse, poiché considerati sufficienti per la previsione dell'evoluzione. Definiamo **vettore di stato**:

$$x(t) = (x_1(t), x_2(t), ..., x_n(t)) \tag{3.4}$$

un vettore contenente le variabili di stato ed in cui la variabile indipendente è il tempo t mentre n è l'ordine del sistema. Sfruttando tali vettori è possibile

descrivere un sistema dinamico mediante un sistema di equazioni differenziali di primo ordine del tipo:

$$\frac{d}{dt}x(t) = F(x(t)) \tag{3.5}$$

con $F(x)$ **funzione vettore** che applicata ad un vettore di stato x ritorna un vettore che ha come j-esima componente $F_j(x_j)$ con F_j una qualche funzione. Tale equazione risulta inoltre descrittrice del movimento di un punto nello **spazio degli stati** n-dimensionale, spazio i cui punti rappresentano univocamente tutti e soli i possibili stati del sistema osservati a certi istanti t. Con lo scorrere del tempo il punto t descrive una curva detta traiettoria del sistema.
Definiamo **stato di equilibrio** un vettore costante $\overline{x}$ verificante la seguente condizione:

$$F(\overline{x}) = 0 \tag{3.6}$$

Uno stato di equilibrio $\overline{x}$ è **uniformemente stabile** se:

$$\forall \varepsilon > 0 \ \exists \, \delta > 0 : \ \|x(0) - \overline{x}\| < \delta \Rightarrow \|x(t) - \overline{x}\| < \varepsilon \tag{3.7}$$

In altre parole una traiettoria del sistema può variare in un piccolo intorno dello stato di equilibrio $\overline{x}$ se il suo stato iniziale è vicino a $\overline{x}$.

Teorema 1 (Lyapunov)**.** *Lo stato di equilibrio $\overline{x}$ è stabile se in un piccolo intorno di $\overline{x}$ esiste una funzione scalare definita positiva $V(x)$ tale che la sua derivata rispetto al tempo è minore o uguale a 0 in quella regione.*

Una funzione $V(x)$ che soddisfa queste proprietà è detta **funzione di Lyapunov** per lo stato di equilibrio $\overline{x}$.
Se consideriamo un sistema dinamico ad una sua condizione di equilibrio stabile corrisponde un minimo dell'energia posseduta dal sistema stesso come conseguenza del fatto che esso tende spontaneamente a minimizzarla. L'energia del sistema viene espressa come una funzione di Lyapunov verificante il teorema di cui sopra.

3.4 Il caso discreto e la funzione energia

La prima rete presentata da Hopfield era una rete discreta, completamente connessa, con aggiornamento asincrono completamente connessa e matrice dei pesi simmetrica.
In questo modello ogni neurone può assumere due stati $v_i = 0$ e $v_i = 1$. Lo stato istantaneo del sistema è specificato elencando gli n valori di v_i e lo stato di ciascun neurone ad un certo istante t è determinato da una formula di attivazione binaria a soglia che modificherà i valori input seguendo questo schema:

$$v_i^t = \begin{cases} 1 & se \sum_{i \neq j} w_{i,j} v_j^{t-1} > \theta_i \\ 0 & se \sum_{i \neq j} w_{i,j} v_j^{t-1} < \theta_i \end{cases} \tag{3.8}$$

dove θ_i è la soglia di attivazione associata al neurone i-esimo.
In modo equivalente, se f_h è una funzione non lineare a soglia:

$$v_i^t = f_h(\sum_{i \neq j} v_j^{t-1} - \theta_1) \qquad i = 1, ..., n \tag{3.9}$$

Questo processo è applicato ad un'unità per volta secondo un ordine stabilito o casualmente ed iterato fino a quando la rete non raggiungerà uno stato di equilibrio che corrisponderà ai valori di output.

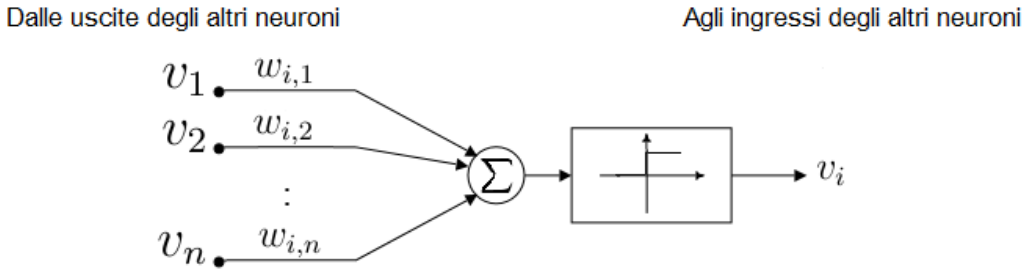

Figura 3.3:

