Università di Pisa
Facoltà di Scienze Matematiche Fisiche e Naturali

Corso di Laurea Specialistica in Scienze Fisiche
Anno Accademico 2006-2007
Tesi di Laurea Specialistica

Dinamica di Latching in una Memoria Associativa di Potts

Candidato: **Eleonora Russo**

Relatore: **Dr. A. Treves**
Relatore: **Dr. R. Mannella**

a Irene

‘BY CONSEQUENCE, or train of thoughts, I understand that succession of one thought to another which is called, to distinguish it from discourse in words, mental discourse. When a man thinketh on anything whatsoever, his next thought after is not altogether so casual as it seems to be. Not every thought to every thought succeeds indifferently. But as we have no imagination, whereof we have not formerly had sense, in whole or in parts; so we have no transition from one imagination to another, whereof we never had the like before in our senses.’

Thomas Hobbes, Leviathan, chap. III.

‘... Id like to sip those richlooking green and yellow expensive drinks those stagedoor johnnies drink with the opera hats I tasted once with my finger dipped out of that American that had the squirrel talking stamps with father he had all he could do to keep himself from falling asleep after the last time after we took the port and potted meat it had a fine salty taste yes because I felt lovely and tired myself and fell asleep as sound as a top the moment I popped straight into bed till that thunder woke me up...’

James Joyce, Ulysses.

Indice

Introduzione

Le neuroscienze cognitive sono un settore di ricerca che studia i meccanismi neurologici che sottendono alla cognizione, ovvero a tutte quelle funzioni come l'attenzione, la percezione, il linguaggio, la motricità, la memoria, ecc... caratteristiche del coinvolgimento del sistema nervoso superiore. I metodi di indagine utilizzati sono molteplici e interdisciplinari; su animali vengono effettuati esperimenti di elettrofisiologia, mentre per l'uomo si ricorre al neuro immaging, come ad esempio la risonanza magnetica funzionale (fMRI) e la tomografia ad emissione di positroni (PET), test neuropsicologici, esperimenti di psicofisica e allo studio di soggetti con lesioni cerebrali. Parallelamente a questi metodi d'indagine, legati principalmente ai settori di psicologia, linguistica e neurobiologia, negli ultimi anni si sta progressivamente rafforzando lo studio delle funzioni cognitive tramite modelli computazionali. Un approccio di questo tipo, chiaramente, non si prefigge di produrre una descrizione dettagliata ed esaustiva dei processi al livello biologico ma può essere utile ad evidenziare le linee guida della dinamica che sottende tali processi.
In particolare, lo scopo di questo lavoro di tesi è costruire un modello di rete neurale che rispecchi il comportamento della corteccia cerebrale in due particolari processi cognitivi. In primo luogo dovremo modellizzare il processo di riconoscimento, ovvero come al momento della presentazione di uno stimolo sensoriale la memoria riconosce lo stimolo attraverso le esperienze passate. Poi, sulla base del precedente modello, descriveremo il processo di latching, ovvero le associazioni libere di idee nate dalla correlazione tra due concetti.
La modellizzazione di questi processi si basa sulla descrizione della corteccia in termini di rete neurale, in cui un 'concetto' coincide con una particolare configu-

razione del network e il processo di riconoscimento di uno stimolo esterno avviene attraverso la rielaborazione di un input codificato.
Come punto di partenza utilizzeremo il modello di Hopfield, memoria associativa ispirata al modello di Ising, che modificheremo gradualmente in base a due fondamentali esigenze: da un lato cercheremo di costruire i nostri modelli mantenendo un'attenzione particolare alle caratteristiche strutturali biologiche del sistema considerato, la corteccia; dall'altro dovremo riuscire a costruire una rete che mimi non solo il processo di recupero di una memoria ma anche il fenomeno di passaggio spontaneo da un concetto all'altro. Rispettivamente, i vincoli legati alla struttura biologica porteranno all'utilizzo di una rete ispirata al modello di Potts; mentre la richiesta di passare spontaneamente da una ad un'altra memoria, comporterà l'introduzione di patterns correlati e di un complesso meccanismo di adattamento della rete.
Sono stati prodotti, dunque, quattro modelli, ognuno dei quali è l'evoluzione del precedente. Per ognuno dei essi è stata sviluppata sia una simulazione numerica che una derivazione teorica delle equazioni dinamiche.
Il modello finale è una rete di Potts in grado di recuperare un pattern tra i memorizzati e da questo passare spontaneamente ad un altro ad esso correlato.

Capitolo 1

La memoria semantica

Per comprendere i fenomeni di recupero di un'informazione memorizzata e di libera associazione occorre in primo luogo cercare di capire in che modo, nel cervello, sia rappresentto un concetto e come la memoria agisca su questo.

Ogni stimolo esterno è percepito da un soggetto attraverso gli organi di senso. Tali organi permettono di codificare l'informazione proveniente dall'esterno rendendola fruibile al sistema nervoso. L'analisi del segnale avviene poi per passaggi consecutivi attraverso neuroni sempre più raffinati e selettivi. Un classico esempio di tale processo è costituito dal sistema visivo.
Nel sistema visivo l'immagine è percepita al livello della retina da recettori (coni e bastoncelli) capaci di discriminare solo l'intensità luminosa e i colori. Attraverso successivi passaggi neuronali troviamo però cellule con selettività sempre maggiore, capaci di distinguere differenze di contrasto, particolari inclinazioni di linee, dimensioni, direzioni, ecc... Questa differenziazione prosegue fino ad un grado di complessità tale da farci trovare, ad esempio, nella corteccia inferotemporale, cellule sensibili alla forma 'volto' più che ad ogni altro stimolo.
Partendo da uno schema organizzativo di questo tipo, particolarmente evidente nelle strutture di decodificazione sensoriale, fu inizialmente ipotizzato che al termine di ogni processo recettivo esistessero *cellule cardinali* capaci di distinguere con esattezza uno specifico oggetto da qualunque altro possibile stimolo. Tali cellule avrebbero dovuto essere così raffinate da attivarsi solo alla presenza di un

unico e particolarissimo segnale, come ad esempio quello causato dalla vista di una persona conosciuta (teoria del *neurone della nonna*) [18]. Questa teoria si è rivelata sbagliata, basti pensare alla quantità di cellule necessarie per rappresentare ciò che conosciamo o che potremmo conoscere.
A livello sperimentale [18], grazie allo studio di pazienti con lesioni cerebrali, si cercò di stabilire quale regione del cervello potesse gestire l'organizzazione della memoria. Nonostante la ricchezza di informazioni che tali studi portarono non si riuscì, però, a focalizzare per la memoria una regione univoca e anatomicamente definita.
Iniziò così a farsi strada l'ipotesi che vi fossero più centri destinati a questo tipo di compito. Si ipotizzò che differenti categorie di concetti, o meglio differenti aspetti di uno stesso concetto, potessero essere rappresentate in luoghi diversi della corteccia e che proprio la rappresentazione combinata di tali caratteristiche potesse formare l'idea completa del concetto.
Oggi gli psicologi cognitivi chiamano *memoria semantica* quella parte della memoria che si occupa proprio della gestione delle rappresentazioni di concetti all'interno del cervello. Nella categorizzazione più diffusa la memoria è divisa in due macroclassi: memoria *implicita* e memoria *esplicita*.
La memoria implicita, o *non dichiarativa*, è un tipo di memoria utilizzata in modo inconscio al momento in cui un soggetto si trova a dover svolgere un compito motorio o percettivo. Questo tipo di memoria è molto rigida e strettamente connessa alle condizioni originali nelle quali è avvenuto l'apprendimento.
La memoria esplicita, o *dichiarativa*, riguarda invece la gestione delle conoscenze di un individuo a livello conscio. Questa è caratterizzata da una grande flessibilità e dall'associazione contemporanea di più informazioni.
La memoria dichiarativa è a sua volta divisa tra *episodica*, legata al ricordo di esperienze soggettive, e *semantica*, legata alla gestione di concetti e conoscenze di natura generale [18]. La memoria semantica è dunque quella parte della memoria a lungo termine, ovvero permanente, che ci permette di pensare agli oggetti svincolandoli dal contesto specifico. Per fare un esempio, se un individuo pensa ad un cane nella corteccia si attiveranno molteplici zone. L'idea di 'cane' infatti attiverà, ad esempio, l'aria relativa alla forma del cane, quella legata al colore o all'odore del cane, ecc... L'attivazione di tutte queste aree contemporaneamente caratterizzerà in modo univoco nel soggetto l'idea 'cane'.

Potremo dunque dire che un concetto è costituito proprio ed esclusivamente dall'attivazione contemporanea di una specifica mappa di aree in tutta la corteccia. La nostra conoscenza delle cose sotto forma di informazioni unitarie è, quindi, il risultato di una serie di processi integrativi di rappresentazioni diverse, localizzate in sedi corticali anatomicamente distinte, ciascuna delle quali rappresenta solo uno degli aspetti del concetto richiamato. Non esiste di conseguenza un unico magazzino della memoria semantica ma, più verosimilmente, una rete di connessioni che, nel complesso della corteccia, rappresentano l'idea unitaria di un concetto.
Utilizzando un linguaggio proprio delle reti neurali possiamo allora dire che: ogni oggetto noto ad un individuo, quindi memorizzato, corrisponde ad una particolare configurazione di neuroni accesi e spenti. Al momento della ricezione di uno stimolo a livello corticale si attiva una configurazione della rete nervosa nota (o no), o meglio già memorizzata (o no), in base alla quale il soggetto riconosce e categorizza lo stimolo.

Partendo da questo tipo di approccio John Hopfield nel 1982 propose un modello di memoria associativa basato sul riconoscimento da parte di una rete neuronale di configurazioni note, dette *patterns*, in seguito ad una perturbazione esterna.

Capitolo 2

Un primo modello

2.1 Il modello di Hopfield, tra biologia e fisica

Il modello di Hopfield fu uno dei primi modelli di memoria associativa applicato alle neuroscienze e riscosse un grande successo grazie alla sua particolare semplicità.
Un sistema con funzione di memoria associativa è un dispositivo capace di richiamare un'informazione precedentemente memorizzata se sollecitato con uno stimolo esterno che presenta un certo grado di somiglianza con tale informazione. In seguito ci riferiremo a tale processo con il nome di **retrieval**.
L'informazione memorizzata è rappresentata da una particolare configurazione della rete, detta **pattern**. Una rete può memorizzare più configurazioni (patterns) contemporaneamente con una capacità che dipende dai parametri con cui è stata costruita la rete stassa, come ad esempio il numero di unità e di connessioni. Il numero massimo di patterns che una rete di N unità può memorizzare è chiamato **storage capacity** ed è indicato dal simbolo α_c.

Il modello di Hopfield è un tipo di Rete Neurale ad Attrattori costituita da N unità binarie σ_i, interconnesse tra loro, ciascuna delle quali schematizza l'attività di un singolo neurone.
L'unità è detta binaria perché può assumere soltanto due valori, tipicamente $+1$ o -1, corrispondenti rispettivamente al neurone acceso o spento.

Le connessioni tra le unità, ovvero la connettività della rete, sono descritte da una matrice w, chiamata matrice delle efficacie sinaptiche (o dei pesi). L'elemento w_{ij} di matrice media l'interazione tra l'unità i e la j stabilendone la forza.

2.1.1 La regola di update

Una volta definita tale matrice di connessioni lo stato di ogni singola unità è determinato dalla formula seguente:

$$\sigma_i := sgn\left(\sum_j^N w_{ij}\sigma_j - U_i\right) \tag{2.1}$$

dove U_i indica la soglia di attivazione relativa all'unità i (che per semplicità prenderemo uguale per tutte le unità).
Questo vuol dire che l'unità σ_i assumerà il valore 1 solo se la somma dei valori delle unità a cui è connessa, moltiplicato per il rispettivo peso sinaptico, dà un valore maggiore della soglia di attivazione U_i. Sarà invece uguale a 0 se questo valore è minore di U_i.
L'insieme dei contributi provenienti dalle unità afferenti all'unità i è indicato come campo complessivo h_i:

$$h_i = \sum_j^N w_{ij}\sigma_j \tag{2.2}$$

da cui

$$\sigma_i := sgn\left(h_i - U_i\right). \tag{2.3}$$

Una tale schematizzazione rispecchia in maniera semplificata ciò che avviene biologicamente al livello neuronale. su ogni dendrite, infatti, ogni neurone può stabilire connessioni sinaptiche con più cellule nervose, dette afferenti. L'attivazione di tali cellule stimola il neurone postsinaptico al livello delle sinapsi depolarizzandolo, localmente. Tale depolarizzazione percorre i dendriti e si somma linearmente al livello del soma con le depolarizzazioni indotte dalle altre cellule afferenti. Se la stimolazione complessiva risulta superiore ad un valore di soglia il neurone rice-

vente si attiva propagando un potenziale d'azione [18].

Ci si riferisce alla formula (2.3) come alla **regola di update** della rete che descrive l'aggiornamento dello stato delle varie uinità.
Tale aggiornamento può essere di due diversi tipi: aggiornamento *sincrono*, in cui ad ogni time step tutti i neuroni sono aggiornati simultaneamente, e aggiornamento *asincrono*, nel quale ad ogni time step viene scelto casualmente un singolo neurone da aggiornare. In quest'ultimo caso per considerare l'aggiornamento di tutta la rete dovremo, di conseguuenza, aspettare un tempo medio di N update.
In tutti i modelli che considereremo nel corso di questo elaborato verrà sempre utilizzato l'aggiornamento asincrono, che ha il vantaggio di essere più plausibile da un punto di vista biologico. I neuroni, infatti, ricevono input con ritardi casuali, dovuti ad esempio a distanze diverse da percorrere lungo gli assoni, per cui anche se inizialmente fossero sincronizzati, questa sincronia verrebbe persa rapidamente. Un aggiornamento sincrono inoltre prevederebbe la presenza, biologicamente immotivata, di una sorta di clock interno all'organismo.

Una rete così definita è dunque un sistema dinamico che, una volta impostato lo stato iniziale, evolve passando attraverso configurazioni differenti.
Ciò avviene fino a che il sistema non si stabilizza in un cosiddetto attrattore o punto fisso; a questo livello la rete infatti continuerà ad aggiornarsi riproponendo sempre la stessa configurazione. Una stessa rete è caratterizzata dall'avere più configurazioni stabili in cui il network può posizionarsi in base ad esempio ad uno stimolo esterno.
Se, come mostrato in figura 2.1, si rappresenta l'energia della configurazione della rete come una superficie unidimensionale, gli attrattori sono rappresentati dai punti di minimo di tale landscape e l'evoluzione dinamica della rete è descritta da un percorso su questa superficie che termina nel fondo di una delle valli [17].

2.1.2 I patterns

Nel caso specifico del modello di Hopfield un attrattore, o **pattern**, è un vettore ξ^μ di lunghezza N. La rete viene costruita in modo da avere p patterns

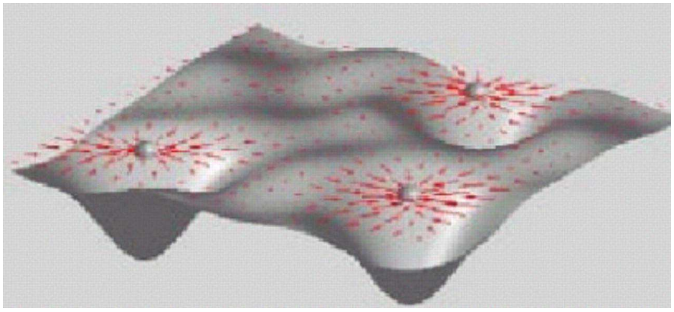

Figura 2.1: Rappresentazione dell'energy landscape di una memoria associativa. I bacini di attrazione dei patterns corrispondono agli avvallamenti del landscape [3].

($\mu = 0...p-1$). Nel modello di Hopfield ogni pattern è costruito in modo da avere metà delle unità attive ($\xi_i^\mu = 1$) distribuite casualmente tra le N totali ed è costruito in modo che l'unità i-esima non sia correlata né con le altre unità dello stesso pattern μ né con quelle di altri.

2.1.3 La regola di Hebb

La capacità di adattamento del sistema nervoso, come ad esempio la capacità di memorizzare eventi e concetti, è dovuta in gran parte alla facoltà che hanno i singoli contatti sinaptici, ciascuno indipendentemente dagli altri, di variare in forza o efficacia col tempo. Vi sono molti meccanismi di potenziamento e depotenziamento del rendimento sinaptico ma sicuramente uno dei maggiori è il fenomeno di long-term potentation (LTP): è stato osservato che se un neurone A contribuisce spesso all'eccitazione del neurone B la sinapsi che collega questi due neuroni si rafforza. Un successivo stimolo proveniente da A avrà quindi maggiori probabilità di suscitare l'attivazione di B. Tale meccanismo è noto come **Legge di Hebb**, psicologo scozzese che, prima ancora di una qualunque evidenza sperimentale, ipotizzò tale comportamento. L'influenza di un neurone sul postsinaptico è determinata, dunque, da quanto spesso questi due neuroni sono stati attivi contemporaneamente.
Aspetto estremamente affascinante di tale teoria è che, in questo modo, proprio le connessioni tra un neurone e un altro sono depositarie della memoria locale

del processo e portano alla creazione di strutture, quelle che Hebb chiama assemblee di cellule, senza la necessità di un disegno superiore che ne guidi la formazione.

In linguaggio matematico possiamo schematizzare l'interazione tra i neuroni della rete tramite la matrice di pesi w. In generale si chiama fase di apprendimento la fase in cui tale matrice subisce modifiche in conseguenza della presentazione di stimoli esterni. Per ogni aggiornamento della rete, durante la fase di apprendimento, la matrice w si modifica in base alla regola di Hebb:

$$\delta w_{ij} = r\ \sigma_i \sigma_j \tag{2.4}$$

dove r è un generico parametro di rate e δw_{ij} è l'incremento, positivo o negativo a seconda della concordanza tra σ_i e σ_j, della forza sinaptica tra le unità i e j.
Grazie a questo processo, gli stimoli più frequenti modificano le connessioni della rete in maniera più significativa, diventando gli attrattori della rete (ovvero i patterns ξ^μ).

Nel modello di Hopfield e in tutti i modelli che costruiremo nel corso di questo elaborato supporremo che la matrice dei pesi sia stata consolidata ddurante una prima fase di apprendimento. Quindi, dopo aver costruito i patterns, utilizzeremo direttamente come matrice di pesi

$$w_{ij} \equiv \frac{1}{N} \sum_{\mu}^{p} \xi_i^\mu \xi_j^\mu \tag{2.5}$$

che deriva dalla (2.4) imponendo la consolidazione dei pattern ξ^μ. Da qui

$$h_i = \sum_j w_{ij} \sigma_j = \frac{1}{N} \sum_j \sum_{\mu}^{p} \xi_i^\mu \xi_j^\mu \sigma_j. \tag{2.6}$$

Alla fase di apprendimento segue poi quella di interrogazione, nella quale la rete reagisce alla presenza di un input esterno senza modificare la propria matrice dei pesi.
Notiamo infine che una simile rigida divisione tra le due fasi non è realistica. Nell'apparato nervoso ogni imput esterno, purchè adeguato, è sempre capace di modificare l'efficacia della sinapsi tra due neuroni. Il motivo della nostra scelta

sta nel fatto che il nostro oggetto di interesse è osservare la dinamica di recupero di un concetto. Siamo dunque disinteressati a come questo sia stato consolidato precedentemente nella memoria. Inoltre siamo interessati ad una memoria di tipo semantico e non episodico, i concetti sono dunque ben consolidati e non così facilmente modificabili. Ci poniamo quindi nella condizione in cui la memorizzazione dei patterns è già avvenuta e si procede all'interrogazione della rete.

2.2 Transizione di fase

Grazie alla funzione energia H

$$H = -\frac{1}{2}\sum_{ij}^{N} w_{ij}\sigma_i\sigma_j = -\frac{1}{2N}\sum_{ij}^{N}\sum_{\mu}^{p}\xi_i^\mu\xi_j^\mu\sigma_i\sigma_j \tag{2.7}$$

che ricorda molto da vicino l'hamiltoniana del modello di Ising, siamo in grado di studiare l'energy landscape del nostro sistema e trovare dunque le configurazioni favorite dalla rete.

Per imporre la stabilità dei patterns occorre fissare la matrice dei pesi che regola le connessioni tra le varie unità. Fissata tale matrice, però, i patterns non rappresentano gli unici attrattori della rete; esistono infatti anche *stati spuri* per cui l'hamiltoniana presenta minimi (locali o globali, in base alla scelta dei parametri). Tali stati spuri si dividono in stati *di mistura*, combinazioni lineari dei pattern, e stati di *spin glass*, non correlati con alcuna combinazione finita di patterns. La presenza di tali stati spuri porta chiaramente ad un malfunzionamento della rete. Se stimolato, infatti, il network potrebbe entrare nel bacino di attrazione di uno di tali stati e fornire come output una configurazione priva di interesse. Per evitare un simile malfunzionamento è necessario studiare lo spazio dei parametri alla ricerca di una regione in cui la rete sia una memoria affidabile.

Senza sviluppare una trattazione analitica completa per lo studio dello spazio dei parametri, riportiamo in breve una discussione euristica delle transizioni di fase che si trovano al variare dei parametri.
Analogamente al caso del modello di Ising per i ferromagneti, anche nel modello di Hopfield si osserva una transizione di fase di seconda specie.

Introduciamo nel modello un parametro T capace di modulare il rumore interno al sistema (discuteremo più in dettaglio tale parametro nel prossimo capitolo) e restringiamoci a studiare la rete per $p \ll N$.
Grazie al formalismo statistico, che ci permette di trattare il sistema come ensemble canonico possiamo ricavare, con qualche passaggio, lo stato medio di attivazione di un'unità della rete

$$\langle \sigma_i \rangle = \tanh \left(\beta \sum_j w_{ij} \langle \sigma_i \rangle \right) \tag{2.8}$$

Assumendo ora che $\langle \sigma_i \rangle$ sia proporzionale al pattern di cui si sta facendo il retrieval, $\langle \sigma_i \rangle \propto m\xi_i^\nu$, otteniamo

$$m = \tanh(\beta m) \tag{2.9}$$

L'equazione (2.9) può essere risolta graficamente (fig. 2.2) e ci dà l'andamento di m al variare di T.

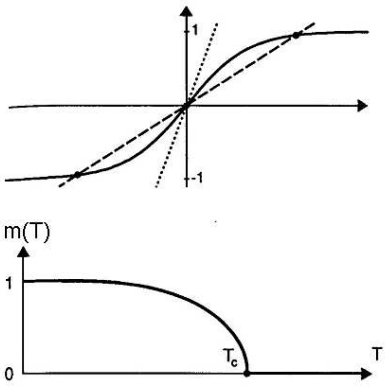

Figura 2.2: Risoluzione grafica dell'eq. (2.9) e andamento del parametro d'ordine $m(T)$ [16].

Per $p \ll N$ abbiamo dunque, per un certo valore T_c del rumore, una transizione di fase di seconda specie.
Non imponendo il limite $p \ll N$ la trattazione diventa molto più lunga e complessa. Viene introdotto il parametro di carico $\alpha \equiv \frac{p}{N}$, che indica quanti patterns

chiediamo alla rete di memorizzare in relazione al numero di unità totali, e si studia l'andamento del nuovo parametro d'ordine al variare di α. L'analisi evidenzia la presenza di una transizione di fase di prima specie per $\alpha = \alpha_c$ (fig. 2.3).

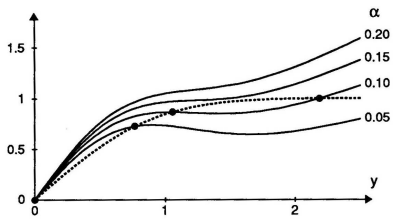

Figura 2.3: Risoluzione grafica dell'equazione del parametro d'ordine della transizione di fase di prima specie al variare di α [16].

In figura 2.4 vengono mostrate le diverse regioni presenti nello spazio dei parametri α e T. Le lettere A, B, C, D indicano regioni caratterizzate da diverse tipologie di attrattori (fig. 2.5) mentre in grigio è stata evidenziata la zona in cui i patterns risultano configurazioni stabili per la rete.

In entrambi i casi di transizione di prima e seconda specie, il parametro d'ordine è legato alla possibilità della rete di richiamare uno dei patterns memorizzati. La rete passa dunque dallo stato *memoria* ad una condizione di inutilizzabilità. Non essendo interessati ad approfondire la capacità massima di un network [20], nell'elaborazione dei nostri modelli ci terremo sempre lontani dalle transizioni di fase e in condizioni di buon funzionamento della rete.

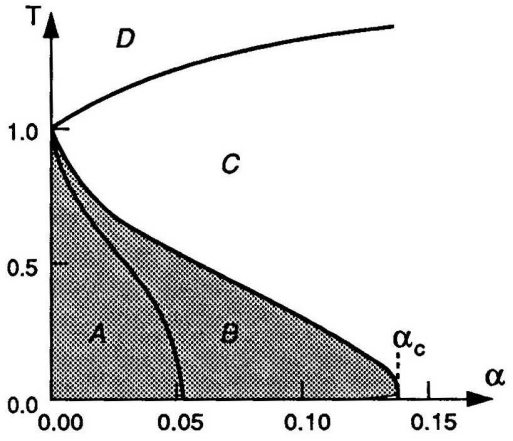

Figura 2.4: Spazio dei parametri. La zona grigia rappresenta i valori dei parametri per cui la rete può fare retrieval, con diversi gradi di qualità, dei patterns. In corrispondenza della linea che separa la zona B dalla C e nel punto $(\alpha = \alpha_c; T = 0)$ si ha una transizione di fase di prima specie; nel punto $(\alpha = 0; T = T_c)$ il sistema ha invece una transizione di fase di seconda specie. Le lettere A, B, C, D indicano regioni con diverse tipologie di attrattori (vedi fig. 2.5)[16].

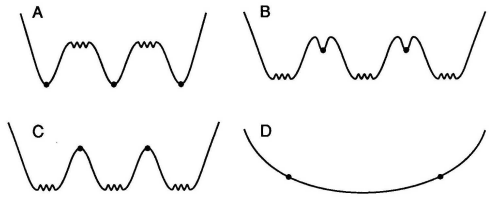

Figura 2.5: Visualizzazione dell'energy-landscape in quattro punti dello spazio dei parametri. Con riferimento alle zone dello spazio dei parametri evidenziate in figura 2.4 si ha: *regione C*, gli unici stati stabili del sistema sono gli stati di spin glass; *regione D*, per temperature così alte scompaiono anche gli stati di spin e la configurazione favorita è quella con $\langle \sigma_i \rangle = 0$; *regione B*, la rete ha sia stati di spin che patterns, ma quest'ultimi risultano ad energia maggiore e quindi meno favoriti dei primi; *regione A*, in questa regione gli stati più stabili del sistema sono i patterns, si ha quindi un buon funzionamento della memoria associativa [16].

