L-systems: Introduzione e modellazione di neuroni

Carlo Francese

22 maggio 2003

Sommario

In questo breve articolo cercherò di mostrare gli l-systems e tre noti algoritmi riguardanti la generazione di modelli reali di neuroni. Infine mostrerò l'applicativo L-Neurons appositivamente sviluppato da Giorgio A. Ascoli e Jeffrey L. Krichmar presso il "Krasnow Institute for Advanced Study - George Mason University" come evoluzioni dei tre algoritmi presentati.

1 L-Systems

1.1 Indroduzione

I sistemi di Lindenmayer, chiamati comunemente L-systems, sono una classe particolare di sistemi dinamici simbolici ove le stringhe ottenute hanno una interpretazione grafica. Vennero introdotti nel 1968 dal biologo olandese Aristid Lindenmayer (1925-1989) quali modelli per la crescita delle piante.

Il processo di riscrittura della stringa avviene in parallelo su ogni simbolo della stringa originaria a differenza di quanto avviene per altri sistemi simbolici come ad esempio le grammatiche di Chomsky ed altre.

Il metodo adottato da Lindenmayer è molto suggestivo e si basa sul "Turtle Graphics" sistema grafico inventato alla fine degli anni '60 da Seymour Papert con il linguaggio LOGO. In questo linguaggio di programmazione,

pensato per i bambini, una tartaruga (a tutti gli effetti un automa) viene guidata sullo schermo per fare disegni. Si parte da un disegno iniziale (che può essere, ad esempio, un segmento od anche una poligonale) il quale viene riprodotto al computer usando delle regole ben precise come ad esempio:

F	Avanzare di un segmento di lunghezza assegnata
f	Avanzare di un segmento di lunghezza assegnata ma senza lasciare traccia
+	Ruotare in senso antiorario di un angolo assegnato
-	Ruotare in senso orario di un angolo assegnato

Ad esempio, per disegnare una semplice figura geometrica (la "trina di Koch"), partendo dal un punto A, potremo dare le seguenti istruzioni: vai avanti di un segmento di lunghezza data (attiviamo in B), ruota in senso orario di 120°, vai avanti di un segmento di lunghezza data (arriviamo in C), ruota in senso antiorario di 60°, vai avanti di un segmento di lunghezza data (arriviamo in D), ruota in senso orario di 120° , vai avanti di un segmento di lunghezza data (arriviamo in E). Se sostituiamo queste istruzioni nel nostro linguaggio otterremmo la stringa "F + F - - F + F".

Ovviamente, affinché la procedura sia effettivamente eseguita da un computer, dovremo dare le istruzioni necessarie per eseguira una rotazione e dovremo immettere come dati iniziali un valore per l'angolo (in questo caso 120° e 60°) ed un valore per la lunghezza del segmento.

Quello che dobbiamo fare è quindi trasformare le sequenze di simboli che generiamo in comandi per la nostra tartaruga.

La figura così disegnata ci può introdurre a Koch ed in particolare al triangolo ed alla trina di Koch. Tenenendo presente che la lunghezza della linea elementare da disegnare, in questo sistema si contrae con un fattore di scala pari ad 1/3 per ogni generazione, non è difficile convincersi che possiamo rappresentare la curva di Koch con un sistema simbolico avente:

$$\begin{aligned} \Sigma &: \quad \{F,+,-\} \\ S &: \quad (F) \\ \delta &: \quad F \rightarrow F + F - - F + F \end{aligned}$$

Questo sistema dunque ci permette di generare frattali ma, per "*far crescere alberi*" abbiamo bisogno di un estensione del nostro sistema di Lindenmayer. Per far ciò abbiamo bisogno di due nuovi simboli: "[" e "]".

Quando si incontra "[" la posizione e l'orientamento della tartaruga vengono salvati su di uno stack, mentre il simbolo "]" fa assumere alla tartaruga la posizione e l'orientamento salvati in precedenza e li toglie dallo stack.

Inoltre possiamo usare il simbolo @0.6 per ridurre di scala i rami dell'albero ad ogni passaggio.

Finora le regole di sostituzioni che abbiamo utilizzato sono totalmente deterministiche ma potremmo facilmente esprimere regole stocastiche.