Nel paragrafo precedente abbiamo detto che la rete di Hopfield può essere vista come un sistema dinamico non lineare che evolve verso una configurazione stabile ed abbiamo accennato ad una particolare funzione, la *funzione energia*. Questa risulta essere una funzione di Lyapunov definita nello spazio di stato e che viene minimizzata con l'evolvere del processo.
Tale funzione assume questa forma:

$$E = -\frac{1}{2}\sum_i \sum_j w_{i,j}\, v_i v_j + \sum_i \theta_i v_i \tag{3.10}$$

Essa è limitata inferiormente ed utilizzando un aggiornamento di tipo asincrono, risulta strettamente monotona decrescente lungo una qualsiasi traiettoria descritta dalla rete di Hopfield nello spazio di stato:

$$\Delta E(t) = E(t+1) - E(t) < 0 \tag{3.11}$$

il che sta a significare che la rete evolve verso una configurazione stabile. Essa risulterà inoltre convergente ad uno stato stabile corrispondente ad un minimo locale della funzione energia.
Lavorando con un aggiornamento di tipo asincrono sarà valutata solo una

generica unità k alla volta e può accadere che:

1. v_k non cambia stato;

2. v_k cambia stato $-1 \longrightarrow 1$, $1 \longrightarrow -1$.

Nel primo caso si ha $E(t+1) = E(t)$ e quindi la funzione rimane invariata. Nel secondo caso la funzione energia cambia secondo il nuovo valore v'_k. Si avrà:

$$E(v) - E(v') = (-\frac{1}{2}\sum_j w_{k,j}\ v_j v_k + \theta_k v_k) - (-\frac{1}{2}\sum_j w_{k,j}\ v_j v'_k + \theta_k v'_k) \tag{3.12}$$

Ma poiché $w_{k,k} = 0$ si ha:

$$E(v) - E(v') = -(v_k - v'k)(\sum_j w_{k,j}\ - \theta_k) \tag{3.13}$$

Osservando il secondo termine e considerando che l'unità cambia il suo stato, l'eccitazione ha un segno diverso che dipende da v_k e $-v'_k$.
Sfruttando la simmetricità della matrice dei pesi e il fatto che ha gli elementi diagonali nulli (3.11) è sempre positiva, infatti:

1. se v_k cambia stato e si ha $-1 \longrightarrow 1$, allora $v_k = -1 \Rightarrow v'_k = 1$, da cui $-(v_k - v'k) > 0$;

2. se v_k cambia stato e si ha $1 \longrightarrow -1$, allora $v_k = 1 \Rightarrow v'_k = -1$, da cui $-(v_k - v'k) > 0$.

Quindi $E(v) > E(v')$ ed allora l'energia diminuirà per ogni cambiamento. Inoltre poiché gli stati possibili sono in numero finito, la rete raggiungerà uno stato in cui l'energia non potrà diminuire ulteriormente e tale stato risulterà stabile, assicurando la convergenza della rete.

3.4.1 Il caso continuo

A differenza del caso discreto, la rete di Hopfield continua può assumere un numero infinito di stati sebbene sia formata da un numero finito di neuroni. Lo stato della rete all'istante t è sempre rappresentato da un vettore v composto dagli n stati v_i dei singoli neuroni, i cui valori sono compresi nell'intervallo reale $0, 1$ se utilizziamo una notazione unipolare, $-1, 1$ (notazione bipolare). Scriviamo lo stato v_i al tempo t come segue:

$$v_i(t) = kv_i(t-1) + \alpha \sum_{j=1}^{n} w_{i,j} F_j(t) \tag{3.14}$$

dove $k, \alpha \epsilon [0, 1]$ ed α è una **costante di attenuazione** mentre:

$$F_j(t) = \frac{1}{1 + e^{-\frac{v_i(t)}{T}}} \tag{3.15}$$

è una funzione di attivazione sigmoide.
La funzione energia assume questa forma:

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j}\, v_i v_j + \sum_{i=1}^{n} \frac{1}{\beta} \int_0^{v_i} F_i^{-1}(v) dv - \sum_{i=1}^{n} \theta_i v_i \tag{3.16}$$

Risulta anch'essa essere una funzione Lyapunoviana e quindi anche in questo caso si ha $\frac{dE}{dt} \leq 0$.

Bibliografia

[1] Prof. Bicego Manuele, *Riconoscimento e recupero dell'informazione per bioinformatica, reti neurali*

[2] Prof. Gambosi Giorgio, *Reti neurali, note dal corso di Machine Learning*, (2010), Packt Publishing

[3] Garreta Raùl, Moncecchi Guillermo, *Learning scikit-learn:Machine Learning in Python*, (2013)

[4] Prof.ssa Labonia Laura, *Storia delle reti neurali artificiali*

[5] Prof.ssa Lazzarini Beatrice, *Introduzione alle reti neurali*, (2015)

[6] Ing. Pioggia Giovanni, *Modelli di sistemi fisiologici*, (2009)