Alma Mater Studiorum · Universita di Bologna

Scuola di Scienza
Dipartimento di Fisica e Astronomia
Corso di Laurea in Fisica

Reti neurali artificiali e apprendimenti basati sulla biofisica dei neuroni

Relatore:
Prof. Gastone Castellani

Presentata da:
Davide De Paoli

Correlatore:
Dott. Nico Curti

Anno accademico 2019/2020

Abstract:

Frank Rosenblatt, padre del percettrone, nel 1962 sottolineò che l' obiettivo ultimo della ricerca nel campo delle reti neurali artificiali doveva essere "indagare le strutture fisiche ed i principi neurodinamici che stanno alla base dell'intelligenza naturale". A differenza di quel che desiderava Rosenblatt, i metodi maggiormente utilizzati e di conseguenza studiati sono quelli supervisionati, non biologicamente plausibili, ritenuti maggiormente efficaci rispetto alle reti che sfruttano metodi di apprendimento ispirati dalla biofisica dei neuroni. L' obiettivo della tesi è quello di analizzare due metodi di apprendimento non supervisionati basati su sistemi neuronali biologici: il modello di L.Bienenstock, N.Cooper e W. Mundro del 1982 (BCM) ed il modello di D.Krotov e J.Hopfield del 2019, e comprenderne le reali capacità. La prima parte della tesi rappresenta un' introduzione al concetto di rete neurale artificiale, al significato di rete multistrato e all' algoritmo di apprendimento della retro propagazione dell' errore, tipico delle reti supervisionate. La seconda parte della tesi illustra il funzionamento della BCM e della rete di Hopfield e Krotov. Nei risultati riguardanti la rete del 2019, viene riportato un confronto tra questo modello ed un modello addestrato tramite retro propagazione dell' errore, con cui riesce a competere nel riconoscimento delle immagini appartenenti a due data set: il MNIST ed il CIFAR-10. Per quanto riguarda la BCM, vengono riportati e discussi i risultati di alcune simulazioni effettuate con la rete modello BCM della libreria Plasticity [20]. Lo scopo delle simulazioni era quello di portare i pesi della rete a memorizzare il maggior numero di pattern differenti possibili, appartenenti al data set MNIST.

Indice:

Capitolo 1: Introduzione

Paragrafo 1.1
Nascita delle reti neurali e presentazione della tesi

Capire come l'ambiente che ci circonda può condizionarci rappresenta la chiave per cogliere quanto affascinante e complesso sia il nostro rapporto con il mondo esterno. È questo l´obiettivo con il quale nascono le prime reti neurali artificiali. Frank Rosenblatt, psicologo e ricercatore dell'Università di Cornell, nel 1956 ideò la prima macchina in grado di simulare il funzionamento di un neurone: il percettrone era una semplice copia artificiale di tali cellule, capace di ricevere ed inviare impulsi elettrici. L´idea prendeva spunto dai lavori antecedenti di McCulloch e Pitts, entrambi specializzati in neuroscienze, che nel 1943 pubblicarono un articolo per mostrare come un semplice sistema di neuroni artificiali potesse essere in grado di eseguire delle funzioni logiche basilari [1].Nella prefazione al suo libro "Principles of Neurodynamics" pubblicato nel 1962 [2], Rosenblatt scrive: "*Per il sottoscritto, il programma del perceptron non ha lo scopo di inventare dispositivi per l'intelligenza artificiale, ma piuttosto di indagare le strutture fisiche e i principi neurodinamici che stanno alla base dell'intelligenza naturale. Un perceptron è prima di tutto un modello cerebrale, non un'invenzione per la pattern recognition*". Le reti neurali si sono dimostrate negli anni utili e versatili, capaci di affrontare alcuni problemi con maggior efficienza rispetto ad altri metodi computazionali. Questo ha portato ad un´inevitabile ricerca della performance piuttosto che della stretta somiglianza con le reti naturali. L´ obiettivo per molti non era più quello di studiare il sistema nervoso modellizzandolo con una rete artificiale, bensì quello di rendere le macchine sempre più veloci ed abili nello svolgere i compiti ad esse assegnati. Nonostante questo, la ricerca di algoritmi che spiegassero come si sviluppano le sinapsi del sistema nervoso non cessò. Per esempio, nel 1982 L.Bienenstock, N.Cooper e W. Mundro pubblicarono un articolo riguardante lo sviluppo della selettività nella corteccia visiva primaria di alcuni mammiferi superiori, come gatti e scimmie [3]. Questo modello, che può essere identificato tramite la sigla BCM, imita ed ottiene i risultati relativi allo sviluppo di un neurone appartenente a questa zona. Nel 2019, D.Krotov e J.Hopfield presentarono un altro modello che sfrutta l' utilizzo di un sistema di inibizione laterale tra neuroni appartenenti allo stesso livello, giungendo ad un sistema capace di imparare senza alcuna supervisione esterna [4]. Tale sistema, oltre ad emulare lo sviluppo dei neuroni in modo simile al caso biologico, ottiene anche ottimi risultati in termini di prestazioni. In questo modo, la rete di Hopfield e Krotov risulta plausibile da un punto di vista biologico ma, in certi casi, performante quasi quanto le reti supervisionate. La ricerca di reti che descrivono lo sviluppo dei neuroni nel sistema nervoso si dimostra quindi interessante anche per coloro che desiderano semplicemente creare macchine con prestazioni sempre migliori.

Con queste premesse, i capitoli della tesi sono sviluppati nel seguente modo. Il primo capitolo contiene un' introduzione al mondo delle reti neurali artificiali, presentando fin da subito la struttura del sistema nervoso da cui derivano. Poi viene data una panoramica sulle caratteristiche

principali di una rete artificiale, sui diversi metodi di apprendimento e sui compiti più comuni che una rete può dover affrontare. Il secondo capitolo tratta da prima il percettrone di Rosenblatt, in particolare quando è presente un solo neurone, per poi analizzare l' algoritmo della retro propagazione dell' errore ed approfondire in questo modo il concetto di reti artificiali multistrato supervisionate. Infine, viene presentato l' esempio delle reti ricorrenti, un modello di rete multistrato che sfrutta la retro propagazione dell' errore come regola di apprendimento. La maggior parte delle informazioni per la stesura del primo e del secondo capitolo traggono ispirazione dal libro "Neural Networks and Learning Machines" di S.Haykin [5]. Per concludere, il capitolo tre introduce due reti non supervisionate già citate: la BCM ed il modello di Hopfield e Krotov del 2019. Poi vengono presentati i risultati ottenuti dal sottoscritto durante dei test sulla BCM mirati a massimizzarne l' efficienza, e viene fatta chiarezza sulle cause del limite riscontrato. Infine viene riportato un confronto tra la proposta del 2019 e reti allenate in modo supervisionato, per verificare quanto tale modello riesca a competere con queste ultime.

Paragrafo 1.2
Struttura e funzionamento del sistema nervoso

Le reti neurali artificiali simulano la struttura alla base del sistema nervoso. Di conseguenza, dare una panoramica sulla sua conformazione può aiutare a comprendere meglio i concetti che verranno affrontati nei prossimi paragrafi. Le informazioni raccolte nel seguente paragrafo sono state raccolte dalle fonti [6] e [7].
Il sistema nervoso possiede la capacità di trasferire informazioni attraverso segnali elettrici, emessi e ricevuti dai neuroni che ne fanno parte. Può essere visto come un sistema a tre stati (Fig.1.1): i recettori convertono gli stimoli sensitivi che il corpo ha ricevuto dall´ambiente in impulsi elettrici che raggiungono il secondo step, la rete neurale. La rete rielabora tali segnali e genera altri impulsi che raggiungono gli effettori. Questi ultimi convertono tali informazioni in azioni.

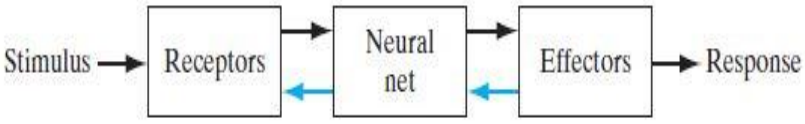

Fig.1.1: Il diagramma a blocchi raffigura il funzionamento alla base del sistema nervoso. Gli stimoli vengono captati dai recettori e convertiti in impulsi elettrici, rielaborati dalla rete ed infine convertiti in azione dagli effettori.

La struttura di un neurone può essere suddivisa in tre parti (Fig. 1.2) : la dendrite, composta da fibre molto ramificate, rappresenta la parte di ingresso per gli stimoli elettrici provenienti da altri neuroni; il soma, il corpo cellulare, raccoglie i segnali ricevuti dai singoli dendriti; l´assone, utilizzato per mandare segnali elettrici ad altre cellule attraverso il potenziale d´azione.

In una situazione di rilassamento della cellula il potenziale misurato tra l´interno e l´esterno della cellula, detto potenziale a riposo, possiede un valore appartenente all' intervallo $[-40\ mV, -90mV]$. Nel momento in cui il corpo cellulare viene sufficientemente stimolato, lungo l´assone si genera una differenza di potenziale di circa $+50\ mV$, per un periodo molto breve di $1-2\ ms$. Questo potenziale genera l´impulso elettrico che si propaga lungo l'assone**,** diretto verso un´altra cellula.

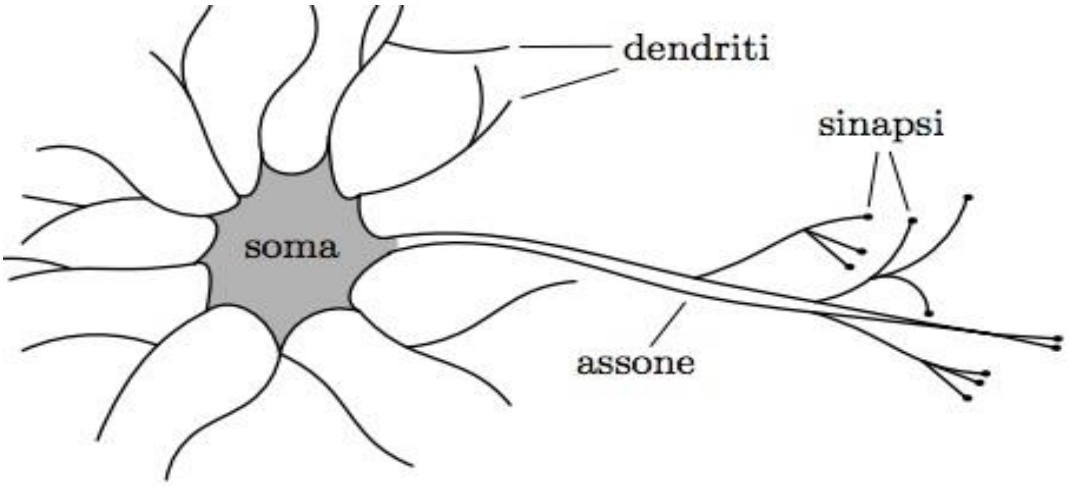

Fig.1.2: L' immagine rappresenta la struttura di un neurone dove sono evidenziate le componenti principali che formano la cellula: il corpo cellulare detto soma, i dendriti, l' assone e le sinapsi.

La differenza di potenziale si genera per via della differenza di concentrazione ionica tra l´interno e l´esterno della cellula. Tali distribuzioni di ioni vengono gestite da due fattori:

- dalle pompe ioniche (per lo più sodio-potassio), proteine interne alla membrana che permettono il passaggio di ioni contro gradiente da una parte all´altra di essa;
- dalla permeabilità selettiva della membrana che concede solo ad alcuni ioni di passare attraverso i canali ionici per spostarsi dove la concentrazione di quelle molecole è minore.

In questo modo si passa da un potenziale di riposo ad un potenziale d´azione con il semplice movimento di cariche dall´interno all´esterno del neurone e viceversa. Il potenziale d´azione è un segnale localizzato (Fig.1.3), che si genera dove l´assone si dirama dal soma e si propaga per tutta la sua lunghezza. Inoltre la sua ampiezza non dipende dall´ intensità di corrente che lo ha provocato. Un' altra sua caratteristica è di essere un segnale del tipo tutto o nulla, quindi o si presenta con un $\Delta V \cong 100\ mV$, oppure non si presenta affatto.

Un´altra componente fondamentale del sistema nervoso sono le giunzioni tra dendriti ed assone, dette sinapsi (Fig.1.4). Il segnale dell´assone produce nella fessura sinaptica il rilascio di un neurotrasmettitore chimico che permette al potenziale d´azione di trasferirsi al dendrite, e quindi al segnale elettrico di raggiungere il soma del neurone postsinaptico. Infatti il neurotrasmettitore rilasciato viene intercettato da proteine di membrana recettoriali che, esattamente seguendo il procedimento precedente, generano un segnale elettrico che percorre il dendrite fino al soma del

neurone ricevente. Il potenziale postsinaptico può essere di tipo eccitatorio o inibitorio in base al tipo di sinapsi (e quindi di mediatore chimico).

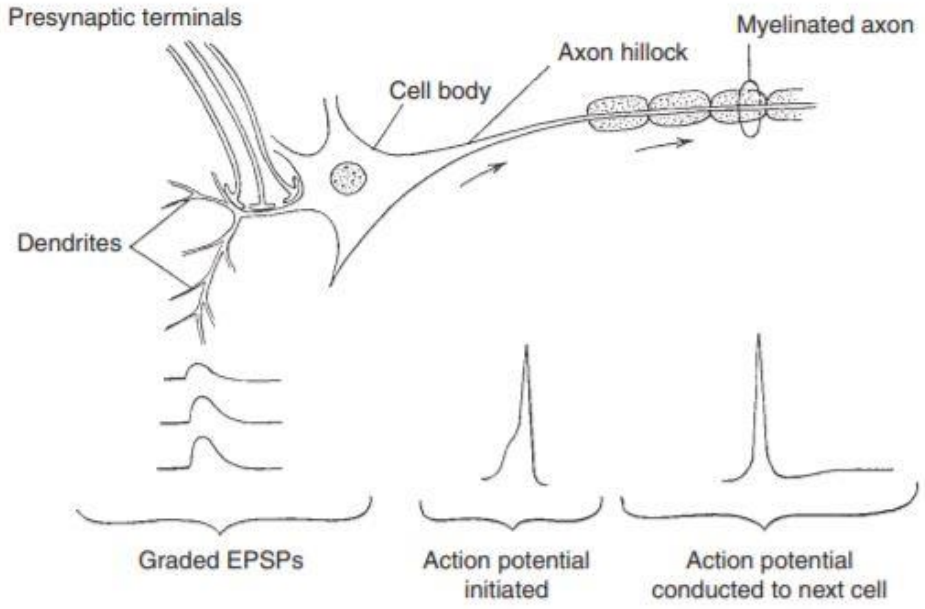

Fig.1.3: L´immagine raffigura la propagazione del potenziale d´ azione lungo le componenti principali del neurone.

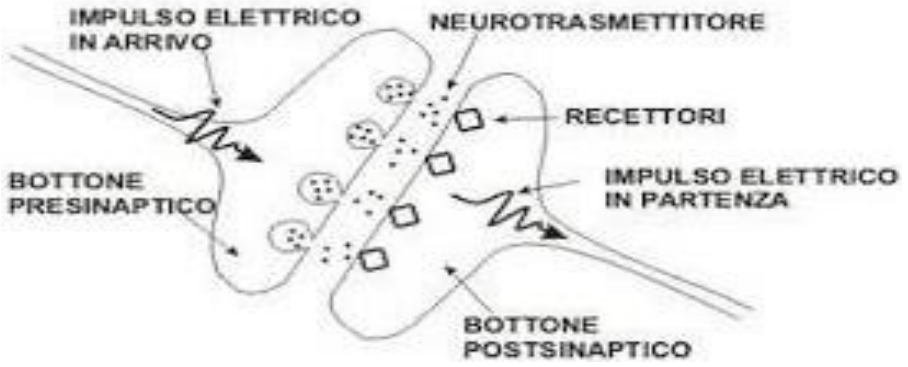

Fig. 1.4: L' immagine rappresenta una sinapsi e le sue principali componenti pre e post-sinaptiche. Inoltre, descrive l' intero processo di transizione del segnale elettrico attraverso la sinapsi, evidenziando la presenza dei neurotrasmettitori e dei recettori.

Le sinapsi sono la chiave per comprendere come si genera la memoria. Lo psicologo D. Hebb nel 1949 pubblica "The Organization of Behavior" [8], un libro che cerca di spiegare i meccanismi alla base dell´apprendimento e specialmente la memoria a lungo termine. Hebb ritiene che l´ apprendimento del sistema nervoso consista in una modifica delle sinapsi come risposta ad ogni stimolo da esso ricevuto. Il meccanismo ipotizzato da Hebb rispetta la seguente regola: "Quando un assone della cellula A è abbastanza vicino da eccitare la cellula B e prende parte ripetutamente e in modo persistente al fatto che essa emetta impulsi, in una o in entrambe le

cellule si verifica un qualche processo di crescita o di mutamento metabolico, tale che aumenta l'efficienza di A nell'attivare l'emissione dell'impulso da parte della cellula B."
In sostanza, se due neuroni possiedono un´ attività fortemente correlata, la sinapsi viene potenziata (e l´efficienza aumentata); in caso contrario se interagiscono di rado, la sinapsi si indebolisce (e l´efficienza si riduce). Definiamo queste azioni con le parole "potenziamento" o "depressione" a lungo termine.

Paragrafo 1.3
Le reti neurali artificiali

Quando si parla di reti neurali artificiali, si fa riferimento a algoritmi che cercano di imitare il funzionamento del sistema nervoso ed il suo approccio ad alcuni problemi o compiti specifici. Tali algoritmi sono composte da tante unità capaci di svolgere compiti semplici, i "neuroni", collegati tra loro da numerose connessioni, le "sinapsi". La struttura nella sua totalità ottiene abilità molto superiori a quelle del singolo neurone, esattamente come possiamo osservare in natura. Nel manuale del 1990 di I.Aleksander e H.Morton intitolato "An introduction to neural computing"[9], viene introdotta la seguente definizione:

"Una rete neurale è un processore massivo strutturato in parallelo e formato da tanti semplici processori unitari, con una naturale propensione ad immagazzinare informazioni e renderle disponibili per essere utilizzate. Assomiglia al cervello per due aspetti:
1) Le informazioni sono acquisite dall´ambiente attraverso processi di apprendimento.
2) Le forze delle connessioni tra i neuroni, anche chiamate "pesi sinaptici", vengono utilizzate per immagazzinare le informazioni acquisite".

Una rete neurale prima di svolgere il compito ad essa assegnato deve imparare come affrontare il problema che le verrà sottoposto. Il metodo di apprendimento della rete indica come la macchina lavora ed accumula le informazioni derivanti dall´esterno, per poi utilizzarle svolgendo così al meglio il proprio lavoro. La memoria di tali sistemi risiede nei pesi: dei valori assegnati alle connessioni tra i neuroni della rete. Questi valori possono cambiare durante la fase di apprendimento in base al metodo che sfrutta la macchina ed agli esempi o agli stimoli che riceve.
La struttura di un neurone può essere schematizzata come in figura (Fig. 1.5). Il segnale ricevuto, o segnale di input, viene rappresentato da un vettore $\boldsymbol{X} = (x_1, x_2, x_3, \dots)$ che associa ogni sua componente ad un diverso legame sinaptico con altri neuroni. I pesi sinaptici vengono raggruppati all´interno del vettore $\boldsymbol{W} = (w_1, w_2, w_3, \dots)$, dove ogni componente indica la forza della connessione tra il neurone ricevente il segnale di input ed il neurone che lo genera; il neurone combina linearmente i segnali di input ed i loro pesi, secondo la formula:

$$v_j = \sum_{j=1} x_j \times w_j + b_k \ . \qquad [1.1]$$

Il valore b_k rappresenta il bias. Il bias in certi casi risulta utile all´interno della regola di apprendimento che gestisce la rete, ma in altri casi viene semplicemente inserito come prima componente del vettore $\boldsymbol{X}$. Infine, la funzione di attivazione rielabora il risultato della sommatoria per generare il segnale di output da inviare al livello successivo della rete. Questa funzione ha anche lo scopo di evitare che si generino segnali troppo forti e difficili da gestire.

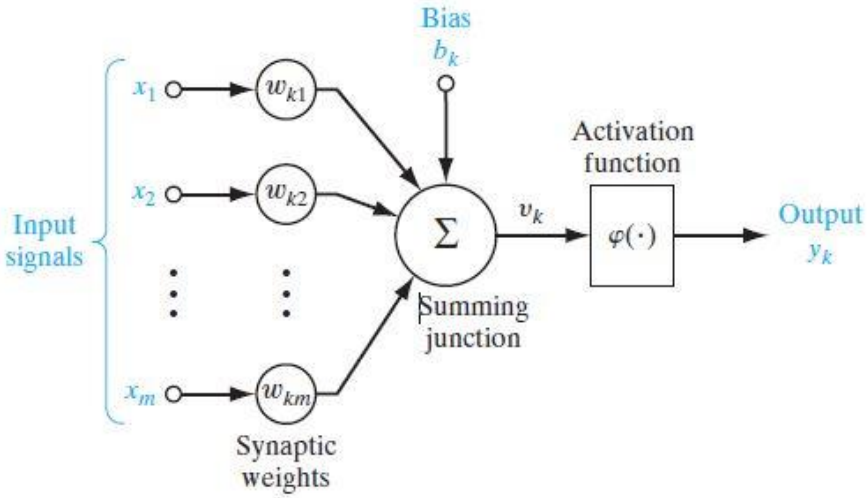

Fig.5 : Il diagramma rappresenta la struttura di un neurone artificiale. Sono quindi raffigurate a sinistra le componenti del segnale di input che raggiungono il neurone, per poi venir combinate linearmente con i rispettivi pesi sinaptici, indicati con la lettera w, e rielaborati dalla funzione di attivazione per generare il segnale di output. Nell' immagine viene anche indicata la presenza di un bias, che viene sommato al valore dato dalla combinazione lineare per ottenere il segnale da rielaborare tramite la funzione di attivazione.

Le reti neurali artificiali possono essere utilizzate per risolvere problemi differenti, ma la struttura base resta più o meno la stessa per tutti i possibili sistemi. Questo permette di sfruttare le stesse teorie e gli stessi algoritmi all´interno di macchine con scopi diversi. Per questo motivo è possibile sfruttare dei moduli, cioè dei blocchi di neuroni che sanno svolgere un determinato lavoro, concatenandoli insieme ad altri con lo scopo di svolgere compiti più complessi rispetto a quelli iniziali.