Tornando alla curva di Koch potremmo scrivere:

F→F + F - - F + F con probabilità α

F→F - F + + F - F con probabilità β; $\alpha+\beta=1$

Le curve che otterremmo, sarebbero sempre diverse al lancio del programma ed assomiglierebbero a queste:

L'esatta "autosimiliarietà" viene perduta e quindi diviene molto difficile definire la dimensione Ds (Dimensione di Similitudine).

Il concetto di dimensione di Similitudine è strettamente legato alla proprietà di autosimiglianza. Consideriamo per esempio un segmento, una parte di piano ed un elemento di volume. Possiamo supporre, senza perdità di generalità, che siano di lunghezza L, area A e volume V pari ad 1. Il segmento può essere diviso in N parti, autosimili, ciascuna di lunghezza $k = \mathbf{L}/\mathbf{N}$ e si ha:

$$L = Nk = 1$$

L'Area può essere divisa in N parti k^2 e si ha:

$$A = Nk^2 = 1$$

Il volume può essere diviso in N parti k^3 e si ha:

$$A = Nk^3 = 1$$

In generale vale la formula $Nk^{Ds} = 1$ ovvero, usando i logaritmi:

$$Ds = \log(N)/\log(1/k)$$

Esaminando l'insieme di Cantor ci si accorge che dopo il primo passo (C_1) abbiamo due coppie identiche dell'insieme ciascuna lunga 1/3 dell'originale, quindi, applicando la formula $Ds = \log(2)/\log(3) = 0.6309$. Notiamo che questo risultato non dipende dalla scala infatti se esaminimo il passo successivo (C_2) avremmo 4 copie di lunghezza 1/9 e quindi, di nuovo $D_s = (2 * \log(2))/(2 * \log(3)) = 0.6309$ e in generale al passo n-esimo:

$$Ds = \log(2^n)/\log(3^n) = (n * \log(2))/(n * \log(3)) = 0.6309$$

Per quanto venga mantenuta una regolarità la curva ottenuta assomiglia ad una costa o meglio dire un confine frattale (fractal boundary), ovvero "*una curva che mostra nuove strutture ad ogni scala di ingrandimento*". In questo caso si parla di *autosomiglianza statistica*.

È importante notare che tutte le curve che consideriamo sono autosimili su un numero finito di ingrandimenti, che è comunque sufficientemente grande da far in modo che la geometria frattale possa essere usata per studiarne le caratteristiche.

Data l'impossibilità di definire Ds per questo genere di curve dobbiamo rivedere ancora una volta la nostra definizione di Dimensione.

1.2 Dimensione

Introdurremo ora il concetto di Dimensione di Hausdorff insieme a due metodi comunemente usati per valutarla: *box counting* e *structured walk*.

Una definizione rigorosa della Dimensione di Hausdorff è materia di studio per un corso di Topologia. In questo contesto cercherò di darne un idea ed introdurre il metodo del box counting.

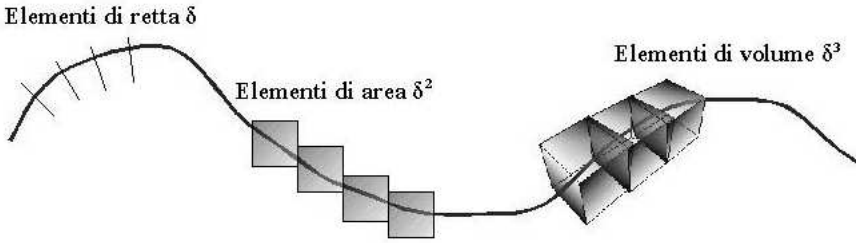
1.2.1 Box Counting

Per prima cosa quando misuriamo la grandezza o l'estensione di un oggetto dobbiamo usare le giuste unità. Per esempio nel caso di una linea, è naturale usare degli elementi di retta di lunghezza δ piuttosto che elementi d'area δ^2 o di volume δ^3.

Nel caso di una linea continua la lunghezza misurata con δfinito è:

$$L_m = \mathbf{N}\delta^1$$

E si ha che $L_m \to L$ per $\delta \to 0$. Se provassimo ad usare elementi d'area o di volume avremmo che il limite per $\delta \to 0$ di $A_m = \mathbf{N}\delta^2$ e di $V_m = \mathbf{N}\delta^3$ cioè l'area ed il volume misurati, risultano uguali a 0.



La giusta scelta per una linea è quindi usare elementi di retta.