Capitolo 3

Unità scalari

3.1 Un semplice modello

Nella corteccia, oltre alla funzionale attività di firing dei neuroni, vi sono continuamente fluttuazioni in attività neuronale dovute all'apertura casuale di vescicole sinaptiche, ritardi nel rilascio di neurotrasmettitori, ecc... Tutti questi elementi collaterali alla trasmissione di informazione possono essere introdotti nel nostro modello tramite un rumore termico, modulato dal parametro temperatura T (o da $\beta = \frac{1}{T}$). Chiaramente, tale temperatura non è in alcun modo legata a l'effettiva temperatura di una parte del cervello. L'introduzione di questo rumore (con il quale teniamo conto di tutte le fluttuazioni, non esplicitamente contemplate, con una scala temporale minore di quella del nostro sistema) permette adesso alla rete di evitare di incappare in minimi spuri e locali, come nel precedente modello, e di poter esplorare, in relazione al valore di temperatura, sempre più regioni del landscape.
Consideriamo allora una rete composta da N neuroni capaci di assumere non più un valore binario ma tutti i valori compresi tra 0 e 1. Un tipo di rete come questa è spesso utilizzata per osservare l'andamento dello stato medio di attività di una rete binaria. Nella nostra analisi preferiamo però considerare la variabile $\sigma_i \in [0, 1]$ come lo stato effettivo del singolo neurone i.
Definiamo il valore dello stato di attività di un neurone ad un tempo $t + \Delta t$ con la probabilità che, ad un update, l'elemento i-esimo di una rete binaria assuma

uno stato attivo. Essendo la rete un sistema con numero di unità fissate possiamo utilizzare il formalismo dell'insieme canonico. La probabilità che l'unità i-esima sia attiva risulta dunque

$$\sigma_i(t+\Delta t) \equiv P(\sigma_i = 1) = \frac{e^{-\beta H(\sigma_i=1)}}{\sum_{\{\sigma_i\}} e^{-\beta H(\sigma_i)}} \tag{3.1}$$

$$\begin{aligned} &= \frac{e^{\beta(h_i+\sum_j^N \sigma_j h_j)}}{\sum_{\{\sigma_i\}} e^{\beta \sum_i j^N \sigma_j h_j}} = \frac{e^{\beta(h_i+\sum_j^N \sigma_j h_j)}}{\sum_{\{\sigma_i\}} e^{\beta(\sigma_i h_i+\sum_j^N \sigma_j h_j)}} \\ &= \frac{e^{\beta h_i}}{\sum_{\{\sigma_i\}} e^{\beta \sigma_i h_i}} \\ &= \frac{e^{\beta h_i}}{e^{\beta U} + e^{\beta h_i}} = \frac{1}{1+e^{-\beta(h_i-U)}} \end{aligned} \tag{3.2}$$

$$P(\sigma_i = 0) = 1 - P(\sigma_i = 1) = \frac{1}{1+e^{\beta(h_i-U)}} \tag{3.3}$$

Notiamo che nel limite per T tendente a zero ritroviamo il modello di Hopfield con la tipica funzione a gradino (fig. 3.1).

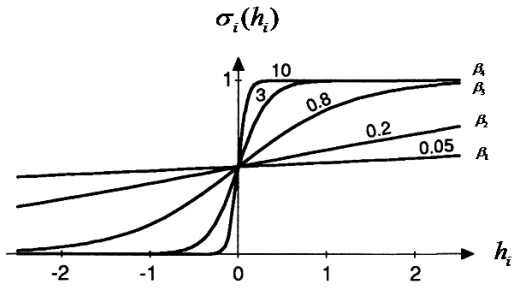

Figura 3.1: Andamento dello stato dell'unità i-esima al variare di del campo h_i per diversi valori del parametro β ($\beta_1 < \beta_4$)[16].

Per quanto riguarda i patterns, lasciamo che le unità, in configurazione di pattern, abbiano valori discreti 0 o 1; imponiamo però che la probabilità di assumere tali valori sia regolata dal parametro a, chiamato ***sparcity*** o *attività media*

($0 < a \ll \frac{1}{2}$). La necessità di introdurre una probabilità di attivazione delle unià di pattern minore di $\frac{1}{2}$ è dovuta al fatto che sperimentalmente è stato osservato che in media le cellule attive nella corteccia di un soggetto, anche durante lo svolgimento di un compito, sono molto meno della metà delle totali [1]. Una rete con una probabilità di attivazione di $\frac{1}{2}$ non è dunque in un regime fisiologico. Sia quindi

$$P(\xi_i^\mu = 0) = 1 - a \tag{3.4}$$
$$P(\xi_i^\mu = 1) = a \tag{3.5}$$

Il campo sentito da ogni neurone, come nel precedente modello, è

$$h_i = \sum_j^N w_{ij}\sigma_j \tag{3.6}$$

con matrice di pesi

$$w_{ij} = \frac{1}{a(1-a)N}\sum_{\mu=1}^{p}(\xi_i^\mu - a)(\xi_j^\mu - a). \tag{3.7}$$

Definiamo inoltre una variabile aggiuntiva m, l'***overlap***, che ci permette di seguire l'evoluzione della rete.

$$m_\mu \equiv \frac{1}{a(1-a)N}\sum_i^N(\xi_i^\mu - a)\sigma_i. \tag{3.8}$$

Tale variabile infatti quantifica la vicinanza del nostro vettore σ ad uno dei patterns memorizzati. Quando avremo $m_\mu(t) = 1$ sapremo che il nostro sistema sarà completamente allineato al pattern μ, ovvero che la rete avrà richiamato alla memoria e riconosciuto tale concetto. Inoltre, essendo un pattern una configurazione energeticamente privilegiata, ci aspettiamo che se $m_\mu(t_0) = 1$ allora sia $m_\mu(t) = 1$ per tutti i tempi maggiori di t_0.

3.1.1 Analisi segnale-rumore

Iniziamo adesso un tipo di analisi che useremo in tutto il corso dell'elaborato. Scriviamo il campo h_i in termini di overlap con i patterns del sistema

$$h_i(t) = \sum_j^N \frac{1}{a(1-a)N} \sum_{\mu=1}^{p} (\xi_i^\mu - a)(\xi_j^\mu - a)\sigma_j(t) = \sum_{\mu=1}^{p} (\xi_i^\mu - a) m_\mu(t) =$$

e mettiamo in evidenza il termine legato al pattern di cui andremo a fare il retrieval:

$$= (\xi_i^{ret} - a) m_{ret}(t) + \sum_{\mu \neq ret}^{p} (\xi_i^\mu - a) m_\mu(t). \tag{3.9}$$

Dal momento che abbiamo ipotizzato che la rete faccia il retrieval del pattern $\mu = ret$ sappiamo che, durante il retrieval, m_{ret} sarà vicino all'unità mentre gli altri m_μ risulteranno infinitesimi. Questo ci spinge a considerare $(\xi_i^{ret} - a)m_{ret}(t)$ come campo medio e la restante parte di campo come rumore:

$$R_i = \frac{1}{a(1-a)N} \sum_{\mu \neq ret}^{p} (\xi_i^\mu - a) \left[\sum_j^N \sigma_j (\xi_j^\mu - a) \right]. \tag{3.10}$$

Analizziamo le caratteristiche statistiche di un tale rumore.
Le variabili ξ_i e ξ_j sono per costruzione indipendenti. Per il teorema del limite centrale sappiamo quindi che per grandi p la variabile R_i tenderà ad avere una distribuzione gaussiana.
Calcoliamo quindi il valor medio, mediando su tutti i pattern possibili:

$$\langle R_i \rangle_\xi = \frac{1}{a(1-a)N} \sum_j^N \sigma_j \sum_{\mu \neq ret}^{p} \langle (\xi_i^\mu - a)(\xi_j^\mu - a) \rangle_\xi \tag{3.11}$$

e poiché ξ_i^μ e ξ_j^μ sono indipendenti, il valor medio risulta nullo:

$$\langle (\xi_i^\mu - a)(\xi_j^\mu - a) \rangle_\xi = \langle \xi_i^\mu - a \rangle \langle \xi_j^\mu - a \rangle = (\langle \xi_i^\mu \rangle - a)(\langle \xi_j^\mu \rangle - a) = (a-a)(a-a) = 0 \tag{3.12}$$

Anche se irrilevante in questo calcolo, notiamo che la media su tutti i pattern possibili non coinvolge lo stato σ_i, in quanto indipendente dal valore di pattern.
Per la varianza

$$\langle R_i^2 \rangle_\xi = \left\langle \frac{1}{[a(1-a)N]^2} \sum_{jy}^N \sigma_j \sigma_y \sum_{\mu,\nu \neq ret}^{p} (\xi_i^\mu - a)(\xi_i^\nu - a)(\xi_j^\mu - a)(\xi_y^\nu - a) \right\rangle_\xi =$$

analogamente a quanto mostrato sopra, i termini con $j \neq y$ e $\mu \neq \nu$ hanno media nulla, quindi

$$= \frac{1}{[a(1-a)N]^2} \sum_{j}^{N} {\sigma_j}^2 \sum_{\mu \neq ret}^{p} \langle (\xi_i^\mu - a)^2 (\xi_j^\mu - a)^2 \rangle_\xi$$

$$= \frac{1}{[a(1-a)N]^2} \sum_{j}^{N} {\sigma_j}^2 \sum_{\mu \neq ret}^{p} \big((1-a)^2 a + a^2 (1-a) \big)^2$$

$$= \frac{(p-1)a^2(1-a)^2}{[a(1-a)N]^2} \sum_{j}^{N} {\sigma_j}^2 \tag{3.13}$$

Definiamo allora una nuova variabile q che ci permetta di monitorare l'attivazione media della rete durante l'evoluzione dinamica

$$q(t) \equiv \frac{1}{aN} \sum_{j}^{N} {\sigma_j}^2; \tag{3.14}$$

in questi termini, possiamo esprimere la varianza come

$$\langle R_i^2 \rangle = \frac{(p-1)a}{N} q \tag{3.15}$$

che, per p grande, diventa

$$\langle R_i^2 \rangle = a\alpha q. \tag{3.16}$$

Torniamo adesso al campo e sostituiamo il rumore R_i con una variabile η_i con distribuzione normale moltiplicata per la deviazione standard di R_i:

$$h_i(t) = (\xi_i^{ret} - a)\ m_{ret}(t) + \sqrt{\alpha a q(t)}\ \eta_i. \tag{3.17}$$

Adesso possiamo mediare sul rumore gaussiano ottenendo:

$$\begin{aligned} \langle h_i(t) \rangle_{rumore} &= (\xi_i^{ret} - a)\ m_{ret}(t) + \int_{-\infty}^{+\infty} \frac{e^{-\frac{\eta_i^2}{2}}}{\sqrt{2\pi}} \sqrt{\alpha a q(t)}\ \eta \ \ d\eta_i \\ &= (\xi_i^{ret} - a)\ m_{ret}(t) \end{aligned} \tag{3.18}$$

3.1.2 Le equazioni del moto

A questo punto vogliamo ricavare delle equazioni differenziali che descrivano la dinamica delle le nostre variabili m_μ e q. Partiamo dalla definizione di overlap

$$m_\mu(t) = \frac{1}{a(1-a)N}\sum_i^N(\xi_i^\mu - a)\sigma_i(t) \tag{3.19}$$

Avendo scelto una dinamica asincrona, al tempo $t + \Delta t$ il valore di m non sarà altro che il valore dell'overlap al tempo t modificato col contributo dato dal neurone y appena aggiornato:

$$m_\mu(t+\Delta t) = m_\mu(t) + \frac{1}{a(1-a)N}(\xi_y^\mu - a)\left(\sigma_y(t+\Delta t) - \sigma_y(t)\right) \tag{3.20}$$

da qui, per incrementi infinitesimi

$$\frac{dm_\mu(t)}{dt} = \frac{m_\mu(t+\Delta t) - m_\mu(t)}{\frac{1}{N}} = \frac{1}{a(1-a)}(\xi_y^\mu - a)\left(\sigma_y(t+dt) - \sigma_y(t)\right) \tag{3.21}$$

Il neurone y è stato però scelto in modo casuale, possiamo quindi mediare su tutti i neuroni senza perdere alcuna informazione

$$\frac{dm_\mu(t)}{dt} = \frac{1}{aN(1-a)}\sum_i^N(\xi_i^\mu - a)\left(\sigma_i(t+dt) - \sigma_i(t)\right) \tag{3.22}$$

da cui, riconoscendo tra le componenti del termine di destra l'overlap

$$\begin{aligned}\frac{dm_\mu(t)}{dt} &= -m_\mu(t) + \frac{1}{aN(1-a)}\sum_i^N(\xi_i^\mu - a)\sigma_i(t+dt) \\ &= -m_\mu(t) + \frac{1}{aN(1-a)}\sum_i^N(\xi_i^\mu - a)\frac{1}{1+e^{-\beta(h_i-U)}}\end{aligned} \tag{3.23}$$

Applichiamo lo stesso procedimento anche alla variabile q:

$$q(t) = \frac{1}{Na}\sum_j^N \sigma_j^2(t) \tag{3.24}$$

$$q(t+\Delta t) = q(t) + \frac{1}{Na}\left(\sigma_y^2(t+\Delta t) - \sigma_y^2(t)\right) \tag{3.25}$$

$$\frac{q(t+\Delta t) - q(t)}{\frac{1}{N}} = \frac{1}{aN}\sum_i^N \left(\sigma_i^2(t+\Delta t) - \sigma_i^2(t)\right) \tag{3.26}$$

$$\frac{dq(t)}{dt} = -q(t) + \frac{1}{aN}\sum_i^N \sigma_i^2(t+\Delta t) \tag{3.27}$$

esplicitando

$$\begin{aligned}
\frac{dm_\mu(t)}{dt} &= -m_\mu(t) + \frac{1}{aN(1-a)}\sum_i^N(\xi_i^\mu - a)\frac{1}{1+e^{-\beta((\xi_i^{ret}-a)\ m_{ret}(t)+\sqrt{\alpha a q(t)}\ \eta - U)}} \\
\frac{dq(t)}{dt} &= -q(t) + \frac{1}{aN}\sum_i^N \frac{1}{(1+e^{-\beta(h_i-U)})^2}.
\end{aligned}$$

A questo punto possiamo usare nuovamente le informazioni che abbiamo sulla composizione dei pattern e sviluppare la sommatoria su tutte le N unità

$$\frac{dm_\mu(t)}{dt} = -m_\mu(t) + \frac{1}{aN(1-a)}\frac{\sum_i^N(\xi_i^\mu - a)}{1+e^{-\beta((\xi_i^{ret}-a)\ m_{ret}(t)+\sqrt{\alpha a q(t)}\ \eta - U)}}$$

con

$$\begin{aligned}
&\frac{\sum_i^N(\xi_i^\mu - a)}{1+e^{-\beta((\xi_i^{ret}-a)\ m_{ret}(t)+\sqrt{\alpha a q(t)}\ \eta - U)}} = \\
&= N\left[\frac{a(1-a)}{1+e^{-\beta((1-a)\ m_{ret}(t)+\sqrt{\alpha a q(t)}\ \eta - U)}} + \frac{(1-a)(-a)}{1+e^{-\beta(-a m_{ret}(t)+\sqrt{\alpha a q(t)}\ \eta - U)}}\right]
\end{aligned}$$

mentre per q

$$\frac{dq(t)}{dt} = -q(t) + \frac{1}{aN}\left[\frac{aN}{\left(1+e^{-\beta((1-a)m_{ret}+\sqrt{\alpha a q(t)}\eta)}\right)^2} - \frac{(1-a)N}{\left(1+e^{-\beta(-a m_{ret}+\sqrt{\alpha a q(t)}\eta)}\right)^2}\right].$$

Adesso, mediando sul rumore gaussiano otteniamo le equazioni differenziali cercate:

$$\left\langle \frac{dm_\mu(t)}{dt} \right\rangle_{\xi_i,\eta} = -m_\mu(t) + \int_{-\infty}^{+\infty} \frac{e^{-\frac{\eta^2}{2}}}{\sqrt{2\pi}} \left[\frac{1}{1+e^{-\beta((1-a)\ m_{ret}(t)+\sqrt{\alpha a q(t)}\ \eta-U)}} + \frac{1}{1+e^{-\beta(-a m_{ret}(t)+\sqrt{\alpha a q(t)}\ \eta-U)}} \right] d\eta$$

$$\left\langle \frac{dq(t)}{dt} \right\rangle_{\xi_i,\eta} = -q(t) + \int_{-\infty}^{+\infty} \frac{e^{-\frac{\eta^2}{2}}}{a\sqrt{2\pi}} \left[\frac{a}{\left(1+e^{-\beta((1-a)m_{ret}+\sqrt{\alpha a q(t)}\eta)}\right)^2} - \frac{1-a}{\left(1+e^{-\beta(-a m_{ret}+\sqrt{\alpha a q(t)}\eta)}\right)^2} \right] d\eta$$

Tali equazioni possono essere semplificate portando a zero la temperatura:

$$\frac{dq(t)}{dt} = -q(t) + \frac{1}{2a} + \frac{a-1}{2a}\mathrm{erf}\left(\frac{a m_{ret}+U}{\sqrt{2\alpha a q(t)}}\right) + \frac{1}{2}\mathrm{erf}\left(\frac{(1-a)m_{ret}-U}{\sqrt{2\alpha a q(t)}}\right)$$

$$\frac{dm_{ret}(t)}{dt} = -m_{ret}(t) + \frac{1}{2}\mathrm{erf}\left(\frac{a m_{ret}+U}{\sqrt{2\alpha a q(t)}}\right) + \frac{1}{2}\mathrm{erf}\left(\frac{(1-a)m_{ret}-U}{\sqrt{2\alpha a q(t)}}\right)$$

3.2 Simulazione e Integrazione Numerica

Le equazioni differenziali sopra trovate sono state risolte numericamente utilizzando il programma di integrazione numerica del software Mathematica (attraverso la funzione di integrazione numerica NDsolve che regola automaticamente il passo di integrazione in modo da mantenere un'accuratezza di 10^{-6}).

Parallelamente è stata sviluppata una simulazione (paragrafo A.1) nella quale, impostate le regole di update (3.2), (3.20), (3.25) e fissato il valore delle costanti, si osserva l'evoluzione temporale della rete.
I risultati ricavati con i due procedimenti sono poi stati confrontati per verificare la consistenza della struttura teorica.

Per stimolare il retrieval forniamo alla rete uno stimolo esterno simile al pattern che vogliamo richiamare. Tale stimolo viene inserito nella simulazione (e chiaramente anche nelle equazioni differenziali) sotto forma di campo esponenzialmente decrescente:

$$h_{ext} = g\ e^{-\frac{t}{\tau}} \tag{3.28}$$

con g parametro di intensità. La funzione di tale campo è di portare lo stato della rete nel bacino di attrazione del pattern da noi scelto per il retrieval.

I pattern sono stati costruiti assegnando lo stato 1, attivo, a Na neuroni scelti casualmente tra gli N. La rete è inizializzata con tutte le unità inattive.

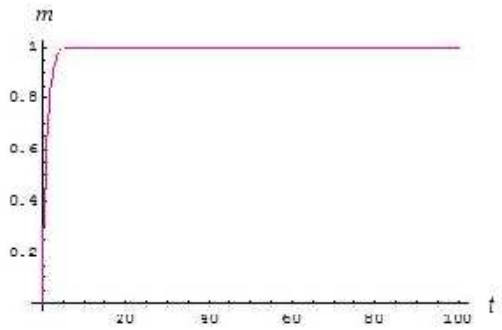

Figura 3.2: Andamento della variabile m_{ret} durante il retrieval del pattern ret. In nero è mostrata la previsione analitica e in rosa i dati ricavati dalla simulazione.

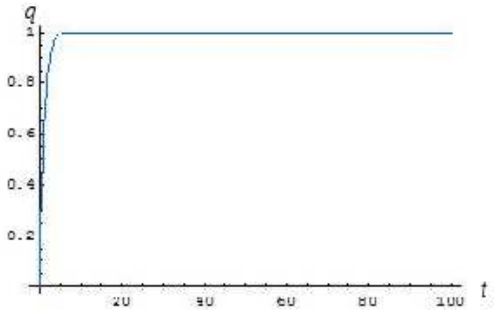

Figura 3.3: Andamento dell'attività media della rete durante il retrieval mostrato in figura 3.2. La previsione analitica è in nero, i dati ricavati dalla simulazione in celeste.

In figura 3.2 e 3.3 possiamo vedere l'andamento della variabile overlap e di q con

l'evolvere del tempo. Dall'immagine si osserva che il valore di m_{ret} sale velocemente ad 1, questo implica che la rete si è allineata col pattern verso cui l'abbiamo spinta ovvero ne ha fatto il retrieval. Come possiamo osservare il pattern di cui si è fatto il retrieval è per la rete un attrattore stabile, infatti, raggiunta l'unità i valori di m e q rimangono costanti nel tempo. Anche la variabile q raggiunge velocemente l'unità, questo ci dice che la rete sta lavorando in un regime di corretta attivazione (la variabile q è stata normalizzata in modo da assumere il valore 1 per un'attivazione media corrispondente alla sparcity).
Il valore delle costanti relative alle figure 3.2 e 3.3 è riportato nella seguente tabella.

N	p	a	U	g	τ	T
10000	1000	0.1	0.2	0.7	$2 * N$	$100 * N$

Dalle immagini si osserva anche un ottimo accordo tra i risultati ottenuti dalla simulazione (in colore) e l'andamento previsto analiticamente (in nero).

Adesso che abbiamo un modello giocattolo di come possa avvenire il richiamo alla memoria di un concetto possiamo porci il problema di come da questo si possa articolare un pensiero. Il modello che abbiamo costruito infatti ci permette al momento di pensare ad un cane, ma prevede poi che si pensi al cane per il resto della nostra vita. Quello che ci aspetteremo invece è che, dopo aver pensato ad un cane, magari a causa di uno stimolo esterno, la nostra attenzione mentale si sposti spontaneamente su altri concetti ad esso collegati. Occorre quindi modificare la dinamica in modo che dall'attrattore cane si possa passare ad un'altro attrattore, come ad esempio 'gatto' o 'cuccia'.
Per permettere questo, che chiameremo fenomeno di **latching**, occorre modificare il nostro modello. Dobbiamo, infatti, trovare il modo di far uscire la rete dall'attrattore inizialmente richiamato e capire, poi, come questa possa cadere spontaneamente nel bacino di un nuovo attrattore.
Rispondiamo nel prossimo paragrafo alla prima di queste due esigenze.

Capitolo 4

Il Pensiero Articolato

Intuitivamente possiamo giustificare che il sistema esca da un attrattore pensando che tutti i neuroni coinvolti nel retrieval di un pattern tendano, dopo un tempo caratteristico, fisiologicamente ad 'adattare', ovvero a spostare la loro soglia sul livello medio di attività dell'unità.
Generalmente ogni neurone, se sottoposto ad una stimolazione costante, tende a spostare la propria soglia di attivazione regolandola sul livello di stimolazione medio. Un tale comportamento, a prima vista immotivato, si rivela di grande utilità permettendo di ampliare notevolmente il range di sensibilità della cellula. Se, per la variazione prolungata delle condizioni esterne, si spostasse il livello di sollecitazione media, molte cellule diventerebbero inutilizzabili perché sempre o molto sopra, quindi costantemente attive, o molto sotto, costantemente inattive, il livello di soglia. Grazie all'adattamento, invece, a patto di una stimolazione sufficientemente prolungata, le cellule sono sensibili, e quindi utili, in un ben più vasto range d'intensità.
Possiamo allora pensare che i neuroni attivi nel retrieval di un pattern non possano verosimilmente restare attivi per sempre ma, dopo un tempo caratteristico, adattino e tendano ad inattivarsi, permettendo alla rete di uscire dalla configurazione di attrattore.

Cerchiamo allora di tradurre il fenomeno di adattamento nel formalismo della nostra rete.

4.1 L'Adattamento

Introduciamo due nuove variabili: θ e $\mathbf{r}$.
La variabile θ_i rappresenta la soglia dinamica che i contributi eccitatori devono superare per attivare il neurone. Questa, con un certo ritardo modulato dalla costante b_2, tenderà a seguire il valore medio dello stato σ_i. La variabile r_i invece rappresenta il nuovo campo, modificato con la soglia dinamica θ_i, che la cellula percepisce per azione delle altre connesse. Anche alla variabile r_i viene associata una costante b_1 che ne moduli la velocità di adattamento. Per ogni time step avremo:

$$r_i(t+\Delta t) = r_i(t) + b_1\left[h_i(t) - \theta_i(t) - r_i(\,t)\right] \tag{4.1}$$

$$\theta_i(t+\Delta t) = \theta_i(t) + b_2\left[\sigma_i(t) - \theta_i(t)\right] \tag{4.2}$$

e la nuova regola di update:

$$\sigma_i(t+\Delta t) = \frac{1}{1+e^{-\beta(r_i(t)-U)}} \tag{4.3}$$

Il procedimento con il quale studiamo la dinamica di m_μ e q è lo stesso usato nel precedente capitolo. Mentre nel caso precedente però potevamo utilizzare l'informazione sulla composizione dei patterns direttamente in h_i (paragrafo 3.1.2) adesso dobbiamo lavorare con r_i, variabile del sistema.
Occorre dunque usare un approccio leggermente diverso.

Il retrieval porta le unità della rete a formare due gruppi con comportamenti dinamici differenti: le unità per cui $\xi_i^{ret} = 1$ e quelle per cui $\xi_i^{ret} = 0$. Al momento in cui il sistema si allineerà al pattern $\mu = ret$ sappiamo infatti che circa Na neuroni saranno attivi e $N(1-a)$ inattivi. Questi due gruppi, tendendo a due stati diversi, avranno dinamiche differenti. Tale divisione era presente anche nel precedente modello solo che era mantenuta implicita nel calcolo di m e q. Adesso però, essendo r_i e θ_i variabili del sistema, fare questo tipo di distinzione è necessario.

Definiamo dunque r_1 e θ_1 campo e soglia relativi alle unità che assumono valore 1 in ξ^{ret} e r_0 e θ_0 relativi a quelle che assumono valore 0 in ξ^{ret}. Abbiamo allora:

$$\begin{aligned} r_1(t+\Delta t) &= r_1(t) + b_1\left[h_1(t) - \theta_1(t) - r_1(t)\right] && (4.4)\\ r_0(t+\Delta t) &= r_0(t) + b_1\left[h_0(t) - \theta_1(t) - r_0(t)\right] && (4.5)\\ \theta_1(t+\Delta t) &= \theta_1(t) + b_2\left[\sigma_1(t) - \theta_1(t)\right] && (4.6)\\ \theta_0(t+\Delta t) &= \theta_0(t) + b_2\left[\sigma_0(t) - \theta_0(t)\right] && (4.7) \end{aligned}$$

in cui

$$\begin{aligned} \sigma_1(t+\Delta t) &= \frac{1}{1+e^{-\beta(r_1(t)-U)}} && (4.8)\\ \sigma_0(t+\Delta t) &= \frac{1}{1+e^{-\beta(r_0(t)-U)}} && (4.9) \end{aligned}$$

e

$$\begin{aligned} h_1(t+\Delta t) &= (1-a)m_{ret} + \sqrt{\alpha a q(t)}\eta && (4.10)\\ h_0(t+\Delta t) &= -a m_{ret} + \sqrt{\alpha a q(t)}\eta && (4.11) \end{aligned}$$

dalle quali, mediando sul rumore in h, si ricavano le equazioni differenziali

$$\begin{aligned} \frac{dm_{ret}}{dt} &= -m_{ret} + \frac{1}{1+e^{\beta(r_1-U)}} - \frac{1}{1+e^{\beta(r_0-U)}}\\ \frac{dq}{dt} &= -q + \frac{1}{\left[1+e^{\beta(r_1-U)}\right]^2} + \frac{(1-a)}{a}\frac{1}{\left[1+e^{\beta(r_0-U)}\right]^2}\\ \frac{dr_0}{dt} &= b_1\left(-a m_{ret} - \theta_0 - r_0\right)\\ \frac{dr_1}{dt} &= b_1\left((1-a)m_{ret} - \theta_1 - r_1\right)\\ \frac{d\theta_0}{dt} &= b_2\left(\frac{1}{1+e^{\beta(r_0-U)}} - \theta_0\right)\\ \frac{d\theta_1}{dt} &= b_2\left(\frac{1}{1+e^{\beta(r_1-U)}} - \theta_1\right) \end{aligned}$$

e le relative per temperatura nulla

$$\begin{aligned}
\frac{dm_{ret}}{dt} &= -m_{ret} + \Theta(r_1 - U) - \Theta(r_0 - U) \\
\frac{dq}{dt} &= -q + \Theta(r_1 - U) + \Theta(r_0 - U) \\
\frac{dr_0}{dt} &= b_1(-am_{ret} - \theta_0 - r_0) \\
\frac{dr_1}{dt} &= b_1((1-a)m_{ret} - \theta_1 - r_1) \\
\frac{d\theta_0}{dt} &= b_2(\Theta(r_0 - U) - \theta_0) \\
\frac{d\theta_1}{dt} &= b_2(\Theta(r_1 - U) - \theta_1)
\end{aligned}$$

con Θ finzione a gradino di Heaviside.