Paragrafo 1.4
I principali metodi di apprendimento

Come già detto nel capitolo precedente, durante la fase di apprendimento la macchina impara a svolgere il proprio compito. Durante questo fase i parametri che caratterizzano la rete neurale (solitamente si parla di pesi ma ce ne possono essere altri, come gli indici di selettività) si addestrano fino a farli convergere ad un valore. Questi parametri, una volta diventati stabili, permetteranno alla rete di immagazzinare le informazioni acquisite durante l'allenamento, per poterle poi utilizzare nel risolvere il problema che le verrà sottoposto. Una rete neurale biologica si sviluppa in base agli stimoli che riceve dall´ambiente, senza dover ricevere esempi generati su misura prima di cominciare a lavorare. Potremmo quasi dire che una rete di questo tipo impara direttamente sul campo. Fare in modo che una rete artificiale cresca nello stesso modo non è semplice, siccome non è permesso aiutarla durante l' apprendimento mostrandole la risposta che dovrà poi riprodurre. Sono infatti più numerosi i tipi addestrati definiti supervisionati, con pacchetti di dati che associano ad uno stimolo iniziale una risposta desiderata, così da istruire la rete su come comportarsi caso per caso. Segue più nello specifico una descrizione circa la differenza tra addestramenti supervisionati e non supervisionati. Le informazioni dei paragrafi seguenti sono state raccolte dalla fonte [9].

Paragrafo 1.4.1
Addestramenti supervisionati

Un´ addestramento supervisionato o anche detto "con maestro", mostrato in figura (Fig.1.6), consiste in uno sviluppo dei pesi sinaptici attraverso un processo di emulazione. Il maestro sa già come gestire i segnali di input dell´ambiente e come dovrebbe rispondere a questi stimoli il sistema da addestrare. Per allenare la rete neurale artificiale basta quindi inviarle segnali che simulino il problema che dovrà affrontare, ed aggiustare i pesi sinaptici tra i neuroni in modo che l´impulso di output sia sempre più prossimo a quello desiderato. Quindi lo scopo dell´apprendimento è minimizzare la differenza tra il segnale di output emesso dalla macchina e quello indicato dall´insegnante. Tale valore viene definito attraverso il segnale di errore . Per valutare quanto la macchina sia efficiente possiamo per esempio calcolare l´errore quadratico medio definito come una funzione dei pesi sinaptici. Tale funzione può essere raffigurata su uno spazio multidimensionale, dove i pesi sono le variabili, e ridefinita iper-superficie di errore. I minimi di tale iper-superficie rappresentano l´obiettivo dell´addestramento, ovvero i set di pesi per i quali la differenza tra il maestro e la macchina è la più piccola possibile. A questo punto diventa importante introdurre anche il gradiente della iper-superficie di errore, che esempio dopo esempio indica la direzione da seguire nello spazio dei pesi per avvicinarsi ai minimi della funzione errore quadratico medio.

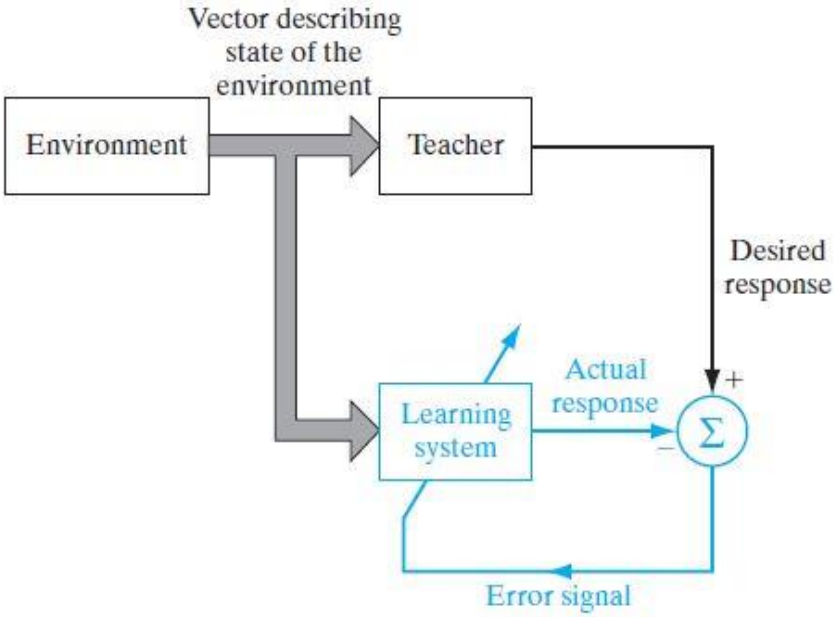

Fig.1.6: Lo schema rappresenta una rete neurale programmata per apprendere in modo supervisionato, con la presenza del maestro che permette alla rete di imparare quale output associare ad ogni segnale di input. Gli stimoli raggiungono sia la rete che il maestro. Le risposte di entrambi permettono poi di generare il segnale di errore, necessario alla rete per l' aggiornamento dei pesi.

Paragrafo 1.4.2
Addestramenti non supervisionati

Un addestramento senza maestro non necessita di esempi input-output preparati in precedenza come l´addestramento supervisionato . La rete impara semplicemente ricevendo degli stimoli dall' ambiente nel quale si trova a dover agire. Le tipologie di apprendimento senza maestro si dividono in addestramenti per rinforzo ed addestramenti non supervisionati. Le reti con apprendimento non supervisionato, o anche dette auto-organizzate, sono capaci di imparare a decodificare la regola statistica che si nasconde sotto agli input ricevuti, senza il bisogno di conoscere in precedenza le risposte giuste a tali segnali. Suzanna Becker nell´ottobre del 1990 pubblica un articolo (Usupervised Learning Procedures for Neural Network) dove spiega come le reti neurali non supervisionate siano ottime per stimare la componente principale di una distribuzione di input, oppure le N componenti più significative. Un altro importante compito che possono svolgere è quello di memorizzare alcuni schemi di input così da poterli richiamare quando necessario, oppure da poterli riconoscere o completare. I sistemi non supervisionati sono in grado di elaborare le Self-Organising-Map (SOM). Le SOM sono dei software capaci di convertire relazioni complesse e non lineari, relative ad un grande numero di dati, in relazioni geometriche molto più semplici. Nell´articolo "Engineering Applications of the Self Organizing Map" del 1996 [10], viene spiegato come la mappa tenda a mantenere le relazioni topologiche degli input. In pratica nella struttura della mappa possiamo andare a riconoscere le varie classi in cui la rete ha suddiviso gli stimoli ricevuti dall´ ambiente, e le relazioni che le legano. Per questo motivo,

se si utilizza una SOM non è tanto importante il segnale di output che la rete elabora, quanto la struttura che viene a formarsi all´interno della rete stessa.
Per quanto riguarda il secondo tipo di addestramento senza maestro, l´addestramento per rinforzo si basa su una continua interazione tra la macchina e gli esempi forniti attraverso delle azioni. Ad ogni azione corrisponde una ricompensa e la macchina cerca di massimizzare tale ricompensa volta per volta. Una ricompensa maggiore viene associata ad un´azione migliore che il sistema compie sull´ambiente in risposta ad un certo input. Nel libro "Reinforcement Learning: An Introduction", scritto da Richard S. Sutton e Andrew G. Barto [11], viene spiegato come una rete di questo tipo, progredendo nello sviluppo, in parte sfrutterà le conoscenze che già possiede per risolvere il compito assegnatole, in parte tenterà nuove azioni per continuare la ricerca della ricompensa più alta possibile. Viene anche sottolineato come sia inizialmente facile confondere l´addestramento per rinforzo con quello non supervisionato, ma la grande differenza si nasconde nell´intento della macchina. Nel primo caso il sistema cerca di schematizzare con una mappa le caratteristiche del set di input che riceve, mentre nel secondo caso l´unico obiettivo con il quale la macchina procede è quello di ricevere una ricompensa alta.

Paragrafo 1.5
I compiti più comuni per una rete neurale artificiale

Le reti neurali artificiali possono trattare diversi problemi, ed in base al compito da portare a termine il sistema sfrutta strutture e metodi di apprendimento su misura. Di seguito analizzeremo i compiti principali che possono esser assegnati ad una macchina di questo tipo.

La Pattern Association sfrutta il concetto di memoria associativa di una rete neurale per ricordare certi schemi di dati e poterli riprodurre. Con questo scopo, viene proposto alla macchina un set di dati in input, che indichiamo con il vettore $\boldsymbol{X_j} = (x_{j_1}, x_{j_2}, x_{j_3}, \dots)$, e fatto memorizzare un vettore di output $\boldsymbol{Y_j} = (y_{j_1}, y_{j_2}, y_{j_3}, \dots)$ corrispondente a X_j tale che :

$$\boldsymbol{X_j} \rightarrow \boldsymbol{Y_j} \text{ per ogni } j = 1, 2, 3, \dots q \qquad [1.2]$$

dove q è il numero di pattern differenti che la rete impara. I pattern che la rete può memorizzare non sono infiniti, quindi q deve esser più piccolo della capacità di memoria di queste macchine. L´associazione che permette il funzionamento di un sistema di questo tipo si differenzia in auto associazione ed etero associazione. Se si parla di auto associazione, alla rete viene proposto ripetutamente lo stesso schema di dati in input e la macchina deve imparare a ripeterlo, quindi in questo caso $\boldsymbol{X_j} = \boldsymbol{Y_j}$. Tale genere di rete sfrutta un apprendimento non supervisionato. Nell´ etero associazione, invece, ad ogni stimolo $\boldsymbol{X_j}$ viene richiesto in output un corrispondente $\boldsymbol{Y_j}$ diverso da $\boldsymbol{X_j}$. In questo caso l'apprendimento necessario è supervisionato. Dopo la prima fase di memorizzazione, la rete è in grado di gestire dei vettori $\boldsymbol{X_j}'$ con piccole differenze rispetto ad $\boldsymbol{X_j}$, associandoli al corretto $\boldsymbol{Y_j}$ secondo la relazione [1.2].

La Pattern Recognition consiste nell' assegnare ad un set di dati una classe piuttosto che un' altra con lo scopo di sintetizzarne le caratteristiche principali. Per essere allenata una rete di questo tipo necessita di input ripetuti con le caratteristiche relative ad una stessa classe. In questo modo impara ad estrarre dai segnali che riceve tali caratteristiche ed associarle alle classi che conosce. Introduciamo quindi il concetto di spazio delle decisioni, che rappresenta l' insieme delle regioni relative alle classi nelle quali la macchina colloca i dati ricevuti. Solitamente la struttura di queste macchine può essere sintetizzata in due reti (Fig. 1.7): la prima rete, non supervisionata, si occupa di estrarre le informazioni degli input ricevuti, trasformando il vettore iniziale in un' altro vettore che ne sintetizza le caratteristiche principali. Questa trasformazione consiste in una riduzione dimensionale, che permette di descrivere i segnali di input trasformati trattenendo solo una parte delle informazioni iniziali. La seconda rete, supervisionata, impara a classificare gli output della prima rete.

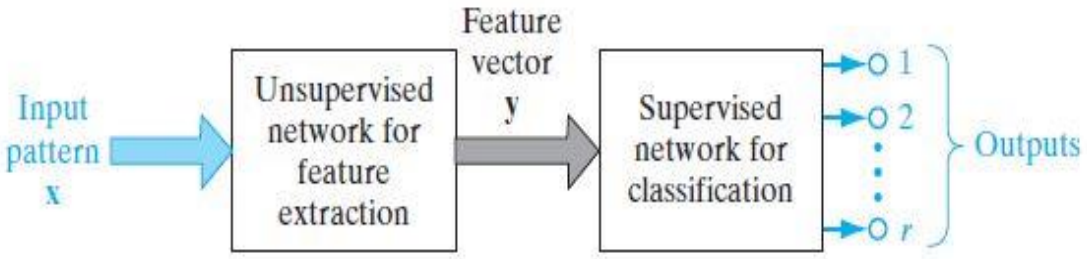

Fig.1.7: L' immagine raffigura la struttura principale di una rete neurale programmata per la Pattern Recognition. Il primo blocco si occupa di estrarre le caratteristiche principali degli esempi. Sfruttando i risultati ottenuti dalla prima parte, il secondo blocco impara in modo supervisionato a classificare gli esempi che gli vengono proposti nei set di dati.

La Function Approximation indica la capacità di una macchina di comprendere la funzione che lega i segnali di input ed output ricevuti. Quindi bisogna fornire alla rete un set di dati, indicati con un vettore $\boldsymbol{X} = (x_1, x_2, x_3, \dots)$, ed un set di output desiderati relativi ad $\boldsymbol{X}$ che definiremo $\boldsymbol{Y} = (y_1, y_2, y_3, \dots)$. Il compito della rete sarà quello di capire come sia fatta la funzione f, tale che:

$$f(\boldsymbol{X}) = \boldsymbol{Y} \qquad [1.3]$$

Tali reti approssimano le mappe input-output che generano durante la fase di apprendimento in funzioni F che descrivono le relazioni presenti tra i segnali in entrata ed in uscita dalla rete. F ed f devono essere legate dalla relazione:

$$F(\boldsymbol{X}) - f(\boldsymbol{X}) < \varepsilon\,, \qquad [1.4]$$

dove ε è un numero piccolo positivo.

Altri compiti, sul quale non è necessario soffermarsi per comprendere l´argomento della tesi ma che sono stati inseriti per completezza, sono ad esempio il Beamforming e quello di controllo. Il primo compito consiste nel distinguere le proprietà spaziali di un segnale che vogliamo analizzare e il rumore di fondo, proveniente da altre fonti, dal quale non possiamo isolarlo (lo stesso sistema

viene adottato in natura per esempio dai pipistrelli per l'eco localizzazione). Con il secondo compito si intende la gestione di alcuni sistemi controllando il loro stato ed intervenendo, se necessario, per mantenere le condizioni di lavoro del soggetto ottimali.

Capitolo 2: apprendimento supervisionato

Paragrafo 2.1
Il percettrone di Rosenblatt

Per parlare di reti formate da molti strati che interagiscono fra loro, come quelle analizzate nel terzo capitolo, è necessario comprendere bene il concetto di rete neurale multistrato. Per questo motivo, in questo paragrafo viene introdotto il percettrone di Rosenblatt, la prima e più semplice rete multistrato ideata durante il secolo scorso.

Secondo l' articolo del 1958 " The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain" scritto da F.Rosenblatt [12], il percettrone nasce come un modello del sistema nervoso, con lo scopo di illustrarne alcune fondamentali proprietà. Il percettrone rappresenta infatti il primo algoritmo capace di descrivere una rete neurale artificiale e quindi un tassello importante per la comprensione del funzionamento di una rete biologica. L' idea di Rosenblatt attinge molto da lavori precedenti come le fonti [8] e [13]. Le caratteristiche alla base del sistema nervoso che Rosenblatt voleva riprodurre possono essere sintetizzate in cinque punti:

- Le connessioni all' interno di un sistema nervoso coinvolte nell' apprendimento e nel riconoscimento non sono le stesse per tutti gli organismi. Inizialmente possiedono una natura randomica, poi nel tempo mutano in base alle esperienze accumulate.

- L' intero sistema possiede infatti una plasticità di base che permette alle connessioni di cambiare mentre la macchina esegue certe attività.

- L' applicazione di rinforzi positivi o negativi all' interno della rete genera una differenziazione delle connessioni tra neuroni più rapida.

- La somiglianza è un concetto alla base di un sistema di questo tipo, poiché permette ad alcune unità di rispondere a molti input differenti purché simili.

Rosenblatt per descrivere il suo algoritmo utilizzò l' esempio di un percettrone simile ad un neurone della corteccia visiva. La struttura di un fotopercettrone, quindi capace di ricevere input visivi, è composta da singole unità chiamate celle, distribuite in più strati e suddivise in quattro parti (Fig.2.1): la prima composta da unità sensoriali, la seconda e la terza rispettivamente chiamate "area di proiezione" ($A1$) ed "area di associazione"($A2$), e l' ultima formata da celle o gruppi di celle definite "risposte"($R1, R2, \dots RK$). Le unità sensoriali appartenenti alla retina, la prima zona del sistema ad entrare in azione, possono generare un impulso proporzionale al segnale di input ricevuto oppure seguire un metodo "tutto o niente", quindi generare un impulso non proporzionale a quello ricevuto solo quando quest' ultimo supera una certa soglia. Gli impulsi prodotti dalla retina raggiungono quindi l' area di proiezione, ed in particolare ogni segnale un set di celle, definito unità $A1$. Nel momento in cui l' impulso ricevuto da una unità $A1$ supera il suo

valore soglia, le celle generano un nuovo impulso sempre del tipo tutto o niente. I segnali vengono così rilevati dalle unità $A2$ appartenenti all' area di associazione. Le connessioni tra le celle delle aree $A1$ ed $A2$ sono totalmente randomiche. Infine, dalla zona $A2$ si passa all' ultimo livello che genererà l' output complessivo della macchina.

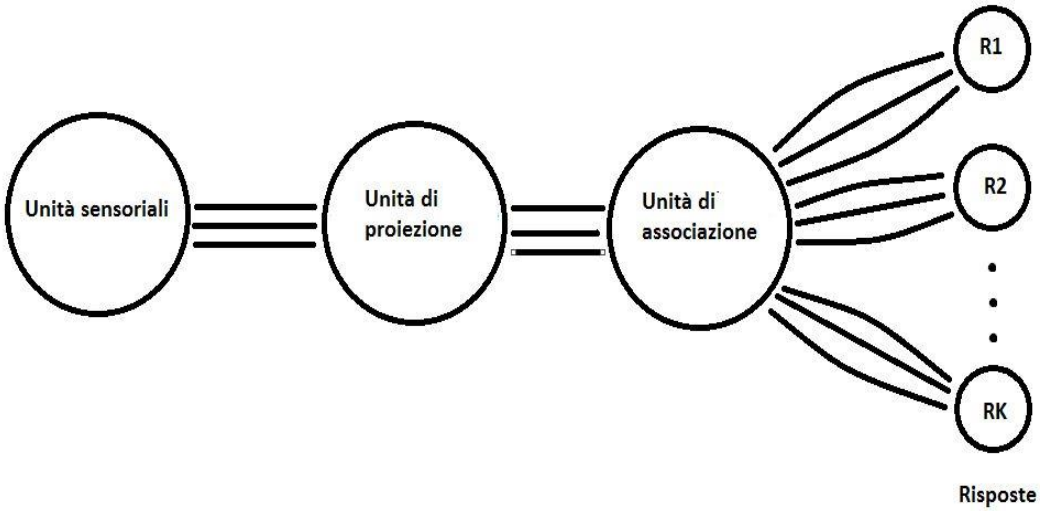

Fig. 2.1: L' immagine schematizza la struttura di un fotopercettrone, evidenziando le quattro parti principali in cui possiamo suddividere la rete.

Inizialmente le connessioni con le unità di risposta sono tutte simili tra loro, ma è a questo punto che entra in gioco il concetto di feedback. Gruppi di unità $A2$, dette sorgenti, mandano segnali alle rispettive unità di risposta, che ad ogni impulso ricevuto possono reagire secondo due differenti regole:

- Ogni risposta possiede un feedback di tipo eccitatorio verso le celle appartenenti alla sua sorgente che potenzia tali connessioni;
- Ogni risposta possiede un feedback di tipo inibitorio verso le celle non appartenenti alla sua sorgente che depotenzia tali connessioni.

In questo modo le risposte tenderanno ad escludersi a vicenda (Fig.2.2). Infatti se un set di sorgenti bersaglia sempre la stessa risposta, questa tenderà a concedere loro un vantaggio su tutte le altre sorgenti, che a loro volta riceveranno un vantaggio da altre risposte. Questa sorta di competizione tra unità rappresenta la base per numerosi metodi di apprendimento supervisionato e non supervisionato.

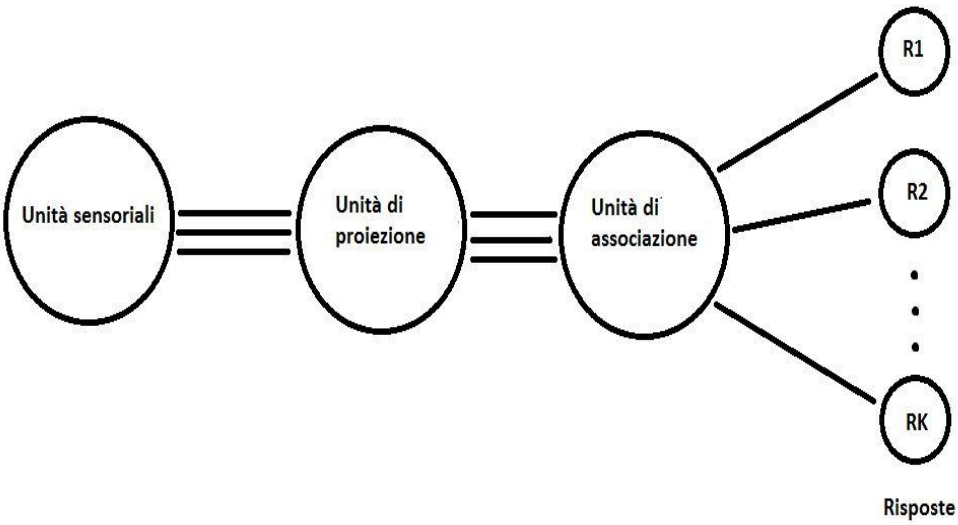

Fig. 2.2: In questa rappresentazione del fotopercettrone ogni risposta resta connessa ad una sola sorgente, per via dell' inibizione reciproca da parte delle unità nello strato precedente.