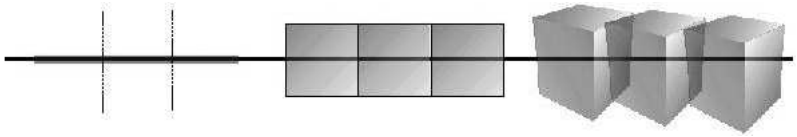
Similmente se provassimo a misurare l'estensione di una superficie usando elementi di retta o di volume otterremmo sempre come limite $\propto$ o 0.

Generalizzando possiamo dire che per esaminare oggetti di dimensione Euclidea $\mathbf{D}e = \mathbf{D}$ avremmo bisogno di elementi estesi $\delta^{\mathbf{D}}$ e che

$$\mathbf{V}^*_m = \mathbf{N}\delta^{\mathbf{D}} \to \mathbf{V}^* \text{ per } \delta \to 0 \ .$$

L'ipervolume $\mathbf{V}^*$ misurato dipende in modo critico da $\mathbf{D}$ ed è $\propto$ se $\mathbf{D} <$ della dimensione dell'oggetto in esame e 0 se maggiore.

La Dimensione di Hausdorff $\mathbf{D_H}$ è definita proprio come il valore critico per il quale il limite passa da 0 a infinito.



Per definire la dimensione di Box Counting, che alcuni chiamano "Capacità" (capacity dimension), procediamo con un esempio:

supponiamo di voler misurare la Dimensione della linea continua nera rappresentata in figura. Per far questo coscruiamo N *scatole* di grandezza R che la contengono. Nel caso di una linea come quella rappresentata possiamo pensare a N segmenti di retta lunghi R,o a N quadrati o cubi di lato R se la linea è rappresentata nel piano o nello spazio.

Chiamiamo *N(R)* il numero minimo di scatole che ci occorrono per ricoprire tutto l'insieme in esame. Facendo diminuire *R*, *N(R)* aumenta.

La Dimensione di Box Counting D_b è definita come quel numero che soddisfa la relazione:

$$N(R) = \lim_{R\to 0} kR^{-D_b}$$

ove k è una costante di proporzionalità corrispondente in questo caso alla lunghezza L del segmento.

Passando ai logaritmi:

$$D_b = \lim_{R\to 0} \frac{-\log N(R)}{\log R} + \frac{\log k}{\log R}$$

Per R abbastanza piccolo la seconda frazione tende a 0 e quindi:

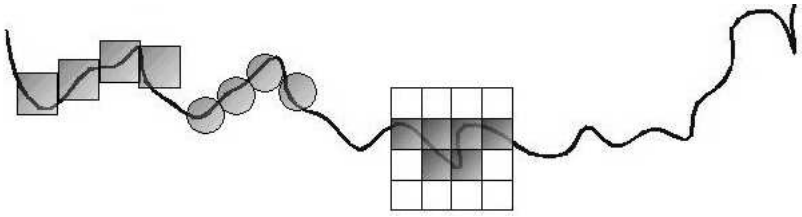
$$D_b = -\lim_{R\to 0} \frac{\log N(R)}{\log R}$$

Quale prima applicazione pensiamo ad un singolo punto. Per ricoprirlo basta una sola scatola quindi $\mathbf{D}_b = 0$.

Per la retta lunga L da cui siamo partiti otteniamo:

$$D_b = -\lim_{R\to 0} \frac{\log(\frac{L}{R})}{\log(R)} = -\lim_{R\to 0} \frac{\log(L)-\log(R)}{\log(R)} = 1$$

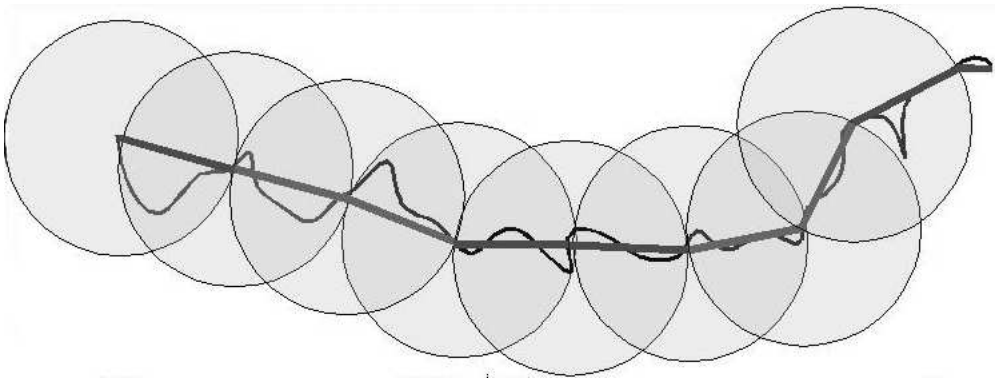
Per misurare $\mathbf{D}_b$ di una curva irregolare come quella rappresentata:



la si ricopre con insiemi regolare di misura R decrescente e si cerca la retta che meglio approssima l'andamento del grafico.