4.2 Simulazione e Integrazione Numerica

Analogamente a quanto fatto per il modello precedente, le equazioni differenziali sono state risolte numericamente utilizzando il software Mathematica (attraverso la funzione di integrazione numerica NDsolve che regola automaticamente il passo di integrazione in modo da mantenere un'accuratezza di 10^{-6}) e, in parallelo, è stata impostata la simulazione allegata (paragrafo A.2).

La rete è inizializzata con tutte le unità inattive ed al tempo t_0 è stato acceso il campo $h_{ext} = \Theta[t - t_0]\ g\ e^{-\frac{t-t_0}{\tau}}$ per stimolare il retrieval di un pattern.
I patterns sono stati costruiti assegnando lo stato 1, attivo, a Na neuroni, scelti casualmente tra gli N. I valori numerici delle costanti utilizzati per le simulazioni e l'integrazione delle equazioni differenziali sono:

N	p	a	U	$b1$	$b2$	g	τ	t_0	T
10000	10	0.1	0.2	0.01	0.02	6	$70 * N$	$200 * N$	$1000 * N$

Nelle immagini in figura 4.1 si osserva l'andamento della variabile overlap m e di q con l'evolvere del tempo. Al momento dell'attivazione del campo la rete fa il retrieval del pattern prescelto portando il valore di m_{ret} e q a 1. Come voluto però, adesso, a causa dell'adaptation dopo un dato tempo la rete esce dall'attrattore del pattern di cui aveva fatto il retrieval e ritorna allo stato di inattività.

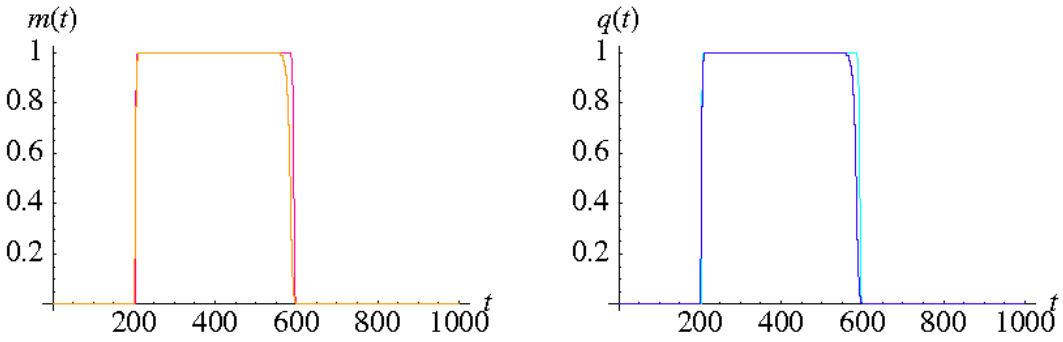

Figura 4.1: Le figure mostrano l'andamento della variabile m_{ret} e q durante il retrieval del pattern ret e il successivo adattamento (per $t \sim 600$). La previsione analitica è tracciata in rosso (m_{ret}) e celeste (q), mentre rispettivamente in arancione e blu sono graficati i risultati della simulazione.

Nelle figure 4.2 e 4.3 è graficato l'andamento delle variabili $\theta(t)$ e $r(t)$. Ogni punto corrisponde al valore di $\theta_i(t)$ (e $r_i(t)$) per un'unità scelta in modo random tra le N (non necessariamente quella appena aggiornata). Per rendere più leggibile i grafici è stato plottato un punto ogni 500 update. Salta immediatamente all'occhio la formazione di due gruppi dinamici di unità. In figura 4.2 abbiamo plottato in blu il valore di θ per la unità con $\xi_i = 1$ e in giallo quello per le unità con $\xi_i = 0$. Possiamo vedere che i diversi colori discriminano bene i due gruppi di unità, questo, assieme al buon accordo con la soluzione analitica (rosa per t_1 e verde per t_0), conferma la correttezza delle nostre ipotesi. Analogamente in figura 4.3 è mostrato in rosa l'andamento di r per le unità con $\xi_i = 1$ e in giallo per quelle con $\xi_i = 0$, mentre la linea blu mostra l'andamento analitico della variabile r_1 e la rosa quello della variabile r_0.

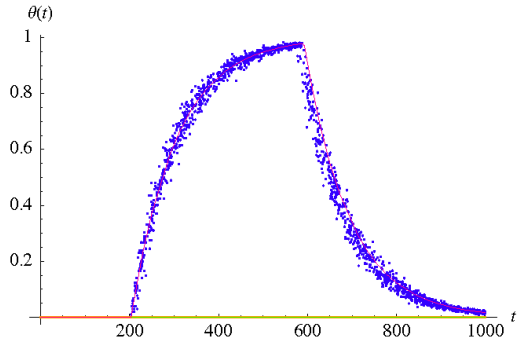

Figura 4.2: Soglia efficacie. Risultati della simulazione: in blu per le unità con $\xi_i = 1$, in giallo per quelle con $\xi_i = 0$. Risultati dell'integrazione numerica: in rosa la variabile θ_1, in verde la variabile θ_0.

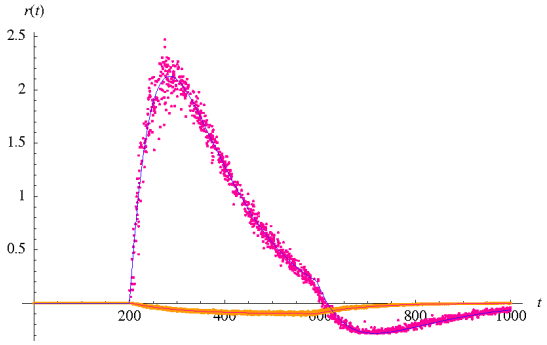

Figura 4.3: Campo. Risultati della simulazione: in rosa per le unità con $\xi_i = 1$, in giallo per quelle con $\xi_i = 0$. Risultati dell'integrazione numerica: in blu la variabile θ_1, in rosa la variabile θ_0.

Capitolo 5

Un nuovo punto di vista

Come annunciato dovremmo adesso riuscire a far passare la rete da un primo ad un secondo attrattore ad esso correlato. Preferiamo però rimandare questo passo per introdurre un'altra sostanziale modifica del nostro modello.

Torniamo allora all'*assemblea di cellule* suggerita da Hebb come conseguenza della sua learning rule e cerchiamo di individuare a livello corticale biologico come possa articolarsi una simile struttura.
Nonostante risulti più intuitivo immaginare una tale assemblea come un gruppo di neuroni adiacenti localizzati in una ristretta zona della corteccia, non bisogna pensare che tale condizione sia necessaria. Si osserva infatti, per tutta l'estensione della corteccia, la presenza di cellule piramidali, fittamente connesse con gli adiacenti interneuroni e più blandamente connesse con altre cellule piramidali dislocate in settori della corteccia anche completamente diversi [7]. Possiamo allora pensare ai componenti di questa assemblea come gruppi locali, magari rappresentanti una caratteristica, collegati da cellule piramidali ad altri gruppi relativi ad aspetti diversi dello stesso concetto [21]. In prima approssimazione consideriamo allora la corteccia come strutturata su due livelli: quello locale, con numerosi collegamenti cellula piramidale-interneurone e interneurone-interneurone, e quello globale, con una rete a maglie più ampie di collegamenti cellula piramidale-cellula piramidale [7].
Se cerchiamo di leggere questa struttura con la lente dei modelli portati avanti

fino a questo punto possiamo vedere la rete locale come un network isolato che, in base agli imput esterni, si allinea ad uno dei suoi S possibili patterns locali. Al livello globale possiamo invece considerare la rete complessiva costituita da unità capaci di assumere con uguale probabilità non più due ma S valori possibili. Quest'ultima rete avrà a sua volta degli attrattori, patterns globali rappresentativi in questo caso dell'intero concetto (fig. 5.1).

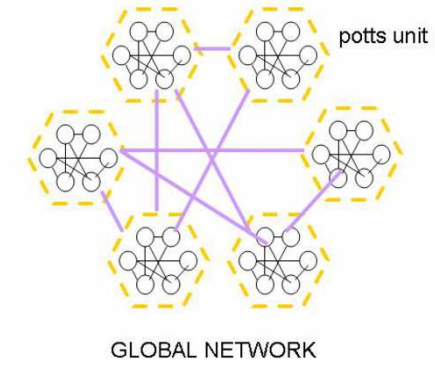

Figura 5.1: Se pensiamo alla corteccia come ad una struttura a due livelli, locale e globale, possiamo pensare le unità della rete come l'output di piccoli networks locali e lo stato relativo come il pattern locale in cui tali networks si trovano, istante per istante.

È da una visione di questo tipo che nasce l'idea di rappresentare la corteccia come rete di Potts.

5.1 La rete di Potts

Cerchiamo quindi di costruire un nuovo modello unendo al vecchio network le idee fondamentali del modello di Potts per gli stati di spin.

5.1.1 La grammatica

Consideriamo una rete di N unità ognuna delle quali ha S stati attivi (labellati con i numeri da 1 a S compreso) e uno inattivo (identificato dall'indice 0). Come

anticipato, ogni stato attivo corrisponde ad uno degli S patterns della rete locale mentre lo stato inattivo ad una completa inattività di tale rete. Tale numerazione, da 1 a S, non ha dunque nessun valore quantitativo e serve esclusivamente per distinguere i diversi attrattori dei network locali.
Analogamente a quanto fatto per la rete scalare, con l'introduzione di una temperatura diversa da zero, imponiamo che le unità assumano, per ogni stato di attività, un valore compreso tra 0 e 1.
Al posto dello scalare σ_i avremo dunque un vettore in S dimensioni (fig. 5.2), in cui ogni componente $\sigma_i^l \in [0,1]$, con $i = 1, ..., N$ e $l = 0, ..., S$, misura quanto bene sia stato richiamato l'aspetto l-esimo nel network locale i. La possibilità di non cadere in nessun attrattore locale è contemplata invece aggiungendo uno stato 'zero'. Infine, pochè al livello locale il network non può avere un contemporaneo e completo retrieval di tutti gli S stati, imponiamo la normalizzazione $\sum_{k=0}^{S} \sigma_i^k = 1$.

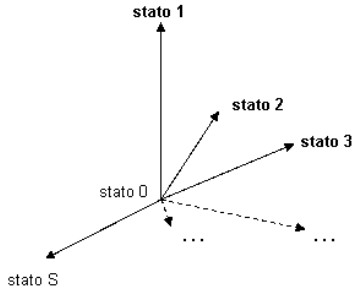

Figura 5.2: Lo stato σ_i dell'unità della rete corrisponde ad un vettore in uno spazio S dimensionale. In caso di $T = 0$ la dinamica di tale vettore è discretizzata e si muove per configurazioni coincidenti con gli assi.

Per quanto riguarda i patterns (globali) della rete assumiamo che, in configurazione di pattern, ogni unità sia completamente in uno stato, ovvero che al livello locale la rete sia in uno specifico attrattore. Ne segue che ogni elemento ξ_i assume un valore, discreto, da 0 a S.
Definiamo adesso la nuova regola di update, che in questo caso dovrà essere applicata ad ogni stato dell'unità aggiornata:

$$\sigma_i^k = \frac{e^{\beta r_i^k}}{\sum_{l=0}^{S} e^{\beta r_i^l}}. \tag{5.1}$$

In particolare notiamo che la somma su tutti gli stati del valore di attivazione è 1.

Per quanto riguarda la matrice dei pesi, rappresentante adesso le connessioni a lungo raggio tra i diversi network locali della corteccia, abbiamo un tensore a quattro indici:

$$J_{ij}^{kl} = \frac{(1-\delta_{ij})}{Na(1-\frac{a}{S})} \sum_{\mu=1}^{p} \left(\delta_{\xi_i^\mu k} - \frac{a}{S}\right)\left(\delta_{\xi_j^\mu l} - \frac{a}{S}\right)(1-\delta_{k0})(1-\delta_{l0}) \tag{5.2}$$

Definiamo il nuovo vettore di overlap m_μ

$$m_\mu \equiv \frac{1}{Na(1-\frac{a}{S})} \sum_{j\neq i}^{N}\sum_{l\neq 0}^{S} \left(\delta_{\xi_j^\mu l} - \frac{a}{S}\right)\sigma_j^l \tag{5.3}$$

e il campo

$$\begin{aligned} h_i^k &= \sum_{j\neq i}^{N}\sum_{l=0}^{S} J_{ij}^{kl}\sigma_j^l = \sum_{j\neq i}^{N}\sum_{l\neq 0}^{S} \frac{(1-\delta_{ij})}{Na(1-\frac{a}{S})} \sum_{\mu=1}^{p} \left(\delta_{\xi_i^\mu k} - \frac{a}{S}\right)\left(\delta_{\xi_j^\mu l} - \frac{a}{S}\right)\sigma_j^l \\ &= \sum_{\mu=0}^{p} m_\mu \left(\delta_{\xi_i^\mu k} - \frac{a}{S}\right) \end{aligned}$$

con $k \neq 0$ e, per $k = 0$

$$h_i^0 = U.$$

Infine scriviamo le equazioni per l'adattamento del campo e della soglia efficace che percepiscono le unità di Potts:

$$\begin{aligned} r_i^k(t+\Delta t) &= r_i^k(t) + b_1[h_i^k(t) - \theta_i^k(t) - r_i^k(t)] \\ \theta_i^k(t+\Delta t) &= \theta_i^k(t) + b_2[s_i^k(t) - \theta_i^k(t)] \end{aligned}$$

dove b_1 e b_2 sono le costanti tempo relative all'adattamento del campo e della soglia.

5.1.2 Analisi segnale-rumore

Definita la nuova 'grammatica' procediamo con la nostra analisi in analogia a quello che abbiamo fatto per i casi precedenti. Per prima cosa isoliamo la parte legata al pattern *ret* da quella relativa agli altri patterns:

$$\begin{aligned} h_i^k &= (\delta_{\xi_i^{ret}k} - \frac{a}{S})m_{ret} + \sum_{\mu\neq ret}^{p} m_\mu(\delta_{\xi_i^\mu k} - \frac{a}{S}) \\ &= (\delta_{\xi_i^{ret}k} - \frac{a}{S})m_{ret} + \frac{1}{Na(1-\frac{a}{S})}\sum_{\mu\neq ret}^{p}\sum_{j\neq i}^{N}\sum_{l\neq 0}^{S}(\delta_{\xi_i^\mu k} - \frac{a}{S})(\delta_{\xi_j^\mu l} - \frac{a}{S})\sigma_j^l \\ &= (\delta_{\xi_i^{ret}k} - \frac{a}{S})m_{ret} + R_k \end{aligned}$$

dove con R_k abbiamo indicato il rumore generato dalle altre configurazioni stabili

$$R_k \equiv \frac{1}{Na(1-\frac{a}{S})}\left(\sum_{\mu\neq ret}^{p}\sum_{j\neq i}^{N}\sum_{l\neq 0}^{S}(\delta_{\xi_i^\mu k} - \frac{a}{S})(\delta_{\xi_j^\mu l} - \frac{a}{S})\sigma_j^l\right) \tag{5.4}$$

Studiamone quindi il valor medio e la varianza per poter definire la nuova variabile q. Come nei casi precedenti (paragrafo 3.1.1) la media del rumore risulta nulla e per la varianza abbiamo:

$$\langle R_k^2\rangle_\xi = \frac{1}{[Na(1-\frac{a}{S})]^2}\left(\sum_{\mu\nu\neq ret}^{p}\sum_{jy\neq i}^{N}\sum_{lh\neq 0}^{S}\sigma_j^l\sigma_y^h\langle(\delta_{\xi_i^\mu k} - \frac{a}{S})(\delta_{\xi_i^\nu k} - \frac{a}{S})(\delta_{\xi_j^\mu l} - \frac{a}{S})(\delta_{\xi_y^\nu h} - \frac{a}{S})\rangle\right)$$

che risulta diversa da 0 solo per $j = y$ e $\mu = \nu$. Lo stesso ragionamento questa volta non può essere fatto per h e l, poiché correlati. Risulta quindi:

$$= \frac{1}{[Na(1-\frac{a}{S})]^2}\left(\sum_{\mu\neq ret}^{p}\sum_{j\neq i}^{N}\sum_{lh\neq 0}^{S}\sigma_j^l\sigma_j^h\langle(\delta_{\xi_i^\mu k} - \frac{a}{S})^2(\delta_{\xi_j^\mu l} - \frac{a}{S})(\delta_{\xi_j^\nu h} - \frac{a}{S})\rangle\right) \tag{5.5}$$

essendo poi

$$\langle(\delta_{\xi_i^\mu k} - \frac{a}{S})^2\rangle = (1-\frac{a}{S})^2\frac{a}{S} + (-\frac{a}{S})^2(1-\frac{a}{S}) = \frac{a}{S}(1-\frac{a}{S})$$
$$= \frac{\frac{a}{S}(1-\frac{a}{S})}{[Na(1-\frac{a}{S})]^2}\left(\sum_{\mu\neq ret}^{p}\sum_{j\neq i}^{N}\sum_{lh\neq 0}^{S}\sigma_j^l\sigma_j^h\langle((\delta_{\xi_j^\mu l} - \frac{a}{S})(\delta_{\xi_j^\nu h} - \frac{a}{S})\rangle\right)$$

dopo una serie di passaggi si arriva al risultato

$$\langle R_k^2\rangle_\xi = \frac{\alpha}{NS^2(1-\frac{a}{S})}\sum_{j\neq i}^{N}\left(\sum_{l\neq 0}^{S}{\sigma_j^l}^2 - \frac{a}{S}(1-\sigma_j^0)^2\right). \tag{5.6}$$

Definiamo quindi la variabile q come

$$q \equiv \frac{1}{Na(1-\frac{a}{S})}\sum_{j\neq i}^{N}\left[\sum_{l\neq 0}^{S}{\sigma_j^l}^2 - \frac{a}{S}(1-\sigma_j^0)^2\right] \tag{5.7}$$

che, anche in questo caso, risulta strettamente legata al livello di attività complessivo della rete. Possiamo infine scrivere:

$$\langle R_k^2\rangle_\xi = \alpha\frac{a}{S^2}q \tag{5.8}$$

e

$$h_i^k = (\delta_{\xi_i^{ret}k} - \frac{a}{S})m_{ret} + \eta\sqrt{\alpha\frac{a}{S^2}q}. \tag{5.9}$$

con η variabile normale.

Abbiamo così definito le due variabili m_μ e q nel caso di rete di Potts.

5.1.3 La dinamica

Occupiamoci adesso della parte dinamica del problema, cerchiamo, ovvero, di ricavare anche in questo caso delle equazioni differenziali che ci permettano di seguire l'evoluzione della rete.

Analogamente a quanto fatto per i casi precedenti consideriamo l'update di una generica unità y. Per la variabile overlap risulta:

$$
\begin{aligned}
m_\mu(t+\Delta t) &= m_\mu(t) + \frac{(1-\delta_{ij})}{Na(1-\frac{a}{S})}\sum_{l\neq 0}^{S}(\delta_{\xi_y^\mu l} - \frac{a}{S})(\sigma_y^l - \sigma_{old_y}{}^l) \\
\frac{m_\mu(t+\Delta t) - m_\mu(t)}{\frac{1}{N}} &= \frac{(1-\delta_{ij})}{a(1-\frac{a}{S})}\sum_{l\neq 0}^{S}(\delta_{\xi_y^\mu l} - \frac{a}{S})\left(\sigma_y^l(t+\Delta t) - \sigma_y^l(t)\right) \\
\frac{dm_\mu(t)}{dt} &= \frac{(1-\delta_{ij})}{a(1-\frac{a}{S})}\sum_{l\neq 0}^{S}(\delta_{\xi_y^\mu l} - \frac{a}{S})\left(\sigma_y^l(t) - \sigma_y^l(t+\Delta t)\right)
\end{aligned}
$$

mediamo su tutte le possibili unità scelte per l'update:

$$
\begin{aligned}
\frac{dm_\mu(t)}{dt} &= \frac{(1-\delta_{ij})}{Na(1-\frac{a}{S})}\sum_{i}^{N}\sum_{l\neq 0}^{S}(\delta_{\xi_i^\mu l} - \frac{a}{S})\left(\sigma_i^l(t) - \sigma_i^l(t+\Delta t)\right) \\
&= -m_\mu(t) + \frac{(1-\delta_{ij})}{Na(1-\frac{a}{S})}\sum_{i}^{N}\sum_{l\neq 0}^{S}(\delta_{\xi_i^\mu l} - \frac{a}{S})\sigma_i^l(t+\Delta t) \\
&= -m_\mu(t) + \frac{(1-\delta_{ij})}{Na(1-\frac{a}{S})}\sum_{i}^{N}\sum_{l\neq 0}^{S}(\delta_{\xi_i^\mu l} - \frac{a}{S})\frac{e^{\beta r_i^l}}{\sum_{n=0}^{S} e^{\beta r_i^n}}.
\end{aligned}
$$

Lo stesso procedimento viene applicato per la trattazione di q:

$$
q(t+\Delta t) = q(t) + \frac{1}{Na(1-\frac{a}{S})}\left[\sum_{l\neq 0}^{S}\sigma_y^l{}^2 - \frac{a}{S}(1-\sigma_y^0)^2 - \left(\sum_{l\neq 0}^{S}\sigma_{old_y}{}^l{}^2 - \frac{a}{S}(1-\sigma_{old_y}{}^0)^2\right)\right]
$$

$$
\frac{dq(t)}{dt} = \frac{1}{a(1-\frac{a}{S})}\left[\sum_{l\neq 0}^{S}\sigma_y^l(t+\Delta t)^2 - \frac{a}{S}(1-\sigma_y^0(t+\Delta t))^2 - \left(\sum_{l\neq 0}^{S}\sigma_y^l(t)^2 - \frac{a}{S}(1-\sigma_y^0(t))^2\right)\right]
$$

$$
\frac{dq(t)}{dt} = -q(t) + \frac{1}{a(1-\frac{a}{S})}\left[\sum_{l\neq 0}^{S}\sigma_y^l(t+\Delta t)^2 - \frac{a}{S}(1-\sigma_y^0(t+\Delta t))^2\right]
$$

mediando su y

$$\frac{dq(t)}{dt} = -q(t) + \frac{1}{Na(1-\frac{a}{S})} \sum_{i}^{N} \left[\sum_{l\neq 0}^{S} \left(\frac{e^{\beta r_i^l}}{\sum_{n=0}^{S} e^{\beta r_i^n}} \right)^2 - \frac{a}{S} \left(1 - \frac{e^{\beta r_i^0}}{\sum_{n=0}^{S} e^{\beta r_i^n})} \right)^2 \right].$$

Come discusso nel paragrafo 4.1 cerchiamo adesso di raggruppare le N unità in sottogruppi con dinamica comune. Nel precedente caso sono stati evidenziati due gruppi dinamici; adesso però, potendo le unità nei patterns assumere $S+1$ distinti valori, il numero di tali gruppi aumenta e la descrizione diventa molto più articolata.
Ricordiamo innanzitutto che se alle unità in configurazione di patters associamo, per comodità, direttamente il numero dello stato in cui queste sono attive, per le stesse in una configurazione generica consideriamo contemporaneamente tutti gli $S+1$ stati. In questi $S+1$ stati l'unità assumerà valori compresi tra 0 e 1 in relazione alla sua vicinanza con uno o un altro attrattore locale.

Analizziamo quindi cosa accade durante il retrieval di un pattern.
Nel momento in cui lo stato del sistema entra nel bacino di un attrattore ogni unità percepisce un'influenza diversa in funzione del valore che questa assume nella configurazione di pattern. Se per l'unità i-esima $\xi_i = 0$ allora il campo (e la soglia) relativo allo stato inattivo di σ_i sarà diverso da quello relativo a un generico stato attivo dello stesso. Abbiamo dunque la distinzione di due gruppi: il gruppo $\{ss\}$, che segue lo stato inattivo di un'unità per cui $\xi_i = 0$, e il $\{sk\}$, che segue gli stati attivi di un'unità con $\xi_i = 0$. Tali stati sono riuniti in uno stesso gruppo in quanto non abbiamo nessun motivo di prevedere una dinamica che privilegi uno stato attivo rispetto ad un'altro sempre attivo.
Nel caso invece di un'unità per cui $\xi_i = x$, dove x è un qualunque stato attivo, si avrà la formazione di tre gruppi: il primo, $\{xs\}$, relativo allo stato inattivo di un'unità con $\xi_i = x$, il secondo, il $\{xx\}$, relativo allo stato attivo x di un'unità con pattern attivo nello specifico stato x, infine il $\{xk\}$, relativo agli stati attivi k di un'unità il cui relativo pattern è attivo ma in uno stato diverso.

Le variabili dinamiche del sistema relative all'unità i sono dunque: $m(t)$, $q(t)$, $r_{ss}(t)$, $r_{sk}(t)$, $r_{xs}(t)$, $r_{xk}(t)$, $r_{xx}(t)$ (i campi), $t_{ss}(t)$, $t_{sk}(t)$, $t_{xs}(t)$, $t_{xk}(t)$, $t_{xx}(t)$ (le

soglie).