Nel seguito del paragrafo viene analizzato nello specifico un percettrone formato da un solo neurone, su modello di quello presentato nel 1943 da W.McCulloch e W.Pitts nell' articolo "A Logical Calculus of the Ideas Immanent in Nervous Activity" [1]. Il percettrone è formato da un corpo centrale che riceve gli stimoli dall' esterno tramite connessioni pesate. Possiamo indicare uno stimolo attraverso un vettore $\boldsymbol{X} = (x_1, x_2, \dots, x_N)$, dove ogni componente si riferisce ad una diversa connessione. Allo stesso modo indichiamo i pesi relativi ad ogni segnale di input con un altro vettore $\boldsymbol{W} = (w_1, w_2, \dots w_N)$. Il corpo centrale somma tutti i segnali che riceve nello stesso tempo attraverso la funzione [1.1], dove il bias rappresenta semplicemente un peso collegato ad un input costante pari a $+1$. Se la sommatoria pesata possiede un valore positivo, allora il percettrone produrrà un output pari a $+1$, in caso contrario l' output varrà -1. In questo modo la macchina è capace di suddividere gli stimoli ricevuti in due classi, indicate come C_1 e C_2, alle quali appartengono rispettivamente gli stimoli che generano un segnale positivo o negativo. Il tutto può essere raffigurato in un piano N-dimensionale che possiede su ogni asse una componente del vettore $\boldsymbol{X}$ con N pari al numero delle sue componenti (Fig. 2.3). Prima di chiedere alla macchina di compiere una classificazione però bisogna addestrarla in modo da ottenere l' output desiderato da ogni stimolo che riceve. Introduciamo un contatore n che indica il numero di esempi presentati alla rete. A questo punto, se le due classi sono linearmente separabili (Fig. 2.4), esiste un set di pesi $\boldsymbol{W}$ tale per cui:

- $\boldsymbol{W} \cdot \boldsymbol{X} > 0$ per ogni $\boldsymbol{X}$ appartenente a C_1;
- $\boldsymbol{W} \cdot \boldsymbol{X} \leq 0$ per ogni $\boldsymbol{X}$ appartenente a C_2.

Questo risultato può essere reinterpretato attraverso il Teorema di Convergenza con Incremento Fisso:

Ipotizziamo che C_1 e C_2 siano classi linearmente separabili e che gli input ricevuti dal percettrone appartengano a C_1 oppure a C_2. Il percettrone converge dopo n_0 esempi, cioè :

$$\boldsymbol{W}(n_0) = \boldsymbol{W}(n_0 + 1) = \boldsymbol{W}(n_0 + 2) = \cdots$$

è un vettore soluzione del sistema per $n_0 \leq n_{max}$, per cui tale che:

- $\boldsymbol{W}(n_0) \cdot \boldsymbol{X} > 0$ *per ogni* $\boldsymbol{X}$ *appartenente a* C_1*;*
- $\boldsymbol{W}(n_0) \cdot \boldsymbol{X} \leq 0$ *per ogni* $\boldsymbol{X}$ *appartenente a* C_2*.*

Un altro aspetto da considerare è la presenza del termine n_{max} introdotto durante la dimostrazione di tale teorema, che può essere recuperata all' interno del paragrafo 1.3 del libro di S.Haykin "Neural Networks and Learning Machines".

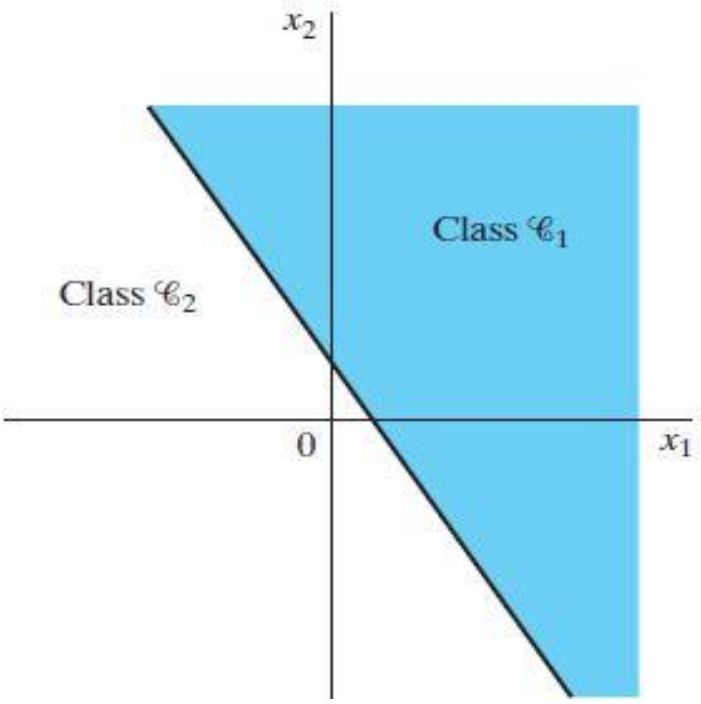

Fig. 2.3 : Su questo piano cartesiano sono rappresentate le due classi C_1 e C_2 che classificano gli elementi di un vettore di input $\boldsymbol{X} = (x_1, x_2)$.

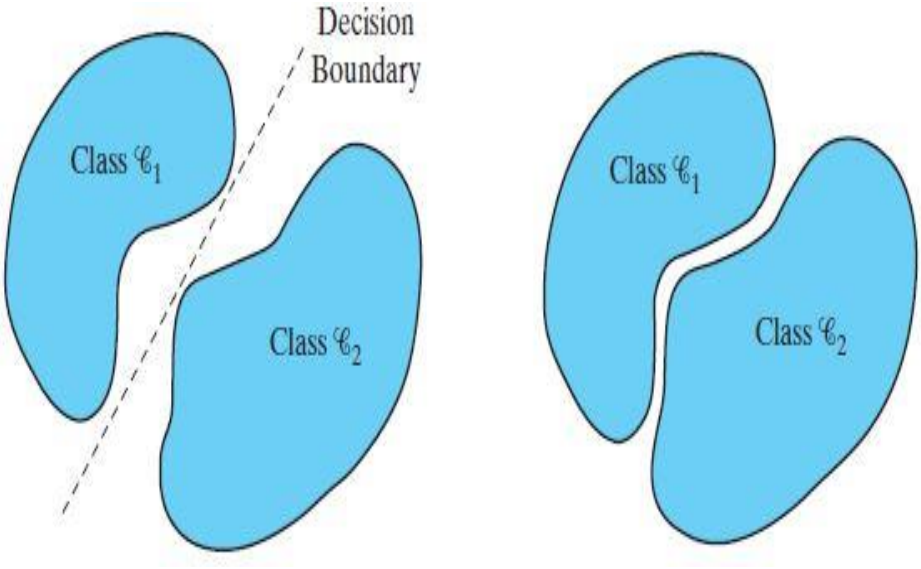

Fig. 2.4: In questa immagine sono rappresentate a sinistra due classi linearmente separabili per cui vale il Teorema di Convergenza con incremento fisso, a destra due classi non linearmente separabili.

Il meccanismo alla base della modifica dei pesi per ottenere $\boldsymbol{W}(n_0)$ sopra citato può esser sintetizzato nella seguente regola:

- Quando $\boldsymbol{W}(n) \cdot \boldsymbol{X}(n) > 0$ ed $\boldsymbol{X}(n)$ appartiene alla classe C_1 , oppure quando $\boldsymbol{W}(n) \cdot \boldsymbol{X}(n) \leq 0$ ed $\boldsymbol{X}(n)$ appartiene alla classe C_2, i pesi non variano fino al prossimo stimolo ricevuto dal percettrone e $\boldsymbol{W}(n) = \boldsymbol{W}(n+1)$;
- Quando $\boldsymbol{W}(n) \cdot \boldsymbol{X}(n) \leq 0$ ed $\boldsymbol{X}(n)$ appartiene alla classe C_1, allora i pesi sinaptici vengono modificati prima dello stimolo successivo nel seguente modo:

$$\boldsymbol{W}(n+1) = \boldsymbol{W}(n) - \eta(n) \cdot \boldsymbol{X}(n) \qquad [2.1]$$

mentre se $\boldsymbol{W}(n) \cdot \boldsymbol{X}(n) > 0$ ed $\boldsymbol{X}(n)$ appartiene alla classe C_2 utilizziamo:

$$\boldsymbol{W}(n+1) = \boldsymbol{W}(n) + \eta(n) \cdot \boldsymbol{X}(n). \qquad [2.2]$$

Introduciamo così il parametro rateo di apprendimento η già citato in precedenza, che indica con quale velocità ma allo stesso tempo con quale imprecisione il percettrone apprende. η può dipendere da n, e quindi variare nel tempo in funzione del numero d' iterazione corrente, ma nel caso proposto in seguito verrà assunto costante. Per raggiungere dei pesi che rispettino le relazioni precedenti, Rosenblatt propone un algoritmo definito di Convergenza del Percettrone, sintetizzato di seguito. I due vettori $\boldsymbol{X}(n) = (+1, x_{n_1}, x_{n_2}, \dots x_{n_N})$ e $\boldsymbol{W}(n) = (b,\ w_{n_1}, w_{n_2}, \dots w_{n_N})$ rappresentano rispettivamente lo stimolo numero n ed i pesi associati a tale stimolo. Come già anticipato il bias funge da peso per un segnale fisso di valore $+1$. I valori scalari $y(n)$ e $d(n)$ indicano la risposta data dalla macchina e la risposta desiderata. $d(n)$ vale $+1$ se $\boldsymbol{X}(n)$ appartiene alla classe C_1 , -1 se $\boldsymbol{X}(n)$ appartiene alla classe C_2.

Per cominciare il vettore dei pesi viene settato pari a 0. Successivamente, dopo ogni stimolo ricevuto dalla macchina viene ripetuto il seguente algoritmo:

$$\boldsymbol{W}(n+1) = \boldsymbol{W}(n) + \eta \cdot [d(n) - y(n)] \cdot \boldsymbol{X}(n) \qquad [2.3]$$

con lo scopo di variare W in modo tale che la risposta data dal percettrone si avvicini, iterazione dopo iterazione, a quella attesa. Una volta applicata la correzione dei pesi, il numero n viene aumentato di una unità ed il processo ripetuto fino ad arrivare ad un $\boldsymbol{W}(n_0)$ capace di soddisfare il Teorema di Convergenza con Incremento Fisso. Ovviamente se il segnale di output corrisponde a quello atteso, il termine moltiplicato per η si annulla ed i pesi restano invariati.

Paragrafo 2.2
La Retro Propagazione dell' errore

Uno degli algoritmi maggiormente utilizzati per addestrare le reti neurali multistrato in modo supervisionato è quello della retro propagazione dell' errore. In questo paragrafo verranno riassunti i meccanismi alla sua base, così da poterlo confrontare in seguito con i metodi non supervisionati che verranno introdotti nel terzo capitolo.

Secondo l' articolo del 2015 di Y. LeCun, Y. Bengio e G.Hinton intitolato "Deep learning" [14], la retro propagazione dell' errore non è altro che un' applicazione della regola della catena ad una "funzione errore" per definire come i parametri interni della rete devono essere modificati. Per comprenderla è necessario prima introdurre una serie di funzioni alla base di questo algoritmo e conoscere in cosa consista il metodo di apprendimento "On line". Si consideri una rete multistrato (Fig.2.5) suddivisa in 3 macro aree: un livello di input composto da unità sensoriali, diversi livelli definiti "nascosti" ed un livello di output, per un totale di M strati.

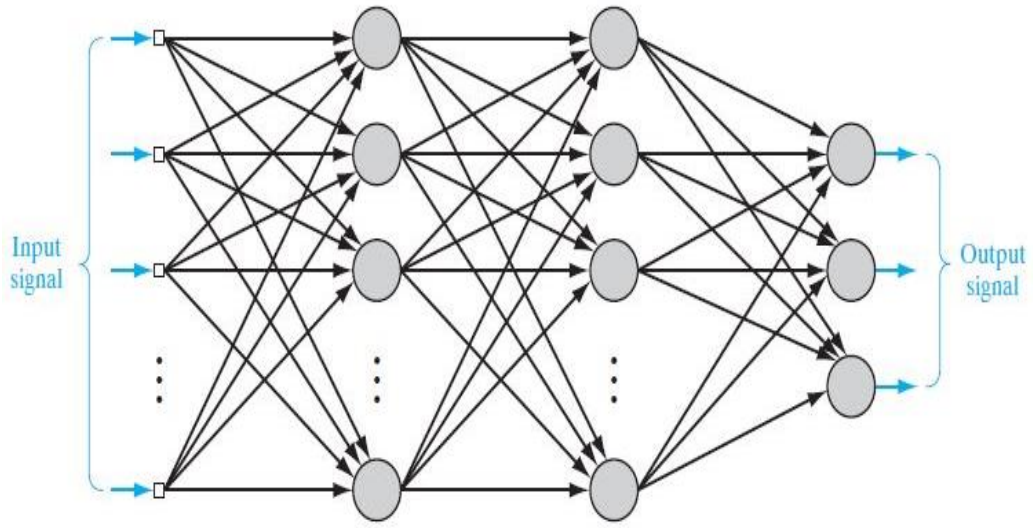

Fig.2.5: Lo schema rappresenta una rete neurale multistrato e ne evidenzia la struttura principale: il primo strato di input, i livelli interni alla rete ed infine uno strato di output.

Tale sistema è strutturato in modo tale che un qualsiasi neurone del livello j possieda connessioni con ogni neurone del livello $j+1$,dove j appartiene all' intervallo $[0, M-1]$. Inoltre l' intera rete viene attraversata da due differenti segnali: il primo consiste nell' insieme degli input che ogni strato riceve da quello precedente, mentre il secondo è relativo al segnale d' errore tra l' output ottenuto da ciascun neurone di un livello e quello desiderato, e percorre la rete in senso opposto al primo. Se la collezione di tutti gli N esempi proposti alla rete durante un' epoca di allenamento viene indicata come:

$$F = \{\boldsymbol{x}(n), \boldsymbol{d}(n)\}_{n=1}^{N} \quad [2.4]$$

dove $\boldsymbol{x}$ rappresenta il vettore di input e $\boldsymbol{d}$ il vettore output desiderato, il segnale di errore prodotto come output dal neurone j può esser rappresentato come:

$$e_j(n) = d_j(n) - y_j(n) \quad [2.5]$$

ricordando che n indica il numero dell' esempio a cui sono associati i valori presi in considerazione. L' energia istantanea d' errore si può definire come:

$$E_j(n) = \frac{1}{2} e_j^2(n) \quad [2.6]$$

e l' energia istantanea d' errore totale diventa:

$$E\,(n) = \sum_{j \in C} \frac{1}{2} e_j^2(n) \quad [2.7]$$

con C l' insieme dei neuroni che popolano il livello di output che genera il segnale d' errore. Il metodo di apprendimento on line consiste nel modificare i pesi sinaptici della rete da addestrare dopo ogni singolo esempio, cercando di minimizzare la funzione dell' energia istantanea d' errore totale E. In questo modo è possibile rilevare piccole differenze di E tra un esempio ed un altro e correggerle. Inoltre, conclusa un' epoca gli esempi in F vengono rimescolati prima di venir sottoposti nuovamente alla macchina. Correggere i pesi dopo ogni esempio evita che il riproporsi di due o più gruppi di dati identici nell' ordine in cui gli esempi vengono proposti alla rete renda ridondante e superflua la sessione di apprendimento.
Come già detto, lo scopo dell' algoritmo della retro propagazione è quello di trovare le correzioni migliori dei pesi per minimizzare $E\,(n)$. Tale procedimento deriva direttamente dalla regola della catena applicata all´energia istantanea d' errore totale. Considerando il caso in cui un singolo neurone j venga raggiunto da un segnale di input, la grandezza campo locale indotto v_j è definita come:

$$v_j(n) = \sum_{i=0}^{N} w_{ji}(n) \times y_i(n) \quad [2.8]$$

dove $w_{ij}(n)$ rappresenta il valore del peso sinaptico associato al neurone i dello strato precedente. Con N viene indicato il numero totale di neuroni appartenenti allo strato precedente a j. Il neurone j produce un segnale pari a:

$$y_j(n) = \varphi_j(v_j(n)) \quad [2.9]$$

con φ_j funzione di attivazione. L' obiettivo della retro propagazione è quello di valutare un $\Delta w_{ji}(n)$ appropriato, proporzionale alla derivata parziale di E rispetto a $w_{ji}(n)$, che corregga il peso della connessione tra i e j dopo l' esempio n. Grazie alla regola della catena possiamo ottenere la seguente espressione:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n) \times \partial e_j(n) \times \partial y_j(n) \times \partial v_j(n)}{\partial e_j(n) \times \partial y_j(n) \times \partial v_j(n) \times \partial w_{ji}(n)} \qquad [2.10]$$

e considerando:

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n)\ , \quad \frac{\partial e_j(n)}{\partial y_j(n)} = -1, \quad \frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'\big(v_j(n)\big), \ \frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n)\ , \qquad [2.11]$$

allora varrà:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n) \times \varphi_j'\big(v_j(n)\big) \times y_i(n). \qquad [2.12]$$

La correzione dei pesi tra il neurone i ed il neurone j si trova attraverso l' equazione:

$$\Delta w_{ji}(n) = -\eta \times \frac{\partial E(n)}{\partial w_{ji}(n)} = \eta \times e_j(n) \times \varphi_j'\big(v_j(n)\big) \times y_i(n)\ , \qquad [2.13]$$

con η rateo di apprendimento. La correzione dei pesi dipende quindi da $e_j(n)$, che a sua volta richiede di conoscere il vettore di output desiderato $\boldsymbol{d}(n)$. Se il neurone j appartiene all' ultimo livello della rete allora l' output desiderato è conosciuto, ma quando j si trova in uno degli strati nascosti, la situazione si fa più complessa. Definiamo il gradiente locale per un neurone j appartenente allo strato di output come:

$$\theta_j(n) = e_j(n) \times \varphi_j'\big(v_j(n)\big), \qquad [2.14]$$

per poi correggere tale gradiente con l' espressione:

$$\theta_j(n) = \varphi_j'\big(v_j(n)\big) \times \textstyle\sum_k \theta_k(n) \times\ w_{kj}(n). \qquad [2.15]$$

nel caso il neurone j appartenesse ad uno strato nascosto. I passaggi par raggiungere questa espressione sono riportati in appendice A. L' espressione corretta per ottenere la correzione dei pesi per quanto riguarda un neurone in un livello nascosto diverrà quindi:

$$\Delta w_{ji}(n) = -\eta \times \varphi_j'\left(v_j(n)\right) \times y_i(n) \times \textstyle\sum_k \theta_k(n) \times\ w_{kj}(n)\ . \qquad [2.16]$$

Quindi, dopo che il segnale sensoriale ricevuto dal primo strato di neuroni ha percorso tutta la rete, essa viene attraversata in senso opposto da un segnale di errore, per mezzo del quale secondo le formule [2.13] e [2.16] è possibile ricavare la correzione dei pesi da attuare prima del prossimo input. Questo procedimento viene ripetuto dopo ogni singolo esempio. η incide molto sulla velocità di apprendimento della rete: essendo direttamente proporzionale a $\Delta w_{ji}(n)$, maggiore sarà il rateo di apprendimento, maggiore diverrà la variazione dei pesi dopo ciascun esempio e più rapidamente si raggiungerà una situazione di minimo di $E(n)$. Il problema diventa

l' incertezza associata ad un η troppo elevato, che non mi permetta di minimizzare l' energia istantanea d' errore totale a sufficienza e generi un andamento dei pesi oscillatorio. Per evitare questa situazione è possibile correggere l' espressione [2.13] con la seguente:

$$\Delta w_{ji}(n) = \alpha \times \Delta w_{ji}(n-1) + \eta \times \theta_j(n) \times \mathrm{y_i(n)} \qquad [2.17]$$

dove viene introdotta la costante momento α. Questa rappresenta un' equazione differenziale al primo ordine e di conseguenza possiede una soluzione del tipo:

$$\Delta w_{ji}(n) = -\eta \sum_{k=0}^{n} \alpha^{n-k} \times \frac{\partial E(k)}{\partial w_{ji}(k)} \qquad [2.18]$$

Siccome la serie deve convergere bisogna che il momento sia compreso all' interno di un intervallo [0,1[. Inoltre, se le derivate $\frac{\partial E(k)}{\partial w_{ji}(k)}$ possiedono lo stesso segno, allora il processo di correzione dei pesi viene accelerato poiché aumenta il modulo di $\Delta w_{ji}(n)$, mentre se hanno segni diversi la discesa verso il minimo della funzione $E(k)$ viene rallentata. Un altro ostacolo che è bene affrontare nell' utilizzo della retro propagazione è la mancanza di un vero e proprio teorema di convergenza, simile a quello di cui si è già parlato nel paragrafo precedente riguardo il percettrone di Rosenblatt. Per questo è necessario introdurre una serie di criteri che permettano di capire quando è bene terminare l' algoritmo. Il primo criterio può essere espresso come segue:

" L' algoritmo di retro propagazione dell' errore viene considerato aver raggiunto dei pesi sinaptici soddisfacenti quando la norma Euclidea del vettore gradiente raggiunge una certa soglia abbastanza piccola."

Dobbiamo quindi ipotizzare in ogni caso sia presente un vettore dei pesi sinaptici w^* che possa essere interpretato come un minimo locale sulla superficie d' errore, nel quale il gradiente dell' energia istantanea d' errore totale soddisfi il criterio precedente. Un secondo criterio può esser così definito:

" L' algoritmo di retro propagazione dell' errore viene considerato essere giunto ad un punto di convergenza quando il rateo di cambiamento dell' errore quadratico medio per epoca è sufficientemente piccolo, all' incirca compreso tra lo 0.1% e l' 1%.".

Un ultimo punto su cui è necessario soffermarsi è la scelta da fare per quanto riguarda la funzione di attivazione $\varphi_j(v_j(n))$. Tra le migliori soluzioni si trovano le sigmoidi non lineari: sono funzioni generalmente continue e derivabili, con una derivata prima non negativa e dotata di un minimo locale ed un massimo locale. La prima forma di sigmoide proposta è la funzione logistica, di cui si parlerà anche più avanti nel corso della trattazione perché molto usata per addestrare sistemi multistrato.

