Francesco Sisini e Valentina Sisini

Reti neurali non supervisionate: il cognitrone di Fukushima

con codice in linguaggio C

Edizioni: i Sisini Pazzi, 2019

INDICE

2 - I NEURONI DELLA VISTA

L'argomento che viene trattato in questo libro ha senza dubbio un grande fascino. La vista, per chi ha la fortuna di esserne provvisto, è un senso di straordinaria importanza che permette e semplifica tantissime attività quotidiane. L'idea di comprenderne i meccanismi naturali per riportarli su sistemi automatici è molto attraente specie in questi anni (per chi scrive corre il 2019) in cui si parla sempre più spesso di robot umanoidi.

Nei decenni passati si sono fatti passi fondamentali per comprendere il meccanismo con il quale i segnali visivi viaggiano come onde luminose (onde elettromagnetiche) e di come tali onde trasmettano detti segnali alle cellule nervose degli occhi. La conoscenza acquisita ha permesso di replicare i meccanismi fisiologici presenti nell'occhio in sistemi elettronici, producendo così gli attuali sensori ottici come ad esempio i CCD e CMOS usati nelle telecamere digitali. Parallelamente è stato studiato anche il meccanismo con il quale, all'interno del cervello, gli stimoli luminosi, ricevuti dalle cellule dell'occhio, vengono trasformati in concetti ed informazioni.

Per comprendere il fenomeno della vista si devono comprendere quindi due fenomeni, uno è il trasporto e la ricezione del segnale visivo, l'altro è l'elaborazione a la concettualizzazione del segnale in informazione.

Dagli anni 60 ad oggi, i dispositivi elettronici e informatici hanno sfruttato diversi tipi di sensori ottici per acquisire segnali luminosi, e algoritmi di tipo task specific uniti a strutture dati *rigide* per trasformare i segnali in informazioni. Per fare un esempio possiamo pensare alle fotocellule catarifrangenti o a barriera che da decine di anni si usano come rilevatori di presenza o di movimento.

Usando algoritmi task specific e strutture dati rigide si possono processare dei segnali a patto però di conoscerne a priori il contenuto

in un software, per replicare almeno in parte, le capacità umane di produrre concetti da immagini anche in assenza di un modello interpretativo costruito a priori. Pensiamo ad un bambino che appena nato vede il viso della mamma e a forza di vederlo capisce da solo che quello è un viso, che la mamma ci vede dagli occhi e che parla dalla bocca. Tutto questo senza che nessuno possa spiegarglielo in quanto non sa ancora parlare e ascoltare.

Strati di neuroni

In che modo il cervello umano provvede alla vista? Questa è la classica domanda semplice con una risposta molto complicata. I meccanismi esatti con cui a noi e alle altre specie animali è dato il senso della vista non sono completamente conosciuti, ma neanche completamente ignoti. Per dare un quadro schematico di come funzioni la vista proviamo a fare un esempio. Supponiamo di essere in una stanza bianca, dove su un tavolo bianco c'è una mela rossa. Supponiamo inoltre che la stanza sia illuminata. La luce che illumina la stanza ha natura elettromagnetica, cioè per la fisica la luce è composta da onde elettromagnetiche che si possono anche chiamare fotoni (la relazione esistente tra le onde e i fotoni è un argomento complesso della fisica quantistica che non è utile ai nostri scopi e non verrà approfondito nel seguito). Le onde vengono riflesse sia dalle pareti della stanza che dalla mela. La maggior parte delle onde subisce diverse riflessioni e per ogni riflessione anche un po' di assorbimento, così ogni onda via via tenderà ad attenuarsi e a 'sparire' o meglio ad essere assorbita e trasformata in calore: questo è il motivo per cui, se la notte spegniamo la luce, dopo un intervallo di tempo impercettibile la stanza si fa buia. Tra tutte le onde presenti nella stanza, una frazione minima verrà anche riflessa verso i nostri occhi. Tra le onde