5.1.4 Un sistema chiuso

Riprendiamo le equazioni trovate per m e q e cerchiamo di esplicitare la dipendenza tra variabili per poter arrivare ad un sistema di 12 equazioni differenziali in 12 variabili.
Per l'overlap:

$$\frac{dm_\mu(t)}{dt} = -m_\mu(t) + \frac{(1-\delta_{ij})}{Na(1-\frac{a}{S})}\sum_i^N\sum_{l\neq 0}^S(\delta_{\xi_i^\mu l} - \frac{a}{S})\frac{e^{\beta r_i^l}}{\sum_{n=0}^S e^{\beta r_i^n}}. \tag{5.10}$$

Sappiamo che, per costruzione, $\xi_i^\mu \neq S$ con probabilità a e $\xi_i^\mu = S$ con probabilità $(1-a)$. Possiamo dunque sviluppare la seguente espressione

$$\begin{aligned}\sum_i^N\sum_{l\neq 0}^S(\delta_{\xi_i^\mu l} - \frac{a}{S})\frac{e^{\beta r_i^l}}{\sum_{n=0}^S e^{\beta r_i^n}} =& N(1-a)\left(\sum_{l\neq 0}^S(-\frac{a}{S})\frac{e^{\beta r_i^l}}{\sum_{n=0}^S e^{\beta r_i^n}}\right)\Bigg|_{\xi=0} \\ &+Na\left((1-\frac{a}{S})\frac{e^{\beta r_i^x}}{\sum_{n=0}^S e^{\beta r_i^n}} + (-\frac{a}{S})\sum_{l\neq 0,x}^S\frac{e^{\beta r_i^l}}{\sum_{n=0}^S e^{\beta r_i^n}}\right)\Bigg|_{\xi=x}\end{aligned}$$

per trovare

$$\frac{dm_\mu(t)}{dt} = -m_\mu(t) - \frac{a(1-a)}{S(1-\frac{a}{S})}\frac{\sum_{l\neq 0}^S e^{\beta r_i^l}}{\sum_{n=0}^S e^{\beta r_i^n}} + \frac{e^{\beta r_i^x}}{\sum_{n=0}^S e^{\beta r_i^n}} - \frac{\frac{a}{S}}{(1-\frac{a}{S})}\frac{\sum_{l\neq 0,x}^S e^{\beta r_i^l}}{\sum_{n=0}^S e^{\beta r_i^n}}.$$

Inoltre per $\xi_i^\mu = S$

$$\sum_{n=0}^S e^{\beta r_i^n} = e^{\beta r_{ss}} + Se^{\beta r_{sk}} \tag{5.11}$$

e per $\xi_i^\mu \neq S$

$$\sum_{n=0}^S e^{\beta r_i^n} = e^{\beta r_{xs}} + e^{\beta r_{xx}} + (S-1)e^{\beta r_{xk}} \tag{5.12}$$

quindi:

$$\begin{aligned}\frac{dm_\mu(t)}{dt} &= -m_\mu(t) - \frac{(1-a)}{S(1-\frac{a}{S})}\frac{Se^{\beta r_{sk}}}{e^{\beta r_{ss}} + Se^{\beta r_{sk}}} \\ &+ \frac{e^{\beta r_{xx}}}{e^{\beta r_{xs}} + e^{\beta r_{xx}} + (S-1)e^{\beta r_{xk}}} \\ &- \frac{\frac{a}{S}}{(1-\frac{a}{S})}\frac{(S-1)e^{\beta r_{xk}}}{e^{\beta r_{xs}} + e^{\beta r_{xx}} + (S-1)e^{\beta r_{xk}}} \end{aligned} \tag{5.13}$$

Analogamente per q:

$$\begin{aligned}\frac{dq(t)}{dt} &= -q(t) + \frac{S(1-S)(1-a)}{a(1-\frac{a}{S})}\frac{e^{2\beta r_{sk}}}{(e^{\beta r_{ss}} + Se^{\beta r_{sk}})^2} \\ &+ \frac{1}{(1-\frac{a}{S})}\frac{2e^{2\beta r_{xx}} + (S-1)Se^{2\beta r_{xk}} + 2(S-1)e^{\beta r_{xx}}e^{\beta r_{xk}}}{(e^{\beta r_{xs}} + e^{\beta r_{xx}} + (S-1)e^{\beta r_{xk}})^2}. \end{aligned} \tag{5.14}$$

Per quanto riguarda le equazioni per i campi e le soglie efficaci l'unica informazione che ci serve sono i valori del campo h e dello stato σ mediati sul rumore, in relazione ai vari gruppi di interesse. Dall'equazione 5.9, considerando che la parte di rumore ha media nulla, si ottiene:

$$\begin{aligned} h_{ss} &= U \\ h_{sk} &= -m_{ret}\frac{a}{S} \\ h_{xs} &= U \\ h_{xx} &= -m_{ret}\frac{a}{S} \\ h_{xk} &= m_{ret}(1-\frac{a}{S}). \end{aligned}$$

Per σ invece abbiamo:

$$\begin{aligned}
\sigma_{ss} &= \frac{e^{\beta r_{ss}}}{e^{\beta r_{ss}} + S e^{\beta r_{sk}}} \\
\sigma_{sk} &= \frac{e^{\beta r_{sk}}}{e^{\beta r_{ss}} + S e^{\beta r_{sk}}} \\
\sigma_{xs} &= \frac{e^{\beta r_{xs}}}{e^{\beta r_{xs}} + e^{\beta r_{xx}} + (S-1)e^{\beta r_{xk}}} \\
\sigma_{xx} &= \frac{e^{\beta r_{xx}}}{e^{(\beta r_{xs}} + e^{\beta r_{xx}} + (S-1)e^{\beta r_{xk}}} \\
\sigma_{xk} &= \frac{e^{\beta r_{xk}}}{e^{\beta r_{xs}} + e^{\beta r_{xx}} + (S-1)e^{\beta r_{xk}}}.
\end{aligned}$$

Il sistema di equazioni infine risulta:

$$\begin{aligned}
\frac{dm_{ret}(t)}{dt} &= -m_{ret}(t) - \frac{(1-a)}{S(1-\frac{a}{S})} \frac{S e^{\beta r_{sk}}}{e^{\beta r_{ss}} + S\beta r_{sk})} \\
&\quad + \frac{e^{\beta r_{xx}}}{e^{\beta r_{xs}} + e^{\beta r_{xx}} + (S-1)e^{\beta r_{xk}}} \\
&\quad - \frac{\frac{a}{S}}{(1-\frac{a}{S})} \frac{(S-1)e^{\beta r_{xk}}}{e^{\beta r_{xs}} + e^{\beta r_{xx}} + (S-1)e^{\beta r_{xk}}} \\
\frac{dq(t)}{dt} &= -q(t) + \frac{S(1-S)(1-a)}{a(1-\frac{a}{S})} \frac{e^{2\beta r_{sk}}}{(e^{\beta r_{ss}} + S e^{\beta r_{sk}})^2} \\
&\quad + \frac{1}{(1-\frac{a}{S})} \frac{2e^{2\beta r_{xx}} + (S-1)S e^{2\beta r_{xk}} + 2(S-1)e^{\beta r_{xx}} e^{\beta r_{xk}}}{(e^{\beta r_{xs}} + e^{\beta r_{xx}} + (S-1)e^{\beta r_{xk}})^2} \\
\frac{dr_{ss}(t)}{dt} &= b_1(U - t_{ss} - r_{ss}) \\
\frac{dr_{sk}(t)}{dt} &= b_1(-\frac{a}{S} m_{ret} - t_{sk} - r_{sk}) \\
\frac{dr_{xs}(t)}{dt} &= b_1(U - t_{xs} - r_{xs}) \\
\frac{dr_{xx}(t)}{dt} &= b_1((1-\frac{a}{S}) m_{ret} - t_{xx} - r_{xx}) \\
\frac{dr_{xk}(t)}{dt} &= b_1(-\frac{a}{S} m_{ret} - t_{xk} - r_{xk})
\end{aligned}$$

$$
\begin{aligned}
\frac{dt_{ss}(t)}{dt} &= b_2\left(\frac{e^{\beta r_{ss}}}{e^{\beta r_{ss}} + Se^{\beta r_{sk}}} - t_{ss}\right) \\
\frac{dt_{sk}(t)}{dt} &= b_2\left(\frac{e^{\beta r_{sk}}}{e^{\beta r_{ss}} + Se^{\beta r_{sk}}} - t_{sk}\right) \\
\frac{dt_{xs}(t)}{dt} &= b_2\left(\frac{e^{\beta r_{xs}}}{e^{\beta r_{xs}} + e^{\beta r_{xx}} + (S-1)e^{\beta r_{xk}}} - t_{xs}\right) \\
\frac{dt_{xx}(t)}{dt} &= b_2\left(\frac{e^{\beta r_{xx}}}{e^{(\beta r_{xs}} + e^{\beta r_{xx}} + (S-1)e^{\beta r_{xk}}} - t_{xx}\right) \\
\frac{dt_{xk}(t)}{dt} &= b_2\left(\frac{e^{\beta r_{xk}}}{e^{\beta r_{xs}} + e^{\beta r_{xx}} + (S-1)e^{\beta r_{xk}}} - t_{xk}\right)
\end{aligned}
$$

5.2 Simulazione e integrazione numerica

In questo caso le equazioni differenziali sono state integrate utilizzando un metodo di Runge-Kutta del quinto ordine con una routine per il controllo del passo di integrazione (adaptive stepsize) (allegato A.4) [24]. La tabella di seguito riporta i valori utilizzati per le costanti:

N	p	S	a	U	$b1$	$b2$	β	g	τ	t_0	T
10000	5	3	0.1	1.5	0.01	0.01	5	6	$70 * N$	$200 * N$	$1000 * N$

Per quanto riguarda la dinamica di adaptation in figura 5.3 osserviamo un comportamento simile a quello visto nel precedente capitolo. Anche la rete di Potts ci permette dunque di avere il retrieval di un pattern e di imporre un processo di adaptation.

Vediamo però che, a differenza di quanto osservato per il modello precedente, nelle immagini in figura 5.4 si distinguono chiaramente 5 gruppi dinamici di unità. Come già discusso nel paragrafo 5.1.3 la formazione di tali gruppi è dovuta sia alla nuova presenza di più stati di attività sia alla casistica più ampia di configurazioni di pattern che questi comportano.

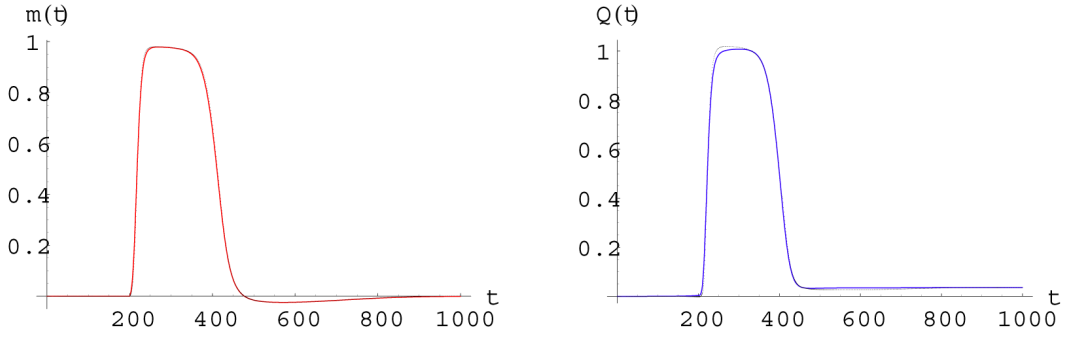

Figura 5.3: Andamento delle variabili $m(t)$ e $q(t)$ in una rete di Potts con adattamento, durante il retrieval di pattern. I valori risultanti dalla simulazione (allegato A.3) per le variabili m e q sono graficati rispettivamente in rosso e blu. In nero è mostrato il risultato analitico.

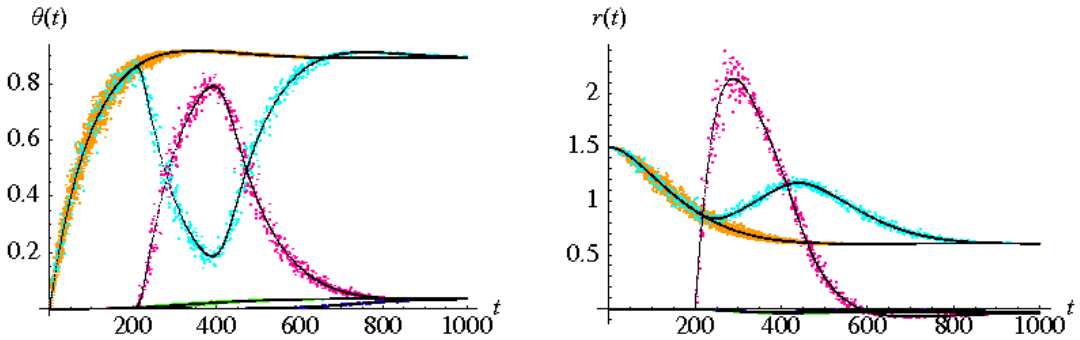

Figura 5.4: Andamento delle variabili $\theta(t)$ e $r(t)$. Abbiamo indicato i risultati della simulazione (allegato A.3) in: giallo il gruppo $\{ss\}$, in celeste il gruppo $\{sk\}$, in blu il gruppo $\{xs\}$, in rosa il gruppo $\{xx\}$ e in verde il gruppo $\{xk\}$. In nero i risultati analitici.

In figura 5.4 abbiamo graficato θ_i e r_i con riferimento alla suddivisione stabilita nel capitolo 5.1.3: in giallo per il gruppo $\{ss\}$, in celeste per il gruppo $\{sk\}$, in blu per il gruppo $\{xs\}$, in rosa per il gruppo $\{xx\}$ e in verde per il gruppo $\{xk\}$.
Il comportamento dei diversi gruppi di unità e il criterio stesso con cui queste si sono divise, segue perfettamente quanto aspettato e previsto dalle equazioni differenziali.

Capitolo 6

Latching o Libera Associazione

Siamo ora finalmente pronti per lavorare sul fenomeno di latching.
A livello intuitivo possiamo pensare che ciò che spinge il pensiero a passare da un concetto ad un'altro sia una sorta di legame esistente tra le due idee. Tale legame, del tutto soggettivo, si forma durante la vita dell'individuo a causa, ad esempio, di esperienze personali (memoria episodica) oppure come conseguenza di conoscenze comuni (memoria semantica).
L'esperienza vissuta dal soggetto al livello corticale porta alla formazione dei pattern. È allora verosimile pensare che il legame intuitivo che percepiamo tra due concetti si traduca, sul piano matematico del nostro network, come correlazione tra i patterns.
Proviamo allora a inserire nella nostra rete non più patterns costruiti in modo random, con unico vincolo quello della sparcity, ma patterns correlati.

6.1 Patterns correlati

Cercando di metterci nella condizione più semplice possibile, costruiamo un set di patterns in cui solo due risultino correlati mentre gli altri $p-2$ siano costruiti in modo random.

Un pattern è un vettore di lunghezza N costruito in modo che solo Na unità siano attive. In un generico pattern tali unità sono distribuite in maniera casuale

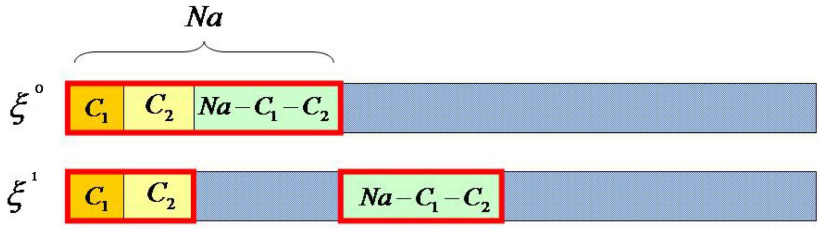

Figura 6.1: Correlazione tra due pattern. Delle Na unità attive del pattern ξ^0: C_1 sono nello stesso stato delle rispettive del pattern ξ^1; C_2 sono attive come le rispettive in ξ^1, ma in uno stato diverso; le restanti $Na - C_1 - C_2$ sono attive a differenza delle rispettive in ξ^1.

tra tutte le N unità. Un modo semplice per formare due pattern correlati è quello di porre le unità attive in modo da creare una corrispondenza tra i due vettori.

Costruiamo allora i due patterns, ad esempio indicati rispettivamente dall'indice $\mu = 0$ e $\mu = 1$, in modo che: C_1 unità, distribuite casualmente tra le N, assumano lo stesso stato attivo sia nel pattern 0 che nell'1; C_2 unità assumano uno stato attivo in entrambi i patterns ma in due stati diversi e, in fine, $Na - C_1 - C_2$ unità siano attive nel pattern 0 ma inattive nel pattern 1 (delle restanti $N(1-a)$ unità ve ne saranno $Na - C_1 - C_2$ attive nel pattern 1 ma non nello 0 e le restanti $N(1-2a) + C_1 + C_2$ saranno inattive in entrambi i patterns) (fig. 6.1).

Con questo nuovo set di patterns ritorniamo al nostro modello di rete.

6.1.1 Piccole correzioni e nuovi gruppi

Per studiare la dinamica di latching occorre che il pattern di cui viene fatto il retrieval sia uno dei due correlati, poniamo che sia il $\mu = 0$. Ci aspettiamo che, inserendo questa unica modifica, per almeno qualche regione dei parametri (dove con parametri intendiamo, ad esempio, a, p, S, C_1...) il retrieval del pattern 0 sia seguito da uno spontaneo retrieval del pattern 1.
Nonostante non occorra introdurre nessuna modifica sul piano dinamico per avere o meno latching, inseriamo, in questo che sarà il nostro modello conclusivo, alcuni

accorgimenti.

Innanzitutto, essendo l'inattività uno stato 'particolare' aggiungiamo una nuova costante tempo $\mathbf{b_3}$ che ci permetta di regolarne separatamente la dinamica. Le nuove equazioni per l'adattamento sono dunque:

$$\begin{aligned} r_i^k(t+1) &= r_i^k(t) + b_1[h_i^k(t) - \theta_i^k(t) - r_i^k(t)] \\ \theta_i^k(t+1) &= \theta_i^k(t) + b_2[s_i^k(t) - \theta_i^k(t)] \\ r_i^0(t+1) &= r_i^0(t) + b_3[U + 1 - s_i^0(t) - r_i^0(t)] \end{aligned}$$

Introduciamo poi nel campo un parametro di **auto-eccitazione** (self-excitation) che dia più stabilità ai patterns richiamati. Tale parametro, regolato dalla costante w, nel momento in cui il sistema esce dall'attrattore del primo pattern rende più veloce e netta la caduta, nel caso questa avvenga, nell'attrattore del secondo. Nello specifico, il termine di self-excitation è costituito da un termine positivo inserito nel campo che favorisce l'attivazione degli stati attivi in proporzione al loro stato di attività

$$+w\sigma_i^k(1 - \delta_{k0}) \tag{6.1}$$

e un termine negativo che per gli stessi stati ne diminuisca l'attivazione in base all'attivazione totale della rete

$$-w\frac{\sum_{l\neq S}^{S}\sigma_i^l}{S}. \tag{6.2}$$

Quest'ultimo parametro serve ad evitare un'iperattivazione ingiustificata della rete. La forma del campo è dunque:

$$h_i^k = \sum_{\mu=0}^{p} m_\mu(\delta_{\xi_i^\mu k} - \frac{a}{S}) + w\left(\sigma_i^k - \frac{\sum_{l\neq S}^{S}\sigma_i^l}{S}\right) \tag{6.3}$$

Oltre all'introduzione di queste piccole modifiche dobbiamo occuparci di correggere la trattazione analitica già imbastita, tenendo in considerazione che al retrieval del primo pattern potrebbe seguire quello del secondo.

Nell'analisi signal-to-noise, infatti, mentre in assenza di latching gli $m_\mu \neq m_{ret=0}$ sono infinitesimali, in caso di latching il contributo di m_1 non è più trascurabile. È dunque necessario estrarre dalla parte di rumore anche m_1.
Utilizziamo allora come campo:

$$h_i^k = m_0(\delta_{\xi_i^0 k} - \frac{a}{S}) + m_1(\delta_{\xi_i^1 k} - \frac{a}{S}) + \eta\sqrt{\alpha\frac{a}{S^2}q} + w\left(\sigma_i^k - \frac{\sum_{l\neq S}^{S}\sigma_i^l}{S}\right). \quad (6.4)$$

Infine, come ultima modifica, occorre rivalutare i raggruppamenti fatti tra le unità nell'elaborazione delle equazioni differenziali. La distinzione fatta in precedenza, basata sullo stato che l'unità assumeva nella configurazione di pattern, in caso di latching deve tener conto anche dello stato assunto nella configurazione del secondo pattern. Inserendo anche questa nuova variabile il numero dei raggruppamenti aumenta vertiginosamente: arriviamo infatti a 15 gruppi e 28 equazioni differenziali.

Cerchiamo dunque di spiegare nel modo più chiaro possibile la separazione dei nuovi gruppi. In caso di latching si possono avere le seguenti eventualità:

- **macrogruppo xs** pattern 0 attivo nello stato x e pattern 1 inattivo;
- **macrogruppo xx** pattern 0 attivo nello stato x e pattern 1 attivo nello stesso stato;
- **macrogruppo xy** pattern 0 attivo nello stato x e pattern 1 attivo ma nello stato y distinto;
- **macrogruppo ss** pattern 0 e pattern 1 entrambi inattivi;
- **macrogruppo sy** pattern 0 inattivo e pattern 1 attivo nello stato y.

All'interno di tali macrogruppi dobbiamo distinguere ulteriori raggruppamenti legati allo stato che andiamo ad osservare nell'unità:

- **nel macrogruppo xs** distinguo tra lo stato inattivo $\{xss\}$, l'attivo x $\{xsx\}$ e l'attivo diverso da x $\{xsk\}$;

- **nel macrogruppo xx** distinguo tra lo stato inattivo $\{xxs\}$, l'attivo x $\{xxx\}$ e l'attivo diverso da x $\{xxk\}$;

- **nel macrogruppo xy** distinguo tra lo stato inattivo $\{xys\}$, l'attivo x $\{xyx\}$, l'attivo y $\{xyy\}$ e l'attivo diverso da x e da y $\{xyk\}$;

- **nel macrogruppo ss** distinguo tra lo stato inattivo $\{sss\}$ e l'attivo $\{ssx\}$;

- **nel macrogruppo sy** distinguo tra lo stato inattivo $\{sys\}$, l'attivo y $\{syy\}$ e l'attivo diverso da y $\{syk\}$;

A questo punto abbiamo tutti gli strumenti necessari per ricavare il nostro sistema di equazioni differenziali.

6.1.2 Le equazioni

Innanzitutto esplicitiamo la variabile stato:

$$
\begin{aligned}
\sigma_{xsx} &= \frac{e^{\beta r_{xsx}}}{e^{\beta(r_{xss}+U)} + e^{\beta r_{xsx}} + (S-1)e^{\beta r_{xsk}}} \\
\sigma_{xsk} &= \frac{e^{\beta r_{xsk}}}{e^{\beta(r_{xss}+U)} + e^{\beta r_{xsx}} + (S-1)e^{\beta r_{xsk}}} \\
\sigma_{xss} &= \frac{e^{\beta r_{xss}+U}}{e^{\beta(r_{xss}+U)} + e^{\beta r_{xsx}} + (S-1)e^{\beta r_{xsk}}} \\
\sigma_{xxx} &= \frac{e^{\beta r_{xxx}}}{e^{\beta(r_{xxs}+U)} + e^{\beta r_{xxx}} + (S-1)e^{\beta r_{xxk}}} \\
\sigma_{xxk} &= \frac{e^{\beta r_{xxk}}}{e^{\beta(r_{xxs}+U)} + e^{\beta r_{xxx}} + (S-1)e^{\beta r_{xxk}}} \\
\sigma_{xxs} &= \frac{e^{\beta r_{xxs}+U}}{e^{\beta(r_{xxs}+U)} + e^{\beta r_{xxx}} + (S-1)e^{\beta r_{xxk}}}
\end{aligned}
$$

$$
\begin{aligned}
\sigma_{xyx} &= \frac{e^{\beta r_{xyx}}}{e^{\beta(r_{xys}+U)} + e^{\beta r_{xyx}} + e^{\beta r_{xyy}} + (S-2)e^{\beta r_{xyk}}} \\
\sigma_{xyy} &= \frac{e^{\beta r_{xyy}}}{e^{\beta(r_{xys}+U)} + e^{\beta r_{xyx}} + e^{\beta r_{xyy}} + (S-2)e^{\beta r_{xyk}}} \\
\sigma_{xyk} &= \frac{e^{\beta r_{xyk}}}{e^{\beta(r_{xys}+U)} + e^{\beta r_{xyx}} + e^{\beta r_{xyy}} + (S-2)e^{\beta r_{xyk}}} \\
\sigma_{xys} &= \frac{e^{\beta r_{xys}+U}}{e^{\beta(r_{xys}+U)} + e^{\beta r_{xyx}} + e^{\beta r_{xyy}} + (S-2)e^{\beta r_{xyk}}} \\
\sigma_{syy} &= \frac{e^{\beta r_{syy}}}{e^{\beta(r_{sys}+U)} + e^{\beta r_{syy}} + (S-1)e^{\beta r_{syk}}} \\
\sigma_{syk} &= \frac{e^{\beta r_{syk}}}{e^{\beta(r_{sys}+U)} + e^{\beta r_{syy}} + (S-1)e^{\beta r_{syk}}} \\
\sigma_{sys} &= \frac{e^{\beta r_{sys}+U}}{e^{\beta(r_{sys}+U)} + e^{\beta r_{syy}} + (S-1)e^{\beta r_{syk}}} \\
\sigma_{ssk} &= \frac{e^{\beta r_{ssk}}}{e^{\beta(r_{sss}+U)} + Se^{\beta r_{ssk}}} \\
\sigma_{sss} &= \frac{e^{\beta r_{sss}+U}}{e^{\beta(r_{sss}+U)} + Se^{\beta r_{ssk}}}
\end{aligned}
$$

Da queste equazioni possiamo poi ricavare il termine di self-excitation

$$
self = w\left(\frac{Se^{\beta r_i^k}}{\sum_l^S e^{\beta(r_i^l)}} - \frac{1}{S}\frac{\sum_{l\neq S}^S e^{\beta(r_i^l)}}{\sum_l^S e^{\beta(r_i^l)}}\right) \tag{6.5}
$$

e il campo

$$
\begin{aligned}
h_{xsx} = (1-\frac{a}{S})m_1 - \frac{a}{S}m_2 &+ w\frac{e^{\beta r_{xsx}}}{e^{\beta(r_{xss}+U)} + e^{\beta r_{xsx}} + (S-1)e^{\beta r_{xsk}}} \\
&- \frac{w}{S}\frac{e^{\beta r_{xsx}} + (S-1)e^{\beta r_{xsk}}}{e^{\beta(r_{xss}+U)} + e^{\beta r_{xsx}} + (S-1)e^{\beta r_{xsk}}} \\
h_{xsk} = -\frac{a}{S}(m_1+m_2) &+ w\frac{e^{\beta r_{xsk}}}{e^{\beta(r_{xss}+U)} + e^{\beta r_{xsx}} + (S-1)e^{\beta r_{xsk}}} \\
&- \frac{w}{S}\frac{e^{\beta r_{xsx}} + (S-1)e^{\beta r_{xsk}}}{e^{\beta(r_{xss}+U)} + e^{\beta r_{xsx}} + (S-1)e^{\beta r_{xsk}}}
\end{aligned}
$$

$$h_{xxx} = (1-\frac{a}{S})(m_1+m_2) + w\frac{e^{\beta r_{xxx}}}{e^{\beta(r_{xxs}+U)}+e^{\beta r_{xxx}}+(S-1)e^{\beta r_{xxk}}} - \frac{w}{S}\frac{e^{\beta r_{xxx}}+(S-1)e^{\beta r_{xxk}}}{e^{\beta(r_{xxs}+U)}+e^{\beta r_{xxx}}+(S-1)e^{\beta r_{xxk}}}$$

$$h_{xxk} = -\frac{a}{S}(m_1+m_2) + w\frac{e^{\beta r_{xxk}}}{e^{\beta(r_{xxs}+U)}+e^{\beta r_{xxx}}+(S-1)e^{\beta r_{xxk}}} - \frac{w}{S}\frac{e^{\beta r_{xxx}}+(S-1)e^{\beta r_{xxk}}}{e^{\beta(r_{xxs}+U)}+e^{\beta r_{xxx}}+(S-1)e^{\beta r_{xxk}}}$$

$$h_{xyx} = (1-\frac{a}{S})m_1 - \frac{a}{S}m_2 + w\frac{e^{\beta r_{xyx}}}{e^{\beta(r_{xys}+U)}+e^{\beta r_{xyx}}+e^{\beta r_{xyy}}+(S-2)e^{\beta r_{xyk}}} - \frac{w}{S}\frac{e^{\beta r_{xyx}}+e^{\beta r_{xyy}}+(S-2)e^{\beta r_{xyk}}}{e^{\beta(r_{xys}+U)}+e^{\beta r_{xyx}}+e^{\beta r_{xyy}}+(S-2)e^{\beta r_{xyk}}}$$

$$h_{xyy} = -\frac{a}{S}m_1 + (1-\frac{a}{S})m_2 + w\frac{e^{\beta r_{xyy}}}{e^{\beta(r_{xys}+U)}+e^{\beta r_{xyx}}+e^{\beta r_{xyy}}+(S-2)e^{\beta r_{xyk}}} - \frac{w}{S}\frac{e^{\beta r_{xyx}}+e^{\beta r_{xyy}}+(S-2)e^{\beta r_{xyk}}}{e^{\beta(r_{xys}+U)}+e^{\beta r_{xyx}}+e^{\beta r_{xyy}}+(S-2)e^{\beta r_{xyk}}}$$

$$h_{xyk} = -\frac{a}{S}(m_1+m_2) + w\frac{e^{\beta r_{xyk}}}{e^{\beta(r_{xys}+U)}+e^{\beta r_{xyx}}+e^{\beta r_{xyy}}+(S-2)e^{\beta r_{xyk}}} - \frac{w}{S}\frac{e^{\beta r_{xyx}}+e^{\beta r_{xyy}}+(S-2)e^{\beta r_{xyk}}}{e^{\beta(r_{xys}+U)}+e^{\beta r_{xyx}}+e^{\beta r_{xyy}}+(S-2)e^{\beta r_{xyk}}}$$

$$h_{syy} = -\frac{a}{S}m_1 + (1-\frac{a}{S})m_2 + w\frac{e^{\beta r_{syy}}}{e^{\beta(r_{sys}+U)}+e^{\beta r_{syy}}+(S-1)e^{\beta r_{syk}}} - \frac{w}{S}\frac{e^{\beta r_{syy}}+(S-1)e^{\beta r_{syk}}}{e^{\beta(r_{sys}+U)}+e^{\beta r_{syy}}+(S-1)e^{\beta r_{syk}}}$$

$$h_{syk} = -\frac{a}{S}(m_1+m_2) + w\frac{e^{\beta r_{syk}}}{e^{\beta(r_{sys}+U)}+e^{\beta r_{syy}}+(S-1)e^{\beta r_{syk}}} - \frac{w}{S}\frac{e^{\beta r_{syy}}+(S-1)e^{\beta r_{syk}}}{e^{\beta(r_{sys}+U)}+e^{\beta r_{syy}}+(S-1)e^{\beta r_{syk}}}$$

$$h_{ssk} = -\frac{a}{S}(m_1+m_2) + w\frac{e^{\beta r_{ssk}}}{e^{\beta(r_{sss}+U)}+Se^{\beta r_{ssk}}} - \frac{w}{S}\frac{Se^{\beta r_{ssk}}}{e^{\beta(r_{sss}+U)}+Se^{\beta r_{ssk}}}.$$