Nel nostro caso può essere definita come segue:

$$\varphi_j\left(v_j(n)\right) = \frac{1}{1+e^{-av_j(n)}} \qquad [2.19]$$

con a che indica un parametro positivo. Una seconda proposta, usata altrettanto spesso nell' ambito delle reti multistrato ed inserita per completezza in questo paragrafo, è la sigmoide nella forma di tangente iperbolica:

$$\varphi_j\left(v_j(n)\right) = a \times \tanh(b \times v_j(n)) \qquad [2.20]$$

con a e b costanti positive. Prima di passare al paragrafo successivo è bene sottolineare un aspetto riguardante l' algoritmo appena analizzato: non è l' unico, e nemmeno il più preciso. Infatti la retro propagazione dell' errore indica solo un' approssimazione della correzione dei pesi migliore che si potrebbe trovare, ed inoltre è caratterizzata da un lento rateo di convergenza, soprattutto per reti molto ampie. Per comprendere meglio questa affermazione serve introdurre l' energia media di errore:

$$E_m\ (N) = \frac{1}{N} \times \sum_{n=1}^{N} E(n)\ . \qquad [2.21]$$

Provando a sviluppare in serie di Taylor E_m si passa a:

$$E_m\ (\boldsymbol{w} + \Delta\boldsymbol{w}) = E_m\ (\boldsymbol{w}) + \boldsymbol{\vartheta}^T(\boldsymbol{w}) \times \Delta\boldsymbol{w} + \frac{1}{2} \times \Delta\boldsymbol{w}^T \times H \times \Delta\boldsymbol{w} + \qquad [2.22]$$
$$(termini\ di\ grado\ superiore\ al\ secondo)$$

dove Δw è una correzione dei pesi dopo un' epoca mentre $\boldsymbol{\vartheta^T}(\boldsymbol{w})$ rappresenta il vettore gradiente locale trasposto, nella forma $\frac{\partial\ E_m(\boldsymbol{w})}{\partial \boldsymbol{w}}$ calcolato nel punto $\boldsymbol{w} = \boldsymbol{w}(n)$. H invece indica l' hessiana, che possiamo anche scrivere come $\frac{\partial\ E_m(\boldsymbol{w})^2}{\partial \boldsymbol{w}^2}$ in $w = w(n)$. A questo punto l' algoritmo di retro propagazione dell' errore darebbe una soluzione del tipo:

$$\Delta\boldsymbol{w}(n) = -\ \eta\ \times\ \boldsymbol{\vartheta}(n) \qquad [2.23]$$

compiendo un' approssimazione lineare della funzione E_m intorno al punto $\boldsymbol{w} = \boldsymbol{w}(n)$. In pratica si prendono così in considerazione i termini di primo grado e grado zero, escludendo gli altri. questo porta ad avere una convergenza dell' algoritmo molto lenta in particolare se le reti sono profonde. Introdurre come abbiamo già visto il termine momento velocizza il procedimento, ma per una soluzione ancora migliore bisogna tener conto almeno del terzo termine dello sviluppo. Il problema è che inserire questi termini nella soluzione non è affatto facile, ne a livello computazionale ne nelle condizioni che è doveroso aggiungere riguardo l' hessiana. Tenendo presente questi aspetti, la soluzione [2.18] non è perfetta ma sicuramente è un' ottima scelta in quanto la retro propagazione dell' errore:

- è semplice da un punto di vista computazionale;
- riesce ad ottenere un buon gradiente di discesa per quanto riguarda l' energia di errore se implementata in una modalità di apprendimento on-line.

Per i motivi appena citati, parlando di apprendimenti supervisionati, d' ora in avanti verrà sempre considerato un metodo di apprendimento che sfrutta la retro propagazione secondo l' algoritmo mostrato in questo capitolo.

Paragrafo 2.3
Un esempio di rete multistrato supervisionata: le reti ricorrenti

La retro propagazione dell' errore è, come già detto, uno dei metodi più comuni quando si parla di allenare una rete neurale multistrato in modo supervisionato. Esistono varie tipologie di reti neurali supervisionate addestrate in questa maniera, con caratteristiche che le differenziano le une dalle altre e di conseguenza adatte a svolgere una vasta gamma di compiti. Per completare il discorso riguardo le reti neurali multistrato supervisionate, di seguito verrà presentato un esempio di rete allenata tramite retro propagazione dell' errore: la rete ricorrente. Le informazioni sotto riportate sono una sintesi di diverse fonti: l' articolo di Y. LeCun, Y. Bengio ed G. Hinton "Deep Learning" [14], il libro di S. Haykin "Neural Networks and Learning Machines" [5] ed infine il libro "Deep Learning for NLP and Speech Recognition" scritto da U.Kamath, J.Liu e J.Whitaker [15]. Le seguenti pagine non pretendono di mostrare un' analisi dettagliata di questo modello, ma semplicemente completano la trattazione dell' algoritmo del paragrafo precedente riportandone un esempio di applicazione.

Per alcuni compiti in particolare è richiesto che la rete ricordi determinati pattern o sequenze che le vengono insegnate, in modo da poterli riprodurre o riconoscere. È il caso di sistemi capaci di completare parole o frasi utilizzati nei programmi di scrittura, per esempio, ma anche di programmi più complessi che possono collegare a certe parole o frasi un altro genere di oggetti come immagini o suoni. Abbiamo introdotto nel capitolo numero uno, tra i problemi più comuni che una rete può dover affrontare, il concetto di "Pattern Association" e di "Pattern Recognition ". Le Reti Neurali Ricorrenti (RNN) risultano fondamentali in entrambi i casi, grazie alla loro memoria. Infatti registrano i precedenti input ricevuti dalla rete ed elaborano i nuovi stimoli anche in funzione di queste informazioni, che vengono trattenute all' interno di strutture particolari meglio descritte in seguito. Per semplificare la descrizione di queste reti, introduciamo un vettore $\boldsymbol{X}$ contenitore di una serie di input che si susseguono nel tempo, quindi indicizzati con un numero discreto temporale tale che $\boldsymbol{X} = (x_1, x_2 , \dots x_T)$. Allo stesso modo indichiamo con il vettore $\boldsymbol{h_t}$ le informazioni immagazzinate nella memoria della rete fino all' istante t, e con il vettore $\boldsymbol{o_t}$ il segnale di output prodotto dalla rete relativo all' istante t.
In questo modo possiamo sintetizzare il processo di elaborazione dati della rete con l' equazione:

$$\boldsymbol{o_t} = f(\boldsymbol{x_t}, \boldsymbol{h_{t-1}}) \qquad [2.24]$$

dove f rappresenta una funzione capace di gestire sia il segnale di input definito esogeno, cioè proveniente da un diverso livello della rete, sia il segnale regresso, che tiene conto delle

informazioni accumulate in memoria. Il diagramma (Fig.2.9) facilita la comprensione della struttura descritta fino ad ora.

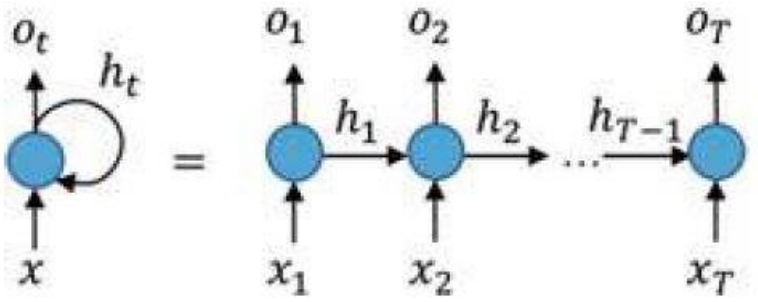

Fig.2.9: Il diagramma raffigura il meccanismo attuato dalle RNN per mantenere all' interno della rete le informazioni conservate nel vettore $\boldsymbol{h_t}$.

Una volta compresa l' idea di base, la formula [2.24] può essere completata inserendo le matrici relative ai pesi legati alle connessioni con il livello precedente (U) ed ai pesi legati al vettore $\boldsymbol{h_t}$ (W), nel seguente modo:

$$o_t = f(U \cdot \boldsymbol{x_t},\ W \cdot \boldsymbol{h_{t-1}})\ . \qquad [2.25]$$

I pesi vengono solitamente trattati durante la fase di apprendimento attraverso l' algoritmo della retro propagazione dell' errore proposto nel paragrafo 2.2.
Un' esempio semplice ma efficace di reti ricorrenti può essere quello proposto in figura (Fig.2.10), nel quale viene presentata la struttura di una rete ricorrente Input-Output. Sono presenti ai lati del percettrone multistrato due file di unità dedicate a trattenere nel tempo un certo numero di segnali. Il sistema non riceve solo input esterni, ma allo stesso tempo lo raggiungono nuovamente diversi segnali, già elaborati in precedenza, dalla fila superiore e altrettanti stimoli dalla fila inferiore che trattengono i risultati delle elaborazioni precedenti. In questo modo il segnale di output dell' intera rete può essere rappresentato come:

$$\boldsymbol{y_{n+1}} = F(\boldsymbol{y_n}, \ldots \boldsymbol{y_{n-q+1}}; \boldsymbol{u_n}, \ldots \boldsymbol{u_{n-q+1}}) \qquad [2.26]$$

che evidenzia la dipendenza di F sia rispetto agli stimoli passati che agli output già prodotti dalla rete.
Nell' equazione [2.26] indichiamo con $\boldsymbol{u_n}$ lo stimolo esterno che raggiunge attualmente la rete, mentre con $\boldsymbol{y_{n+1}}$ il vettore di output generato dalla rete come risposta a $\boldsymbol{u_n}$.

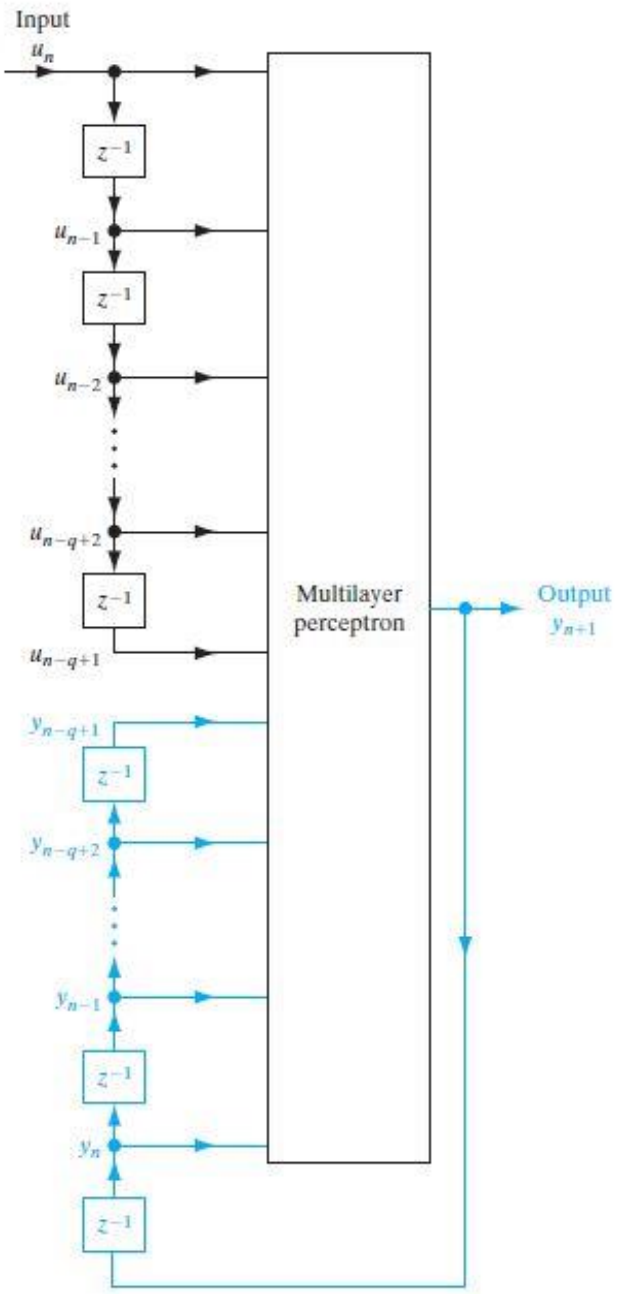

Fig.2.10: Lo schema rappresenta una RNN del tipo Input-Output. Sulla sinistra del percettrone sono facilmente distinguibili le due linee progettate per la conservazione della memoria. Quella nera si occupa di trattenere le informazioni relative ai segnali di input esogeni precedenti, mentre la linea blu tiene conto delle risposte precedenti della rete.

Capitolo 3: apprendimenti non supervisionati

Paragrafo 3.1
Il modello BCM

All' interno del paragrafo corrente viene descritto il modello di rete BCM, un sistema non supervisionato che cerca di simulare lo sviluppo dei neuroni biologici. Nel sottoparagrafo 3.3.1 verranno poi riportati i risultati ottenuti dal sottoscritto durante dei test effettuati sfruttando un modello di rete BCM.

Nel 1982 venne pubblicato "*Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex*",scritto da E.Bienenstock , L.Cooper e P.Munro [3].L' articolo presenta un nuovo algoritmo per descrivere lo sviluppo di un neurone appartenente alla corteccia visiva primaria nei vertebrati superiori, in particolare nelle prime settimane di vita di alcuni cuccioli di gatto. Lo scopo era quello di trovare una legge per lo sviluppo dei pesi sinaptici che fosse compatibile con i dati raccolti durante questo periodo critico. L' intero lavoro si basa sul concetto di selettività, cioè sulla capacità dei neuroni di rispondere in modo diverso rispetto ad altri a stimoli comuni.
Prima di andare nello specifico della regola di crescita dei pesi, è necessario introdurre un formalismo adatto alla sua trattazione ed un indice, definito di selettività. I valori utilizzati per i seguenti parametri sono da considerare come dei valori medi in un intervallo di tempo centrato nell' istante t, e riferiti al neurone k che appartiene alla corteccia visiva primaria. In questo modo, fissando un' istante nel tempo, l' integrazione dei segnali di input da parte di k sarà del tipo puramente spaziale. Il segnale di output prodotto dal neurone nell' istante t varrà:

$$c(t) = \sum_j m_j(t) d_j(t) \qquad [3.1]$$

dove m_j rappresenta il peso sinaptico tra il neurone presinaptico j e k mentre d_j il segnale di input prodotto da j. La sommatoria considera tutti i neuroni appartenenti al livello precedente a k e con esso collegati.
Sia i pesi che i segnali di input possono essere collezionati in due vettori, in modo da riscrivere la [3.1] in dicitura vettoriale:

$$c(t) = \boldsymbol{m}(t) \cdot \boldsymbol{d}(t) \ . \qquad [3.2]$$

$\boldsymbol{m}(t)$ viene definito stato del neurone al tempo t, e descrive la relazione con tutti i neuroni dello strato precedente. Il valore di $c(t)$ viene qui calcolato attraverso una semplice sommatoria, ma

potrebbe dover essere ricavato con una funzione più complessa, come una sigmoide. Nonostante questo, non cambiando in modo significativo i risultati riportati in seguito, è sufficiente trattarlo come in [3.2]. L' indice di selettività è un numero compreso nell' intervallo [0,1] che definisce quanto il neurone k risponde all' input d e può essere definito come:

$$Sel_d(k) = 1 - \frac{risposta\ media\ di\ k\ allo\ stimolo\ d}{risposta\ massima\ di\ k\ allo\ stimolo\ d} \qquad [3.3]$$

che è lecito riscrivere in funzione dello stato del sistema in un dato istante nel seguente modo:

$$Sel_d(m) = 1 - \frac{E[\boldsymbol{m}\cdot\boldsymbol{d}]}{\max(\boldsymbol{m}\cdot\boldsymbol{d})}\ . \qquad [3.4]$$

Sel_d risulta un valore molto importante perché permette di comprendere l' attività di un' unità della rete rispetto alle altre, da quali stimoli viene sollecitata maggiormente o da quali, al contrario, non viene attivata. Permette anche un confronto tra i neuroni che, a causa di uno sviluppo sinaptico competitivo, possono diventare più legati ad alcuni segnali di input invece che ad altri. $E\ [\boldsymbol{m}\cdot\boldsymbol{d}]$ indica il valore atteso del segnale di risposta, e dipende dalla distribuzione dello stimolo $\boldsymbol{d}$. Tale distribuzione può essere di due tipi: discreta o continua. Per raggiungere i risultati in appendice B verrà utilizzata una distribuzione discreta. L' immagine in figura (Fig.3.1) aiuta la comprensione di questo indice.
La legge proposta per la modifica dei pesi sinaptici, dipendente dal tempo, è la seguente:

$$m'_j\,(t) = \varphi\big(c(t)\big)d_j(t) - \varepsilon m_j(t) \qquad [3.5]$$

nella quale φ rappresenta una funzione scalare dell' attività post sinaptica, mentre il secondo termine di destra indica una riduzione uniforme di tutte le giunzioni. Questo termine può essere trascurato durante la trattazione dell' algoritmo ponendo ε piccolo a sufficienza. Insieme alla funzione φ è bene introdurre anche una funzione di soglia $\theta_M(t)$ tale per cui deve valere:

$$\varphi\big(c(t)\big) < 0\ se\ c(t) <\ \theta_M(t)\ \ \text{mentre}\ \ \varphi\big(c(t)\big) > 0\ se\ \ c(t) > \theta_M(t)\ ; \qquad [3.6]$$

in questo modo quando un neurone viene sollecitato a sufficienza da produrre un $c(t)$ capace di superare la funzione di soglia, e $d_j(t)$ è positivo, allora il peso sinaptico con il neurone j crescerà; al contrario quando $c(t)$ non supera la soglia, nonostante $d_j(t)$ sia positivo, il valore della giunzione calerà. Questo sistema viene quindi definito di competizione temporale tra i pattern di ingresso dell' unità j. Se θ_M fosse costante il problema potrebbe essere quello di non avere un segnale in uscita capace di superare la soglia, e di conseguenza di portare il neurone a non venir stimolato da nessun tipo di segnale dopo aver calato fino a zero tutti i suoi pesi sinaptici.

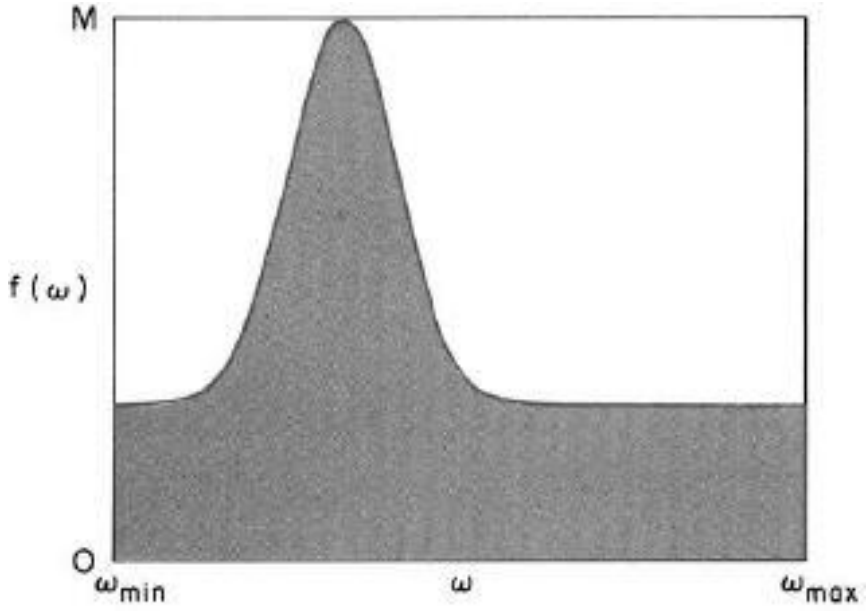

Fig.3.1: Nel grafico è rappresentata la selettività di una cellula verso una gamma di stimoli con orientazione compresa tra ω_{max} e ω_{min}. La selettività, in questo caso ricavabile come:

$$Sel_d(N) = 1 - \frac{1}{M(\omega_{max} - \omega_{min})} \int_{\omega_{min}}^{\omega_{max}} f(\omega) d\omega \quad ,$$

evidenzia una forte risposta della cellula in particolare su una piccola gamma di stimoli corrispondenti al picco nel grafico.

Ora il problema diventa quello di capire come si deve comportare la funzione $\varphi(c(t), \bar{c}(t))$ che dipende dalla risposta del neurone al tempo t e da una media dei segnali prodotti fino a quel momento, che può essere indicata come:

$$\overline{c}(t) = \boldsymbol{m}(t)\, \overline{\boldsymbol{d}}(t) \qquad [3.7]$$

con $\overline{\boldsymbol{d}}$ valore medio dei segnali di input ricevuti dal neurone. La forma precisa della funzione φ non è importante, siccome i risultati ottenuti utilizzando la [3.5] non varierebbero. Il segno di φ invece deve essere ben definito, e dipende dalla relazione tra $\theta_M(t)$ e $c(t)$ come visto in precedenza; per questo è necessario trovare un' espressione per la funzione di soglia.
La soluzione proposta nell' articolo come funzione di soglia vale:

$$\theta_M(t) = (\frac{\overline{c}(t)}{c_0})^p \ \ \bar{c} \qquad [3.8]$$

e di conseguenza si avrà che:

$$sign\left(\varphi\big(c(t),\bar{c}(t)\big)\right) = sign\left(c - \left(\frac{\bar{c}(t)}{c_0}\right)^p \ \bar{c}\right) \quad \text{per } c > 0 \qquad [3.9]$$

mentre quando $c(t) = 0$, allora $\varphi\big(0,\bar{c}(t)\big)$ è sempre uguale a zero (difficilmente $c < 0$). Nell' equazione sono presenti due costanti, c_0 e p, che devono essere ricavate sperimentalmente e che è possibile variare per ottenere per esempio ratei di convergenza diversi. Inoltre la sua struttura permette andamenti di questo tipo:

- se inizialmente $\bar{c} \ll c_0$ ma la risposta del neurone agli input di un certo tipo è favorevole , questo porterà ad una crescita dei pesi sinaptici. Allo stesso tempo crescerà $\bar{c}$ fino ad arrivare ad uno stato stabile;
- se la risposta è fin da subito sfavorevole, i pesi sinaptici verranno calati fino a raggiungere $c(t) = 0$, e di conseguenza una φ pari a zero che interrompe definitivamente la crescita della sinapsi.

Parlando di punti di stabilità potrebbe sorgere facilmente il dubbio di come si possa essere certi della loro esistenza. In appendice B si trova una rapida spiegazione di come questo sia possibile.
Per le prove sperimentali viene utilizzata una barra luminosa che cambia orientazione, input dopo input, in modo casuale. Tutti i segnali luminosi non provenienti da questa sorgente vengono considerati, all' interno dei dati raccolti, come rumore di fondo. I soggetti studiati sono cuccioli di mammifero durante le prime settimane di vita, nel così definito periodo critico per lo sviluppo delle sinapsi della corteccia visiva primaria. Per quanto riguarda il set di stimoli, una distribuzione di questo tipo, discreta, uniforme e su di una simmetria circolare, può essere definita un' ambiente circolare. I parametri che caratterizzano d sono il numero di neuroni del primo strato N, il numero di stimoli diversi che definisce la distribuzione discreta K, e una misura che indichi la geometria alla base dei segnali di input, per esempio $min\ cos\ (\boldsymbol{d^1}, \boldsymbol{d^i})$, con $0 < i \leq K$. In particolare nei test sono stati usati i seguenti valori: $N = 37$, e $k \in [12{,}60]$. Per dare un' idea dello sviluppo di sistemi in questo ambiente, è riportato un esempio in figura (Fig.3.2). Nel caso di un monoculare i risultati ottenuti verificano le predizioni teoriche, ovvero:

- lo stato converge velocemente ad uno dei punti fissi;
- esistono più punti fissi, e tutti possiedono la stessa selettività;
- la curva asintotica che indica la crescita della selettività è sempre unimodale;
- esiste un punto fisso stabile differente per ogni orientazione, e se non ci sono particolari preferenze iniziali, tutte le orientazioni hanno la stessa probabilità di divenire quelle preferite in termini di selettività.