che raggiungeranno i nostri occhi ce ne saranno sia di riflesse dalle pareti che di riflesse dalla mela. Queste onde attraverseranno il cristallino degli occhi per poi raggiungere lo strato di cellule nervose (neuroni) che compongono la retina. I neuroni della retina vengono eccitati dagli impulsi luminosi ricevuti e reagiscono producendo un segnale differente in base all'intensità e alla frequenza dell'onda elettromagnetica da cui sono stati eccitati. È importante ricordare che la frequenza elettromagnetica è la caratteristica fisica da cui dipende la nostra esperienza 'cromatica'. Questo segnale è trasmesso dalla retina verso la corteccia cerebrale dove un altro strato di neuroni provvede a generare l'immagine nel cervello. In realtà alcuni passaggi sono un pochino più complessi, ma va da sé che chi fosse curioso di capirli più a fondo avrà tutte le possibilità di studiare esaustivamente l'argomento sui libri dedicati o in rete dove al momento esistono sezioni dedicate chiare e dettagliate.

In figura (1) è data una semplice rappresentazione dello schema oggetto, retina e corteccia cerebrale.

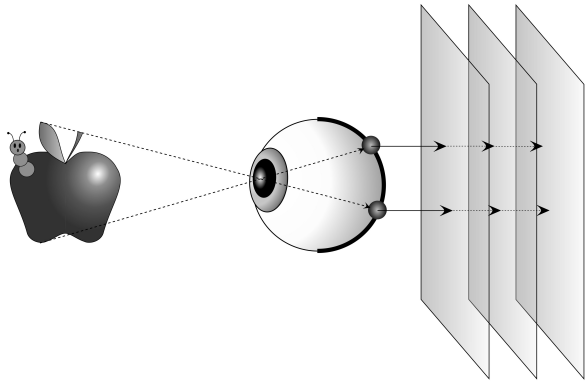

Fig. 1 - L'immagine della mela è trasmessa ai neuroni della retina dalla luce. I segnali nervosi sono trasmessi dalla retina verso i neuroni della corteccia attraverso i nervi e le

1975, come Rosenblatt 1958, tenta un approccio al problema basandosi sulla fiducia nella natura, e che potremmo riassumere con le seguenti parole: "Se lo schema funziona non cambiarlo", quindi se il sistema che permette la concettualizzazione dei segnali ottici è basato su una rete multistrato di neuroni, proviamo a creare un modello con la stessa struttura e vediamo cosa otteniamo. Nel prossimo paragrafo vediamo come rappresentare in linguaggio C i diversi strati neurali. In figura (2) è riportata una rielaborazione ispirata alla rappresentazione originale del cognitrone di Fukushima.

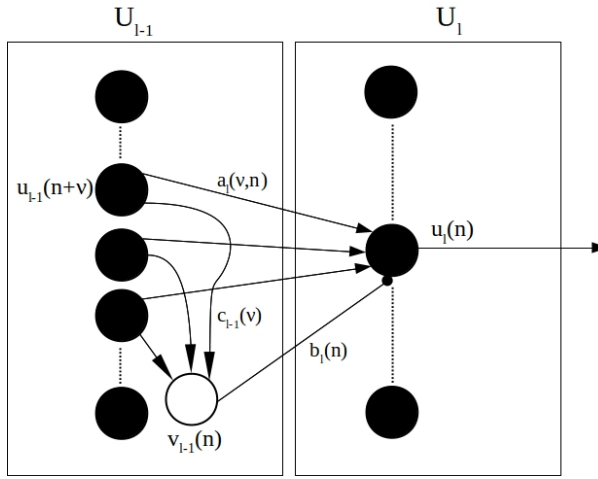

Fig. 2 - Struttura base del cognitrone. Sono mostrate le interconnessioni neurali tra due strati contigui.

Rappresentazione in C

Nel volume precedente abbiamo introdotto i tipi di dato base del linguaggio C. Il C però prevede anche un uso più avanzato dei dati permettendo di creare strutture complesse di più tipi base. Così, ad esempio, se vogliamo creare una variabile che rappresenti le coordinate

neurone $u_0(1,3)$ oltre a connettersi al $u_1(1,3)$ e al $u_1(1,1)$ si connetterà anche al $u_1(1,2)$.