Essendo

$$\frac{dm_\mu}{dt} = -m_\mu + \frac{(1-\delta_{ij})}{Na(1-\frac{a}{S})}\sum_i^N\sum_{l\neq 0}^S(\delta_{\xi_i^\mu l}-\frac{a}{S})\sigma_i^l \tag{6.6}$$

ricaviamo la forma completa del sistema finale:

$$
\begin{aligned}
\frac{dm_1}{dt} &= -m_1 + \frac{1}{Na(1-\frac{a}{S})}[C1\left((1-\frac{a}{S})\sigma_{xxx} - \frac{a}{S}(S-1)\sigma_{xxk}\right) + \\
&\quad C2\left((1-\frac{a}{S})\sigma_{xyx} - \frac{a}{S}\sigma_{xyy} - \frac{a}{S}(S-2)\sigma_{xyk}\right) + \\
&\quad (Na - C1 - C2)\left(-\frac{a}{S}\sigma_{syy} - \frac{a}{S}(S-1)\sigma_{syk}\right) + \\
&\quad (Na - C1 - C2)\left((1-\frac{a}{S})\sigma_{xsx} - \frac{a}{S}(S-1)\sigma_{xsk}\right) + \\
&\quad (N(1-2a) + C1 + C2)\,(-a\sigma_{ssk})] \\
\\
\frac{dm_2}{dt} &= -m_2 + \frac{1}{Na(1-\frac{a}{S})}[C1\left((1-\frac{a}{S})\sigma_{xxx} - \frac{a}{S}(S-1)\sigma_{xxk}\right) + \\
&\quad C2\left(\frac{a}{S}\sigma_{xyx} + (1-\frac{a}{S})\sigma_{xyy} - \frac{a}{S}(S-2)\sigma_{xyk}\right) + \\
&\quad (Na - C1 - C2)\left((1-\frac{a}{S})\sigma_{syy} - \frac{a}{S}(S-1)\sigma_{syk}\right) + \\
&\quad (Na - C1 - C2)\left(-\frac{a}{S}\sigma_{xsx} - \frac{a}{S}(S-1)\sigma_{xsk}\right) + \\
&\quad (N(1-2a) + C1 + C2)\,(-a\sigma_{ssk})] \\
\\
\frac{dq}{dt} &= -q + \frac{1}{Na(1-\frac{a}{S})}[C1\left({\sigma_{xxx}}^2 + (S-1)\sigma_{xxk} - \frac{a}{S}(1-\sigma_{xxs})^2\right) + \\
&\quad C2\left({\sigma_{xyx}}^2 + {\sigma_{xyy}}^2 + (S-2){\sigma_{xyk}}^2 - \frac{a}{S}(1-\sigma_{xys})^2\right) + \\
&\quad (Na - C1 - C2)\left({\sigma_{xsx}}^2 + (S-1){\sigma_{xyk}}^2 - \frac{a}{S}(1-\sigma_{xys})^2\right) + \\
&\quad (Na - C1 - C2)\left({\sigma_{syy}}^2 + (S-1){\sigma_{syk}}^2 - \frac{a}{S}(1-\sigma_{sys})^2\right) + \\
&\quad (N(1-2a) + C1 + C2)\left(S{\sigma_{ssk}}^2 - \frac{a}{S}(1-\sigma_{sss})^2\right)] \\
\frac{dr_{xsx}}{dt} &= b_1\,(h_{xsx} - t_{xsx} - r_{xsx}) \\
\frac{dr_{xsk}}{dt} &= b_1\,(h_{xsk} - t_{xsk} - r_{xsk}) \\
\frac{dr_{xss}}{dt} &= b_3\,(1 - \sigma_{xss} - r_{xss}s)
\end{aligned}
$$

$$
\begin{aligned}
\frac{dr_{xxx}}{dt} &= b_1 \left(h_{xxx} - t_{xxx} - r_{xxx}\right) \\
\frac{dr_{xxk}}{dt} &= b_1 \left(h_{xxk} - t_{xxk} - r_{xxk}\right) \\
\frac{dr_{xxs}}{dt} &= b_3 \left(1 - \sigma_{xxs} - r_{xxs}\right) \\
\frac{dr_{xyx}}{dt} &= b_1 \left(h_{xyx} - t_{xyx} - r_{xyx}\right) \\
\frac{dr_{xyy}}{dt} &= b_1 \left(h_{xyy} - t_{xyy} - r_{xyy}\right) \\
\frac{dr_{xyk}}{dt} &= b_1 \left(h_{xyk} - t_{xyk} - r_{xyk}\right) \\
\frac{dr_{xys}}{dt} &= b_3 \left(1 - \sigma_{xyx} - r_{xys}\right) \\
\frac{dr_{syy}}{dt} &= b_1 \left(h_{syy} - t_{syy} - r_{syy}\right) \\
\frac{dr_{syk}}{dt} &= b_1 \left(h_{syk} - t_{syk} - r_{syk}\right) \\
\frac{dr_{sys}}{dt} &= b_3 \left(1 - \sigma_{sys} - r_{sys}\right) \\
\frac{dr_{ssk}}{dt} &= b_1 \left(h_{ssk} - t_{ssk} - r_{ssk}\right) \\
\frac{dr_{sss}}{dt} &= b_3 \left(1 - \sigma_{sss} - r_{sss}\right) \\
\frac{dt_{xsx}}{dt} &= b_2 \left(\sigma_{xsx} - t_{xsx}\right) \\
\frac{dt_{xsk}}{dt} &= b_2 \left(\sigma_{xsk} - t_{xsk}\right) \\
\frac{dt_{xxx}}{dt} &= b_2 \left(\sigma_{xxx} - t_{xxx}\right) \\
\frac{dt_{xxk}}{dt} &= b_2 \left(\sigma_{xxk} - t_{xxk}\right) \\
\frac{dt_{xyx}}{dt} &= b_2 \left(\sigma_{xyx} - t_{xyx}\right) \\
\frac{dt_{xyy}}{dt} &= b_2 \left(\sigma_{xyy} - t_{xyy}\right) \\
\frac{dt_{xyk}}{dt} &= b_2 \left(\sigma_{xyk} - t_{xyk}\right)
\end{aligned}
$$

$$\begin{aligned}
\frac{dt_{syy}}{dt} &= b_2\left(\sigma_{syy} - t_{syy}\right) \\
\frac{dt_{syk}}{dt} &= b_2\left(\sigma_{syk} - t_{syk}\right) \\
\frac{dt_{ssk}}{dt} &= b_2\left(\sigma_{ssk} - t_{ssk}\right)
\end{aligned}$$

6.2 Simulazione e integrazione numerica

Anche in quest'ultimo caso le equazioni differenziali sono state integrate numericamente utilizzando un metodo di Runge-Kutta del quinto ordine con una routine per il controllo del passo di integrazione (adaptive stepsize) (Appendice A.6). In Appendice A.5 è riportato il testo della simulazione per rete di Potts con adattamento e latching.

Anche se non è stata effettuata un'analisi sistematica di tutto lo spazio dei parametri ci si aspetta che la regione dei parametri che consente latching sia minore di quella per la quale si ha retrieval [28]. Lo studio di tale regione è difficoltoso in quanto l'analisi teorica portata avanti fino a questo momento non fornisce delle prescrizioni specifiche a tale riguardo. Per il momento è stata riscontrata l'esistenza di tre tipi di latching in tre regioni differenti. Tali tipologie sembrano inoltre essere le uniche possibili [28].

In caso di latching, quello che si osserva è che al momento in cui lo stato del sistema inizia ad allontanarsi dalla configurazione del pattern di cui si era effettuato il retrieval, le poche unità ancora in overlap col primo pattern fanno da chiave per il retrieval del secondo. Questo avviene a causa della correlazione imposta tra i due attrattori e dell'adattamento delle unità che impedisce una seconda attivazione del primo pattern. Distinguiamo dunque i tre tipi di latchig in base all'overlap che il sistema ha con il primo pattern al momento del retrieval del secondo:

- latching patologico
- latching alto

- latching basso

Nel latching patologico si ha una serie infinita di retrieval alternati del primo e secondo pattern. Un tale fenomeno è, dal punto di vista cognitivo, poco chiaro e quindi lo tralasciamo.
Discutiamo, invece, più in dettaglio le altre due tipologie di latching.

6.3 Latching Alto

In figura 6.2 possiamo osservare un esempio di latching alto.
Questo tipo di latching si osserva per valori di $C_1 \ll C_2$, ovvero tra pattern che hanno molte unità attive nello stesso stato. Per questo motivo non c'è necessità che la rete si allontani troppo dal primo pattern per cadere nel bacino di attrazione del secondo. Questo spiega un alto valore delle variabili m_0 e m_1 al momento del latching.

Le immagini 6.3, 6.4, 6.5, 6.6 mostrano rispettivamente l'attivazione della rete, la soglia efficace, il campo sentito negli stati attivi e il campo sentito negli stati inattivi per valori dei parametri:

N	p	S	a	U	$b1$	$b2$	$b3$	C_1	C_2
10000	2	3	0.25	0.1	0.005	0.001	0.0001	$a*N*0.19$	$a*N*0.01$

w	β	g	τ	t_0	T
0.8	5	3	$70*N$	$500*N$	$6000*N$

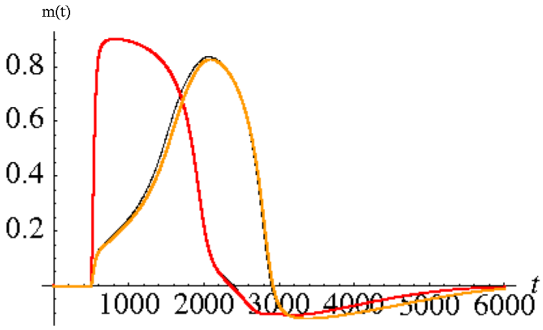

Figura 6.2: Andamento dell'overlap col pattern 0 in rosso e col pattern 1 in giallo. In nero il risultato dell'integrazione numerica per le variabili m_0 e m_1.

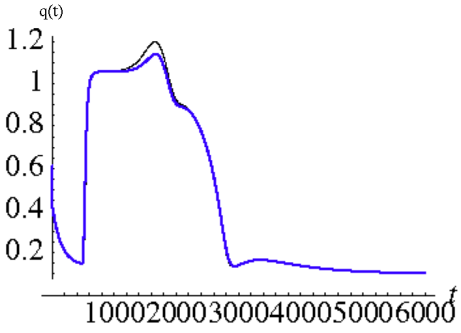

Figura 6.3: Andamento di $q(t)$ in blu come risultato della simulazione e in nero come risultato dell'integrazione numerica.

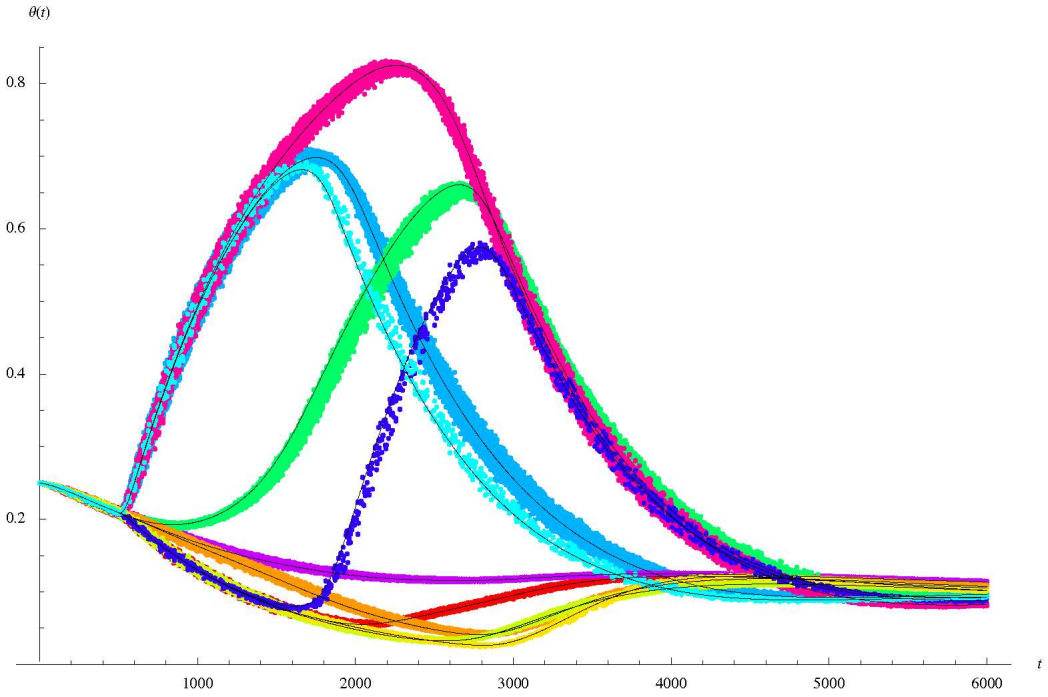

Figura 6.4: Soglia efficacie. In nero il risultato dell'integrazione numerica per tutti i gruppi dinamici. I dati ricavati dall'integrazione numerica sono invece in: azzurro per t_{xsx}, rosso per t_{xsk}, ciclamino per t_{xxx}, verde chiaro per t_{xxk}, celeste per t_{xyx}, blu per t_{xyy}, giallo per t_{xyk}, viola per t_{ssk}, verde per t_{syy} e arancione per t_{syk}.

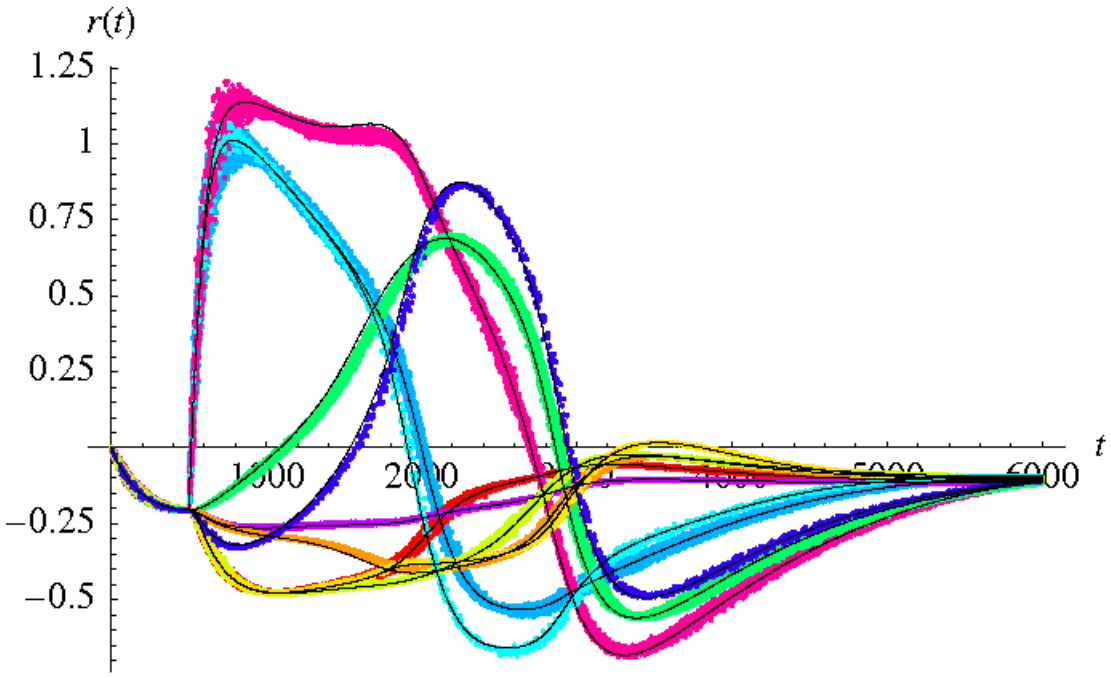

Figura 6.5: Campo sentito dagli stati attivi. In nero il risultato dell'integrazione numerica per tutti i gruppi dinamici. I dati ricavati dall'integrazione numerica sono invece in: azzurro per r_{xsx}, rosso per r_{xsk}, ciclamino per r_{xxx}, verde chiaro per r_{xxk}, celeste per r_{xyx}, blu per r_{xyy}, giallo per r_{xyk}, viola per r_{ssk}, verde per r_{syy} e arancione per r_{syk}.

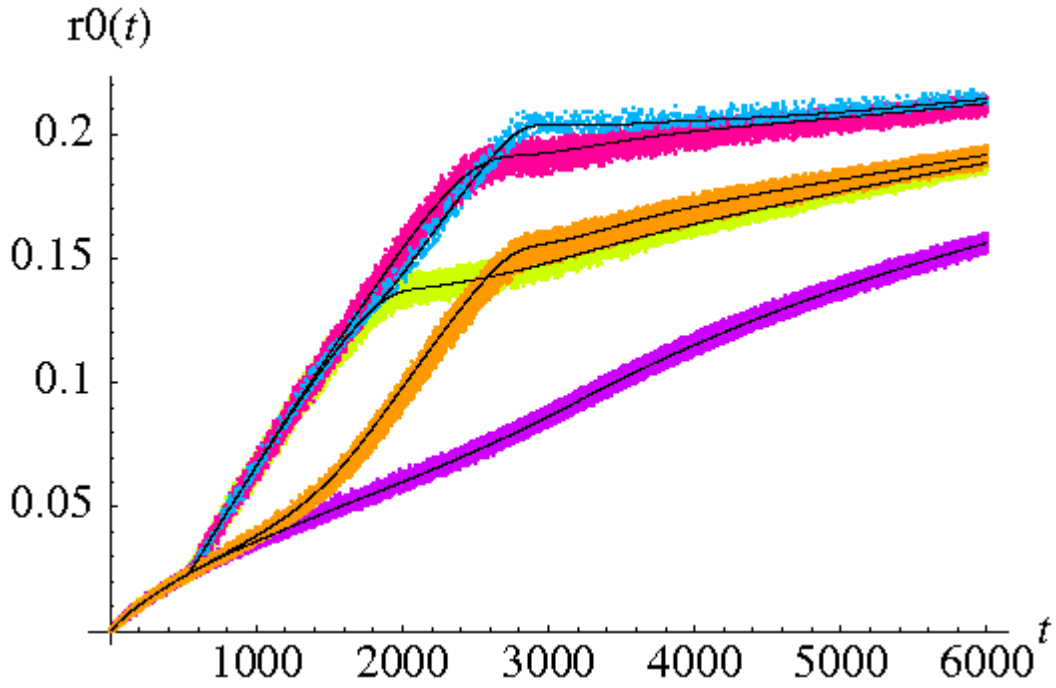

Figura 6.6: Campo sentito dagli stati inattivi. In nero il risultato dell'integrazione numerica per tutti i gruppi dinamici. I dati ricavati dall'integrazione numerica sono invece in: verde per r_{xss}, ciclamino per r_{xxs}, celeste per r_{xys}, viola per r_{sss} e arancione per r_{sys}.

6.4 Latching Basso

La figura 6.7 mostra un esempio di latching basso.
In questo tipo di latching i valori dei parametri C_1 e C_2 sono simili e molto bassi ($C_1 \sim C_2 \ll Na$). I due pattern risultano così quasi esclusivi poiché quasi tutte le unità attive del primo corrispondono ad unità inattive del secondo e viceversa. Al momento in cui, per l'adattamento, la rete esce dal bacino del primo pattern l'overlap col secondo è quasi nullo. L'anticorrelazione che c'è tra i due attrattori permette comunque al primo pattern di essere chiave per il retrieval del secondo. I valori dei parametri relativi alle figure 6.7, 6.8, 6.9, 6.10 e 6.11 sono:

N	p	S	a	U	$b1$	$b2$	$b3$	C_1	C_2
10000	2	3	0.25	0.1	0.05	0.001	0.0005	$a*N*0.01$	$a*N*0.01$

w	β	g	τ	t_0	T
0.8	5	3	$70*N$	$500*N$	$6000*N$

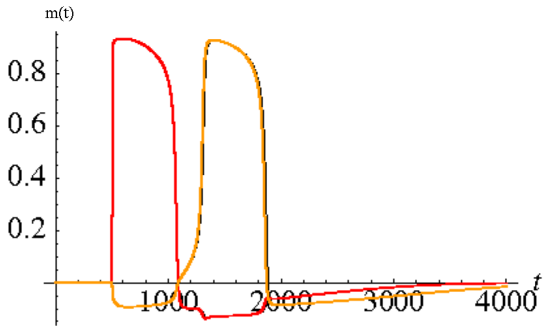

Figura 6.7: Andamento dell'overlap col pattern 0 in rosso e col pattern 1 in giallo. In nero il risultato dell'integrazione numerica per le variabili m_0 e m_1.

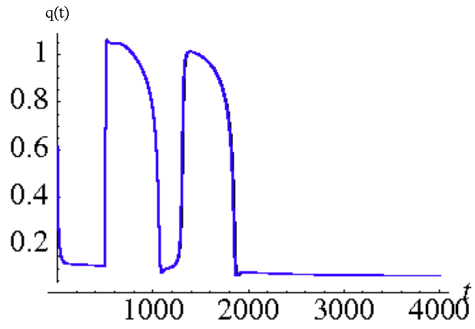

Figura 6.8: Andamento di $q(t)$ in blu come risultato della simulazione e in nero come risultato dell'integrazione numerica.

Le figure 6.9, 6.10 e 6.11 mostrano l'andamento della soglia e del campo all'evolvere del tempo.

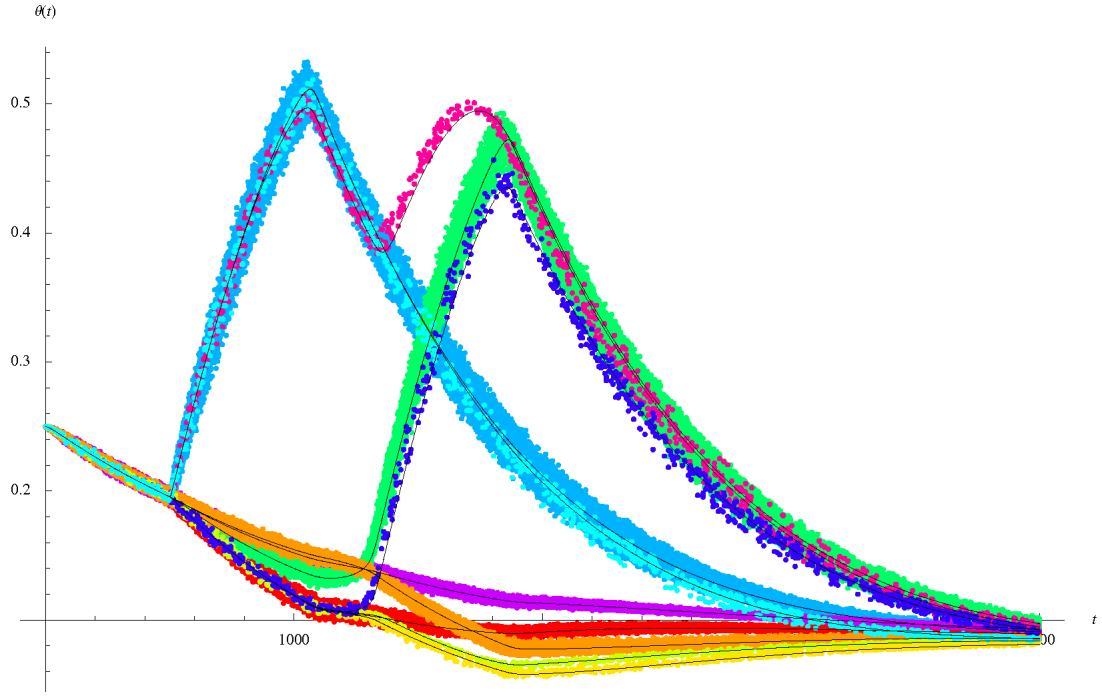

Figura 6.9: Soglia efficacie. In nero il risultato dell'integrazione numerica per tutti i gruppi dinamici. I dati ricavati dall'integrazione numerica sono invece in: azzurro per t_{xsx}, rosso per t_{xsk}, ciclamino per t_{xxx}, verde chiaro per t_{xxk}, celeste per t_{xyx}, blu per t_{xyy}, giallo per t_{xyk}, viola per t_{ssk}, verde per t_{syy} e arancione per t_{syk}.

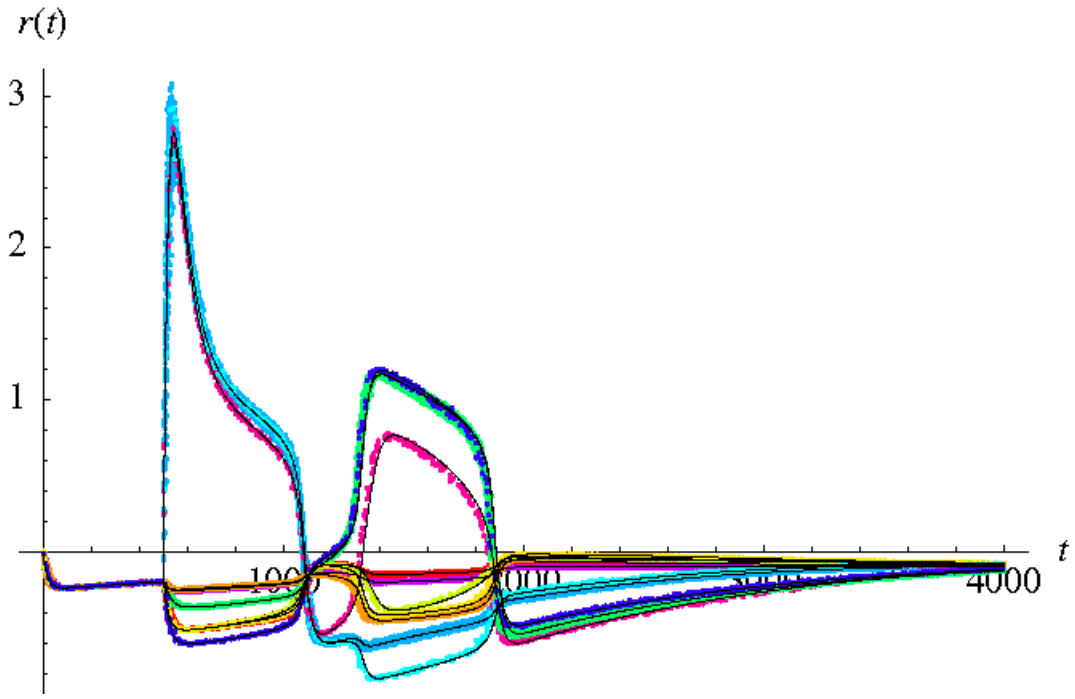

Figura 6.10: Campo sentito dagli stati attivi. In nero il risultato dell'integrazione numerica per tutti i gruppi dinamici. I dati ricavati dall'integrazione numerica sono invece in: azzurro per r_{xsx}, rosso per r_{xsk}, ciclamino per r_{xxx}, verde chiaro per r_{xxk}, celeste per r_{xyx}, blu per r_{xyy}, giallo per r_{xyk}, viola per r_{ssk}, verde per r_{syy} e arancione per r_{syk}.