Attraverso il teorema 3 in appendice B è sicuro che, se i vettori di input sono tra loro ortogonali o quasi, lo stato di stabilità verrà sicuramente raggiunto. In questo caso gli stimoli appartenenti all' ambiente circolare non sono per forza ortogonali, ma non potendo dimostrare in qualche modo che questi punti fissi stabili non vengano raggiunti e siccome a livello sperimentale il sistema raggiunge sempre uno stato di equilibrio, si può dire che il teorema 3 valga anche in questo caso. Inoltre, bisogna aggiungere che i dati sperimentali marcano anche l' importanza delle sinapsi inibitorie, fondamentali per rendere la selettività più forte. Per dimostrarlo, basta portare un

sistema con input bidimensionale a diventare selettivo. Successivamente, settare tutte le componenti negative dello stato a 0 e notare un calo drastico della selettività.

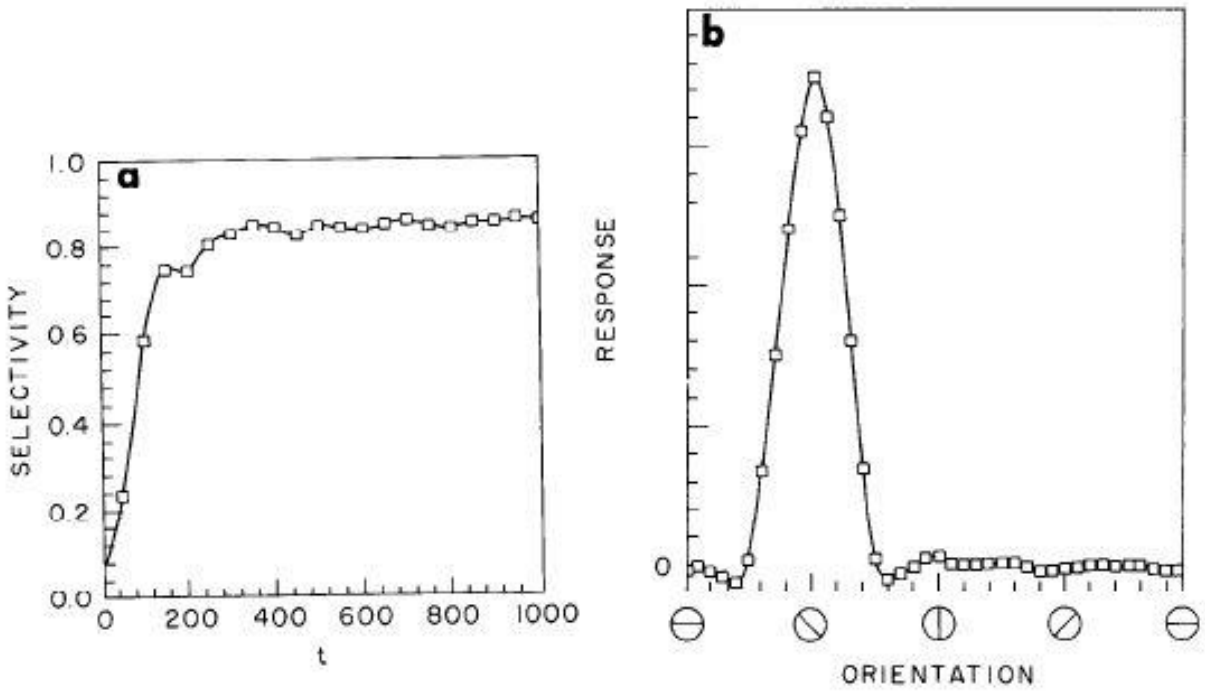

*Fig.3.2: I grafici rappresentano lo sviluppo della selettività di una cellula in un ambiente circolare. I parametri scelti sono K=40 , N=37. In particolare, l' immagine **a** indica la crescità del valore del coefficiente Sel_d nel tempo, mentre l' immagine **b** raffigura come alla fine del processo di apprendimento la cellula preferisca una piccola parte degli stimoli su tutti gli altri.*

Nel caso specifico in cui è presente un ambiente con un solo stimolo $\boldsymbol{d^1}$, il sistema raggiunge il punto fisso stabile $\boldsymbol{m^1}$. Poi, se compare uno stimolo differente d non esattamente ortogonale a d^1, la risposta dipende sia dal $\cos(\boldsymbol{d^1}, \boldsymbol{d})$, sia dal fattore ε presente in [3.5], che in questo caso è bene tornare a considerare.
Nel caso di una cellula binoculare, invece, il segnale di output prodotto vale:

$$c(t) \;=\; \boldsymbol{m_r}(t) \cdot \boldsymbol{d_r}(t) \;+\; \boldsymbol{m_l}(t) \cdot \boldsymbol{d_l}(t)\,. \qquad [3.10]$$

A questo punto si aprono molte possibilità, come chiudere un occhio e permettere lo sviluppo solo dell' altro, oppure utilizzare un $\boldsymbol{d_r}(t)$ differente da $\boldsymbol{d_l}(t)$ ed esistono altri casi anormali studiati all' interno dell' articolo.
Per quanto riguarda questa breve trattazione, sarà sufficiente riportare i risultati generali della ricerca:

- la cellula che riceve entrambi gli input (destro e sinistro) tende a raggiungere uno stato di selettività binoculare, preferendo la stessa orientazione in entrambi gli occhi (salvo casi particolari);

- nel caso in cui il sistema sia privato di un occhio, la cellula svilupperà una selettività verso un' orientamento ma in modo monoculare, situazione già affrontata in precedenza;
- nel caso in cui il sistema sia privato di entrambi gli occhi, la cellula perde la sua selettività ma continua ad essere in grado di rispondere, e rimane binoculare.

La proposta del 1982 riesce a predire, con buoni riscontri con la realtà, lo sviluppo di una cellula appartenente alla zona 17 di un sistema nervoso appartenente ai cuccioli di mammifero. In particolare introduce il concetto di selettività, che permette di capire come la cellula nel tempo sviluppi una preferenza verso alcuni input ai quali risponde in modo più evidente. Per la crescita della selettività verso alcuni stimoli piuttosto che altri, è fondamentale tanto la presenza delle sinapsi eccitatorie quanto quella delle sinapsi inibitorie, senza le quali il suo valore rimarrebbe molto più basso. La BCM non considera però l' inibizione laterale nella rete, data dall' interazione tra neuroni di uno stesso livello, e rende così possibile la presenza di numerose cellule con un' alta selettività verso stessi stimoli. Per questo motivo un' intera rete sviluppata esclusivamente sull' algoritmo appena proposto non raggiungerebbe una buona efficienza. È importante sottolineare quindi che, fino ad ora, si è parlato di competizione tra i pattern di input ma nessuna competizione tra le unità nascoste è mai stata presentata. Nel prossimo capitolo vedremo come la proposta di Hopfield e Krotov prenderà in considerazione anche l' inibizione laterale e permetterà di ottenere reti migliori in termini di efficienza, ma comunque non supervisionate e biologicamente attendibili.

Paragrafo 3.2
Il modello di Hopfield e Krotov

Nelle seguenti pagine verrà descritto il modello di rete proposto da Hopfield e Krotov: sempre un modello non supervisionato ma che, durante una seconda fase dell' addestramento, sfrutta la retro propagazione dell' errore per lo sviluppo delle sinapsi. La differenza sostanziale con il modello della BCM è la mancanza della funzione di soglia, non necessaria nella rete proposta in questo paragrafo.

Nel 2019 J.Hopfield e D.Krotov pubblicarono un articolo intitolato "Unsupervised learning by competing hidden units" per presentare il loro sistema di apprendimento [4], innovativo perchè non supervisionato ma ugualmente capace di stare al passo con le reti allenate tramite retro propagazione dell' errore. Lo scopo principale era quello di mostrare che la rete da loro ideata fosse in grado di competere con una rete supervisionata nel riconoscere le immagini dei data set con i quali veniva allenata. La macchina proposta nell' articolo è composta da due parti:

- i primi strati della rete si sviluppano tramite un algoritmo non supervisionato, che sfrutta una crescita delle sinapsi locale, quindi dovuta esclusivamente all' attività pre e post sinaptica del neurone;
- il resto dei livelli, invece, vengono addestrati attraverso la retro propagazione dell' errore, mantenendo fissi i pesi della prima parte.

In questo senso, possiamo descrivere l' attività dell' intera rete nel seguente modo:

$$\begin{cases} h_\mu = f\left(W_{\mu i} v_i\right) \\ c_\alpha = \tanh(\beta S_{\alpha\mu} h_\mu) \end{cases} \quad , \quad \text{con } f(x) = \begin{cases} x^n \ , \ x \geq 0 \\ 0 \ , \ x < 0 \end{cases} \qquad [3.11]$$

dove $W_{\mu i}$ sono i pesi appresi attraverso la prima fase dell' addestramento, mentre $S_{\alpha\mu}$ sono i parametri ottenuti durante la seconda fase dello sviluppo. h_μ rappresenta l' attività del neurone nascosto μ, mentre v_i quella del neurone appartenente al primo livello visibile i e per ultimo, c_α indica il segnale complessivo di output della rete. β rappresenta semplicemente una costante numerica, mentre n viene indicata come la potenza della funzione di attivazione. Nonostante sia ancora presto per comprendere i grafici in figura (Fig. 3.3), possono comunque servire per comprendere a grandi linee in che modo questo valore modifica la performance finale della macchina. Il sistema [3.11] può essere facilmente rappresentato come in figura (Fig. 3.4).

Nel paragrafo precedente è stata introdotto un certo tipo di competizione, quello tra pattern che concorrono per diventare i preferiti di una cellula. Questo si traduce nella cellula che risponderà con particolare intensità ad alcuni stimoli piuttosto che ad altri. Nell' algoritmo che verrà introdotto, al contrario, la competizione sarà tra le stesse cellule: se una cellula sviluppa una forte selettività verso un tipo di impulso, farà in modo che le altre non possano fare lo stesso. Questo rende difficile per due cellule sviluppare una selettività per uno stesso stimolo, poiché durante il processo di crescita delle sinapsi si ostacolerebbero a vicenda.

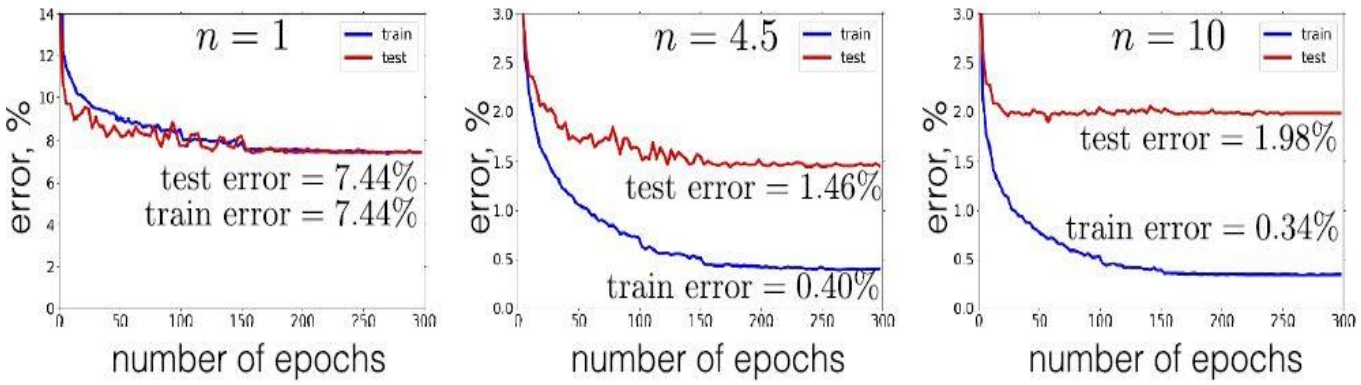

Fig.3.3: In questa immagine è evidente come la potenza della funzione di attivazione influenzi la percentuale di errore della macchina addestrata. Per ottenere buone prestazioni non basta aumentare a dismisura questo valore, ma ricercare l' n esatto che massimizzi l' efficienza, in questo caso rappresentato dal valore 4.5 come nel grafico centrale.

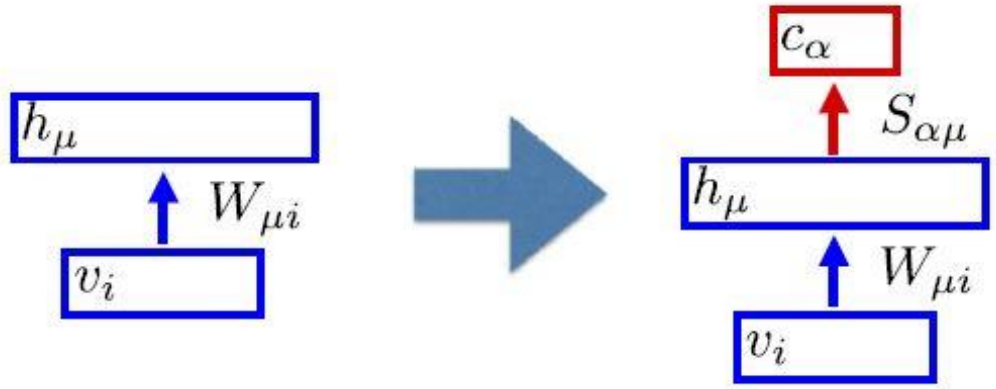

Fig.3.4: lo schema raffigura le due sezioni della rete neurale presentata in questo capitolo: sulla destra la parte non supervisionata, quella di sinistra allenata con un apprendimento supervisionato.

Per semplicità, almeno per il momento, ipotizziamo di avere un sistema con una sola unità nascosta. A questo punto indichiamo come prodotto interno tra due vettori la seguente operazione:

$$\langle \boldsymbol{X}, \boldsymbol{Y} \rangle = \sum_{i,j} \eta_{ij} X_i Y_j \quad \text{dove} \quad \eta_{ij} = |W_i|^{p-2} \delta_{ij} \qquad [3.12]$$

dove il simbolo δ_{ij} rappresenta la delta di Kronecker. Il parametro p può essere interpretato in questo modo: se p=2 si ottiene il classico prodotto scalare tra due vettori, mentre pian piano che aumenta si dà sempre più importanza ai pesi. La legge per lo sviluppo dei pesi studiata in questo articolo prende spunto dalla legge di Oja, ricavata e spiegata in appendice C. Può essere scritta nel seguente modo:

$$\tau_L \frac{dW_i}{dt} = g(Q)\,(R^p v_i - \langle \boldsymbol{W}, \boldsymbol{v}\rangle W_i) \qquad \text{dove} \qquad Q = \frac{\langle \boldsymbol{W}, \boldsymbol{v}\rangle}{\langle \boldsymbol{W}, \boldsymbol{W}\rangle^{\frac{p-1}{p}}} \quad . \qquad [3.13]$$

τ_L definisce il tempo di scala dell' apprendimento della rete, $\boldsymbol{v}$ indica il segnale proveniente dai neuroni visibili. La funzione $g(Q)$ rappresenta una funzione di attivazione non lineare. R è una costante particolare: indica il raggio della sfera alla quale i pesi tendono a convergere se si aspetta un tempo sufficientemente lungo. Infatti, considerando $t \to \infty$, se si calcola la norma con il prodotto interno mostrato in precedenza, varrà:

$$|W_1|^p + |W_2|^p + \cdots + |W_N|^p = R^p \quad . \qquad [3.14]$$

Per dimostrarlo basta ricavare la derivata nel tempo di $\langle \boldsymbol{W}, \boldsymbol{W}\rangle$ ed ottenere l' espressione:

$$\tau_L \frac{d\langle \boldsymbol{W}, \boldsymbol{W}\rangle}{dt} = p\; g(Q)\;\; \langle \boldsymbol{W}, \boldsymbol{v}\rangle\; (R^p - \langle \boldsymbol{W}, \boldsymbol{W}\rangle\,) \quad , \qquad [3.15]$$

che evidenzia come lo sviluppo del prodotto si interrompa solo raggiunto un valore di $\langle \boldsymbol{W}, \boldsymbol{W}\rangle$ pari a R^p, provato che $g(Q)\,\langle \boldsymbol{W}, \boldsymbol{v}\rangle \geq 0$. Questa condizione vale per una grande gamma di funzioni di attivazione.
In generale, quando la rete possiede numerose unità nascoste, la [3.12] può essere riscritta come:

$$\langle \boldsymbol{X}, \boldsymbol{Y}\rangle_\mu = \sum_{i,j} \eta_{ij}{}^{(\mu)}\, X_i Y_j \quad \text{con} \quad \eta_{ij}{}^{(\mu)} = |W_i|^{p-2}\, \delta_{ij}\ , \qquad [3.16]$$

dove μ indica un' indice per riferirsi alle diverse unità, ed ognuna di esse svilupperà i propri pesi attraverso la regola [3.13].
Adesso verrà introdotta l' altra parte fondamentale di questo algoritmo, cioè l' inibizione laterale che ogni cellula applica sulle altre dello stesso livello.
La regola generale può essere rappresentata così:

$$\tau \frac{dh_\mu}{dt} = I_\mu - \;\omega_{inh}\; \sum_{\gamma \neq \mu} r\big(h_\gamma\big) - h_\mu \qquad [3.17]$$

dove con I_μ si indica il prodotto $\langle \boldsymbol{W_\mu}, \boldsymbol{v}\rangle_\mu$, mentre con $r\big(h_\gamma\big)$ il valore $max\big(h_\mu, 0\big)$. Infine ω_{inh} è un parametro che indica la forza dell' inibizione globale. Di solito viene settato in modo da permettere solo ad una piccola parte delle unità nascoste, una volta raggiunto lo stato finale dello sviluppo, di avere un' attività positiva. Da sottolineare anche la presenza di un tempo, τ ,

differente da quello in formula [3.13]. In questo caso si usa un tempo di scala riferito all' apprendimento di un solo neurone, quindi molto più piccolo rispetto a τ_L. I valori delle attività dei vari neuroni nascosti h_μ vengono poi utilizzati per lo sviluppo dei pesi sinaptici attraverso la funzione $g(h)$.
La funzione di attivazione $g(h)$ possiede una struttura del tipo:

$$g(h) = \begin{cases} 0, & h < 0 \\ -\Delta, & 0 \leq h < h_* \\ 1, & h_* \leq h \end{cases} \qquad [3.18]$$

nella quale h_* rappresenta il valore di soglia che l' attività del neurone deve superare per poter crescere. In caso contrario, se la risposta della cellula è già negativa, resta invariata, mentre se è positiva ma non abbastanza da passare la soglia, allora viene depotenziata. In questo modo fin dall' inizio vengono avvantaggiati solo pochi neuroni sufficientemente attivi nei confronti di un certo stimolo, mentre gli altri vengono immediatamente allontanati. Per spiegare bene il processo di inibizione laterale, si può fare riferimento alle immagini di (Fig.3.5) che semplifica l' intero procedimento, passo per passo.

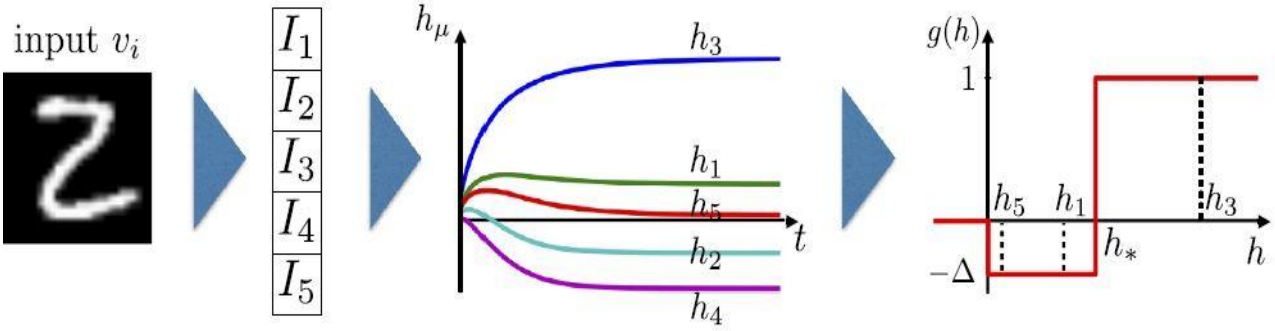

Fig.3.5: durante il primo passaggio raffigurato in questa immagine, il segnale di input v_i viene convertito in un set di correnti di input. La regola di inibizione laterale sfrutta queste correnti per modificare l' attività dei neuroni nel tempo, come illustrato nel secondo passaggio. Infine nell' ultimo grafico si osserva il risultato della funzione di attivazione, che influirà sulla crescita dei pesi sinaptici secondo la formula [3.13].

È possibile velocizzare l' intero apprendimento utilizzando una regola che gestisce lo sviluppo dei pesi strutturata in questo modo:

$$g(i) = \begin{cases} 1, & se\ i = K \\ -\Delta, & se\ i = K - k\,, \\ 0, & altrove. \end{cases} \qquad [3.19]$$

dove i indicizza i neuroni che hanno risposto allo stimolo, ordinandoli dalla risposta meno forte alla più forte quando $i = K$. k viene definito parametro di rango, e parte da uno fino ad arrivare al numero di neuroni che hanno risposto meno uno. In questo modo viene potenziata una ed una sola sinapsi, quella avvantaggiata fin dall' inizio dell' addestramento, mentre le altre vengono calate immediatamente.

Il confronto tra la rete proposta da Hopfield e Krotov ed una rete allenata in modo supervisionato attraverso la retro propagazione dell' errore, trattato nel medesimo articolo, verrà discusso nel seguente paragrafo, dopo i risultati riguardanti i test effettuati sulla BCM.