In figura (7) è data una rappresentazione geometrica delle aree di connessione del neurone (2,2) e del neurone (3,3) dello strato 1.

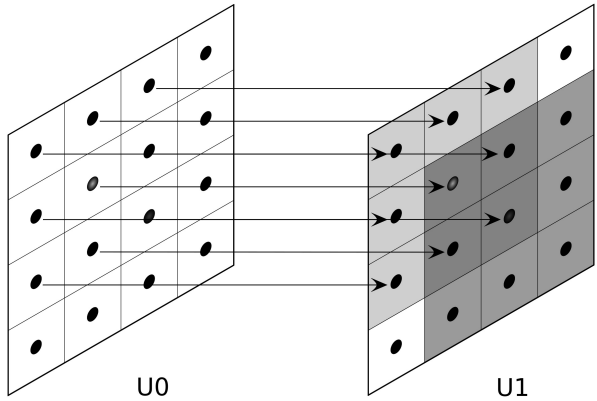

Fig. 7 - La figura evidenzia le aree di connessione del neurone (2,2) e del neurone (3,3) dello strato 1.

Stabilire l'area di connessione per un neurone di un livello U_l, equivale quindi a stabilire una *sottomatrice* sul livello U_l stesso. Gli elementi del livello U_{l-1} corrispondenti agli elementi della sottomatrice, sono quelli connessi al detto neurone. Possiamo dare la seguente definizione:

L'area di connessione di un neurone $u_l(i,j)$ ***di indice (i,j) è l'insieme di indici (m,n) tale che esista la connessione*** $a_l((m,n),(i,j))$**.**

Quindi l'area di connessione rappresenta l'estensione dei dendriti di un neurone sul proprio livello di appartenenza. Più un neurone è *esteso*, maggiore è il numero di collegamenti sinaptici che riesce a stabilire con i neuroni del livello precedente.

Questa descrizione così formale è un po' pesante, ma è stata necessaria perché altrimenti il seguito del modello sarebbe diventato davvero incomprensibile. Come il lettore può immaginare, presto cominceremo a 'cacciar giù' formule. Se il modello matematico a cui queste si riferiscono non è ben chiaro fin da subito, dette formule risulteranno incomprensibili. Quindi coraggio, rileggete il paragrafo con carta e matita alla mano e andate avanti finché non vi torna!

Il concetto di *sottomatrice* è mostrato in figura (8).

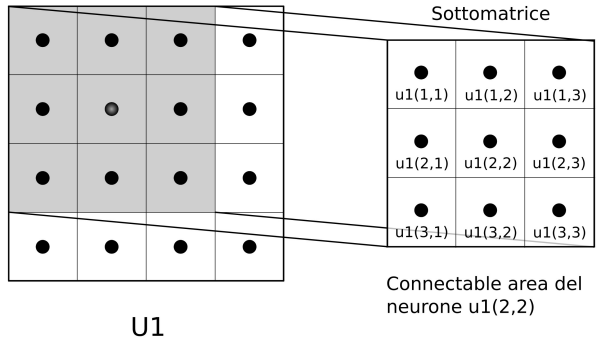

Fig. 8 - La figura evidenzia la matrice 3×3 ottenuta dell'area di connessione di un neurone.

Visto che ora il modello è un pochino più chiaro, facciamo un altro passo avanti. L'area di connessione esiste per ogni neurone del modello, quindi, piuttosto che definire una sottomatrice per ogni neurone, è più comodo definirla in termini di scostamenti relativi rispetto all'indice del neurone. Consideriamo per esempio che tutti i neuroni abbiano un'area di connessione corrispondente ad un quadrato di lato 3. Prendiamo come esempio il neurone $u_1(6,6)$: la sua area di connessione è data dagli indici (5,5), (5,6), (5,7), (6,5), (6,6), (6,7), (7,5), (7,6) e (7,7). Possiamo far riferimento a questi indici usando degli scostamenti relativi rispetto

all'indice (6,6) che indicheremo con δi e δj.