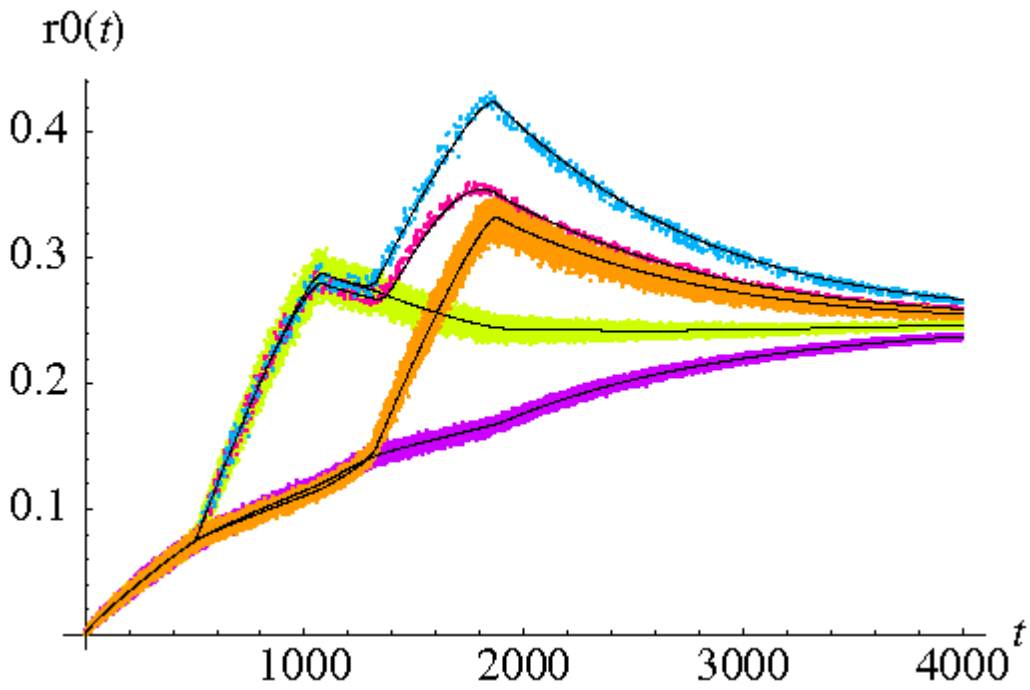

Figura 6.11: Campo sentito dagli stati inattivi. In nero il risultato dell'integrazione numerica per tutti i gruppi dinamici. I dati ricavati dall'integrazione numerica sono invece in: verde per r_{xss}, ciclamino per r_{xxs}, celeste per r_{xys}, viola per r_{sss} e arancione per r_{sys}.

6.4.1 Nota

I grafici relativi al latching alto e al latching basso sono stati prodotti utilizzando una rete con due soli patterns (correlati). Si potrebbe allora dubitare che tale latching sia dovuto alla correlazione dei patterns e pensare che sia ascrivibile più ad una generale 'facilità' della rete di passare in modo casuale da un attrattore all'altro. Per testare questa possibilità è stata modificata la simulazione aggiungendo altri pattern scorrelati. È stato osservato che il latching, comunque presente, rimane dei tre tipi sopra descritti ed avviene comunque solo tra i due patterns correlati.

Capitolo 7

Conclusioni

In questo elaborato siamo riusciti a produrre una rete di unità di Potts capace di immagazzinare particolari configurazioni e recuperarle al momento della somministrazione di un adeguato stimolo.
Siamo partiti dal più generale modello di Hopfield (capitolo 2), per lo studio di una rete associativa, e abbiamo introdotto una dinamica di adattamento (capitolo 4), propria delle cellule del sistema nervoso. Abbiamo poi generalizzato tale modello, derivato da quello di Ising, in funzione delle perculiarità anatomiche della corteccia cerebrale introducendo le unità di Potts (capitolo 5). In ultimo, al fine di modellizzare la dinamica di libera associazione, abbiamo creato delle configurazioni della rete correlate (capitolo 6), a mimare la relazione tra due concetti in un processo cognitivo.
Questa impostazione ha portato, come voluto, al manifestarsi di un processo di latching, ovvero di passaggio da una configurazione memorizzata ad un'altra ad essa correlata. In particolare si sono osservate tre distinte tipologie di latching per tre regioni dello spazio dei parametri.
È stata, inoltre, svolta una trattazione analitica che ci ha permesso di ricavare un sistema di equazioni differenziali, risolubile numericamente, relativo a tutte le variabili dinamiche del sistema.
Le soluzioni di tale sistema sono risultate in perfetto accordo con i dati ricavati parallelamente per mezzo di una simulazione di evoluzione della rete.

Se lo studio di una memoria associativa, anche attraverso l'utilizzo di reti di Potts, si inserisce in un ampio ambito di ricerca sui modelli di reti di memoria, l'introduzione e lo studio della dinamica di latching è un elemento innovativo nel settore. In particolare, il nostro piccolo modello vuole riprodurre, nel suo complesso, un'unità dinamica di base del processo di associazione spontanea di idee e si inserisce all'interno di un progetto più grande che spera di poter legare il meccanismo di latching alle dinamiche del linguaggio.
Tale speranza si fonda sulla possibilità che il linguaggio, sia a livello semantico che sintattico, mimi la struttura delle relazioni tra i concetti e quindi la grammatica del pensiero. In un progetto ambizioso, dunque, con questo ulteriore salto potremmo cercare di collegare il latching alle strutture proprie del linguaggio.
A tal fine, risulterà fondamentale, in futuro, trovare degli osservabili che siano in grado di legare direttamente il modello impostato a dati sperimentali. Ad esempio, interessante sviluppo potrebbe essere quello di danneggiare la rete, modificando magari la connettività tra le unità, e confrontare le prestazioni di questa con quelle di pazienti con traumi o disfunzioni cerebrali.
Altro interessante esercizio sarebbe poi studiare un modello in cui il numero S di possibili stati dell'unità vari da unità a unità. Se, infatti, questi rappresentano i possibili output che un particolare aspetto può assumere (come, ad esempio, 'giallo', 'verde', 'blu', per l'aspetto 'colore') non c'è allora motivo che aspetti diversi abbiano lo stesso numero di output.
Infine, ulteriori sviluppi del modello dovrebbero andare a studiare in dettaglio le proprietà delle diverse tipologie di latching e capire come queste possano essere messe in relazione ai meccanismi di elaborazione concettuale.

Ringraziamenti

Innanzitutto vorrei ringraziare i miei due relatori Alessandro Treves e Riccardo Mannella per la presenza, disponibilità e pazienza che mi hanno concesso in questi mesi di lavoro.

Un grande ringraziamento va poi alla mia famiglia, che mi ha dato la forza per affrontare le difficoltà, il cui supporto è stato costante e prezioso. Un grande bacio a Lucio, che mi ha regalato pensieri felici.

Un enorme ringraziamento va a tutti i miei sorprendenti amici, che mi sono stati vicini col loro affetto e che sono sempre stati pronti ad aiutarmi in ogni momento. Grazie veramente di cuore.

Abbraccio e ringrazio Erika, per la sua dolcezza e per aver condiviso con me ogni minuto di gioia e tristezza di questi mesi. Ringrazio poi Emilio, per il suo intuito scientifico e per tutto quello che mi ha insegnato.
E Athena, Valentina, Gergo e Andre, per l'allegria e ironia con cui hanno saputo sostenermi.
Ringrazio Sara e la sua incauta fiducia nel mio gusto estetico. Carmelo e Raffo, che mi hanno accudito e hanno rappresentato la mia dimensione familiare a Trieste.

Tocca poi a tutti i miei amici 'pisani':
Federico, mio tragicomico grande amico, ti ringrazio per il tuo grande affetto e per essere stato sempre e senza ombra di dubbio dalla mia parte. Giulia, per aver saputo affrontare anche i momenti più difficili con coraggio e passione. Hasina, per la tua allegria e gentilezza. Ringrazio poi Linda e Giulietta, due donne forti, ironiche e attive, che mi hanno incitato nei momenti di difficoltà.

E ancora Alistar, che mi ha subito concesso la sua preziosa amicizia, e Luca, che mi ha adottato a distanza.
Come dimenticare poi Alessio e Carla, che in questi anni, stoicamente, hanno tollerato le mie assenze e mancanze.

Ringrazio infine Goffredo a cui devo l'infinita bellezza di questi anni, che mi è stato vicino e che ha vissuto con me. Grazie di tutto.

Appendice A

I codici

A.1 Simulazione per modello di unità scalare

```
int       xi[N][p];
int       s[N], sold[N];
double    m[p];
int       retr=1;
double    h[N];
double    q;

FILE *fpattern;
FILE *fdynamics;
FILE *fconst;

extern void generate_one_pattern(int);
extern void save_pattern(int);
extern void initializing();
extern void update(int, int);
extern void save_dynamics(int);
extern double mmax();
extern double calcolaq();
extern void save_const();

/*****************************    DYNAMICS   *****************************/

int main()
{
int i, j, k, mu;
double n, prova, pro, H;

fpattern = fopen("spatt.txt","w");
```

```
fdynamics = fopen("sdyn.dat","w");
fconst = fopen("sconst.dat","w");

save_const();

for(i=0; i<p; i++)
{
generate_one_pattern(i);
save_pattern(i);
}

initializing();

for(j=0;j<T;j++)
{
i=(int) floor(N*((double)(rand()/(RAND_MAX+1.0))));
update(i,j);
save_dynamics(j);
}

fclose (fconst);
fclose (fdynamics);
fclose (fpattern);
}


/*******************************************************/
////////////////////////////////////////////////////////

extern int       xi[N][p];
extern int       s[N], sold[N];
extern double    m[p];
extern int       retr;
extern double    h[N];
extern double    q;


extern FILE *fpattern;
extern FILE *fdynamics;
extern FILE *fconst;

void generate_one_pattern(int);
void save_pattern(int);
void initializing();
void update(int, int);
void save_dynamics(int);
double mmax();
double calcolaq();
void save_const();
```

```
double mmax()
{
int mu;
double max;

if(0==retr)
max=m[1];
else
max=m[0];

for(mu=0;mu<p;mu++)
{
if(mu!=retr)
{
if(m[mu]>max)
{
m[mu]=max;
}
}}
return max;
}


void update(int i, int n)
{
int    mu;

h[i]=0.;
for(mu=0;mu<p;mu++)
{
if(mu==retr)
h[i]+=((double)(xi[i][mu]==0)-a)*(m[mu]-
((double)(xi[i][mu]==0)-a)*(double)(s[i]==0)/((double)N*a*(1-a)))
+g*exp(-(n/((double)tau)))*((double)(xi[i][mu]==0));
else
h[i]+=((double)(xi[i][mu]==0)-a)*(m[mu]-((double)(xi[i][mu]==0)-a)
*(double)(s[i]==0)/((double)N*a*(1-a)));
}

sold[i]=s[i];
if(h[i]>U)
s[i]=0;
else
s[i]=1;

for(mu=0;mu<p;mu++)
{
m[mu]+=(((double)(xi[i][mu]==0)-a)*((double)(s[i]==0)-
(double)(sold[i]==0)))/((double)N*a*(1.-a));
}
q+=((s[i]==0)*(s[i]==0)-(sold[i]==0)*(sold[i]==0))/((double)N*a);
```

```
}
void initializing()
{
int i, mu;
double n;

for(i=0;i<N;i++)
{
s[i]=S;
sold[i]=s[i];
}

for(i=0;i<N;i++)
{
h[i]=0.;
}

for(mu=0;mu<p;mu++)
{
m[mu]=0.;
}
for(mu=0;mu<p;mu++)
{
for(i=0;i<N;i++)
{
n=((double)(xi[i][mu]==0)-a)*(double)(s[i]==0);
m[mu]=m[mu]+n;
}
m[mu]=m[mu]/(a*(1.-a)*(double)N);
//printf(" m%d:%.1f \n", mu, m[mu]);
}

q=0;
for(i=0;i<N;i++)
{
q+=(double)((s[i]==0)*(s[i]==0));
//printf(" %.2f \n", q);
}
q=q/(a*(double)N);
//printf(" %.2f \n", q);
}


double calcolaq()
{
int i;

q=0;
for(i=0;i<N;i++)
{
q+=(double)((s[i]==0)*(s[i]==0));
```

```
}
q=q/(a*(double)N);
return q;
}


void save_pattern(int mu)
{
int i;
for(i=0;i<N;i++)
{
fprintf(fpattern, "%d ", xi[i][mu]);
}
fprintf(fpattern, "\n  \n");
}

void save_const()
{
fprintf(fconst, "%d  \n%d  \n%d  \n%.3f  \n%.4f  \n%.3f
\n%d  \n%d  \n", N, p, S, a, U, g, tau, T);
}


void save_dynamics(int n)
{
int i;
double t;
t=(double)n/(double)N;
if((n%100)==0)
{
fprintf(fdynamics, "%.4f %.4f %.4f %.4f\n",
t, q, m[retr], mmax());
}}



void generate_one_pattern(int mu)
{
     int i,j,k,l;
     for(i=0;i<N;i++)
{
xi[i][mu]=S;
}
for(i=0;i<(int) round(a*N);i++)
{
k=0;
while(k!=S)
{
j=(int) floor(N*((double)(rand()/(RAND_MAX+1.0))));
k=xi[j][mu];
}
```

```
xi[j][mu]=(int) floor(S*((double)(rand()/(RAND_MAX+1.0))));
}}
```

A.2 Simulazione per modello di unità scalare con adattamento

```
int       xi[N][p];
int       s[N], sold[N];
double    m[p];
int       retr=0;
double    h[N], rk[N], r0[N];
double    q;
double   theta[N];
double   n0=200*N;
int   aa, bb;

FILE *fpattern;
FILE *fdynamics;
FILE *fconst;
FILE *ftheta0;
FILE *ftheta1;
FILE *frk1;
FILE *frk0;

extern void generate_one_pattern(int);
extern void save_pattern(int);
extern void initializing();
extern void update(int, int);
extern void save_dynamics(int);
extern double mmax();
extern double calcolaq();
extern void save_const();

/**********************    DYNAMICS   ***********************/

int main()
{
int i, j, k, mu;
double n, prova, pro, H;

fpattern = fopen("spatt.txt","w");
fdynamics = fopen("sdyn.dat","w");
fconst = fopen("sconst.dat","w");
ftheta0 = fopen("theta0.dat","w");
ftheta1 = fopen("theta1.dat","w");
frk1 = fopen("rk1.dat","w");
frk0 = fopen("rk0.dat","w");
```

```
save_const();

for(i=0; i<p; i++)
{
generate_one_pattern(i);
save_pattern(i);
}
initializing();

for(j=0;j<T;j++)
{
i=(int)floor(N*((double)(rand()/(RAND_MAX+1.0))));
update(i,j);
save_dynamics(j);
}

fclose (frk0);
fclose (frk1);
fclose (ftheta1);
fclose (ftheta0);
fclose (fconst);
fclose (fdynamics);
fclose (fpattern);
}


/*********************************************************************/
/////////////////////////////////////////////////////////////////////
extern int       xi[N][p];
extern int       s[N], sold[N];
extern double    m[p];
extern int       retr;
extern double    h[N], rk[N], r0[N];
extern double    q;
extern double   theta[N];
extern double   n0;
extern int   aa, bb;

extern FILE *fpattern;
extern FILE *fdynamics;
extern FILE *fconst;
extern FILE *ftheta0;
extern FILE *ftheta1;
extern FILE *frk1;
extern FILE *frk0;

void generate_one_pattern(int);
void save_pattern(int);
void initializing();
void update(int, int);
```

```
void save_dynamics(int);
double mmax();
double calcolaq();
void save_const();
//////////////////////////////////////////////////////////////////////
double mmax()
{
int mu;
double max;

if(0==retr)
max=m[1];
else
max=m[0];

for(mu=0;mu<p;mu++)
{
if(mu!=retr)
{
if(m[mu]>max)
{
max=m[mu];
}
}}
return max;
}


void update(int i, int n)
{
int    mu;
h[i]=0.;
for(mu=0;mu<p;mu++)
{
if(mu==retr)
h[i]+=((double)(xi[i][mu]==0)-a)*(m[mu]-(((double)(xi[i][mu]==0)-a)
*(double)(s[i]==0)/((double)N*a*(1.-a))))+(double)(n>n0)*g*exp(-
((n-n0)/((double)tau)))*((double)(xi[i][mu]==0));
else
h[i]+=((double)(xi[i][mu]==0)-a)*(m[mu]-(((double)(xi[i][mu]==0)-a)
*(double)(s[i]==0)/((double)N*a*(1.-a))));
}
theta[i]+=b2*((double)(s[i]==0)-theta[i]);
rk[i]+=b1*(h[i]-theta[i]-rk[i]);

sold[i]=s[i];

if(rk[i]>U)
s[i]=0;
else
s[i]=1;
```

```
for(mu=0;mu<p;mu++)
{
m[mu]+=(((double)(xi[i][mu]==0)-a)*((double)(s[i]==0)
-(double)(sold[i]==0)))/((double)N*a*(1.-a));
}
q+=((s[i]==0)*(s[i]==0)-(sold[i]==0)*(sold[i]==0))/((double)N*a);
}


void initializing()
{
int i, mu;
double n;
for(i=0;i<N;i++)
{
s[i]=S;
sold[i]=s[i];
}
for(i=0;i<N;i++)
{
h[i]=0.;
rk[i]=0.;
}
for(i=0;i<N;i++)
{
theta[i]=0.;
}
for(mu=0;mu<p;mu++)
{
m[mu]=0.;
}
for(mu=0;mu<p;mu++)
{
for(i=0;i<N;i++)
{
n=((double)(xi[i][mu]==0)-a)*(double)(s[i]==0);
m[mu]=m[mu]+n;
}
m[mu]=m[mu]/(a*(1.-a)*(double)N);
}
q=0;
for(i=0;i<N;i++)
{
q+=(double)((s[i]==0)*(s[i]==0));
}
q=q/(a*(double)N);
}


double calcolaq()
```

```
{
int i;
q=0;
for(i=0;i<N;i++)
{
q+=(double)((s[i]==0)*(s[i]==0));
}
q=q/(a*(double)N);
return q;
}


void save_pattern(int mu)
{
int i;
for(i=0;i<N;i++)
{
fprintf(fpattern, "%d ", xi[i][mu]);
}
fprintf(fpattern, "\n  \n");
}


void save_const()
{
fprintf(fconst, "%d  \n%d  \n%d  \n%.3f  \n%.4f  \n%.3f  \n%d
\n%d  \n%.3f  \n%.3f \n", N, p, S, a, U, g, tau, T, b2, b1);
}


void save_dynamics(int n)
{
int i, j;
double t;

t=(double)n/(double)N;
if((n%500)==0)
{
fprintf(fdynamics, "%.4f %.4f %.4f %.4f\n", t, q,
 m[retr], mmax());
j=(int) floor(N*((double)(rand()/(RAND_MAX+1.0))));
if(xi[j][retr]==0)
fprintf(ftheta1, "%.4f %.4f\n", t, theta[j]);
else
fprintf(ftheta0, "%.4f %.4f\n", t, theta[j]);

j=(int) floor(N*((double)(rand()/(RAND_MAX+1.0))));

if(xi[j][retr]==0)
fprintf(frk1, "%.4f %.4f\n", t, rk[j]);
else
```

```
fprintf(frk0, "%.4f %.4f\n", t, rk[j]);
}}

void generate_one_pattern(int mu)
{
int i,j,k,l;
for(i=0;i<N;i++)
{
xi[i][mu]=S;
}
for(i=0;i<(int) round(a*N);i++)
{
k=0;
while(k!=S)
{
j=(int) floor(N*((double)(rand()/(RAND_MAX+1.0))));
k=xi[j][mu];
}
xi[j][mu]=(int) floor(S*((double)(rand()/(RAND_MAX+1.0))));
}}
```

A.3 Potts adaptation simulazione

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include</home/russo/Desktop/backup/adaptation/const_potts.h>
#include</home/russo/Desktop/backup/adaptation/funzioni_potts.cpp>


int       xi[N][p];
double    s[N][S+1], sold[N][S+1];
double    m[p];
int       retr=0;
double    h[N][S+1], r[N][S+1], r0[N][S+1];
double    q;
double   theta[N][S+1];
double   n0;
char   Bt[20];
char   Br[20];
char   Bs[20];

double    den=a*(1.-as)*(double)N;

FILE *fpattern;
FILE *fdynamics;
FILE *fconst;
```

```
FILE *pat;

FILE *Rss;
FILE *Rsk;
FILE *Rxs;
FILE *Rxk;
FILE *Rxx;

FILE *Tss;
FILE *Tsk;
FILE *Txs;
FILE *Txk;
FILE *Txx;

FILE *Sss;
FILE *Ssk;
FILE *Sxs;
FILE *Sxk;
FILE *Sxx;

FILE *At;
FILE *Ar;
FILE *As;

extern void save_const();
extern void generate_one_pattern(int);
extern void generate_two_pattern();
extern void read_pattern();
extern void save_pattern(int);
extern void initializing();
extern void update_stato(int, int);
extern void update_mq(int);
extern void save_mq(double);
extern void save_tr(int, double);
extern double mmax();
extern double calcolaq();
extern double calcolam(int);

/*****************    DYNAMICS   ********************/

int main()
{
int i, n, k, mu;
double t;

fpattern = fopen("spatt.txt","w");
fdynamics = fopen("s_mq.dat","w");
fconst = fopen("sconst.dat","w");

Rss = fopen("Rss.dat","w");
Rsk = fopen("Rsk.dat","w");
```

```
Rxs = fopen("Rxs.dat","w");
Rxk = fopen("Rxk.dat","w");
Rxx = fopen("Rxx.dat","w");

Tss = fopen("Tss.dat","w");
Tsk = fopen("Tsk.dat","w");
Txs = fopen("Txs.dat","w");
Txk = fopen("Txk.dat","w");
Txx = fopen("Txx.dat","w");

Sss = fopen("Sss.dat","w");
Ssk = fopen("Ssk.dat","w");
Sxs = fopen("Sxs.dat","w");
Sxk = fopen("Sxk.dat","w");
Sxx = fopen("Sxx.dat","w");

n0=200*N;

save_const();

/// PATTERNS RANDOM ///
for(mu=0; mu<p; mu++)
{
generate_one_pattern(mu);
save_pattern(mu);
}

initializing();

for(n=0;n<T;n++)
{
i=(int)floor(N*((double)(rand()/(RAND_MAX+1.0))));

update_stato(i,n);
update_mq(i);

t=(double)n/(double)N;

if((n%tempostampa)==0)
{
save_mq(t);
save_tr(n, t);
}
}


fclose (Rss);
fclose (Rsk);
fclose (Rxs);
fclose (Rxk);
fclose (Rxx);
```

```
fclose (Tss);
fclose (Tsk);
fclose (Txs);
fclose (Txk);
fclose (Txx);

fclose (Sss);
fclose (Ssk);
fclose (Sxs);
fclose (Sxk);
fclose (Sxx);

fclose (fconst);
fclose (fdynamics);
fclose (fpattern);
}

/*************************************************************/
////////////////////////////////////////////////////////////////

extern int       xi[N][p];
extern double    s[N][S+1], sold[N][S+1];
extern double    m[p];
extern int       retr;
extern double    h[N][S+1], r[N][S+1], r0[N][S+1];
extern double    q;
extern double   theta[N][S+1];
extern double   n0;
extern char   Bt[20];
extern char   Br[20];
extern char   Bs[20];

extern double   den;

extern FILE *fpattern;
extern FILE *fdynamics;
extern FILE *fconst;
extern FILE *pat;
extern FILE *Ar;
extern FILE *At;
extern FILE *As;

extern FILE *Rss;
extern FILE *Rsk;
extern FILE *Rxs;
extern FILE *Rxk;
extern FILE *Rxx;

extern FILE *Tss;
extern FILE *Tsk;
```

```
extern FILE *Txs;
extern FILE *Txk;
extern FILE *Txx;

extern FILE *Sss;
extern FILE *Ssk;
extern FILE *Sxs;
extern FILE *Sxk;
extern FILE *Sxx;

void save_const();
void generate_one_pattern(int);
void generate_two_pattern();
void read_pattern();
void save_pattern(int);
void initializing();
void update_stato(int, int);
void update_mq(int);
void save_mq(double);
void save_tr(int, double);
double mmax();
double calcolaq();
double calcolam(int);

//////////////////////////////////////////////////////

void save_const()
{
fprintf(fconst,"%d  \n%d  \n%d  \n%.3f  \n%.4f  \n%.3f  \n%d  \n%d
\n%.3f  \n%.3f \n %d \n %lf  \n", N, p, S, a, U, g, tau, T, b2, b1, beta, n0/N);
}

void generate_one_pattern(int mu)
{
int i,j,k;

for(i=0;i<N;i++)
{
xi[i][mu]=S;
}

for(i=0;i<(int) round(a*N);i++)
{
k=0;
while(k!=S)
{
j=(int) floor(N*((double)(rand()/(RAND_MAX+1.0))));
k=xi[j][mu];
}

xi[j][mu]=(int) floor(S*((double)(rand()/(RAND_MAX+1.0))));
```

```
}
}


void save_pattern(int mu)
{
int i;

for(i=0;i<N;i++)
{
fprintf(fpattern, "%d ", xi[i][mu]);
}
fprintf(fpattern, "\n  \n");

}


void initializing()
{
int i, k, mu;
double qs, ma, maa;


for(i=0;i<N;i++)
{
for(k=0;k<S;k++)
{
s[i][k]=0.;
sold[i][k]=s[i][k];
}
s[i][S]=1.;
sold[i][S]=s[i][S];
}
for(mu=0;mu<p;mu++)
{
m[mu]=0.;
}
for(mu=0;mu<p;mu++)
{
maa=0.;
for(i=0;i<N;i++)
{
ma=0.;
for(k=0;k<S;k++)
{
ma+=((double)(xi[i][mu]==k)-as)*s[i][k];
}
maa+=ma;
}
m[mu]=maa/den;
}
```

```
q=0.;
for(i=0;i<N;i++)
{
qs=0.;
for(k=0;k<S;k++)
{
qs+=(s[i][k])*(s[i][k]);
}
q+=qs-as*(1.-s[i][S])*(1.-s[i][S]);
}
q=q/den;

for(i=0;i<N;i++)
{
h[i][S]=U;
r[i][S]=h[i][S];
theta[i][S]=0.;
for(k=0;k<S;k++)
{
h[i][k]=0.;
  for(mu=0;mu<p;mu++)
  {
  h[i][k]+=m[mu]*((double)(xi[i][mu]==k)-as);
  }
r[i][k]=h[i][k];
theta[i][k]=0.;
}
}
}


void update_stato(int i, int n)
{
int   mu, k, l, smax;
double Z, rmax, hmax;
double II[p];

rmax=r[i][S];
smax=S;

for(k=0;k<S;k++)
{
  h[i][k]=0.;
  for(mu=0;mu<p;mu++)
  {
II[mu]=0.;
for(l=0;l<S;l++)
{
II[mu]+=((double)(xi[i][mu]==l)-as)*s[i][l];
```

```
}
II[mu]=II[mu]/den;


  h[i][k]+=((double)(xi[i][mu]==k)-as)*(m[mu]-II[mu])+
  (double)(n>n0)*g*exp(-((n-n0)/((double)tau)))*
  (double)((xi[i][mu]==k)*(mu==retr));
  }

  sold[i][k]=s[i][k];

  theta[i][k]+=b2*(s[i][k]-theta[i][k]);
  r[i][k]+=b1*(h[i][k]-theta[i][k]-r[i][k]);

  if(r[i][k]>rmax)
  {
  rmax=r[i][k];
  smax=k;
  }
}

h[i][S]=U;
sold[i][S]=s[i][S];

theta[i][S]+=b2*(s[i][S]-theta[i][S]);
r[i][S]+=b1*(h[i][S]-theta[i][S]-r[i][S]);

//////////////    UPDATE PER T=0    ///////////////
/*
for(k=0;k<=S;k++)
{
s[i][k]=1.*(smax==k)+0.;
}
*/