Paragrafo 3.3
Risultati e discussione

Il seguente paragrafo si compone di due sotto paragrafi. Il primo è dedicato alla parte della tesi sperimentale, mentre il secondo riporta i risultati della rete proposta da Hopfield e Krotov confrontata con una rete supervisionata.
Per quanto riguarda la parte sperimentale, l' obiettivo era quello di allenare un modello di rete BCM per riuscire a massimizzarne l' efficacia nel distinguere diverse immagini. Una volta raggiunto il risultato migliore possibile, si è scelto di indagare sulle difficoltà che i neuroni della rete incontravano nel tentare di convergere a tanti pattern differenti. Le osservazioni fatte sono anche frutto di un confronto con dati pubblicati in articoli o blog scientifici.

Paragrafo 3.3.1
Risultati ed osservazioni sulla BCM

Nel paragrafo 3.1 è stato introdotto il modello della BCM come un sistema non supervisionato capace di descrivere lo sviluppo di neuroni appartenenti alla corteccia visiva primaria. Nella proposta del 1982, la rete non prendeva in considerazione le interazioni laterali tra neuroni del medesimo livello. Questo aspetto è stato poi introdotto nel 1988 da Cooper e Scofield [17], utilizzando una matrice di questi tipo: $L = (L_{ij})$ dove il valore L_{ij} rappresenta il peso sinaptico tra il neurone i ed il neurone j appartenenti allo stesso livello della rete. A questo punto, l' output generato dalla rete varrà:

$$c = \boldsymbol{m}\,\boldsymbol{d} + \boldsymbol{L}\,\boldsymbol{c} \qquad [3.20]$$

e anche la crescita dei pesi dovrà di conseguenza tener conto di questo cambiamento. Per verificare quanto effettivamente il modello BCM sia capace di memorizzare, attraverso i pesi sinaptici, immagini differenti durante il periodo di addestramento, sono stati effettuati una serie di test utilizzando la libreria Plasticity [20]. L' obiettivo dei test era quindi quello di studiare la selettività dei neuroni addestrati e tentare di massimizzarne l' eterogeneità, così da rendere la rete capace di riconoscere il maggior numero di pattern differenti. Con il termine eterogeneità della selettività, viene inteso la presenza di neuroni all' interno della rete specializzati nel riconoscere differenti stimoli. Questo è un aspetto importante della rete, poiché una volta allenata , se le viene mostrato un numero da riconoscere, i neuroni con i pesi sinaptici che hanno memorizzato tale simbolo rispondono con maggior intensità rispetto a tutte le altre unità, e permettono il corretto riconoscimento della cifra. L' introduzione dell' espressione [3.20] nel modello BCM del 1982 rappresenta quindi uno studio interessante, dal momento che le interazioni laterali all' interno della rete permetterebbero, almeno in teoria, di evitare che più neuroni convergano agli stessi pesi sinaptici. Il modello BCM della libreria Plasticity permette di assegnare diverse variabili che concedono all' utilizzatore di personalizzare la propria rete in base alle necessità. Le variabili maggiormente analizzate durante i test sono state:

1) l'ottimizzatore, cioè l' algoritmo per la correzione dei pesi durante l' allenamento;
2) la distribuzione iniziale dei pesi;
3) la funzione di attivazione;
4) Il valore dell' interazione laterale, con la quale i neuroni inibivano i vicini ostacolandoli nel convergere verso uno stesso stimolo.

Inoltre lo strato di output del sistema addestrato era formato da 100 unità, veniva allenato per un totale di 10 epoche, ed infine l' aggiornamento dei pesi sinaptici avveniva, per la maggior parte delle simulazioni, ogni 1000 esempi mostrati alla rete. La rete è stata allenata sfruttando il data set MNIST, una vasta collezione di immagini rappresentanti cifre scritte a mano, da 0 a 9. Di conseguenza un dato numero è presente nel data set scritto in molti modi differenti. Ogni figura è formata da 784 variabili, schematizzabili in una matrice 28x28 , ed ogni segnale fa riferimento ad un pixel del numero (Fig.3.6). Per buona parte della raccolta dati effettuata con la rete BCM, la

matrice di input non era composta semplicemente da segnali binari, ma conteneva anche valori compresi in un intervallo tra 0 ed 1 (come è evidente dall' immagine). Nelle tabelle delle pagine successive, viene sempre indicato se gli esempi forniti alla rete sono stati rappresentati tramite matrici binarie oppure no. Utilizzando i metodi della libreria Plasticity è permesso, a fine allenamento, generare un' immagine che rappresenta le matrici delle connessioni sinaptiche di ciascun neurone. In particolare, nelle immagini inserite in seguito, il colore rosso rappresenta sinapsi di segno positivo, il blu rappresenta sinapsi di segno negativo, mentre il colore bianco indica un peso sinaptico nullo. Attraverso queste immagini è stato possibile studiare il comportamento della rete, quali numeri venivano più facilmente riconosciuti da essa e, al contrario, verso quali cifre i neuroni tendevano a specializzarsi con meno frequenza.

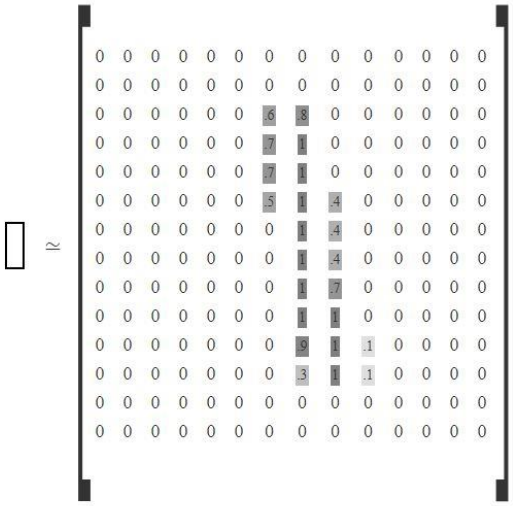

Fig.3.6: I numeri nella matrice di input, in questo esempio, rappresentano i segnali inviati ad ogni neurone della rete relativi al numero uno, scritto in una delle sue tante versioni.

Per raggiungere la massima eterogeneità in termini di selettività, il primo passo è stato quello di testare la rete utilizzando l' intero data set MNIST e le diverse funzioni di attivazione fornite dalla libreria (Fig.3.7). Sicuramente, i risultati più chiari sono stati raggiunti utilizzando la funzione ReLU, che di conseguenza è diventata una costante nelle simulazioni successive. In seguito, i test si sono concentrati sulle variabili del modello algoritmo di ottimizzazione e distribuzione iniziale dei pesi. Dopo ogni addestramento, è quindi stato necessario contare quanti neuroni convergevano ad ogni cifra e quanti simboli differenti comparivano per ogni numero. Lo scopo era quello di trovare la combinazione migliore, che permettesse ai neuroni di convergere ad un vasto numero di simboli differenti e che non generasse troppe sinapsi negative, soprattutto posizionate lontano dalla sagoma della cifra. I risultati ottenuti durante i vari test sono riportati nella tabella (Tab.3.1). La combinazione migliore trovata è quella data dall' ottimizzatore Momentum, con momento pari ad 1, e da una distribuzione normale di He per i pesi iniziali, con deviazione standard pari ad 1 e centrata sullo zero. La normale di He è risultata la scelta più appropriata nonostante mantenesse un' eterogeneità della selettività minore rispetto alle altre distribuzioni, siccome restituiva una

convergenza con presenza minore di sinapsi inibitorie. Ulteriori test sono stati effettuati una volta inserite anche le interazioni laterali, fino a questo momento lasciate da parte (Tab.3.2). Durante queste simulazioni, è sempre stata utilizzata una distribuzione normale dei pesi iniziali, centrata sullo 0 e con una deviazione standard pari ad 1. Dai dati raccolti si nota come, una volta introdotte, alcuni neuroni siano riusciti effettivamente a cambiare il simbolo al quale convergere, portando la rete a riconoscere più caratteri distinti. Infatti, nel caso del modello utilizzato, un valore negativo di questa variabile dovrebbe portare gli altri neuroni a specializzarsi verso pattern differenti. Con un valore pari a -0.01 è stato raggiunto un totale di 37 simboli differenti riconosciuti, ma andando oltre questa soglia l' immagine diventava troppo poco chiara ed inutilizzabile. L' obiettivo è quindi diventato quello di ricercare un valore prossimo a -0.01 capace di fornire una distribuzione dei pattern di convergenza più uniforme, ma questo ha comportato un calo dei simboli totali distinti.

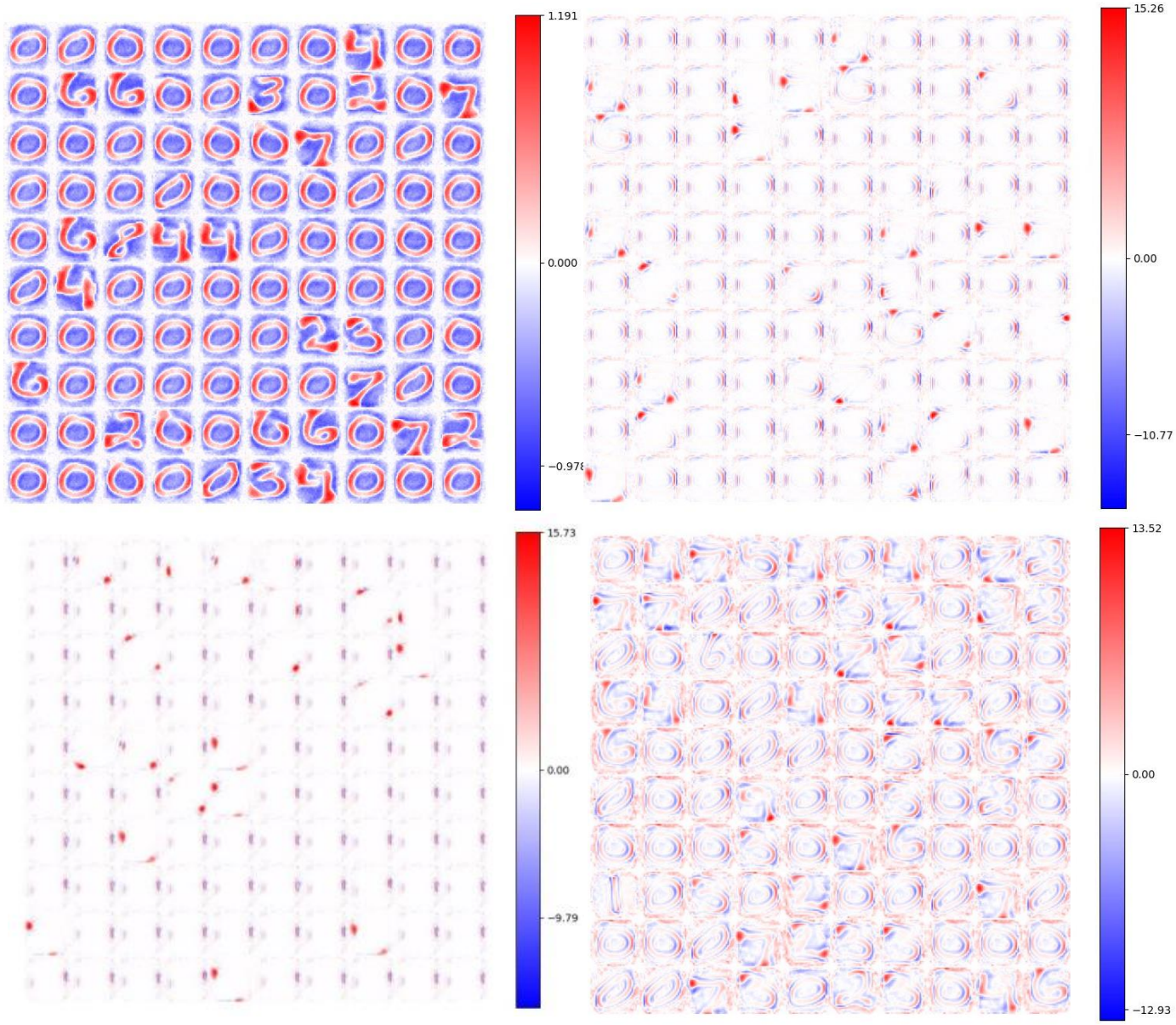

Fig.3.7: Alcuni esempi delle funzioni di attivazione disponibili nella libreria Plasticity. In alto a sinistra la ReLU, a destra la Elu, in basso a sinistra la Linear ed in basso a destra la ReLie. Gli esempi sono stati passati alla rete tramite matrici di input non binari.

Ottimizzatore	Pesi	n°0	n°1	n°2	n°3	n°4	n°5	n°6	n°7	n°8	n°9	Tot. cifre	Tot-simboli
Adam	U	52	1	10	4	8	0	13	7	3	2	9	19
Adam	GU	47	2	11	5	8	1	13	9	2	2	10	24
Adam	HU	35	5	9	7	7	3	21	10	1	2	10	26
Adam	LU	40	2	12	5	6	2	18	10	2	3	10	25
Adam	N	19	9	15	9	9	0	14	19	1	5	9	31
Adam	GN	51	1	14	9	7	2	5	8	2	1	10	23
Adam	HN	34	5	20	5	10	0	9	14	2	1	9	27
Momentum	N	23	8	17	8	13	1	10	18	1	1	10	34
Momentum	GN	42	11	12	5	5	0	8	14	1	2	9	25
Momentum	HN	30	10	16	8	7	0	9	17	1	2	9	27
Momentum	HU	34	11	9	9	8	1	15	10	0	3	9	29
Momentum	TN	23	9	16	2	13	5	17	13	0	2	9	32
Nesterov M	N	23	7	18	9	14	2	9	16	1	1	10	31
Nesterov M	GN	43	10	13	6	4	0	6	16	1	1	9	24
Nesterov M	HN	32	10	17	7	8	0	7	16	1	2	9	27
Nesterov M	HU	34	10	10	8	9	1	15	10	0	3	9	28

Tab. 3.1: In tabella sono riassunti i risultati dei test effettuati sul modello BCM variandone l'ottimizzatore e la distribuzione dei pesi iniziali. È indicato il valore della ricorrenza di ogni cifra tra quelle memorizzate dai pesi dei neuroni a fine allenamento, il numero totale di cifre e di simboli differenti trovati. Gli esempi sono stati passati alla rete tramite matrici di input non binari. Nei casi in cui sono presenti gli ottimizzatori Momentum o Nesterov Momentum, il parametro momento vale sempre 0.9. Sono state utilizzate le seguenti abbreviazioni: uniforme(U),normale(N),troncata(T), di Glorot(G), di He(H), di LeCun(L).

Interazioni laterali	n°0	n°1	n°2	n°3	n°4	n°5	n°6	n°7	n°8	n°9	Tot. cifre	Tot-simboli
+0.0001	23	8	17	8	13	1	10	18	1	1	10	34
-0.0001	23	8	17	8	13	1	10	18	1	1	10	34
+0.001	23	8	17	9	14	1	10	16	1	1	10	34
-0.001	24	7	18	8	13	1	10	17	1	1	10	34
+0.01	X	X	X	X	X	X	X	X	X	X	X	X
-0.01	22	9	15	10	12	3	11	16	1	1	10	37
-0.015	21	10	15	9	12	3	11	17	1	1	10	36
-0.012	21	9	15	9	12	3	12	18	2	1	10	36

Tab.3.2: La tabella riporta i risultati ottenuti sfruttando l' inibizione laterale implementata nel modello BCM della libreria Plasticity. Il modello utilizzava un ottimizzatore Momentum con momento pari a 0.9 ed una distribuzione Normal con std pari ad 1. Gli esempi sono stati passati alla rete tramite matrici di input non binari.

Osservando i risultati raccolti in tabella 3.1 e 3.2, è evidente come i neuroni convergano molto più spesso verso alcuni pattern piuttosto che altri. Per esempio, in tutti i test eseguiti le immagini al quale convergono un maggior numero di unità sono sempre relative allo 0, ma anche il numero 2 ed il numero 7 sono quasi sempre tra i più memorizzati dai pesi sinaptici. Al contrario, ci sono cifre che difficilmente riescono ad attrarre a se i pesi sinaptici della rete, come il numero 5, l' 8 ed il 9. Per riuscire a quantificare ciò, è stata fatta una veloce analisi su un campione di 10 test effettuati variando l' ordine con il quale gli esempi venivano presentati alla rete(Tab.3.3).

	n°1	n°2	n°3	n°4	n°5	n°6	n°7	n°8	n°9	n°0
media	9.3	6.5	10.6	7.3	1.1	10.6	14.1	0.7	1.1	38.7
std	2.3	2.4	2.7	1.8	1.4	2.8	2.5	0.8	0.9	5.1

Tab.3.3: In tabella sono riassunti i risultati relativi alle ricorrenze delle cifre da 0 a 9 su un campione di 10 test, ottenuto sfruttando il modello BCM con l' ottimizzatore Momentum (m=1) e la distribuzione dei pesi Normale di He. Gli esempi sono stati passati alla rete tramite matrici di input non binari.

La stessa identica analisi è stata effettuata anche presentando alla rete gli esempi sotto forma di matrici binarie. In questo modo, il problema che la rete doveva affrontare nel riconoscere diversi pattern ai quali convergere è risultato molto semplificato, come possiamo notare osservando gli ottimi risultati riportati nelle tabelle sottostanti (Tab.3.4 e Tab.3.5). Numeri come l' 8, il 9 ed il 5 risultano sempre tra i meno ricorrenti, mentre numeri come lo 0 ed il 7 restano tra i maggiori simboli di convergenza per i pesi sinaptici, però in generale la distribuzione dei neuroni nel convergere verso le 10 cifre risulta più omogenea rispetto a quelle rappresentate in tabella 3.3 e 3.2.

	n°1	n°2	n°3	n°4	n°5	n°6	n°7	n°8	n°9	n°0
media	12.2	6.4	6.9	9.5	2.6	12	13.1	0.6	1.5	35.2
std	3.9	2.4	2.6	2.6	1.1	2.2	2.5	0.5	1.6	5.9

Tab.3.4: Dati relativi a 10 simulazioni fatte utilizzando l' ottimizzatore Momentum (m=1), la distribuzione normale di He per settare i pesi iniziali, e, per quanto riguarda gli esempi contenuti nel data set, fornendo alla rete esempi rappresentati esclusivamente da input binari (quindi una matrice di input composta solo da 0 ed 1).

	n°1	n°2	n°3	n°4	n°5	n°6	n°7	n°8	n°9	n°0
media	14.7	7.9	9	8.6	3.4	13.2	14.1	0.6	3	25.5
std	2.8	3	2.4	2.2	0.9	3.2	2.5	0.9	0.9	3.5

Tab.3.5: Dati relativi a 10 simulazioni fatte utilizzando le stesse ed identiche variabili della tabella precedente, input binari ed implementando un' inibizione laterale tra neuroni appartenenti allo stesso livello pari a -0.01.

Visti i dati raccolti fino a questo momento, era necessario cercare di capire per quale motivo i pesi dei neuroni ricordassero più frequentemente alcuni numeri specifici, e quindi comprendere le relazioni che legano i vari esempi proposti nel data set MNIST. Per fare ciò, sono stati effettuati ulteriori test con data set ridotti, comprensivi di due o massimo tre cifre, per vedere se, almeno in questi casi, i neuroni della rete fossero capaci di convergere in modo uniformemente distribuito verso i pattern di input possibili (Tab.3.6 e Tab.3.8). Inoltre sono stati anche effettuati dei tentativi con data set formati da un' unica cifra, per verificare che la rete non avesse problemi a riconoscere nessun numero in particolare, se non confuso con altri numeri inseriti nel data set. In questi casi, è stato interessante notare come alcuni cifre scritte a mano fossero talmente simili ad altre da portare i neuroni fuori strada. Per esempio, allenando la rete con solo la cifra 1, è possibile trovare tra i pattern di convergenza delle immagini simili a 7, oppure tra i 5 trovare alcuni simboli confondibili con un 6 (Fig.3.8).

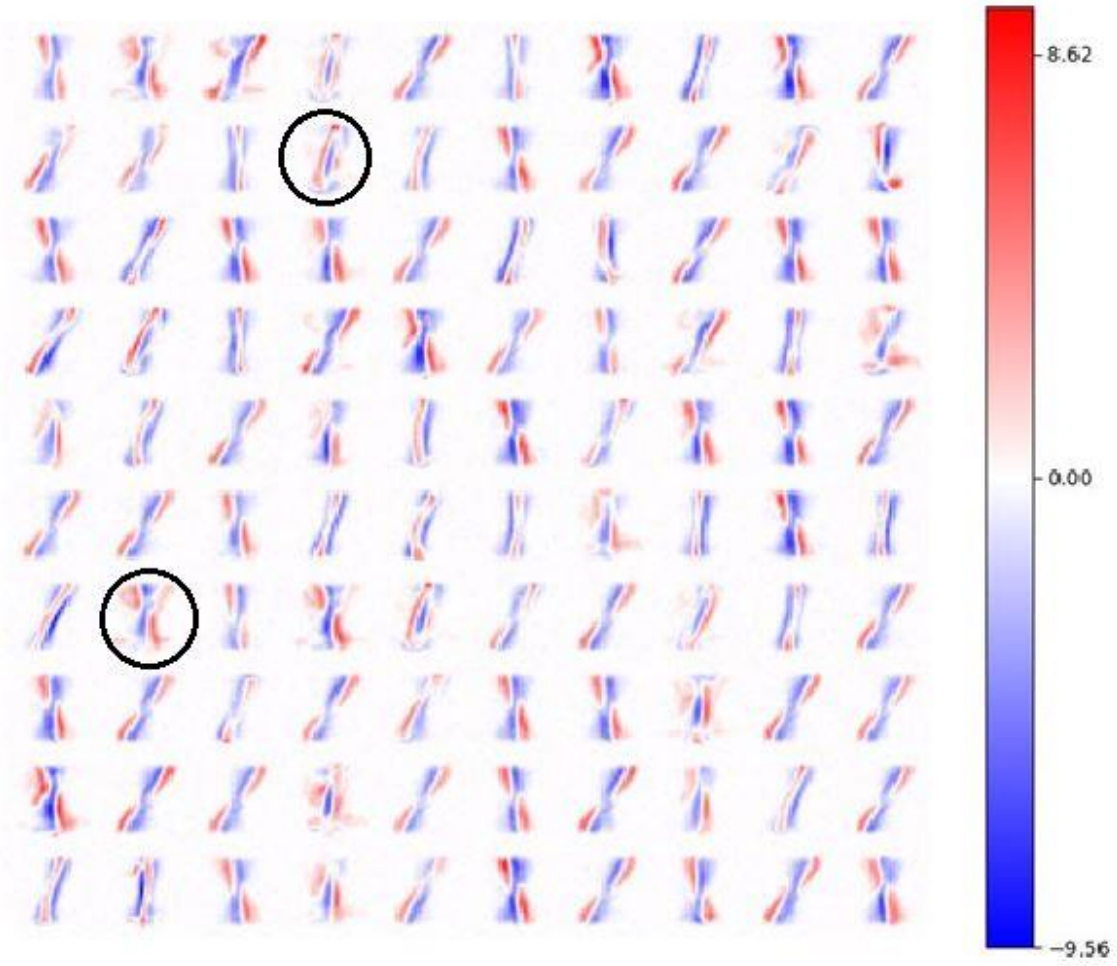

Fig.3.8: L' immagine riporta il risultato dell' addestramento del modello BCM tramite un data set composto solo dai numeri 1 del MNIST. Si possono notare come alcuni numeri assomiglino ad altre cifre, e possano facilmente essere confusi anche ad occhio umano nonostante gli esempi proposti alla rete fossero tutti numeri 1. Gli esempi sono stati passati alla rete tramite matrici di input non binari.

x-y	n°x	n°y	x-y	n°x	n°y	x-y	n°x	n°y
0-1	63	37	**2-7**	61	39	**6-8**	58	42
0-2	48	52	**2-8**	62	38	**6-9**	44	56
0-3	60	40	**2-9**	59	41			
0-4	49	51				**7-8**	59	41
0-5	51	48	**3-4**	48	52	**7-9**	51	49
0-6	50	50	**3-5**	36	64			
0-7	50	50	**3-6**	51	49	**8-9**	39	61
0-8	56	44	**3-7**	45	55			
0-9	55	45	**3-8**	49	51			
			3-9	42	58			
1-2	18	82						
1-3	26	74	**4-5**	61	39			
1-4	24	76	**4-6**	52	48			
1-5	29	71	**4-7**	52	48			
1-6	30	70	**4-8**	58	42			
1-7	25	75	**4-9**	59	41			
1-8	28	72						
1-9	31	79	**5-6**	46	54			
			5-7	40	60			
2-3	63	37	**5-8**	48	52			
2-4	60	40	**5-9**	44	56			
2-5	59	41						
2-6	64	36	**6-7**	53	47			

Tab.3.6: Nella tabella sono riassunti i risultati ottenuti allenando la rete con un data set ridotto, formato da sole sue cifre per volta. Gli esempi sono stati passati alla rete tramite matrici di input non binari.