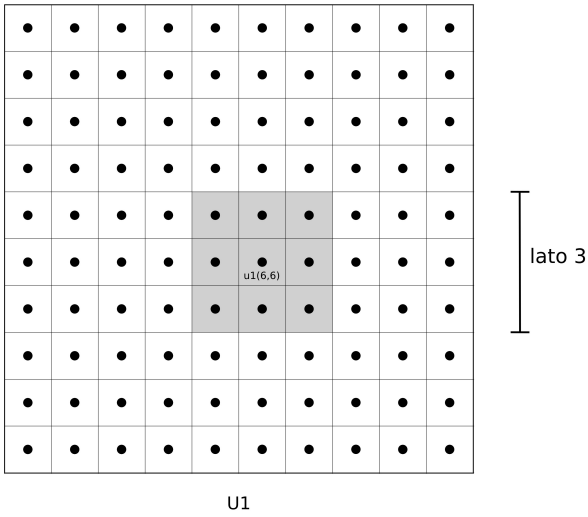

Fig. 9 - La figura evidenzia l'area di connessione del neurone (6,6).

Usando le variazioni (δ), la coppia di indici (5,5) diventa (-1,-1) in quanto per $\delta i = -1$ e $\delta j = -1$ abbiamo:

$$(5,5) = (6 + \delta i, 6 + \delta j)$$

Tutte le coppie di indici che esprimono la submatrice dell'area di connessione possono essere scritte usando gli scostamenti relativi $(\delta i, \delta j)$. Questa notazione è più comoda perché, una volta stabilita la forma e l'estensione dell'area di connessione, avremo che l'insieme degli indici che la descrive è lo stesso per tutti i neuroni, quindi non dovremo ridefinre la submatrice per ogni neurone.

Per fare un esempio, se continuassimo ad usare la submatrice quadrata di lato 3, allora l'insieme degli indici che la descrive sarebbe: (-1,-1), (-1,0), (-1,1), (0,-1), (0,0), (0,1), (1,-1), (1,0) e (1,1). Queste coppie sono le stesse

contare anche quei neuroni che non *esistono*. Se avete tempo e voglia, provate a fare una prova. Togliete la condizione e provate a compilare il codice, scoprirete che il compilatore compila tutto. Vi domandate: cosa succede a *runtime*? Questo è il bello e il brutto del C. Non posso sapere cosa succederà a runtime sulle vostre macchine, perchè questo dipende dalla specifica configurazione di memoria che avrete al momento dell'esecuzione del programma. Quello che vi posso garantire è che senza quel controllo andrete a leggere dei byte che sono al di fuori dell'area di memoria riservata al vostro array e il cui contenuto esula dal vostro controllo. Se va tutto bene, il programma eseguirà una violazione di accesso che verrà registrata e avrà come conseguenza l'interruzione del programma, ma, se va tutto male, l'algoritmo tratterà dei dati *casuali* come se fossero le risposte dei vostri neuroni e tutto a vostra insaputa... occhio, perché il C è bello, ma è traditore!

Una novità: i neuroni inibitori

Fin'ora abbiamo sempre considerato che l'effetto del segnale trasmesso dai neuroni presinaptici fosse quello di eccitare i neuroni postsinaptici ad essi collegati innalzando il loro potenziale interno così da indurre in essi la produzione di un nuovo segnale, scatenando una catena di segnali che collega strato a strato.

In realtà, proprio negli anni in cui Fukushima produsse i risultati della sua ricerca, si capì che alcuni neuroni partecipano al processo di trasmissione non eccitando, ma inibendo i neuroni postsinaptici.

In pratica l'effetto dei neuroni inibitori è quello di stabilire un *valore di soglia*: se il potenziale supera il valore di soglia allora il neurone *spara*, altrimenti non spara.

L'effetto di un neurone inibitore presinaptico è di aumentare il valore di soglia che deve essere raggiunto dal potenziale del neurone postsinaptico perché questo si attivi ed emetta (spari) il segnale in uscita.