//////////////    UPDATE PER T!=0    ///////////////

Z=0.;
for(k=0;k<=S;k++)
{
Z+=exp(beta*(r[i][k]-rmax));
}

for(k=0;k<=S;k++)
{
s[i][k]=exp(beta*(r[i][k]-rmax))/Z;
}
}


void update_mq(int i)
```

```
{
int   l, mu,j;
double qs,Q ,QS;


for(mu=0;mu<p;mu++)
{
for(l=0;l<S;l++)
{
m[mu]+=((double)(xi[i][mu]==l)-as)*(s[i][l]-sold[i][l])/den;
}
}


qs=0.;
for(l=0;l<S;l++)
{
qs+=((s[i][l])*(s[i][l]))-((sold[i][l])*(sold[i][l]));
}
q+=(qs-as*((1.-s[i][S])*(1.-s[i][S])-(1.-sold[i][S])*(1.-sold[i][S])))/den;
}


void save_mq(double t)
{
fprintf(fdynamics, "%.4f %.4f %.4f %.4f\n", t, q, m[retr], mmax());
}


void save_tr(int n, double t)
{
int j, l, k;


j=(int) floor(N*((double)(rand()/(RAND_MAX+1.0))));


/// ////////////////   DIVIDO PER STATI
/*
for(k=0;k<=S;k++)
{
sprintf(Bt,"theta_%d.dat",k);
At=fopen(Bt,"a");
sprintf(Br,"r_%d.dat",k);
Ar=fopen(Br,"a");
sprintf(Bs,"S_%d.dat",k);
As=fopen(Bs,"a");
fprintf(As, "%.4f %.4f\n", t,s[j][k]);
fprintf(Ar, "%.4f %.4f\n", t,r[j][k]);
fprintf(At, "%.4f %.4f\n", t,theta[j][k]);
fclose(As);
```

```
fclose(At);
fclose(Ar);
}*/
/// ////////      DIVIDO PER PATTERN


if(xi[j][retr]==S)
{
fprintf(Rss, "%.4f %.4f\n", t, r[j][S]);
fprintf(Tss, "%.4f %.4f\n", t, theta[j][S]);
fprintf(Sss, "%.4f %.4f\n", t, s[j][S]);

for(k=0;k<S;k++)
{
fprintf(Rsk, "%.4f %.4f\n", t, r[j][k]);
fprintf(Tsk, "%.4f %.4f\n", t, theta[j][k]);
fprintf(Ssk, "%.4f %.4f\n", t, s[j][k]);
}
}
else
{
fprintf(Rxs, "%.4f %.4f\n", t, r[j][S]);
fprintf(Txs, "%.4f %.4f\n", t, theta[j][S]);
fprintf(Sxs, "%.4f %.4f\n", t, s[j][S]);

for(k=0;k<S;k++)
{
   if(xi[j][retr]==k)
   {
   fprintf(Rxx, "%.4f %.4f\n", t, r[j][k]);
   fprintf(Txx, "%.4f %.4f\n", t, theta[j][k]);
   fprintf(Sxx, "%.4f %.4f\n", t, s[j][k]);
   }
   else
   {
   fprintf(Rxk, "%.4f %.4f\n", t, r[j][k]);
   fprintf(Txk, "%.4f %.4f\n", t, theta[j][k]);
   fprintf(Sxk, "%.4f %.4f\n", t, s[j][k]);
   }
}
}

}


double mmax()
{
int mu;
double max;

if(retr==0)
```

```
max=m[1];
else
max=m[0];

for(mu=0;mu<p;mu++)
{
if(mu!=retr)
{
if(m[mu]>max)
{
max=m[mu];
}
}
}
return max;
}
```

A.4 Codice di integrazione numerica per rete di Potts con adattamento

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include "nrutil.h"
#define SAFETY 0.9
#define PGROW -0.2
#define PSHRNK -0.25
#define ERRCON 1.89e-4
#define MAXSTP 10000
#define TINY 1.0e-30

void derivs(double,double [],double []);
void odeint(double [],int,double,double,double,double,double,int *,int *,
void (*)(double, double [], double []),void (*)(double [],double [],int,
double *,double,double,double [],double *,double *,void (*)(double,double [],double [])));
void rkck(double [],double [],int,double,double,double [],double [],
void (*)(double,double [],double []));
void rkqs(double [],double [],int,double *,double, double,double [],
double *,double *,void (*)(double,double [],double []));

double a,U,g,b2,b1,N,p,S,tau,T,beta,x0;
double as, m1, m2, q1, q2;
int kmax, kount, nvar;
double *xp, **yp, dxsav;

FILE *cost;
```

```
FILE *dinamica;


int main()
{
int i;
int nok;
int nbad;

nvar=12;
kmax=8000;
dxsav=0.001;

xp=vector(1,kmax);
yp=matrix(1,nvar,1,kmax);

cost=fopen("/scratch/russo/simulazioni/potts/sconst.dat","r");
fscanf(cost, "%lf", &N);
fscanf(cost, "%lf", &p);
fscanf(cost, "%lf", &S);
fscanf(cost, "%lf", &a);
fscanf(cost, "%lf", &U);
fscanf(cost, "%lf", &g);
fscanf(cost, "%lf", &tau);
fscanf(cost, "%lf", &T);
fscanf(cost, "%lf", &b2);
fscanf(cost, "%lf", &b1);
fscanf(cost, "%lf", &beta);
fscanf(cost, "%lf", &x0);
fclose(cost);

double ystart[13]={0.,0.,0., U,  0.,  U,  0.,  0., 0.,  0.,  0.,  0.,  0.};

as=a/S;
m1=(1.-a)/((1.-as)*S);
m2=as/(1.-as);
q1=(S*(1.-S)*(1.-a))/(a*(1.-as));
q2=1./(1.-as);

tau=tau/N;

odeint(  ystart, nvar, 0., T/N,  0.000001,  0.01,  0.0,&nok,&nbad,derivs,rkqs);

dinamica=fopen("tutti.dat","w");
for(i=1;i<kmax;i++)
{
fprintf(dinamica, "%lf %lf %lf %lf
%lf %lf %lf %lf %lf
%lf %lf %lf %lf\n",
xp[i], yp[1][i], yp[2][i], yp[3][i], yp[4][i], yp[5][i], yp[6][i],
yp[7][i], yp[8][i], yp[9][i], yp[10][i], yp[11][i], yp[12][i]);
```

```
}
fclose(dinamica);
}

void odeint(double ystart[], int nvar, double x1, double x2, double eps,
double h1,double hmin, int *nok,int *nbad,void (*derivs)(double, double [], double []),
void (*rkqs)(double [], double [], int, double *, double, double, double [],
double *, double *, void (*)(double, double [], double [])))
{
int nstp,i;
double xsav,x,hnext,hdid,h;
double *yscal,*y,*dydx;
yscal=vector(1,nvar);
y=vector(1,nvar);
dydx=vector(1,nvar);

x=x1;
h=SIGN(h1,x2-x1);
*nok =0;
*nbad=0;
kount = 0;

for (i=1;i<=nvar;i++)
y[i]=ystart[i];
if (kmax > 0)
xsav=x-dxsav*2.0;
for (nstp=1;nstp<=MAXSTP;nstp++)
{
(*derivs)(x,y,dydx);
for (i=1;i<=nvar;i++)
yscal[i]=fabs(y[i])+fabs(dydx[i]*h)+TINY;
if (kmax > 0 && kount < kmax-1 && fabs(x-xsav) > fabs(dxsav))
{
xp[++kount]=x;
for (i=1;i<=nvar;i++)
yp[i][kount]=y[i];
xsav=x;
}
if ((x+h-x2)*(x+h-x1) > 0.0)
h=x2-x;
(*rkqs)(y,dydx,nvar,&x,h,eps,yscal,&hdid,&hnext,derivs);
if (hdid == h)
++(*nok);
else
++(*nbad);
if ((x-x2)*(x2-x1) >= 0.0)
{
for (i=1;i<=nvar;i++)
ystart[i]=y[i];
if (kmax)
{
```

```
xp[++kount]=x;
for (i=1;i<=nvar;i++)
yp[i][kount]=y[i];
}
free_vector(dydx,1,nvar);
free_vector(y,1,nvar);
free_vector(yscal,1,nvar);
return;
}
if (fabs(hnext) <= hmin)
nrerror("Step size too small in odeint");
h=hnext;
}
nrerror("Too many steps in routine odeint");
}


void rkqs(double y[], double dydx[], int n, double *x, double htry, double eps,
double yscal[], double *hdid,double *hnext, void (*derivs)(double, double [], double []))
{
int i;
double errmax,h,htemp,xnew,*yerr,*ytemp;
yerr=vector(1,n);
ytemp=vector(1,n);
h=htry;
for (;;)
{
rkck(y,dydx,n,*x,h,ytemp,yerr,derivs);
errmax=0.0;
for (i=1;i<=n;i++)
{errmax=FMAX(errmax,fabs(yerr[i]/yscal[i]));
}
errmax /= eps;

if (errmax <= 1.0)
break;
htemp=SAFETY*h*pow(errmax,PSHRNK);

h=(h >= 0.0 ? FMAX(htemp,0.1*h) : FMIN(htemp,0.1*h));
xnew=(*x)+h;
if (xnew == *x) nrerror("stepsize underflow in rkqs");
}
if (errmax > ERRCON) *hnext=SAFETY*h*pow(errmax,PGROW);
else *hnext=5.0*h;
*x += (*hdid=h);
for (i=1;i<=n;i++) y[i]=ytemp[i];
free_vector(ytemp,1,n);
free_vector(yerr,1,n);
}
```

```
void rkck(double y[], double dydx[], int n, double x, double h, double yout[],
double yerr[], void (*derivs)(double, double [], double []))
{
int i;
static double a2=0.2,a3=0.3,a4=0.6,a5=1.0,a6=0.875,b21=0.2,
b31=3.0/40.0,b32=9.0/40.0,b41=0.3,b42 = -0.9,b43=1.2,
b51 = -11.0/54.0, b52=2.5,b53 = -70.0/27.0,b54=35.0/27.0,
b61=1631.0/55296.0,b62=175.0/512.0,b63=575.0/13824.0,
b64=44275.0/110592.0,b65=253.0/4096.0,c1=37.0/378.0,
c3=250.0/621.0,c4=125.0/594.0,c6=512.0/1771.0,
dc5 = -277.00/14336.0;
double dc1=c1-2825.0/27648.0,dc3=c3-18575.0/48384.0,
dc4=c4-13525.0/55296.0,dc6=c6-0.25;
double *ak2,*ak3,*ak4,*ak5,*ak6,*ytemp;
ak2=vector(1,n);
ak3=vector(1,n);
ak4=vector(1,n);
ak5=vector(1,n);
ak6=vector(1,n);
ytemp=vector(1,n);
for (i=1;i<=n;i++)
ytemp[i]=y[i]+b21*h*dydx[i];
(*derivs)(x+a2*h,ytemp,ak2);
for (i=1;i<=n;i++)
ytemp[i]=y[i]+h*(b31*dydx[i]+b32*ak2[i]);
(*derivs)(x+a3*h,ytemp,ak3);
for (i=1;i<=n;i++)
ytemp[i]=y[i]+h*(b41*dydx[i]+b42*ak2[i]+b43*ak3[i]);
(*derivs)(x+a4*h,ytemp,ak4);
for (i=1;i<=n;i++)
ytemp[i]=y[i]+h*(b51*dydx[i]+b52*ak2[i]+b53*ak3[i]+b54*ak4[i]);
(*derivs)(x+a5*h,ytemp,ak5);
for (i=1;i<=n;i++)
ytemp[i]=y[i]+h*(b61*dydx[i]+b62*ak2[i]+b63*ak3[i]+b64*ak4[i]+b65*ak5[i]);
(*derivs)(x+a6*h,ytemp,ak6);
for (i=1;i<=n;i++)
yout[i]=y[i]+h*(c1*dydx[i]+c3*ak3[i]+c4*ak4[i]+c6*ak6[i]);
for (i=1;i<=n;i++)
yerr[i]=h*(dc1*dydx[i]+dc3*ak3[i]+dc4*ak4[i]+dc5*ak5[i]+dc6*ak6[i]);

free_vector(ytemp,1,n);
free_vector(ak6,1,n);
free_vector(ak5,1,n);
free_vector(ak4,1,n);
free_vector(ak3,1,n);
free_vector(ak2,1,n);
}


void derivs(double x, double y[], double dydx[])
{
```

```
double M1, M2, M3,Q1,Q1square,Q2up,Q2down,Q2downsquare, QQQ2, QQQ1;
/// EMME ///

M1=1.+exp(beta*(y[3]-y[4]))/S;
M2=1.+exp(beta*(y[5]-y[7]))+exp(beta*(y[6]-y[7]));
M3=1.+(exp(beta*(y[5]-y[6]))+exp(beta*(y[7]-y[6])))/(S-1.);

dydx[1]=-y[1] -m1/M1 +1./M2 -m2/M3;
/// Q ///

Q1=exp(beta*(y[3]-y[4]))+S;
Q1square=Q1*Q1;
QQQ2=y[6]-y[7];
QQQ1=y[5]-y[7];
Q2up=2.+(S-1.)*S*exp(beta*2.*QQQ2)+2.*(S-1.)*exp(beta*QQQ2);
Q2down=1.+exp(beta*QQQ1)+(S-1.)*exp(beta*QQQ2);
Q2downsquare=Q2down*Q2down;
dydx[2]=0.;
-y[2]+ q1/Q1square+ q2*Q2up/Q2downsquare;
/// ////
dydx[3]=b1*(U-y[8]-y[3]);
dydx[4]=b1*(-as*y[1]-y[9]-y[4]);
dydx[5]=b1*(U-y[10]-y[5]);
dydx[6]=b1*(-as*y[1]-y[11]-y[6]);
dydx[7]=b1*((1.-as)*y[1]+(double)(x>x0)*g*exp(-((x-x0)/((double)tau)))-y[12]-y[7]);
dydx[8]=b2*((1./(1.+S*exp(beta*(y[4]-y[3]))))-y[8]);
dydx[9]=b2*(1./(S+exp(beta*(y[3]-y[4]))))-y[9]);
dydx[10]=b2*(1./(1.+exp(beta*(y[7]-y[5]))+(S-1.)*exp(beta*(y[6]-y[5]))))-y[10]);
dydx[11]=b2*(1./(exp(beta*(y[5]-y[6]))+exp(beta*(y[7]-y[6]))+(S-1.))-y[11]);
dydx[12]=b2*(1./(exp(beta*(y[5]-y[7]))+1.+(S-1.)*exp(beta*(y[6]-y[7])))-y[12]);
}
```

A.5 Simulazione per rete di Potts con latching

```
int       xi[N][p];
double    s[N][S+1], sold[N][S+1];
double    m[p];
double    mS[p];
int       retr=0;
double    h[N][S], r[N][S+1];
double    q;
double   theta[N][S];
double   n0;
char   Bt[20];
char   Br[20];
char   Bs[20];
double    den=a*(1.-as)*(double)N;

FILE *fpattern;
```

```
FILE *fdynamics;
FILE *fconst;
FILE *pat;
FILE *At;
FILE *Ar;
FILE *As;

FILE *Rxss;FILE *Rxsx;
FILE *Rxsk;FILE *Rxxs;
FILE *Rxxx;FILE *Rxxk;
FILE *Rxys;FILE *Rxyx;
FILE *Rxyy;FILE *Rxyk;
FILE *Rsss;FILE *Rssk;
FILE *Rsys;FILE *Rsyy;
FILE *Rsyk;

FILE *Txsx;FILE *Txsk;
FILE *Txxx;FILE *Txxk;
FILE *Txyx;FILE *Txyy;
FILE *Txyk;FILE *Tssk;
FILE *Tsyy;FILE *Tsyk;

FILE *Sxss;FILE *Sxsx;
FILE *Sxsk;FILE *Sxxs;
FILE *Sxxx;FILE *Sxxk;
FILE *Sxys;FILE *Sxyx;
FILE *Sxyy;FILE *Sxyk;
FILE *Ssss;FILE *Sssk;
FILE *Ssys;FILE *Ssyy;
FILE *Ssyk;

extern void save_const();
extern void generate_one_pattern(int);
extern void generate_two_pattern();
extern void read_pattern();
extern void save_pattern(int);
extern void initializing();
extern void update_stato(int, int);
extern void update_mq(int);
extern void save_mq(double);
extern void save_tr(int, double);
extern double mmax();
extern double calcolaq();
extern double calcolam(int);

/*************************    DYNAMICS   **********************/

int main()
{
int i, n, mu;
double t;
```

```
fdynamics = fopen(mq,"w");
fconst = fopen("sconst.dat","w");

Rxss = fopen("dati/Rxss.dat","w");
Rxsx = fopen("dati/Rxsx.dat","w");
Rxsk = fopen("dati/Rxsk.dat","w");
Rxxs = fopen("dati/Rxxs.dat","w");
Rxxx = fopen("dati/Rxxx.dat","w");
Rxxk = fopen("dati/Rxxk.dat","w");
Rxys = fopen("dati/Rxys.dat","w");
Rxyx = fopen("dati/Rxyx.dat","w");
Rxyy = fopen("dati/Rxyy.dat","w");
Rxyk = fopen("dati/Rxyk.dat","w");
Rsss = fopen("dati/Rsss.dat","w");
Rssk = fopen("dati/Rssk.dat","w");
Rsys = fopen("dati/Rsys.dat","w");
Rsyy = fopen("dati/Rsyy.dat","w");
Rsyk = fopen("dati/Rsyk.dat","w");

Sxss = fopen("dati/Sxss.dat","w");
Sxsx = fopen("dati/Sxsx.dat","w");
Sxsk = fopen("dati/Sxsk.dat","w");
Sxxs = fopen("dati/Sxxs.dat","w");
Sxxx = fopen("dati/Sxxx.dat","w");
Sxxk = fopen("dati/Sxxk.dat","w");
Sxys = fopen("dati/Sxys.dat","w");
Sxyx = fopen("dati/Sxyx.dat","w");
Sxyy = fopen("dati/Sxyy.dat","w");
Sxyk = fopen("dati/Sxyk.dat","w");
Ssss = fopen("dati/Ssss.dat","w");
Sssk = fopen("dati/Sssk.dat","w");
Ssys = fopen("dati/Ssys.dat","w");
Ssyy = fopen("dati/Ssyy.dat","w");
Ssyk = fopen("dati/Ssyk.dat","w");

Txsx = fopen("dati/Txsx.dat","w");
Txsk = fopen("dati/Txsk.dat","w");
Txxx = fopen("dati/Txxx.dat","w");
Txxk = fopen("dati/Txxk.dat","w");
Txyx = fopen("dati/Txyx.dat","w");
Txyy = fopen("dati/Txyy.dat","w");
Txyk = fopen("dati/Txyk.dat","w");
Tssk = fopen("dati/Tssk.dat","w");
Tsyy = fopen("dati/Tsyy.dat","w");
Tsyk = fopen("dati/Tsyk.dat","w");
fpattern = fopen("dati/spatt.txt","w");

n0=500*N;
printf("pippo");
save_const();
```

```
/// DUE PATTERN CORRELATI ////
generate_two_pattern();

for(mu=0;mu<p;mu++)
{
if(mu!=0&&mu!=1)
{
generate_one_pattern(mu);
}
}


for(mu=0;mu<p;mu++)
{
save_pattern(mu);
}

initializing();


for(n=0;n<T;n++)
{
i=(int)floor(N*((double)(rand()/(RAND_MAX+1.0))));

update_stato(i,n);
update_mq(i);
t=(double)n/(double)N;

if((n%tempostampa)==0)
{
save_mq(t);
save_tr(n, t);
}
}

fclose (fpattern);
fclose (Rxss);fclose (Rxsx);
fclose (Rxsk);fclose (Rxxs);
fclose (Rxxx);fclose (Rxxk);
fclose (Rxys);fclose (Rxyx);
fclose (Rxyy);fclose (Rxyk);
fclose (Rsss);fclose (Rssk);
fclose (Rsys);fclose (Rsyy);
fclose (Rsyk);fclose (Txsx);
fclose (Txsk);fclose (Txxx);
fclose (Txxk);fclose (Txyx);
fclose (Txyy);fclose (Txyk);
fclose (Tssk);fclose (Tsyy);
fclose (Tsyk);
fclose (Sxss);fclose (Sxsx);
fclose (Sxsk);fclose (Sxxs);
```

```
fclose (Sxxx);fclose (Sxxk);
fclose (Sxys);fclose (Sxyx);
fclose (Sxyy);fclose (Sxyk);
fclose (Ssss);fclose (Sssk);
fclose (Ssys);fclose (Ssyy);
fclose (Ssyk);

fclose (fconst);
fclose (fdynamics);
}


/*****************************************************/
//////////////////////////////////////////////////////

extern int       xi[N][p];
extern double    s[N][S+1], sold[N][S+1];
extern double    m[p];
extern double    mS[p];
extern int       retr;
extern double    h[N][S], r[N][S+1];
extern double    q;
extern double   theta[N][S];
extern double   n0;
extern char   Bt[20];
extern char   Br[20];
extern char   Bs[20];

extern double   den;

extern FILE *fpattern;
extern FILE *fdynamics;
extern FILE *fconst;
extern FILE *pat;
extern FILE *Ar;
extern FILE *At;
extern FILE *As;

extern FILE *Rxss;extern FILE *Rxsx;
extern FILE *Rxsk;extern FILE *Rxxs;
extern FILE *Rxxx;extern FILE *Rxxk;
extern FILE *Rxys;extern FILE *Rxyx;
extern FILE *Rxyy;extern FILE *Rxyk;
extern FILE *Rsss;extern FILE *Rssk;
extern FILE *Rsys;extern FILE *Rsyy;
extern FILE *Rsyk;

extern FILE *Txsx;extern FILE *Txsk;
extern FILE *Txxx;extern FILE *Txxk;
extern FILE *Txyx;extern FILE *Txyy;
extern FILE *Txyk;extern FILE *Tssk;
```

```
extern FILE *Tsyy;extern FILE *Tsyk;

extern FILE *Sxss;extern FILE *Sxsx;
extern FILE *Sxsk;extern FILE *Sxxs;
extern FILE *Sxxx;extern FILE *Sxxk;
extern FILE *Sxys;extern FILE *Sxyx;
extern FILE *Sxyy;extern FILE *Sxyk;
extern FILE *Ssss;extern FILE *Sssk;
extern FILE *Ssys;extern FILE *Ssyy;
extern FILE *Ssyk;

void save_const();
void generate_one_pattern(int);
void generate_two_pattern();
void read_pattern();
void save_pattern(int);
void initializing();
void update_stato(int, int);
void update_mq(int);
void save_mq(double);
void save_tr(int, double);
double mmax();
double calcolaq();
double calcolam(int);

//////////////////////////////////////////////////////

void save_const()
{
fprintf(fconst,"%d  \n%d  \n%d  \n%lf  \n%lf  \n%lf
\n%d  \n%d  \n%lf  \n%lf \n%lf \n%d \n%lf \n%d \n%d
\n%lf \n\n", N, p, S, a, U, g, tau, T, b2, b1,
b3,beta, n0/N, C1, C2, w);
}

void generate_one_pattern(int mu)
{
int i,j,k;

for(i=0;i<N;i++)
{
xi[i][mu]=S;
}

j=0;
i=0;
while(i<(int)round(a*N))
{
j=(int) floor(N*((double)(rand()/(RAND_MAX+1.0))));

if(xi[j][mu]==S)
```

```
{
xi[j][mu]=(int) floor(S*((double)(rand()/(RAND_MAX+1.0))));
i++;
}
}}

void generate_two_pattern()
{
int i, j, k, c1, c2, temp, d;

for(i=0;i<N;i++)
{
xi[i][0]=S;
xi[i][1]=S;
}
c1=0;
c2=0;
d=0;
i=0;
while(i<(int)round(a*N))
{
j=(int) floor(N*((double)(rand()/(RAND_MAX+1.0))));

if(xi[j][0]==S)
{
xi[j][0]=(int) floor(S*((double)(rand()/(RAND_MAX+1.0))));
i++;
}

if(c1<=C1)
{
xi[j][1]=xi[j][0];
  c1++;
}
else
{
if(c2<=C2)
{
temp=xi[j][0];
while(temp==xi[j][0])
{
temp=(int) floor(S*((double)(rand()/(RAND_MAX+1.0))));
}
xi[j][1]=temp;
c2++;
}
}
}

while(d<=((int)round(a*N)-C1-C2))
{
```

```
j=(int) floor(N*((double)(rand()/(RAND_MAX+1.0))));

if((xi[j][0]==S)&&(xi[j][1]==S))
{
xi[j][1]=(int) floor(S*((double)(rand()/(RAND_MAX+1.0))));
d++;
}
}}

void read_pattern()
{
int i, mu;
pat=fopen("/scratch/russo/simulazioni/potts/programma vijay/patterns.dat","r");

for(mu=0;mu<p;mu++)
{
for(i=0;i<N;i++)
{
fscanf(pat, "%d", &xi[i][mu]);
}
}
fclose(pat);

}

void save_pattern(int mu)
{
int i;

for(i=0;i<N;i++)
{
fprintf(fpattern, "%d ", xi[i][mu]);
}
fprintf(fpattern, "\n %d, %d, %d \n",N,mu,xi[N-1][mu]);
}


void initializing()
{
int i, k, mu, D[N], j;
double qs, ma, maa, maS, maaS;
for(i=0;i<N;i++)
{
for(k=0;k<=S;k++)
{
s[i][k]=1./(double)(S+1);
sold[i][k]=s[i][k];
}
}
/// M ///
for(mu=0;mu<p;mu++)
```

```
{
m[mu]=0.;
}
for(mu=0;mu<p;mu++)
{
maa=0.;
for(i=0;i<N;i++)
{
ma=0.;
for(k=0;k<S;k++)
{
ma+=((double)(xi[i][mu]==k)-as)*s[i][k];
}
maa+=ma;
}
m[mu]=maa/den;
}
/// Q ///
q=0.;
for(i=0;i<N;i++)
{
qs=0.;
for(k=0;k<S;k++)
{
qs+=(s[i][k])*(s[i][k]);
}
q+=qs-as*(1.-s[i][S])*(1.-s[i][S]);
}
q=q/den;
/// H , R , T  ///
for(i=0;i<N;i++)
{
r[i][S]=0.;

for(k=0;k<S;k++)
{
h[i][k]=0.;
  for(mu=0;mu<p;mu++)
  {
  h[i][k]+=m[mu]*((double)(xi[i][mu]==k)-as);
  }
r[i][k]=h[i][k];
theta[i][k]=s[i][k];
}
}
}

void update_stato(int i, int n)
{
int   mu, k, l, smax;
double Z, rmax;
```

```
double II[p], self;

rmax=r[i][S];

/// update per tutti gli stati diversi da S ///////
self=0.; // calcolo la auto eccitazione
for(l=0;l<S;l++)
{
self+=s[i][l];
}
self=(w*self)/S;
for(k=0;k<S;k++)
{

  h[i][k]=0.;
  for(mu=0;mu<p;mu++)
  {
II[mu]=0.;
for(l=0;l<S;l++)
{
  II[mu]+=((double)(xi[i][mu]==l)-as)*s[i][l];
}
II[mu]=II[mu]/den;

h[i][k]+=((double)(xi[i][mu]==k)-as)*(m[mu]-II[mu])+
(double)(n>n0)*g*exp(-((n-n0)/((double)tau)))
*(double)((xi[i][mu]==k)*(mu==retr));
  }
h[i][k]+=w*s[i][k]-self;

  sold[i][k]=s[i][k];
  theta[i][k]+=b2*(s[i][k]-theta[i][k]);
  r[i][k]+=b1*(h[i][k]-theta[i][k]-r[i][k]);


  if(r[i][k]>rmax)
  {
  rmax=r[i][k];
  }
}

/// ////////// update rS e sold per S ///
sold[i][S]=s[i][S];
r[i][S]+=b3*(1.-s[i][S]-r[i][S]);

/// //////////    UPDATE stato PER T!=0    ///////////////
Z=0.;
for(k=0;k<S;k++)
{
Z+=exp(beta*(r[i][k]-rmax));
}
```

```
Z+=exp(beta*(r[i][S]+U-rmax));

for(k=0;k<S;k++)
{
s[i][k]=exp(beta*(r[i][k]-rmax))/Z;
}
s[i][S]=exp(beta*(r[i][S]-rmax+U))/Z;
}

void update_mq(int i)
{
int   l, mu,j;
double qs,Q ,QS;

for(mu=0;mu<p;mu++)
{
for(l=0;l<S;l++)
{
m[mu]+=((double)(xi[i][mu]==l)-as)*(s[i][l]-sold[i][l])/den;
}}

qs=0.;
for(l=0;l<S;l++)
{
qs+=((s[i][l])*(s[i][l]))-((sold[i][l])*(sold[i][l]));
}
q+=(qs-as*((1.-s[i][S])*(1.-s[i][S])-(1.-sold[i][S])*(1.-sold[i][S])))/den;
}


void save_mq(double t)
{
fprintf(fdynamics, "%.4f %.4f %.4f %.4f
%.4f\n", t, q, m[retr], mmax(), m[1]);
}

void save_tr(int n, double t)
{
int j, l, k;
j=(int) floor(N*((double)(rand()/(RAND_MAX+1.0))));
/// /////////////////      DIVIDO PER PATTERN

if(xi[j][retr]==S)
{

if(xi[j][1]==S) ///   SS
{
fprintf(Rsss, "%.4f %.4f\n", t, r[j][S]);
fprintf(Ssss, "%.4f %.4f\n", t, s[j][S]);
for(k=0;k<S;k++)
{
```

```
fprintf(Rssk, "%.4f %.4f\n", t, r[j][k]);
fprintf(Tssk, "%.4f %.4f\n", t, theta[j][k]);
fprintf(Sssk, "%.4f %.4f\n", t, s[j][k]);
}
}
else ///   SY
{
fprintf(Rsys, "%.4f %.4f\n", t, r[j][S]);
fprintf(Ssys, "%.4f %.4f\n", t, s[j][S]);
for(k=0;k<S;k++)
{
   if(xi[j][1]==k)
   {
fprintf(Rsyy, "%.4f %.4f\n", t, r[j][k]);
fprintf(Tsyy, "%.4f %.4f\n", t, theta[j][k]);
fprintf(Ssyy, "%.4f %.4f\n", t, s[j][k]);
}
else
{
fprintf(Rsyk, "%.4f %.4f\n", t, r[j][k]);
fprintf(Tsyk, "%.4f %.4f\n", t, theta[j][k]);
fprintf(Ssyk, "%.4f %.4f\n", t, s[j][k]);
  }
}
}
}
else ///se xi 1 attiva
{
if(xi[j][1]==S) ///   XS

fprintf(Rxss, "%.4f %.4f\n", t, r[j][S]);
fprintf(Sxss, "%.4f %.4f\n", t, s[j][S]);
for(k=0;k<S;k++)
{
  if(xi[j][0]==k)
   {
fprintf(Rxsx, "%.4f %.4f\n", t, r[j][k]);
fprintf(Txsx, "%.4f %.4f\n", t, theta[j][k]);
fprintf(Sxsx, "%.4f %.4f\n", t, s[j][k]);
}
else
{
fprintf(Rxsk, "%.4f %.4f\n", t, r[j][k]);
fprintf(Txsk, "%.4f %.4f\n", t, theta[j][k]);
fprintf(Sxsk, "%.4f %.4f\n", t, s[j][k]);
  }
}
}
else if(xi[j][1]==xi[j][0]) ///   XX
{
fprintf(Rxxs, "%.4f %.4f\n", t, r[j][S]);
```

```
fprintf(Sxxs, "%.4f %.4f\n", t, s[j][S]);
for(k=0;k<S;k++)
{
  if(xi[j][0]==k)
   {
fprintf(Rxxx, "%.4f %.4f\n", t, r[j][k]);
fprintf(Txxx, "%.4f %.4f\n", t, theta[j][k]);
fprintf(Sxxx, "%.4f %.4f\n", t, s[j][k]);
}
else
{
fprintf(Rxxk, "%.4f %.4f\n", t, r[j][k]);
fprintf(Txxk, "%.4f %.4f\n", t, theta[j][k]);
fprintf(Sxxk, "%.4f %.4f\n", t, s[j][k]);
  }
}
}
else ///   XY
{
fprintf(Rxys, "%.4f %.4f\n", t, r[j][S]);
fprintf(Sxys, "%.4f %.4f\n", t, s[j][S]);
for(k=0;k<S;k++)
{
   if(xi[j][0]==k)
   {
fprintf(Rxyx, "%.4f %.4f\n", t, r[j][k]);
fprintf(Txyx, "%.4f %.4f\n", t, theta[j][k]);
fprintf(Sxyx, "%.4f %.4f\n", t, s[j][k]);
}
else if(xi[j][1]==k)
   {
fprintf(Rxyy, "%.4f %.4f\n", t, r[j][k]);
fprintf(Txyy, "%.4f %.4f\n", t, theta[j][k]);
fprintf(Sxyy, "%.4f %.4f\n", t, s[j][k]);
}
else
{
fprintf(Rxyk, "%.4f %.4f\n", t, r[j][k]);
fprintf(Txyk, "%.4f %.4f\n", t, theta[j][k]);
fprintf(Sxyk, "%.4f %.4f\n", t, s[j][k]);
  }
}
}
}

}

double mmax()
{
int mu;
double max;
```

```
/// ////////////////
max=m[2];
for(mu=2;mu<p;mu++)
{
if(m[mu]>max)
{
max=m[mu];
}
}
/// ////////////////
return max;
}

double calcolaq()
{
int i, l;
double Q, QS;

Q=0.;
for(i=0;i<N;i++)
{
QS=0.;
for(l=0;l<S;l++)
{
QS+=(s[i][l])*(s[i][l]);
}
Q+=QS-as*(1.-s[i][S])*(1.-s[i][S]);
}
Q=Q/den;
return Q;
}

double calcolam(int mu)
{
int i, l;
double M, MS;

M=0.;
for(i=0;i<N;i++)
{
MS=0.;
for(l=0;l<S;l++)
{
MS+=((double)(xi[i][mu]==l)-as)*s[i][l];
}
M+=MS;
}
M=M/den;
return M;
}
```