1.2.2 Structured Walk

Un'altro metodo usato per misurare la dimensione frattale di un insieme è chiamato Structured Walk.

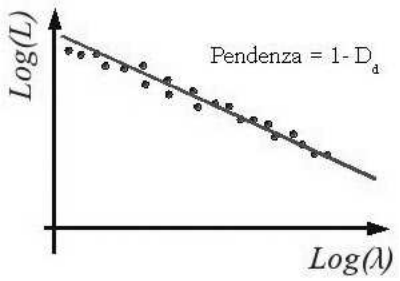


I passi da seguire sono i seguenti:

1. Scegliere un raggio λ.
2. Tracciare un cerchio centrato all'inizio della curva, il primo punto in cui il cerchio incontra la curva sarà il centro per il centro successivo.
3. Procedere in questo modo fino alla fine della curva.

Alla fine si avranno disegnato N cerchi misurando così la lunghezza della curva come $\mathbf{L} \approx \mathbf{N}\lambda$.

Ripetendo il processo con diversi valori di λ e graficando $\log(\mathbf{L})$ vs $\log(\lambda)$ possiamo ottenere, dalla pendenza della retta un'altra misura della dimensione frattale, chiamata in inglese "Divider Dimension". Venne trovata da L.F. Richardson analizzando le lunghezze dei confini fra stati. Il grafico è noto come *Richardson plot*.



2 Algoritmi fondamentali per la modellazione di neuroni

2.1 Introduzione

Negli ultimi dieci anni, la crescita cellulare di molte classi di neuroni è stata simulata grazie a modelli computerizzati estremamente accurati in termini di dettagli fisiologici (tipi e densità dei canali ionici, proprietà delle membrana dei dendriti, ecc.).

La morfologia dei dendriti purtroppo o non tiene conto dei dati sperimentali (modelli comportamentali anatomaticamente precisi) oppure è semplificata con "grossolane" approssimazioni o addirittura nessuna delle due e purtroppo raramente giunge a modelli biologicamente plausibili.

Questa mancanza di un modello neuroanatomico è sorprendente di fronte ad una sempre maggior approvazione tra i neuroscienziati sul fatto che la morfologia dei dentridi giochi un ruolo importante nell'integrazione neuronale e sui recenti progressi di computer grafica e nelle applicazione di modellazione 3D.

Un tentativo di risposta a questa nuova esigenza ci viene fornito da Giorgio A. Ascoli e Jeffrey L. Krichmar, i quali hanno sviluppato L-Neuron (LN), un software per la generazione e lo studio di modelli di neuroni anatomicamente plausibili.

In letteratura scientifica si trovano una serie di regole anatomiche correlate con i parametri morfologici locali (ad esempio il diametro e la lunghezza dei rami) che hanno provato di essere potenti e parsimoniosi descrittori degli aspetti specifici della topologia dei dentridi. LN integra queste regole ormai riconoscute con dei vincoli geometrici globali. L'implementazione di LN adotta un approccio molto simile a quello degli L-systems, il noto formalismo matematico particolarmente utile per descrivere i rami delle piante, degli alberi, dei frattali, e di altre strutture ricorsive.

2.2 Approcci correnti

La morfologia dei dendriti può essere caratterizzata quantitativamente a più livelli. Tipicamente in un file "neuro-anatomico" (es. Neurolucida) un singolo dendrite viene descritto come una serie di scompartimenti cilindrici (i rami). Ogni ramo è rappresentato nel file come una linea contenente un tag numerico, coordinate spaziali Cartesiane (x, y e z), un diametro ed infine un tag con riferimento al "padre". Questo tipo di rappresentazione "Cartesiana" rappresenta una descrizione accurata della morfologia dei dendriti ma non è compatta e supporta poche informazioni "intuitive" per l'utente.