Il modello matematico del neurone deve essere modificato per tener conto di questa *novità* biologica! Purtroppo è sempre così, quando crediamo di aver capito tutto, arriva qualcosa a scombinare le carte.
Per semplicità poniamo la variabile e pari al valore dell'espressione (1) in modo da avere:

$$e = \sum_{\delta i, \delta j} a_1 \left[(i+\delta i, j+\delta j), (i,j)\right] \; u_0(i+\delta i, j+\delta j) \tag{4}$$

dove e sta per *excitatory*, cioè indica il contributo eccitatorio dei neuroni presinaptici afferenti a $u_1(i,j)$.
Trattiamo i neuroni inibitori in maniera analoga agli eccitatori e stabiliamo che nel modello del cognitrone (lo stabilì Fukushima) il contributo inibitorio fosse proporzionale al segnale del neurone, con costante di proporzionalità data, analogamente a quanto fatto per i neuroni eccitatori, dal peso della connessione.
Indichiamo il contributo inibitorio con la variabile h (*hinibitory*) e abbiamo:

$$h = b_1(i,j)\; v_0(i,j) \tag{5}$$

dove $b_1(i,j)$ è il peso della connessione e $v_0(i,j)$ è il segnale di output del neurone inibitorio (i,j).
Avrete sicuramente notato che diversamente dell'espressione (1), la (5) contiene il contributo di un unico neurone. È proprio così, non ci sono errori. Nel cognitrone originale, ogni neurone eccitatore ha connesso a sé un solo neurone inibitore.

Arrivati a questo punto dobbiamo inserire l'espressione h dentro al potenziale. In prima approssimazione potremmo scrivere per il potenziale totale, cioè la somma algebrica del potenziale eccitatore e quello inibitore, l'espressione seguente:

$$p_1(i,j) = e - h \tag{6}$$

In tale espressione, h rappresenta esattamente il valore di soglia che il potenziale eccitatore deve superare perché si inneschi il segnale di uscita. Infatti insieme alla (6) che definisce la funzione f nella (2), definiamo anche la funzione g nella (3) come segue:

$$g(x) = \phi(x) = \begin{cases} x & (x \geq 0) \\ 0 & (x < 0) \end{cases} \tag{7}$$

quindi, sostituendo l'argomento $e - h$ al posto della x nella (7) si ha:

$$\phi(e-h) = \begin{cases} e-h & (e \geq h) \\ 0 & (e < h) \end{cases} \tag{8}$$

dove è chiaro il ruolo di *soglia* giocato dall'inibizione. Fukushima comunque non era soddisfatto di questa espressione per il potenziale in quanto presenta un problema che si manifesta con il tempo, una specie di 'usura'. Vediamo di capire. Esattamente come accade nel modello del percettrone, che abbiamo incontrato nel precedente volume, i pesi delle connessioni neurali variano a causa dei meccanismi di rinforzo sinaptico, e possono crescere in maniera indefinita. La crescita indefinita non è accettabile se si cerca di mantenere una corrispondenza con i principi biologici. Per questo Fukushima propone la seguente espressione per il potenziale:

$$\left(\frac{1+e}{1+h} - 1 \right) \tag{9}$$

che si riduce alla (6) per $h \ll 1$ infatti, eseguendo la differenza nella (9)

si ha:

$$\frac{1+e-1-h}{1+h} = \frac{e-h}{1+h} \approx e-h \tag{10}$$

e si approssima a:

$$\frac{e}{h} - 1 \tag{11}$$

per $e, h \gg 1$. L'espressione (11) ci dice che il potenziale è maggiore di zero quando la componente eccitatoria supera quella inibitoria, ma esso rimane comunque limitato anche al crescere indefinito di e ed h.

Con la forma data dall'espressione (9) per il potenziale, si ha che il segnale di output del generico neurone $u_l(i,j)$ è dato da:

$$u_l(i,j) = \tag{12}$$
$$= \phi\left(\frac{1+\sum_{\delta i,\delta j} a_l((i+\delta i, j+\delta j),(i,j))\; u_{l-1}(i+\delta i, j+\delta j)}{1+b_l(i,j)\; v_{l-1}(i,j)} - 1\right)$$

In figura (12) è rappresentata la relazione tra i neuroni eccitatori e quelli inibitori.