I dati in tabella 3.6 sono l' esito di un solo addestramento, per cui non possono essere considerati dati molto attendibili, dal momento che la convergenza dei neuroni dipende anche dall' ordine in cui l' unità della rete riceve gli esempi durante l' addestramento. Prendendo sei casi, scegliendo tra le combinazioni più interessanti, e ripetendo solo per questi la simulazione 12 volte, sono state calcolate le medie e le deviazioni standard di questi campioni (Tab.3.7).

n° x + n° y	Media neuroni conversi al n° x	Deviazione standard
n°0 + n°2	56.6	5.4
n°0 + n°3	56.5	5.1
n°0 + n°7	52.8	5.0
n°1 + n°2	25.0	4.6
n°2 + n°6	56.8	5.1
n°3 + n°5	42.5	3.5

Tab.3.7: Media e deviazione standard di 10 simulazioni fatte con data set ridotto ad una coppia di cifre per volta. Gli esempi sono stati passati alla rete tramite matrici di input non binari.

In questo modo , eccetto che per alcuni casi isolati e rari, può esser ipotizzato che tra il numero di neuroni che converge ad una cifra ed il numero di neuroni che converge all' altra non corrono più di 20 unità. Guardando i dati raccolti in tabella 3.6, se non consideriamo i casi relativi al numero 1, per la maggior parte degli abbinamenti abbiamo una distribuzione più uniforme rispetto ai dati della tabella 3.3, dove l' 8 per esempio raggiunge una ricorrenza media pari a 0.8. Questo può essere interpretato nel seguente modo: i neuroni della rete, se devono distinguere esclusivamente due numeri, ottengono risultati migliori rispetto a doverne distinguere un numero maggiore. Infatti, se invece che addestrare la rete con due sole cifre ne aumentiamo la quantità, alcune di queste prenderanno il sopravvento e diventeranno un punto di convergenza dei pesi preferito rispetto ad altre, creando un divario bene evidente. Per rendere meglio l' idea, in tabella 3.8 sono riportati i risultati relativi ad alcuni test a tre cifre ripetuti quattro volte ciascuno, in modo da capirne a grandi linee il comportamento. In questi esempi abbiamo solitamente un numero che attrae più neuroni rispetto agli altri due (come lo 0 in 0-6-7 o in 0-8-3), oppure due cifre che si spartiscono in modo equo la maggior parte dei neuroni e lasciano la terza da parte (come nel caso di 6-7-9). Quindi maggiori sono le cifre da distinguere, meno uniformemente convergono i neuroni.

x-y-z	n°x	n°y	n°z
0-6-7	33	35	34
0-6-7	41	30	29
0-6-7	41	35	24
0-6-7	45	26	29
6-7-9	43	40	17
6-7-9	43	42	14
6-7-9	53	34	13
6-7-9	47	37	16
0-8-3	47	32	21
0-8-3	54	24	22
0-8-3	48	29	23
0-8-3	50	27	23

Tab.3.8: La tabella raccoglie i risultati relativi ad alcuni test a tre cifre, riportati come esempio, ripetuti ciascuno 4 volte. Gli stimoli sono stati passati alla rete tramite matrici di input non binarie.

Per rendere più consistenti le osservazioni fatte fino ad ora, i risultati ottenuti sono stati confrontati con altre analisi effettuate sul data set MNIST. All' interno dell' articolo di M.Wu e Z.Zhang intitolato "*Handwritten Digit Classification using the MNIST Data Set*" [18], il MNIST è stato studiato tramite sei classificatori diversi. Gli istogrammi riportati di seguito (Fig.3.9) indicano gli errori associati a ciascuna cifra (il 10 indica lo 0) nel venir riconosciuta dal classificatore in uso. I classificatori con risultati migliori, in termini di percentuale di errore, sono quelli relativi agli istogrammi in seconda fila (vedi tabella 2 dell' articolo), per cui ha senso concentrarsi principalmente si di essi. Si presentano due evidenti picchi: il numero 7 ed il numero 9 nella fase di

addestramento, 8 e 9 nella fase di test. Un' ulteriore evidenza è la bassissima percentuale di errore con il quale viene sempre riconosciuto lo 0 (10 in figura), o il numero 6 nella fase di test. Questo non giustifica il fatto che alcuni pattern come il 9 siano poco presenti tra i punti di convergenza dei pesi sinaptici nei risultati raccolti, però conferma che tra loro le immagini siano in certi casi parecchio confondibili anche per altri sistemi, e questo rende il compito della rete BCM più complesso. Un altro modo per rappresentare questo concetto può essere quello di sfruttare una matrice di confusione (Tab.3.9).Anche in questo caso possiamo notare come il 9 ed il 7 siano i due numeri maggiormente confusi con altre cifre.

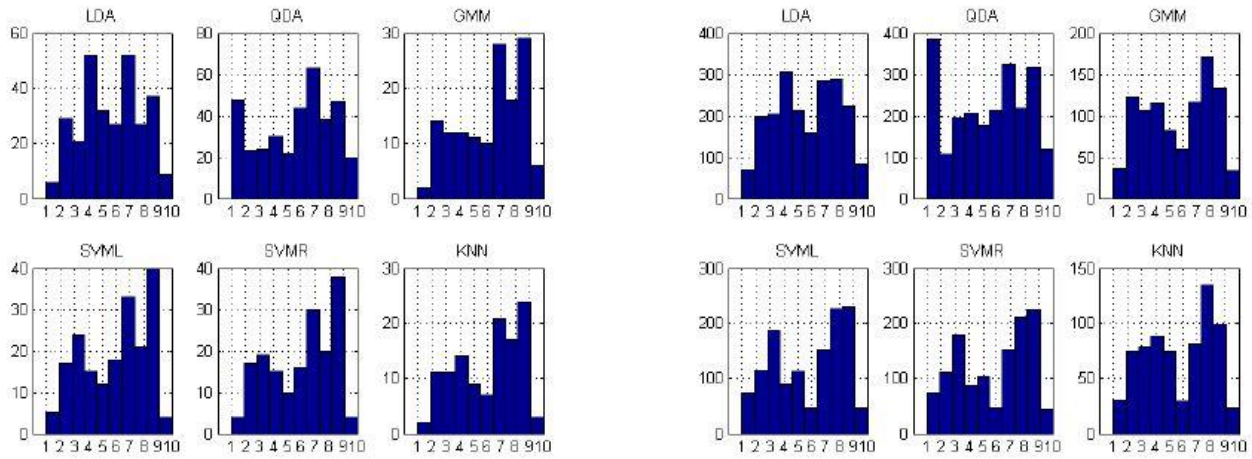

Fig.3.9: Gli istogrammi di sinistra rappresentano la frequenza di errore durante la fase di allenamento, mentre quelli di destra la frequenza di errore durante la fase di test.
I classificatori utilizzati sono i seguenti: analisi discriminante lineare (LDA), analisi discriminante quadratica(QDA), Gaussian Mixture Models (GMM), Support Vector Machine con una funzione kernel lineare(SVML), SVM con una funzione kernel a base radiale (SVMR) and k-NN(con k = 3).

Predicted\True	1	2	3	4	5	6	7	8	9	10
1	99.82%	0.29%		0.20%		0.31%	0.97%		0.50%	
2	0.09%	98.64%	0.20%				0.68%	0.10%		0.10%
3			98.91%		0.90%		0.19%	0.31%	0.79%	0.10%
4		0.10%		98.78%				0.31%	0.50%	
5					98.88%	0.31%		0.21%	0.10%	
6	0.09%	0.10%		0.10%	0.11%	99.06%				0.20%
7		0.68%	0.50%				97.37%	0.10%	0.50%	
8		0.10%	0.20%			0.10%	0.10%	98.36%	0.50%	0.20%
9			0.20%	0.92%			0.68%	0.31%	97.03%	
10		0.10%			0.11%	0.21%		0.31%	0.10%	99.39%

Tab.3.9: La matrice di confusione riporta la frequenza con le quali un numero viene scambiato con un altro. I dati sono stati raccolti tramite il classificatore Gaussian Mixture Models.

Un ulteriore confronto può essere fatto tramite una riduzione delle dimensioni del data set, passando da 728 dimensioni per ogni esempio del MNIST a numeri molto minori. Le tecniche prese

in considerazione sono state l' analisi delle componenti principali (PCA) e la t-distributed stochastic neighbor embedding (t-SNE). Lo scopo della PCA è quello di semplificare il numero di variabili per descrivere le componenti del data set, cercando di mantenere più informazioni possibili nonostante la riduzione delle dimensioni del sistema (Fig.3.10). La seconda tecnica utilizzata è quella della t-SNE, il cui scopo è quello di rappresentare i punti in uno spazio a dimensioni ridotte, ma cercando di mantenere una distanza tra i punti proporzionata a quella nello spazio originale (nel nostro caso, uno spazio originale di 728 dimensioni).

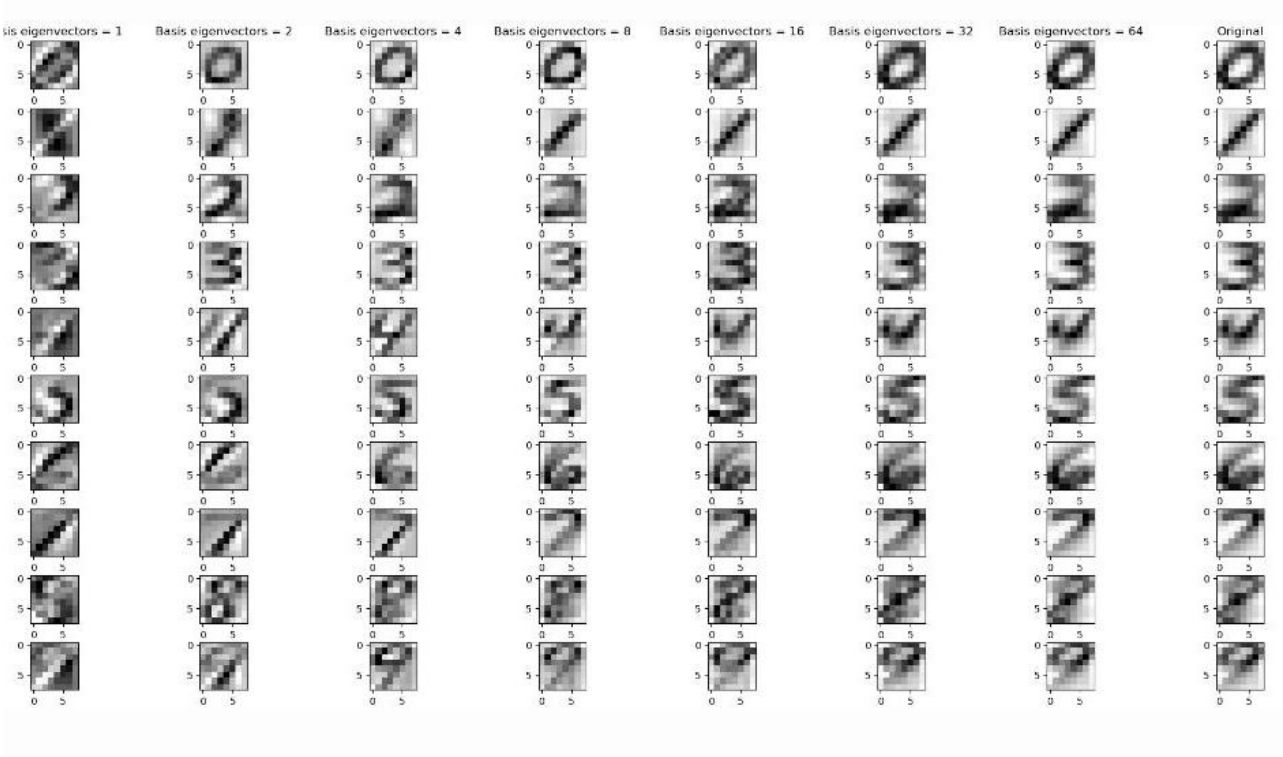

Fig.3.10: Nell' immagine sono rappresentate tutte le cifre, partendo dal considerare una sola dimensione (a sinistra) fino ad arrivare a considerarne 64 (penultima colonna a destra). L' immagine ed i dati per ottenerla sono stati pubblicati il 16 Nov 2019 da Casey Juanxi Li sul sito casey.li.

Nella figura 3.10 è visibile anche ad occhio nudo come i tratti di alcuni numeri, se semplificati, possono facilmente essere confusi con quelli di altri, e quindi non bisogna stupirsi se anche il modello BCM utilizzato fatichi a distinguere numeri differenti. Per esempio, il numero 5, considerando due sole dimensioni, rappresenta meglio la cifra zero, e lo stesso vale per l' 1 o il 9, che potrebbero essere confusi con un 7 oppure con un 4. L' immagine seguente riporta i risultati dell' applicazione di PCA e t-SNE sul MNIST (Fig.3.11). I grafici rappresentano gli esempi del data set su due sole dimensioni per permetterne la rappresentazione. Nonostante ciò, come notiamo in figura 3.10, una buona parte delle caratteristiche della maggioranza delle cifre (0,2,3,6,7,8) viene mantenuta anche con solo due dimensioni, e per questo i grafici qui sotto riportati riescono a dare un' idea delle relazioni tra i numeri del data set. Le immagini possono essere interpretate in questo modo: numeri vicini condividono caratteristiche simili, mentre le cifre isolate possiedono caratteristiche più difficili da confondere, e di conseguenza verranno meno scambiati con simboli

relativi a numeri differenti. Come per gli istogrammi in figura 3.8, lo zero in entrambi i grafici ricopre una posizione molto laterale e lontana dal centro, mentre per esempio l' 8 resta piuttosto centrale. Ecco perché all' interno degli istogrammi, durante la fase di test, rappresenta sempre un picco.

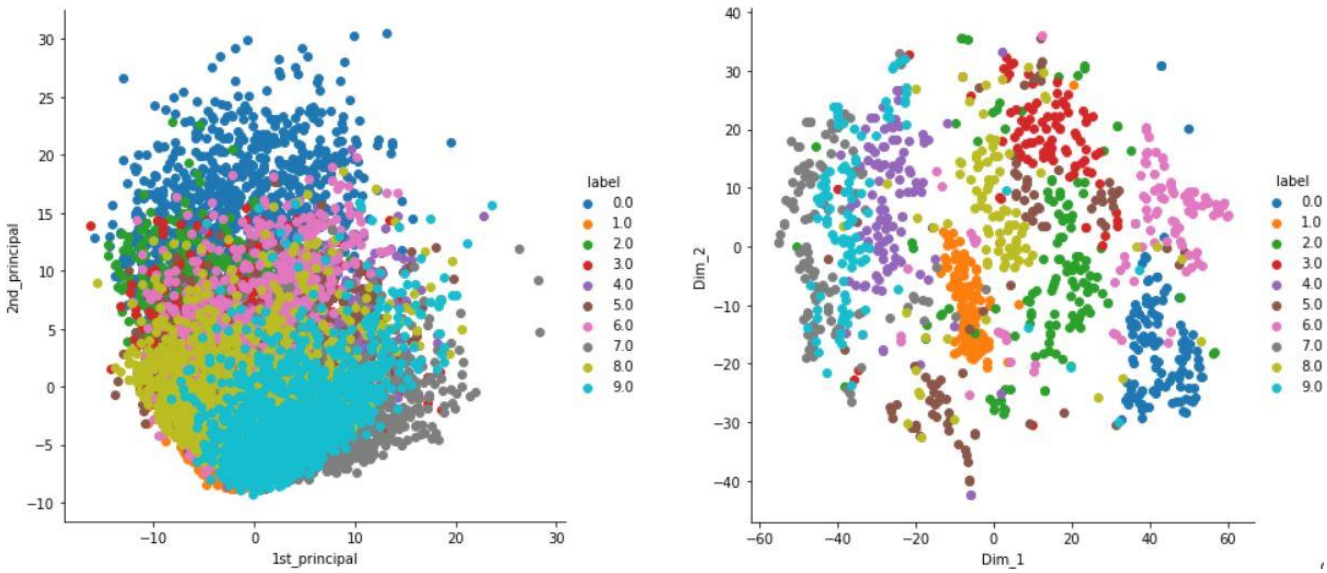

Fig.3.11: Il grafico di sinistra rappresenta l' analisi effettuata tramite PCA, mentre il grafico di destra l' analisi effettuata tramite t-SNE. Ogni punto colorato indica una certa cifra, come indica la legenda a lato dei grafici. Le due immagini sono state caricate sul sito Medium da Rana singh.

Paragrafo 3.3.2
Risultati della proposta di Hopfield e Krotov

Per quanto riguarda il confronto tra il modello proposto nel paragrafo 3.2 e le reti addestrate tramite apprendimenti supervisionati, nel resto del paragrafo verranno inseriti i risultati riportati all' interno dell' articolo di Hopfield e Krotov del 2019.
Il primo set di dati utilizzato sulla rete è il MNIST, che consiste in 60.000 esempi: 50.000 utilizzati per l' addestramento dei pesi, 10.000 per regolare p, k e Δ. Poi questi esempi vengono rimescolati per riproporre alla rete 60.000 casi e concludere l' allenamento. Gli ultimi 10.000 sono utilizzati come verifica. In figura (Fig.3.6) sono rappresentati i pesi di due reti composte ciascuna da 2.000 unità nascoste, la prima del tipo proposto da Hopfield e Krotov, la seconda allenata con la retro propagazione dell' errore. Se la rete allenata tramite retro propagazione dell' errore durante la fase di allenamento raggiunge un errore pari allo 0.00%, la rete biologicamente plausibile non scende sotto lo 0.40%. Nonostante ciò, durante la fase di test quest' ultima raggiunge una percentuale d' errore addirittura più bassa della concorrente. Infatti si parla di un errore pari a 1.46%, poco sotto all' 1.50% della rivale. Di conseguenza, può essere concluso che la rete di Hopfield e Krotov riesce ad estrarre delle caratteristiche degli esempi particolarmente importanti, che le permettono di competere nel classificare le immagini del data set con la rete che sfrutta la retro propagazione dell' errore.

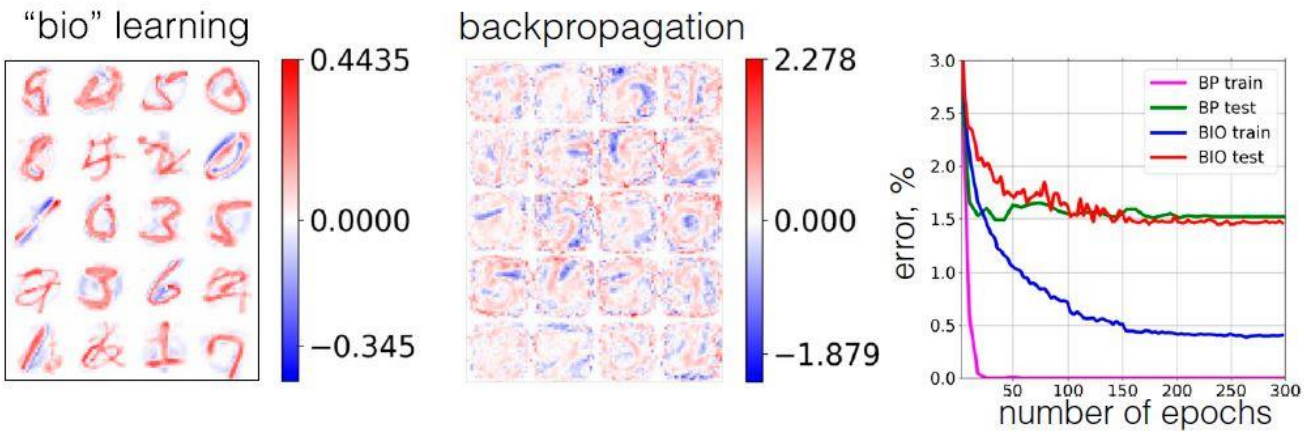

Fig.3.6: l' immagine riporta i risultati ottenuti testando una rete allenata con il metodo di Hopfield e Krotov ed una rete supervisionata. In particolare, sono stati selezionati in modo casuale 20 neuroni sui 2.000 della rete, e le immagini rappresentano come i loro pesi si sono specializzati verso un pattern piuttosto che un altro. Sul grafico a destra sono riportati gli andamenti delle percentuali di errore in fase di addestramento ed in fase di test per entrambi i sistemi.