A.6 Codice di integrazione numerica per rete di Potts con latching

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include "nrutil.h"
#define SAFETY 0.9
#define PGROW -0.2
#define PSHRNK -0.25
#define ERRCON 1.89e-4
#define MAXSTP 10000
#define TINY 1.0e-30

void derivs(double,double [],double []);

void odeint(double [],int,double,double,double,double,double,int *,int *,
void (*)(double, double [], double []),void (*)(double [],double [],int,
double *,double,double,double [],double *,double *,void (*)(double,double [],double [])));

void rkck(double [],double [],int,double,double,double [],double [],
void (*)(double,double [],double []));

void rkqs(double [],double [],int,double *,double, double,double [],
double *,double *,void (*)(double,double [],double []));

double a,U,g,b2,b1,b3,N,p,S,tau,T,beta,x0,C1,C2,w;
double as,as1,asS,na12,n2a,solito,ws;
int kmax, kount, nvar;
double *xp, **yp, dxsav;

FILE *cost;
FILE *dinamica;

int main()
{
int i;
int nok;
int nbad;

nvar=28;
kmax=2242;
dxsav=0.00001;

xp=vector(1,kmax);
yp=matrix(1,nvar,1,kmax);

cost=fopen("/scratch/russo/simulazioni/potts/sconst.dat","r");
```

```
fscanf(cost, "%lf", &N);
fscanf(cost, "%lf", &p);
fscanf(cost, "%lf", &S);
fscanf(cost, "%lf", &a);
fscanf(cost, "%lf", &U);
fscanf(cost, "%lf", &g);
fscanf(cost, "%lf", &tau);
fscanf(cost, "%lf", &T);
fscanf(cost, "%lf", &b2);
fscanf(cost, "%lf", &b1);
fscanf(cost, "%lf", &b3);
fscanf(cost, "%lf", &beta);
fscanf(cost, "%lf", &x0);
fscanf(cost, "%lf", &C1);
fscanf(cost, "%lf", &C2);
fscanf(cost, "%lf", &w);
fclose(cost);

double ystart[28+1]={0.,0.00,-0.0011,0.6136,
0.0001,0.0001,0.0001,1./(S+1.),1./(S+1.),
0.0001,0.0001,0.0001,1./(S+1.),1./(S+1.),
0.0001,0.0001,0.0001,0.0001,1./(S+1.),1./(S+1.),1./(S+1.),
0.0001,0.0001,1./(S+1.),
0.0001,0.0001,0.0001,1./(S+1.),1./(S+1.)};

as=a/S;
as1=1.-as;
asS=-as*(S-1.);
na12=N*a-C1-C2;
n2a=N*(1.-2.*a)+C1+C2;
solito=1./(a*N*(1.-as));
ws=w/S;
tau=tau/N;

odeint(  ystart, nvar, 0.0, T/N,  0.000001,  0.001,  0.0,&nok,&nbad,derivs,rkqs);


dinamica=fopen("tutti.dat","w");
for(i=1;i<kmax;i++)
{
fprintf(dinamica, "%lf %lf %lf %lf %lf %lf %lf %lf %lf
%lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf
%lf %lf %lf %lf %lf %lf %lf %lf %lf\n",
xp[i],yp[1][i],yp[2][i],yp[3][i],yp[4][i],yp[5][i],yp[6][i],yp[7][i],yp[8][i],
yp[9][i],yp[10][i],yp[11][i],yp[12][i],yp[13][i],yp[14][i],yp[15][i],yp[16][i],
yp[17][i],yp[18][i],yp[19][i],yp[20][i],yp[21][i],yp[22][i],yp[23][i],yp[24][i],
yp[25][i],yp[26][i],yp[27][i],yp[28][i]);
}
fclose(dinamica);
}
```

```
void odeint(double ystart[], int nvar, double x1, double x2, double eps,
double h1,double hmin, int *nok,int *nbad,void (*derivs)(double, double [], double []),
void (*rkqs)(double [], double [], int, double *, double, double, double [],
double *, double *, void (*)(double, double [], double [])))
{
int nstp,i;
double xsav,x,hnext,hdid,h;
double *yscal,*y,*dydx;


yscal=vector(1,nvar);
y=vector(1,nvar);
dydx=vector(1,nvar);

x=x1;
h=SIGN(h1,x2-x1);
*nok =0;
*nbad=0;
kount = 0;

for (i=1;i<=nvar;i++)
y[i]=ystart[i];
if (kmax > 0)
xsav=x-dxsav*2.0;
for (nstp=1;nstp<=MAXSTP;nstp++)
{
(*derivs)(x,y,dydx);
for (i=1;i<=nvar;i++)
yscal[i]=fabs(y[i])+fabs(dydx[i]*h)+TINY;
if (kmax > 0 && kount < kmax-1 && fabs(x-xsav) > fabs(dxsav))
{
xp[++kount]=x;
for (i=1;i<=nvar;i++)
yp[i][kount]=y[i];
xsav=x;
}
if ((x+h-x2)*(x+h-x1) > 0.0)
h=x2-x;
(*rkqs)(y,dydx,nvar,&x,h,eps,yscal,&hdid,&hnext,derivs);
if (hdid == h)
++(*nok);
else
++(*nbad);
if ((x-x2)*(x2-x1) >= 0.0)
{
for (i=1;i<=nvar;i++)
ystart[i]=y[i];
if (kmax)
{
xp[++kount]=x;
for (i=1;i<=nvar;i++)
```

```
yp[i][kount]=y[i];
}
free_vector(dydx,1,nvar);
free_vector(y,1,nvar);
free_vector(yscal,1,nvar);
return;
}
if (fabs(hnext) <= hmin)
nrerror("Step size too small in odeint");
h=hnext;
}
nrerror("Too many steps in routine odeint");
}


void rkqs(double y[], double dydx[], int n, double *x, double htry, double eps,
double yscal[], double *hdid,double *hnext, void (*derivs)(double, double [], double []))
{
int i;
double errmax,h,htemp,xnew,*yerr,*ytemp;

yerr=vector(1,n);
ytemp=vector(1,n);
h=htry;
for (;;)
{
rkck(y,dydx,n,*x,h,ytemp,yerr,derivs);
errmax=0.0;
for (i=1;i<=n;i++)
{errmax=FMAX(errmax,fabs(yerr[i]/yscal[i]));
}
errmax /= eps;

if (errmax <= 1.0)
break;
htemp=SAFETY*h*pow(errmax,PSHRNK);

h=(h >= 0.0 ? FMAX(htemp,0.1*h) : FMIN(htemp,0.1*h));
xnew=(*x)+h;
if (xnew == *x) nrerror("stepsize underflow in rkqs");
}
if (errmax > ERRCON) *hnext=SAFETY*h*pow(errmax,PGROW);
else *hnext=5.0*h;
*x += (*hdid=h);
for (i=1;i<=n;i++) y[i]=ytemp[i];
free_vector(ytemp,1,n);
free_vector(yerr,1,n);
}


void rkck(double y[], double dydx[], int n, double x, double h, double yout[],
```

```
double yerr[], void (*derivs)(double, double [], double []))
{
int i;
static double a2=0.2,a3=0.3,a4=0.6,a5=1.0,a6=0.875,b21=0.2,
b31=3.0/40.0,b32=9.0/40.0,b41=0.3,b42 = -0.9,b43=1.2,
b51 = -11.0/54.0, b52=2.5,b53 = -70.0/27.0,b54=35.0/27.0,
b61=1631.0/55296.0,b62=175.0/512.0,b63=575.0/13824.0,
b64=44275.0/110592.0,b65=253.0/4096.0,c1=37.0/378.0,
c3=250.0/621.0,c4=125.0/594.0,c6=512.0/1771.0,
dc5 = -277.00/14336.0;
double dc1=c1-2825.0/27648.0,dc3=c3-18575.0/48384.0,
dc4=c4-13525.0/55296.0,dc6=c6-0.25;
double *ak2,*ak3,*ak4,*ak5,*ak6,*ytemp;
ak2=vector(1,n);
ak3=vector(1,n);
ak4=vector(1,n);
ak5=vector(1,n);
ak6=vector(1,n);
ytemp=vector(1,n);
for (i=1;i<=n;i++)
ytemp[i]=y[i]+b21*h*dydx[i];
(*derivs)(x+a2*h,ytemp,ak2);
for (i=1;i<=n;i++)
ytemp[i]=y[i]+h*(b31*dydx[i]+b32*ak2[i]);
(*derivs)(x+a3*h,ytemp,ak3);
for (i=1;i<=n;i++)
ytemp[i]=y[i]+h*(b41*dydx[i]+b42*ak2[i]+b43*ak3[i]);
(*derivs)(x+a4*h,ytemp,ak4);
for (i=1;i<=n;i++)
ytemp[i]=y[i]+h*(b51*dydx[i]+b52*ak2[i]+b53*ak3[i]+b54*ak4[i]);
(*derivs)(x+a5*h,ytemp,ak5);
for (i=1;i<=n;i++)
ytemp[i]=y[i]+h*(b61*dydx[i]+b62*ak2[i]+b63*ak3[i]+b64*ak4[i]+b65*ak5[i]);
(*derivs)(x+a6*h,ytemp,ak6);
for (i=1;i<=n;i++)
yout[i]=y[i]+h*(c1*dydx[i]+c3*ak3[i]+c4*ak4[i]+c6*ak6[i]);
for (i=1;i<=n;i++)
yerr[i]=h*(dc1*dydx[i]+dc3*ak3[i]+dc4*ak4[i]+dc5*ak5[i]+dc6*ak6[i]);

free_vector(ytemp,1,n);
free_vector(ak6,1,n);
free_vector(ak5,1,n);
free_vector(ak4,1,n);
free_vector(ak3,1,n);
free_vector(ak2,1,n);
}


void derivs(double x, double y[], double dydx[])
{
double m1xx, m1xy, m1sy, m1xs, m1ss;
```

```
double m2xx, m2xy, m2sy, m2xs, m2ss;
double qxx, qxy, qsy, qxs, qss;
double selfxs, selfxx, selfxy, selfss, selfsy;
double Zxs, Zxx, Zxy, Zss, Zsy;
double Sxss,Sxsx,Sxsk,Sxxs,Sxxx,Sxxk,Sxys,Sxyx,Sxyy,Sxyk,Ssss,Sssk,Ssys,Ssyy,Ssyk;
double hxsx,hxsk,hxxx,hxxk,hxyx,hxyy,hxyk,hssk,hsyy,hsyk;

Zxs=1.+exp(beta*(y[4]-y[5]+U))+(S-1.)*exp(beta*(y[6]-y[5]));
Zxx=1.+exp(beta*(y[9]-y[10]+U))+(S-1.)*exp(beta*(y[11]-y[10]));
Zxy=1.+exp(beta*(y[14]-y[15]+U))+exp(beta*(y[16]-y[15]))+(S-2.)*exp(beta*(y[17]-y[15]));
Zss=exp(beta*(y[21]-y[22]+U))+S;
Zsy=1.+exp(beta*(y[24]-y[25]+U))+(S-1.)*exp(beta*(y[26]-y[25]));


Sxss=(exp(beta*(y[4]-y[5]+U)))/Zxs;
Sxsx=1./Zxs;
Sxsk=(exp(beta*(y[6]-y[5])))/Zxs;

Sxxs=(exp(beta*(y[9]-y[10]+U)))/Zxx;
Sxxx=1./Zxx;
Sxxk=(exp(beta*(y[11]-y[10])))/Zxx;

Sxys=(exp(beta*(y[14]-y[15]+U)))/Zxy;
Sxyx=1./Zxy;
Sxyy=(exp(beta*(y[16]-y[15])))/Zxy;
Sxyk=(exp(beta*(y[17]-y[15])))/Zxy;

Ssss=(exp(beta*(y[21]-y[22]+U)))/Zss;
Sssk=1./Zss;

Ssys=(exp(beta*(y[24]-y[25]+U)))/Zsy;
Ssyy=1./Zsy;
Ssyk=(exp(beta*(y[26]-y[25])))/Zsy;


/// EMME  1 ///
m1xx=as1*Sxxx+asS*Sxxk;
m1xy=as1*Sxyx-as*Sxyy-as*(S-2.)*Sxyk;
m1sy=-as*Ssyy+asS*Ssyk;
m1xs=as1*Sxsx+asS*Sxsk;
m1ss=-as*S*Sssk;

dydx[1]=-y[1]+solito*(C1*m1xx+C2*m1xy+na12*m1sy+na12*m1xs+n2a*m1ss);

/// EMME  2 ///
m2xx=as1*Sxxx+asS*Sxxk;
m2xy=-as*Sxyx+as1*Sxyy-as*(S-2.)*Sxyk;
m2sy=as1*Ssyy+asS*Ssyk;
m2xs=-as*Sxsx+asS*Sxsk;
m2ss=-as*S*Sssk;
```

```
dydx[2]=-y[2]+solito*(C1*m2xx+C2*m2xy+na12*m2sy+na12*m2xs+n2a*m2ss);

/// Q ///
qxx=Sxxx*Sxxx+(S-1.)*Sxxk*Sxxk-as*(1.-Sxxs)*(1.-Sxxs);
qxy=Sxyx*Sxyx+Sxyy*Sxyy+(S-2.)*Sxyk*Sxyk-as*(1.-Sxys)*(1.-Sxys);
qsy=Ssyy*Ssyy+(S-1.)*Ssyk*Ssyk-as*(1.-Ssys)*(1.-Ssys);
qxs=Sxsx*Sxsx+(S-1.)*Sxsk*Sxsk-as*(1.-Sxss)*(1.-Sxss);
qss=S*Sssk*Sssk-as*(1.-Ssss)*(1.-Ssss);

dydx[3]=-y[3]+solito*(C1*qxx+C2*qxy+na12*qsy+na12*qxs+n2a*qss);

/// /// h /// ///

selfxs=ws*(Sxsx+(S-1.)*Sxsk);
selfxx=ws*(Sxxx+(S-1.)*Sxxk);
selfxy=ws*(Sxyx+Sxyy+(S-2.)*Sxyk);
selfss=w*Sssk;
selfsy=ws*(Ssyy+(S-1.)*Ssyk);

/*
selfxs=ws/Zxs;
selfxs=selfxs*(1.+(S-1.)*exp(beta*(y[6]-y[5])));
selfxx=ws/Zxx;
selfxx=selfxx*(1.+(S-1.)*exp(beta*(y[11]-y[10])));
selfxy=ws/Zxy;
selfxy=selfxy*(1.+exp(beta*(y[16]-y[15]))+(S-2.)*exp(beta*(y[17]-y[15])));
selfss=ws/Zss;
selfsy=ws/Zsy;
selfsy=selfsy*(1.+(S-1.)*exp(beta*(y[26]-y[25])));
*/

hxsx=as1*y[1]-as*y[2]+w*Sxsx-selfxs+(double)(x>x0)*g*exp(-((x-x0)/((double)tau)));
hxsk=-as*(y[1]+y[2])+w*Sxsk-selfxs;

hxxx=as1*(y[1]+y[2])+w*Sxxx-selfxx+(double)(x>x0)*g*exp(-((x-x0)/((double)tau)));
hxxk=-as*(y[1]+y[2])+w*Sxxk-selfxx;

hxyx=as1*y[1]-as*y[2]+w*Sxyx-selfxy+(double)(x>x0)*g*exp(-((x-x0)/((double)tau)));
hxyy=-as*y[1]+as1*y[2]+w*Sxyy-selfxy;
hxyk=-as*(y[1]+y[2])+w*Sxyk-selfxy;

hssk=-as*(y[1]+y[2])+w*Sssk-selfss;

hsyy=-as*y[1]+as1*y[2]+w*Ssyy-selfsy;
hsyk=-as*(y[1]+y[2])+w*Ssyk-selfsy;

/// xs ////
dydx[5]=b1*(hxsx-y[7]-y[5]);
dydx[6]=b1*(hxsk-y[8]-y[6]);

dydx[4]=b3*(1.-Sxss-y[4]);
```

```
dydx[7]=b2*(Sxsx-y[7]);
dydx[8]=b2*(Sxsk-y[8]);

/// xx ////
dydx[10]=b1*(hxxx-y[12]-y[10]);
dydx[11]=b1*(hxxk-y[13]-y[11]);

dydx[9]=b3*(1.-Sxxs-y[9]);

dydx[12]=b2*(Sxxx-y[12]);
dydx[13]=b2*(Sxxk-y[13]);

/// xy ////
dydx[15]=b1*(hxyx-y[18]-y[15]);
dydx[16]=b1*(hxyy-y[19]-y[16]);
dydx[17]=b1*(hxyk-y[20]-y[17]);

dydx[14]=b3*(1.-Sxys-y[14]);

dydx[18]=b2*(Sxyx-y[18]);
dydx[19]=b2*(Sxyy-y[19]);
dydx[20]=b2*(Sxyk-y[20]);

/// ss ////
dydx[22]=b1*(hssk-y[23]-y[22]);

dydx[21]=b3*(1.-Ssss-y[21]);

dydx[23]=b2*(Sssk-y[23]);

/// sy ////
dydx[25]=b1*(hsyy-y[27]-y[25]);
dydx[26]=b1*(hsyk-y[28]-y[26]);

dydx[24]=b3*(1.-Ssys-y[24]);

dydx[27]=b2*(Ssyy-y[27]);
dydx[28]=b2*(Ssyk-y[28]);

}
```

Bibliografia

[1] Abeles M., Vaadia E., Bergman H., Firing patterns of single units in the prefrontal cortex and neural network models, Networks: Computation in neural systems, Vol.1, Issue 1, Jan 1990, pp. 13-25.

[2] Amati D. and Shallice, T. (2007). On the emergence of modern humans. Cognition, 103:358385.

[3] Amit D. J., Modeling brain function, the world of actractor neural networks, Cambridge University Press, 1989.

[4] Barsalou L. W. (2005). Continuity of the conceptual system across species. Trends Cogn Sci., 9(7):30911.

[5] Barton R. (2007). Evolutionary specialization in mammalian cortical structure. Journal of Evolutionary Biology, 20:15041511.

[6] Bolle D., Cools R., Dupont P., and Huyghebaert, J. (1993). Mean-field theory for the Q-state Potts-glass neural network with biased patterns . Journal of Physics A: Mathematical General, 26:549562.

[7] Braitenberg V. and Schuz A. (1991). Anatomy of the Cortex: Statistics and Geometry. Springer Verlag.

[8] Buhmann J., Divko R., Schulten K., Associative memory with high information content, Phys. Rev. A, March 1, 1989, vol. 39, n. 5.

[9] Derrida B., Gardner E. , Zippelius A., An exactly solvable asymmetric neural network model, Europhys. Lett., 4 (2), pp. 167-173 (1987).

[10] Deridda B., Distribution of activities in a diluted neural network, J. Phys. A: Math. Gen. 22 (1989) 2069-2080.

[11] Fuster J. M. (1999). Memory in the Cerebral Cortex: An Empirical Approach to Neural Networks in the Human and Nonhuman Primate. MIT Press.

[12] Garagnani M., Wennekers T., and Pulvermller F. (2007). A neuronal model of the language cortex. Neurocomputing, 70:19141919.

[13] Gardner E., Derrida B., Optimal storage properties of neural network models, J. Phys. A: Gen. 21 (1988) 271-284.

[14] Gil-da-Costa R., Braun A., Lopes M., Hauser M. D., Carson R. E., Herscovitch P., and Martin, A. (2004). Toward an evolutionary perspective on conceptual representation: Speciesspecific calls activate visual and affective processing systems in the macaque. PNAS, 101:17516. 23

[15] Hauser M., Chomsky N., and Fitch W. (2002). The Faculty of Language: What Is It, Who Has It, and How Did It Evolve? Sience, 298:15691579.

[16] Hertz J.,Krogh A., Palmer R. , Introduction to the theory of neural computation, Addison-Wesley Publishing Company, 1994.

[17] Hopfield, J.J., Neural networks and physical systems with emergent collective computational abilities, Proc.Natl. Acad. Sci. USA, vol. 79, pp. 2554-2558, Aprile 1982.

[18] Kandel E. R., Schwartz J. H., Jessell T. M., Principi di neuroscienze, Casa Editrice Ambrosiana, 2003.

[19] Kanter I. (1988). Potts-glass models of neural networks. Phys. Rev. A , 37:27392742.

[20] Kropff E.,Treves A., The storage capacity of Potts models for semantic memory retrieval, Journal of Statistical Mechanics, 2:P08010, 2005.

[21] Kropff E.,Treves A., The complexity of latching transitions in large scale cortical networks, Natural Computing, 10.1007/s11047-006-9019-3, Springer, 2006.

[22] Kropff E.,Treves A., Uninformative memories will prevaile, Effective storage of correlated representations and its consequences, Los Alamos archives, 2007.

[23] McClelland J.L., O'Reilly R. C., McNaughton B. L., Why there are complementary lerning systemy in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. Psychological Review, 1995 Jul, 102(3): 419-457.

[24] Numerical Recipes in C: the art of scientific computing, Cambridge University Press, 1992.

[25] Papp G., Treves A., Network analysis of hippocampal subfield, Jan 10, 2007.

[26] Pillay P. and Manger P. R. (2007). Order-specific quantitative patterns of cortical gyrification. European Journal of Neuroscience, 25:27052712.

[27] Pulvermuller F., A brain perspective on language mechanisms: from discrete neuronal ensembles to serial order, Neurobiology 67 (2002), 85-111.

[28] Russo E., Namboodiri M.K. ,Treves A., Kropff E., Free association transitions in models of cortical latching dynamics, New Journal of Physics (submitted).

[29] Treves A., Come funziona la memoria, Mondadori, 1998.

[30] Treves A., Frontal latching networks: a possible neural basis for infinite recursion, Cognitive Neuropsychology, 2005, 22 (3/4), 276-291.

[31] Tsodyks M. V. and FeigelMan M. V. (1988). The enhanced storage capacity in neural networks with low activity level. Europhysics Letters, 6:101105.