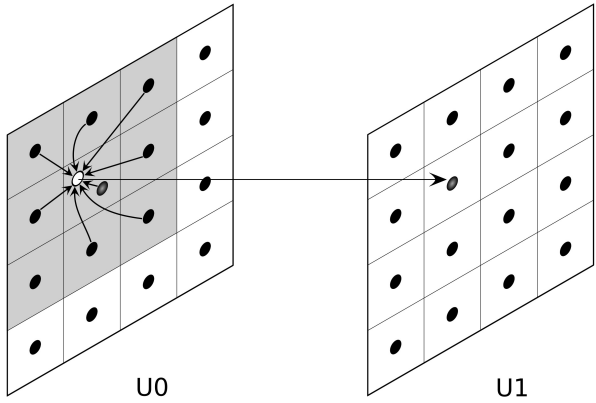

Fig. 12 - Il neurone inibitore (2,2) è stimolato dai segnali dei neuroni della sua area di connessione che giace sul suo stesso strato. Esso invia il segnale inibitore al neurone

6 - UN COGNITRONE FUNZIONANTE

Volete una bella notizia? Ve la do io: abbiamo tutto il codice per scrivere la prima implementazione del cognitrone e avere qualche bella soddisfazione.
Si, sarà diverso dal percettrone, proprio un altro sapore. Certo anche le reti *supervised* hanno il loro fascino. La possibilità di configurare un insieme di matrici per implementare in modo nascosto un algoritmo, semplicemente limitandoci a fare da istruttori è molto allettante. Però in qualche modo sembra quasi troppo semplice, diciamo che, una volta svelato, il mistero delle NN supervisionate potrebbe lasciare, non delusi, però troppo. Sì, diciamo che è molto matematico alla fine. Intendiamoci, anche le reti artificiali non supervisionate sono matematica, però i meccanismi che le regolano sono più nascosti, per cui appaiono più affascinanti.

Cosa dobbiamo fare adesso? Dobbiamo mettere insieme il codice che abbiamo sviluppato fin'ora eseguendo una serie di chiamate nella funzione `main` in modo da replicare il fenomeno di trasmissione degli stimoli dalla retina alla corteccia cerebrale. Cerchiamo di farlo nel modo più vicino a quanto descritto dal nostro Fukushima nel suo articolo, o *paper* come viene definito in gergo un articolo scientifico. Non potremo farlo completamente uguale perché egli non pubblicò su quell'articolo anche il codice Fortran, per cui la nostra implementazione si basa su ciò che lui ha descritto. Comunque questo è il vero spirito scientifico: si cerca di replicare un esperimento basandosi sulla descrizione data da un altro gruppo di ricerca.

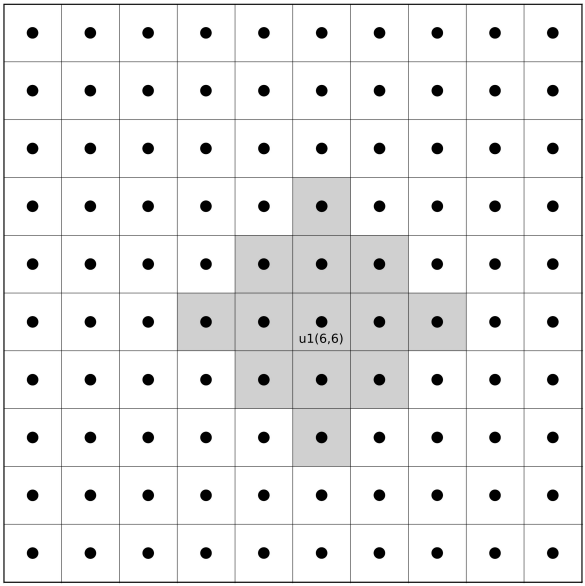

Fig. 13 - I neuroni di vicinato del neurone (6,6) sono evidenziati in grigio.