Il secondo set di dati sfruttato si chiama CIFAR-10. CIFAR-10 contiene 60.000 immagini a colori suddivise in 10 classi diverse. Le 10 classi rappresentano aeroplani, automobili, uccelli, gatti, cervi, cani, rane, cavalli, navi e camion. Questa volta gli esempi totali sono 50.000, divisi in 45.000 di allenamento e 5.000 per testare la macchina. Una volta regolati p, k e Δ, si passa a 50.000 esempi per concludere l' apprendimento e 10.000 esempi per verificarne l' efficacia. Come in precedenza, le due differenti reti composte da 2.000 unità nascoste sono state addestrate e messe alla prova per confrontarne i risultati. Su un tempo pari a 100 epoche, durante la fase di test il metodo supervisionato ottiene un errore pari a 44.74% mentre quello privo di supervisione pari a 49.25%. In questo caso la rete allenata tramite retro propagazione può essere ulteriormente migliorata per arrivare ad un 41.32%. Il confronto tra i risultati ottenuti viene mostrato in figura (Fig. 3.7).

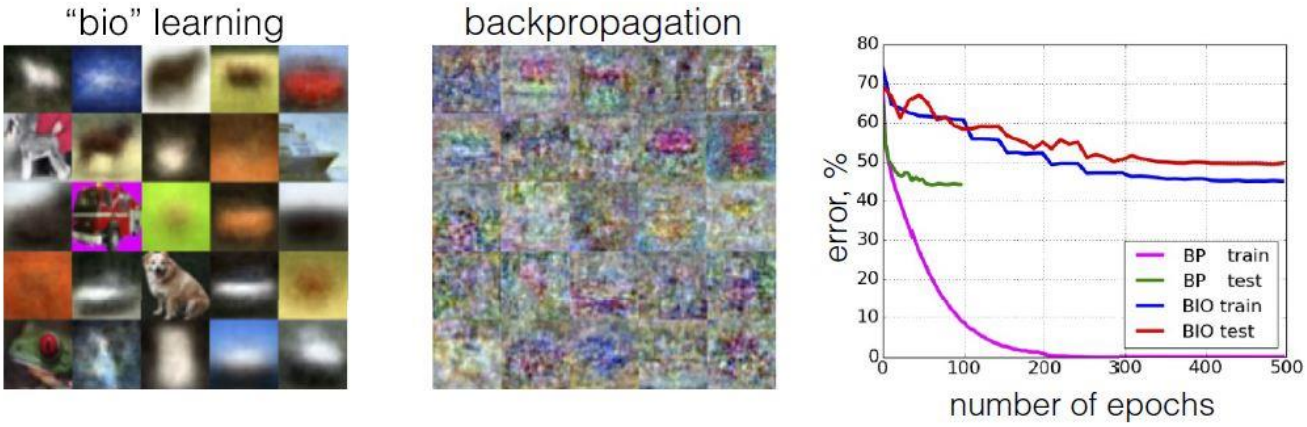

Fig.3.7: In questa figura vengono confrontati i risultati ottenuti da due distinti sistemi, uno allenato con il modello di Hopfield e Krotov, l' altro tramite retro propagazione dell' errore. Nell' immagine sono mostrati i pesi di convergenza di 25 neuroni scelti casualmente tra i 2.000 della rete. Sul grafico a destra sono riportati gli andamenti delle percentuali di errore in fase di addestramento ed in fase di test per entrambi i sistemi. Da notare che il test sulla retro propagazione è stato fatto solo su 100 epoche, tempo sufficiente per ottimizzare l' errore della macchina (i dettagli di questa osservazione sono spiegati nell' appendice B dell' articolo).

Paragrafo 3.4
Conclusioni

Durante la parte iniziale della tesi sono stati introdotti i concetti di rete neurale artificiale, di apprendimento supervisionato e non supervisionato ed è stato spiegato cosa si intende con il termine rete neurale multistrato. È stata introdotta la retro propagazione dell' errore, uno degli apprendimenti supervisionati maggiormente usati per l' addestramento di reti artificiali supervisionate, ed è stato mostrato l' esempio delle reti ricorrenti come reti supervisionate che sfruttano questo algoritmo per sviluppare i propri pesi sinaptici.

Durante il paragrafo precedente, abbiamo riportato i test effettuati su una rete BCM allenata tramite il data set MNIST. È chiaro come i pesi dei neuroni della rete tendessero a convergere verso alcuni numeri precisi. Per chiarirne il motivo, è stato necessario approfondire il legame tra le immagini del data set per comprenderne meglio le relazioni. Nel MNIST sono raccolti una grande quantità di simboli scritti a mano, e nonostante rappresentino solo dieci diverse cifre, la calligrafia di una persona può variare notevolmente da quella di un' altra. Come se non bastasse, ci sono numeri che tendono ad assomigliarsi notevolmente, come il 5 ed il 6, oppure l' 1 ed il 7. Di conseguenza, questi due fattori portano i neuroni della rete a confondere tra loro i diversi numeri. Se utilizziamo esempi relativi a due sole cifre, sono maggiori i casi in cui troviamo una buona distribuzione tra le due variabili con le quali alleniamo la rete, ma inserendo un' ulteriore cifra all' interno del data set, i neuroni non si specializzano più con distribuzioni così omogenee, e tendono a preferire sempre gli stessi numeri come punti di convergenza rispetto agli. Come si è visto, semplificando il problema ed utilizzando matrici composte da input binari il risultato ottenuto migliora, ed i neuroni convergono in modo più eterogeneo ai diversi pattern. Inoltre, inserendo le inibizioni laterali all' interno dell' algoritmo, si raggiunge un livello di eterogeneità in termini di convergenza dei pesi ancora migliore. Infatti in tabella 3.5, senza considerare la cifra 8, tutte le altre sono presenti con una media superiore a 3 ricorrenze, compreso il 5 ed il 9 che in tabella 3.3 non superavano rispettivamente un valore medio pari a 2.6 ed 1.5 ricorrenze. Possiamo infine dire che i pesi dei neuroni del modello BCM non riescono a convergere secondo una distribuzione perfettamente uniforme alle 10 cifre del data set MNIST a causa della forte somiglianza di alcune immagini al suo interno.

Per quanto riguarda invece il sistema neurale proposto da Hopfield e Krotov nel 2019, le analisi raccolte nel paragrafo precedente rappresentano un' ottimo punto di arrivo per questa trattazione: una rete biologicamente plausibile e capace di portare a termine tanto bene alcuni compiti quanto le reti supervisionate. Riassumendo le caratteristiche che rendono questa rete così simile al sistema nervoso, tale macchina impara in modo non supervisionato, sfruttando un processo di variazione dei pesi locale ed utilizzando un sistema di inibizione laterale. Infatti, riprendendo il concetto di selettività introdotto dal sistema BCM, quando un neurone viene attivato in modo particolarmente forte da uno stimolo e la sua selettività aumenta, inibisce gli altri tramite una funzione di attivazione che non permette loro di competere per lo stesso input.

Un' osservazione necessaria è la seguente: la rete si comporta bene se testata con set di allenamento del tipo MNIST oppure CIFAR-10, ma non si può prevedere lavori in modo ugualmente soddisfacente in altre occasioni. In effetti un simile discorso viene affrontato anche all' interno dell' articolo "*Assessing the scalability of biologically-motivated deep learning algorithms and architectures*" del 2018 [19]. Anche in questo caso la retro propagazione dell' errore viene confrontata con altri metodi di allenamento, biologicamente plausibili, ottenendo ottimi risultati durante i test che sfruttavano data set MNIST e CIFAR-10. Nonostante questo, provando le reti attraverso un data set ImageNet è facile rendersi conto di quanto ancora sia ampio il divario tra insegnamenti supervisionati e non (Tab.3.8).

METHOD	TOP-1	TOP-5
DTP, PARALLEL	98.34	94.56
DTP, ALTERNATING	99.36	97.28
SDTP, PARALLEL	99.28	97.15
FA	93.08	82.54
BACKPROPAGATION	**71.43**	**49.07**
BACKPROPAGATION, CONVNET	**63.93**	**40.17**

Tab.3.8: La tabella riassume i risultati dei test su ImagineNet ai quali sono state sottoposte diverse reti biologicamente plausibili (per di più varianti della DTP) e due modelli di reti allenate tramite retro propagazione dell' errore. I risultati sono rappresentati sotto forma di errori percentuale. Il divario tra le prime 4 righe (apprendimenti non supervisionati) e le ultime due (apprendimenti supervisionati) è evidente.

L'idea di Hopfield e Krotov si dimostra così interessante soprattutto per i risultati ottenuti. Infatti evidenziano che, in certi casi, reti supervisionate e non possono effettivamente competere, mentre per anni si dette per scontato che le macchine maggiormente performanti fossero a prescindere quelle allenate tramite retro propagazione dell' errore.
Rosenblatt sperava che un giorno il suo percettrone permettesse di comprendere meglio lo sviluppo ed il funzionamento del sistema nervoso umano. Durante la trattazione è stato illustrato come, da quella semplice struttura, si siano poi sviluppate idee molto più articolate e reti assai complesse, spesso però lontane dal concetto di "biologicamente plausibile". Sono gli studi come quello del 2019 che permettono di fare passi avanti nel descrivere i processi delle reti neurali biologiche, e che magari un giorno permetteranno di capire e conoscere con precisione queste macchine così efficienti ed articolate.

Appendice A:

L' equazione [2.14] può essere riscritta come :

$$\theta_j(n) = \frac{\partial E(n)}{\partial y_j(n)} \times \varphi_j'\left(v_j(n)\right), \qquad [A.1]$$

mentre integrando l' equazione [2.7] si ottiene:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \times \frac{\partial e_k(n)}{\partial y_j(n)} \qquad [A.2]$$

dove l' indice j rappresenta il neurone nascosto, mentre l' indice k della sommatoria identifica i neuroni del livello di output. Utilizzando nuovamente la regola della catena viene trovato il seguente risultato:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \times \frac{\partial e_k(n)}{\partial v_k(n)} \times \frac{\partial v_k(n)}{\partial y_j(n)} \qquad [A.3]$$

Per come è stato definito il segnale di errore prodotto come output dal neurone k, la derivata $\frac{\partial e_k(n)}{\partial v_k(n)}$ può immediatamente essere scritta come $-\varphi_k'\big(v_k(n)\big)$. Differenziando l' equazione [2.8] per cercare la derivata $\frac{\partial v_k(n)}{\partial y_j(n)}$, si arriva al risultato:

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k e_k(n) \times \varphi_k'\big(v_k(n)\big) \times w_{kj}(n) \qquad [A.4]$$

E per finire basta inserire l' equazione [A.1] nella [2.21] per poter scrivere il gradiente locale per un neurone j appartenente ad uno strato nascosto come:

$$\theta_j(n) = \varphi_j'\left(v_j(n)\right) \times \sum_k \theta_k(n) \times w_{kj}(n)\,. \qquad [A.5]$$

Appendice B:

Ipotizziamo una distribuzione discreta per i segnali di input formata da k possibili stimoli, con una possibilità pari ad $1/k$ per ciascun segnale. I risultati che possiamo ottenere attraverso stimoli di questo tipo sono i seguenti:

- i punti di equilibrio sono localmente stabili se e solo se sono i punti di selettività più alta rispetto alla distribuzione di input utilizzata;
- dato uno stato iniziale $m(t)$, la probabilità che questo converga ad un punto di massima selettività in un tempo t che va all' infinito è pari al 100%.

Le prime osservazioni verranno fatte ipotizzando due soli input distinti, d^1 e d^2, con una probabilità pari ad ½ per entrambi. La massima selettività vale quindi ½ se il neurone risponde a solo uno dei due stimoli, mentre è minima quando il neurone risponde indifferentemente ad entrambi gli stimoli. A questo punto possiamo introdurre il primo lemma seguito dal primo teorema:

lemma 1: siano d^1 e d^2 linearmente indipendenti e tali che $P[d^1] = P[d^2] = 1/2$. Allora per ogni valore di φ che rispetti l' equazione [3.9] esisteranno 4 punti fissi per l' equazione [3.5] indicati come $m^0, m^1, m^2, m^{1,2}$, rispettivamente con selettività pari a $0, \frac{1}{2}, \frac{1}{2}, 0$;
teorema 1: oltre alle condizioni inserite nel lemma 1, valga $cos(d^1, d^2) \geq 0$. Allora gli unici due punti fissi stabili sono m^1 ed m^2, e qualsiasi sia lo stato di partenza, i pesi sinaptici convergeranno quasi sicuramente ad uno dei due punti fissi stabili.

Il teorema richiede quindi che i due segnali di input non siano ortogonali. Mentre il teorema appena citato assicura la convergenza ad un punto stabile, il secondo teorema assicura che un neurone, una volta sviluppata la propria selettività a certi input, non possa modificarla:

teorema 2: sotto le stesse condizioni del teorema 1, esiste attorno ad ogni punto fisso stabile una regione relativa a tale punto. Se lo stato del neurone entra in una di queste regioni, è destinato a convergere in quel punto e non può più uscire dalla regione. (Fig.B1)

Tutte queste osservazioni possono anche essere fatte per una gamma di input maggiore, passando da d^1, d^2 a $d^1 \dots d^k$, situazione ovviamente più vicina a quella reale. Così ritroviamo il primo lemma nella forma:

lemma 2: siano $d^1, \dots, d^k$ linearmente indipendenti e tali che $P[d^1] = \cdots = P[d^k] = 1/k$. Allora per ogni valore di φ che rispetti l' equazione [3.9] esisteranno 2^k punti fissi per l' equazione[3.6] con selettività pari a $0, \frac{1}{k}, \frac{2}{k}, \dots, (k-1)/k$. In particolare i punti fissi con selettività $(k-1)/k$ saranno k.

In questo caso la selettività massima sarà $(k-1)/k$ che indica una risposta positiva del neurone ad un solo input tra $d^1, \dots, d^k$.

Infine il primo teorema può essere reinterpretato come:

teorema 3: oltre alle condizioni inserite nel lemma 2, siano $d^1, \dots, d^k$ *tutti ortogonali tra loro oppure molto vicini ad esserlo. Allora tutti i* k *punti fissi con selettività massima sono punti stabili, e qualsiasi sia lo stato di partenza, i pesi sinaptici convergeranno quasi sicuramente ad uno di questi punti.*

Sono presenti anche in questo caso delle regioni attorno ai punti stabili che intrappolano lo stato del sistema al loro interno e non gli permettono di variare la propria selettività verso altri punti.
In generale simulazioni fatte al computer suggeriscono che, anche nel caso in cui $d^1 \dots d^k$ non fossero quasi ortogonali tra loro, i punti fissi di massima selettività resterebbero stabili, ma analiticamente questo punto non verrà trattato.

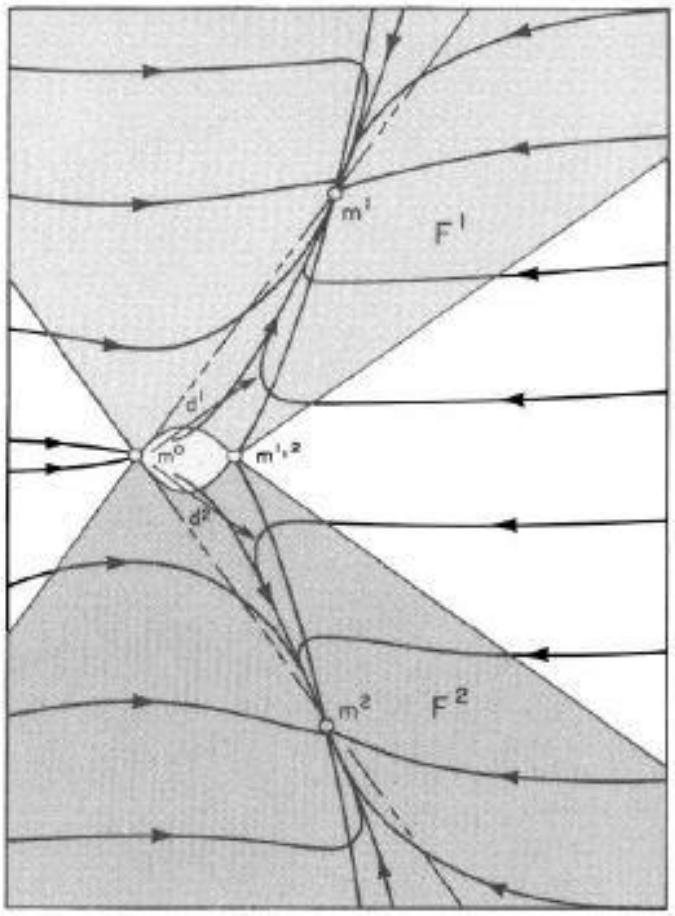

Fig.B1: *Nel grafico sono rappresentate le due aree attorno ai punti fissi stabili* m^1, m^2 *con un colore più scuro, mentre l' area tra i punti instabili* $m^0, m^{1,2}$*con un colore chiaro. Questo serve ad indicare che lo stato del sistema, una volta entrati in un' area scura, è destinato a convergere al punto stabile appartenente a quell' area. Le linee sottolineano il comportamento dello stato del sistema all' interno dello spazio dei pesi sinaptici.*

Appendice C:

Nel 1982 Erkki Oja pubblica un' articolo intitolato "A Simplified Neuron Model as a Principal Component Analyzer" [16], all' interno del quale ricava la celebre regola di Oja per lo sviluppo dei pesi sinaptici. In questa appendice saranno riportati solo i passaggi principali per dare un' idea delle operazioni da seguire per ottenere tale regola.
Si comincia da un classico sistema formato da un neurone che computa i segnali ricevuti nel modo seguente:

$$O = \sum_{i=1}^{n} \mu_i \varepsilon_i \qquad \text{[C.1]}$$

dove O rappresenta il segnale di output, μ_i i pesi delle sinapsi con gli n neuroni del livello precedente ed ε_i gli input ricevuti.
Viene utilizzata poi un' equazione di apprendimento pari a :

$$\mu_i(t+1) = \frac{\mu_i(t) + \gamma\, O(t)\varepsilon_i(t)}{\{\sum_{i=1}^{n}[\mu_i(t) + \gamma\, O(t)\varepsilon_i(t)]^2\}^{1/2}} \qquad \text{[C.2]}$$

con γ che rappresenta semplicemente uno scalare. A questo punto, assumendo che γ sia abbastanza piccolo, basta espandere in serie di Taylor ed ottenere:

$$\mu_i(t+1) = \mu_i(t) + \gamma\, O(t)[\varepsilon_i(t) - O(t)\, \mu_i(t)] + \vartheta(\gamma^2) \qquad \text{[C.3]}$$

la quale, trascurando il termine $\vartheta(\gamma^2)$, può essere riscritta come:

$$\mu_i(t+1) - \mu_i(t) = \gamma\, O(t)[\varepsilon_i(t) - O(t)\, \mu_i(t)]\ . \qquad \text{[C.4]}$$

È possibile ricondurre questo risultato all' espressione [3.13] confrontando singolarmente i termini nelle stesse posizioni, che a grandi linee dipendono dai medesimi parametri.

Bibliografia:

[1] W.S.McCulloch e W.H.Pitts, “A logical calculus of the ideas immanent in nervous activity”, Bulletin of Mathematical Biophysics 5, 115–133 (1943)

[2] F. Rosenblatt, “Principles of Neurodynamics”, Spartan Books -1962.

[3] EL Bienenstock, LN Cooper and PW Munro “Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex”, J Neurosciences 1982 Jan;2(1):32-48.

[4] D. Krotov and J.J.Hopfield, “Unsupervised learning by competing hidden units”, PNAS April 16, 2019 116 (16) 7723-7731

[5] S.Haykin, “Neural Networks and Learning Machines”, S.Haykin, ISBN 10: 0131471392 ISBN 13: 9780131471399
Casa editrice: Pearson College Div, 2008

[6] J.R.Daube and D.I.Rubin, “Clinical Neurophysiology”, Oxford press, Contemporary Neurology Series, third edition, 2009.

[7] T.Cantelmi, M.De Murtas, E.Ernani, L.Zanello “Modellizzazione neurale dei disordini mentali - Deterioramento della Memoria nell'Alzheimer e disturbo formale del pensiero nella schizofrenia.”, Scione Editore, Roma 2000

[8] D. Hebb “The Organization of Behavior”, Wiley: New York; 1949

[9] I.Aleksander e H.Morton , “An introduction to neural computing” , Publisher: 2Van Nostrand Reinhold Co.115 Fifth Ave. New York, NYUnited States ISBN:978-0-442-31218-31990.

[10] T.Kohonen, E.Oja, O.Simula, A.Visa e J.Kangas **“**Engineering applications of the self-organizing map”, Computer Science, 1996 - DOI:10.1109/5.537105 Corpus ID: 61685094

[11] R.S.Sutton e A.G.Barto “Reinforcement Learning: An Introduction”, Second Edition MIT Press, Cambridge, MA, 2018

[12] F.Rosenblatt**,**“The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain”, Psychol Rev. Nov;65(6):386-408. 1958.

[13] V.Hayek, “The Sensory Order” University of Chicago Press, 1952.

[14] Y.LeCun, Y.Bengio e G.Hinton, “Deep learning”, *Nature*, volume 521, pages436–444 (2015)

[15] U.Kamath, J.Liu e J.Whitaker, “Deep Learning for NLP and Speech Recognition”, Springer 2019.

[16] E.Oja, “A Simplified Neuron Model as a Principal Component Analyzer”, J Math Biol. 1982;15(3):267-73.

[17] L.N.Cooper e C.L.Scofield, “Mean-field theory of a neural network”, Proc Natl Acad Sci U S A. 1988 Mar; 85(6): 1973–1977.

[18] M.Wu e Z.Zhang, “*Handwritten Digit Classification using the MNIST Data Set*”, Academia.edu 2010.

[19] S.Bartunov, A.Santoro, B.A.Richards, L.Marris, G.E.Hinton e T.Lillicrap, “*Assessing the scalability of biologically-motivated deep learning algorithms and architectures*”, 32nd Conference on Neural Information Processing Systems

[20] N. Curti, S. Gasperini, M. Ceccarelli, "*plasticity - Unsupervised Neural Networks with biological-inspired learning rules*", Github, https://github.com/Nico-Curti/plasticity, 2020