Diversamente in una classica analisi neuroanatomica i dendriti sono sono creati sulla base di distribuzioni statistiche di parametri geometrici e topologici come ad esempio le biforcazioni asimmetriche. Questo tipo di descrizione è più accessibile ai neuroscienziati ma diversamente dal formato "Cartesiano", contiene soltanto informazioni sull'anatomia comune di un gruppo di dendriti e quindi non è adatto a fornire un modello completo dettagliato.

2.2.1 Modelli di Hillman e Tamori

Un altro tipo di descrizione fu proposta nel 1979 da D.E. Hillman il quale affermò che un piccolo insieme di parametri, da lui chiamati "fondamentali", potrebbero essere sufficienti a descrivere completamente e precisamente un intero dentride neuronale. Questo approccio prende vantaggio da una serie di correlazioni locali, come la "power rule" di Rall, legando il diametro di due rami figlio al diametro della biforcazione del padre.

Nella descrizione di Hillman un dentride dovrebbe partire con un "diametro iniziale" dello stelo. A questo punto lo stelo dovrebbe allungarsi per una certa "lenght" (lunghezza) e dovrebbe restringersi in accordo ad un "taper rate" (tasso di rimpicciolimento). Se il diametro del ramo era più grande di un certo "threshold" (soglia), il ramo dovrebbe biforcarsi generando due nuovi rami il cui diametro è completamente determinato dalla "power rule" di Rall a dal "ratio" (rapporto) del diametro del figlio. Il procedimento automaticamente continua considerando i due nuovi rami come due nuovi steli. Se, iterando il procedimento, il diametro scende sotto una certa soglia ("thre-

shold") il ramo crescerà per un ulteriore "terminal lenght" (lunghezza finale) per poi terminare.

L'importanza dell'approcio di Hillman sta nel fatto che questa descrizione può essere facilmente implementata in un algoritmo auto-consistente. Hillman suggerisce che l'angolo di biforcazione dovrebbe costituire un parametro fondamentale aggiuntivo il quale dev'essere ricavato da dati sperimentali.

Nel 1993 Y. Tamori modifica questo algoritmo introducendo il concetto di "effective volume" (volume effettivo) per calcolare l'angolo di biforcazione dagli altri parametri fondamentali.

2.2.2 Modello di Burke

Un altro algoritmo ricorsivo nasce nel 1993 grazie a Burke ed ai suoi collaboratori, i quali implicitamente ricavano la lunghezza di un ramo nella probabilità di allungarsi o biforcarsi e sostituiscono la "power rule" di Rall con la distribuzione sperimentale del diametro dei rami figlio.

2.2.3 Tabella comparativa

Algoritmo	Calcola Diametro	Misura Diametro	Calcola Angolo	Misura Angolo
Hillman	X			X
Tamori	X		X	
Burke		X		X

2.3 Modello di Ascoli/Krichmar

L-Neurons (LN) implementa tutti i sopra citati algoritmi aggiungendovi due parametri angolari per renderizzare in tre dimensioni le ramificazioni: "elevation" (altitudine) e "azimuth". Ed inoltre aggiunge un termine moltiplicativo alla "power rule" di Rall, che potremmo definire come un "nuovo parametro fondamentale" riutilizzando la terminologia adottata da Hillman.

Dunque possiamo dire, che l'algoritmo di Burke descrive più accurata-

mente il processo di allungamento del ramo rispetto all'algoritmo di Hillman/Tamori. Difatti nella descrizione di Hillman/Tamori i rami sono approssimati tramite dei cilindri diritti tra due biforcazioni. Di conseguenza, nei neuroni virtuali generati tramite questo algoritmo, la lunghezza del cammino dei dendriti è solitamente troppo corto oppure la grandezza totale dell'albero è troppo grande. LN corregge questo problema introducendo un parametro di "segmentazione" il quale regola il numero di "pezzi" che costituiscono un ramo tra due punti di biforcazione e quando il loro cammino è più diritto o tortuoso.

Infine LN aggiunge un ulteriore parametro (opzionale) esterno direzionale: "bias" (inclinazione) detto anche "tropism". Il "tropism" è usato negli L-systems per simulare l'effetto della gravità o del vento nella crescita delle piante. In LN, diversamente, il "tropism" è utilizzato per simulare l'effetto del fattore "neurotrofia" e può essere sia direzionale (lungo un certo vettore) oppure centrifugo.