Stimoli originali

L'esperimento originale prevedeva che alla rete fossero presentati ciclicamente cinque stimoli corrispondenti ai *pattern* delle cifre "0", "1", "2", "3", "4".

In questo lavoro ci siamo adoperati per riprodurre fedelmente i pattern esatti usati da Fukushima. In realtà la forma esatta usata per rappresentare le cifre non è significativa ai fini dell'esperimento, ma aggiunge un certo fascino sapere di stare usando esattamente le stesse forme.

```
#endif
```

Il codice presentato contiene solo delle informazioni per il preprocessore e per il linker, quindi non serve compilarlo, mentre l'implementazione delle funzioni è codificata nel sorgente che segue:

Listato cgn_lib.c

```
/**
   Il FILE cgn_lib.c è parte del
   Cognitrone: sistema di pattern recongnition
   Copyright (C) 2018  Francesco Sisini (francescomichelesisini@gmail.com)
   This program is free software: you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation, either version 3 of the License, or
   (at your option) any later version.
   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.
   You should have received a copy of the GNU General Public License
   along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "cgn_testate.h"

void inibizione_laterale(STRATO *Ul, AREA *ar)
{
  for(int n=0;n<NEURONS; n++)
    {
      int i=n/NCOLS;
      int j=n%NCOLS;
      double e=0;
      double h=0;
      double g=0;
      for(int ij=0;ij<H_HA;ij++)
        {
          int di=ar->g_i[ij];
          int dj=ar->g_j[ij];
          int l=i+di;
          int m=j+dj;
          if(l>=0&&l<NCOLS&&m>=0&&m<NCOLS)
            {
              int nu=l*NCOLS+m;
              h+=Ul->u[n].g[ij]*(Ul->u[nu].segnale);
              g+=Ul->u[n].g[ij];
            }
        }
      e=Ul->u[n].segnale;
      Ul->u[n].segnale_tmp=phi((e-h)/(1+h));
      /* BORDI */
      Ul->u[n].segnale_tmp*=g;
    }
  for(int n=0;n<NEURONS; n++)
    {
      Ul->u[n].segnale=Ul->u[n].segnale_tmp;
    }
}

void trasmetti_indietro(STRATO *from, STRATO *to,AREA *ar)
{
  double e=0;
  double h=0;
  for(int n=0;n<NEURONS; n++)
    {
      int i=n/NCOLS;
```

ha la forma del tipo

$$s_1 > s_2 + \epsilon$$

Questa è una delle insidie del C. Se volete capire a cosa serve addizionare quella ϵ, provate a toglierla e a ricompilare. Vi accorgerete che, sebbene il risultato sia tutto sommato accettabile, si è persa la simmetria nella propagazione del segnale. In questo contesto non voglio divagare approfondendo la rappresentazione dei dati di tipo `float` nel C, ma anche questo è un aspetto *traditore* del C che spesso fa preferire altri linguaggi di programmazione nello sviluppo di applicazioni scientifiche.

Quello che abbiamo ottenuto è senz'altro interessante, però manca ancora qualcosa per stupirci, vediamo di aggiungerlo con il prossimo esperimento.

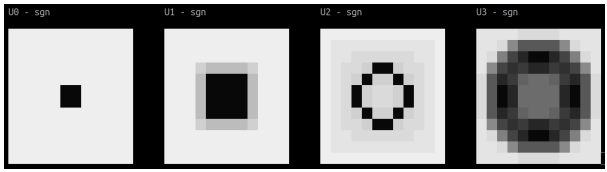

Fig. 14 - In figura è rappresentato lo screenshot di un ciclo dell'eleboarazione del cognitrone che viene stimolato con un segnale puntiforme.

Propagazione inversa

Quello che abbiamo visto fino adesso ha sicuramente un incredibile fascino, però è normale che rimanga un dubbio: cosa ha imparato davvero il cognitrone durante i venti cicli?
Spieghiamoci meglio, la domanda che ci sorge spontanea è: in che modo la configurazione di risposta sul quarto strato del cognitrone è collegata allo stimolo visivo presentato sulla retina?

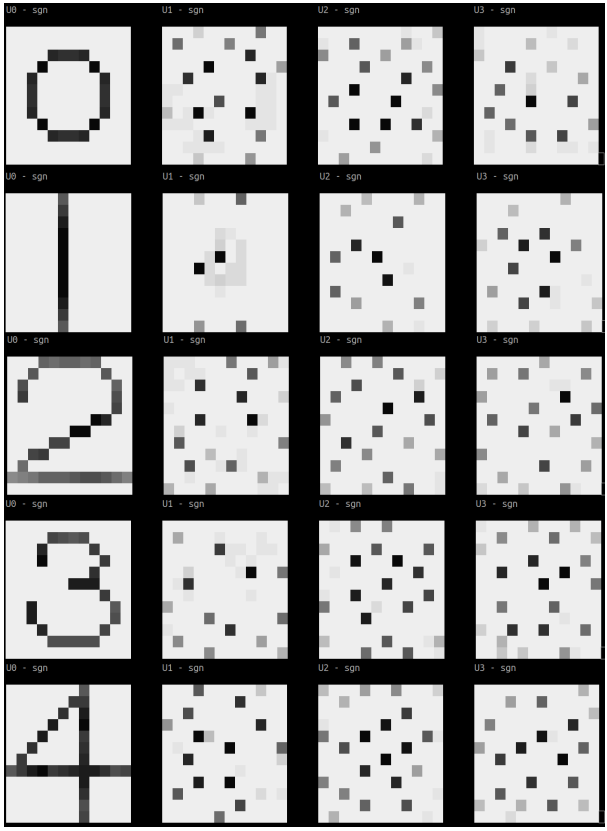

Fig. 20 - In figura sono reppresentati gli screenshot di un ciclo dell'elaboarazione dell'esperimento di Fukoshima.

La differenza rispetto alla versione 6 è nel segnale di stimolo. Come si vede, la sequenza di costrutti `if` provvede a fornire ciclicamente gli stimoli "0", "1", "2", "3", "4", in modo che il cognitrone sia sottoposto ad uno stimolo variabile e non sempre allo stesso. Domanda: e ora? Funzionerà ancora la propagazione inversa? Per rispondere non ci resta che provare!

Dalla tastiera premiamo il tasto invio in modo che prosegua il ciclo di apprendimento (avete notato la `getchar` no?) e al posto dello "0" compare

9 - CONSIDERAZIONI

Abbiamo visto che il cognitrone è un sistema capace di auto-organizzarsi per apprendere l'abilità di distinguere delle immagini; in pratica quello che succede è che i neuroni dei suoi strati si specializzano per eccitarsi solo in conseguenza della visione di una specifica immagine (o pattern).
In questo paragrafo analizziamo le implicazioni e le possibili applicazioni di questa capacità.

Confronto con algoritmi specifici

La prima osservazione che giustamente salta in mente è che il risultato ottenuto sfruttando la capacità auto-organizzativa del cognitrone si poteva ottenere per mezzo di un algoritmo specifico. Se lo scopo era quello di realizzare un sistema in grado di riconoscere le cifre "0", "1", "2", "3" e "4", si poteva utilizzare un semplice algoritmo di confronto che al presentarsi dello stimolo (pattern) ne verificasse la corrispondenza con delle immagini pre-caricate in memoria. In pratica per realizzare un tale sistema è sufficiente annidare due cicli for che confrontino uno ad uno gli elementi (pixel) dell'immagine presentata alla retina con delle immagini che possono essere archiviate su un database. Banale no? Si, ma anche no. Se disponiamo di un database, cioè di un sistema di memoria che ha come unico scopo quello di archiviare l'informazione e di un sistema di organizzazione di tale informazione, allora si, il compito è banale. Cosa succede però, se non abbiamo a disposizione questo database?
Se ci stiamo occupando di realizzare un sistema informatico *standard* la domanda è puramente accademica, infatti anche dovendo implementare l'algoritmo su un sistema embedded si potrebbe sempre trovare una soluzione per memorizzare le immagini in un database, magari custom,