UNIMORE-FIM INFORMATICA

Apprendimento ed Evoluzione in Sistemi Artificiali

Appunti delle lezioni

Lorenzo Balugani (@EvilBalu)

Quarto Semestre

INTRODUZIONE

La materia studia il modo in cui la natura supera determinati ostacoli, e applica queste soluzioni a sistemi artificiali, e esploreremo l'informatica in quanto manipolazione di informazione. Difatti, con il termine informatica si indica l'insieme dei processi e delle tecnologie che rendono possibile la raccolta, la creazione, l'immagazzinamento e la diffusione dell'informazione, dove questa è l'atto di informare o venir informati (comunicazione della conoscenza). La natura riesce anche a trasmettere e elaborare informazioni (anche incoerenti tra di loro): ad esempio il sistema solare calcola la posizione dei pianeti, le proteine in soluzione calcolano la struttura 3D con la minima energia libera e le creature evolvono (vengono usate informazioni sulle specie esistenti e sul loro successo per introdurre modifiche).

Inizialmente, un programma era visto come una sottospecie di macchina, dalla progettazione e dal controllo dettagliato ciascuna dei cui elementi hanno un determinato ruolo: esempio di macchina è la macchina banale (von Foerster), che svolge una o al più poche funzioni diverse (il computer non è una macchina banale, in quanto ha uno stato interno sul quale può ragionare prima di dare un output). Il software tuttavia non è una macchina, in quanto in ambito ingegneristico non ci sono metodi come il debugging (il che implica che l'approccio ingegneristico ha debolezze), anche se il software ha limiti ben definiti. La logica non è sempre in grado di gestire incoerenze, mentre reti neurali possono farlo: queste non possono essere "programmate", ma possono venire istruite in un determinato modo. I sistemi informatici classici sono quindi basati sulla logica (qualche forma di), mentre quelli naturali sono basati sull'evoluzione dinamica (non evolvono a regole fisse).

ORIGINI DELL'INFORMATICA

L'informatica ha origine ben prima della nascita dei calcolatori economici, ad esempio si possono vedere le antiche biblioteche e archivi come le prime basi di dati e gli abachi o i regoli calcolatori come prime macchine di calcolo. Le prime macchine da calcolo meccaniche giungono nel 1600, con la pascalina e versione di Leibniz e nel 900 arrivano le calcolatrici elettriche ed elettroniche, inoltre nei primi anni del 1900 nascono le prime macchine in grado di memorizzare dati. Ma non solo le macchine ad essere in grado di eseguire i conti, in quanto ci sono anche gli umani (il termine computer è nato per descrivere l'attività di esseri umani che fanno dei conti) che possono dividere operazioni complesse tra di loro, risultando in operazioni semplici affidate a molteplici individui.
Nei secoli, in particolare, nasce il progetto di "meccanizzare" il pensiero (sempre per opera di Leibniz, che sosteneva che le guerre nascevano a causa di incomprensioni tra nazioni e di calcolo errato delle conseguenze) nel quale prenderà parte anche Boole, che rendendosi conto di quanto sia complicato propone di meccanizzare il pensiero mediante una successione di proposizioni, riducendo tutto ad una logica basata sul vero e sul falso e sui connettivi logici AND, OR e NOT. Le nozioni di Boole rimarranno dormienti a lungo, fino a Babbage con il suo difference engine (solo teorica, realizzata poi da un tipografo) e l'analytical engine (una macchina universale, mai realizzata a causa dell'eccessiva complessità), per la quale verranno scritti programmi teorici da parte di Ada Lovelace. Da allora ai giorni nostri, ci sono state le ricerche sulla formalizzazione della matematica, gli studi di Turing e di Von Neumann (che ha l'idea di separare la memoria dal calcolo), che portano alla nascita del calcolatore moderno. Con il calcolatore moderno, vengono rispolverate tutta una serie di teorie che finalmente possono essere applicate in un nuovo modo, come ad esempio quella algoritmica e si definiscono linguaggi per programmare queste macchine: prima con interruttori, poi con il linguaggio macchina, con l'assembler e infine con i linguaggi ad alto livello (l'evoluzione da linguaggio macchina ai linguaggi di terza generazione comporta un aumento di livello di astrazione e di autonomia dai dettagli dell'hw), e delle tecniche di programmazione moderni (subroutine, funzioni, moduli, oggetti…).
Un programma viene visto come un insieme di agenti (parti che hanno scopo) sw che vanno coordinati tra loro, ancora basati sulla logica (controllo). In questo corso si vedrà una logica diversa, quella dell'autocontrollo, con sistemi che apprendono e moduli i le cui funzioni non sono stati predefinite e possono cambiare, migliorando le loro interazioni con l'ambiente (sistema dinamico).

COME CALCOLA LA NATURA

Ciò che accomuna un corpo che cade, l'universo intero ed una cellula è il fatto che siano tutti quanti sistemi dinamici, ovvero cambiano nel tempo, e i cambiamenti a cui sono soggetti non sono casuali ma sono dovuti a delle leggi. Un esempio di questo è lo studio delle stelle, il cui moto è regolare e prevedibile: Aristotele divide il mondo tra il mondo sublunare e i cieli, il primo imprevedibile e il secondo prevedibile. Con il tempo, però, si scopre che i pianeti non hanno un moto regolare (infatti questo è un moto retrogado), e si deve accettare il fatto che la Terra non è al centro del sistema solare (inizialmente ci si mette una pezza con il sistema tolemaico, che mantiene la terra al centro dell'universo, viene poi proposto il modello eliocentrico di Copernico che passerà con molta fatica). Galileo, nei suoi sforzi di mostrare la veridicità del sistema di Copernico, mostra come sia possibile descrivere in termini matematici anche i moti terrestri, e Newton mostra che la legge che regola la caduta dei gravi è la stessa che causa l'orbita della Luna: i successi della meccanica fanno di questa disciplina il paradigma per le altre scienze, con le leggi della dinamica che vengono estese ai mezzi continui e applicati a fenomeni elettrici e magnetici (le leggi vengono usate per prevedere la realtà, ad esempio Nettuno viene scoperto puntando i telescopi in un punto calcolato con le leggi). La meccanica mostra quindi nei secoli la sua potenza predittiva e esplicativa, le cui equazioni (ad esempio, F=ma) sono sia deterministiche (conoscendo la posizione e la velocità iniziale di un corpo il suo futuro è ad esempio determinato, e la sua posizione è teoricamente nota ad ogni istante) che reversibili (ad esempio, l'inversione temporale nell'equazione di Newton non rompe l'equazione). Se vi sono tanti corpi che interagiscono muovendosi il calcolo è complicato, ma in assenza di eventi esterni il futuro è determinato (la creatura in grado di prevedere tutte le interazioni delle particelle dell'universo prende il nome di demone di Laplace). Se si osservano solo alcune variabili di un sistema, può capitare di vedere comportamenti apparentemente casuali (moto browniano), ma per il determinismo sono le variabili che non stiamo osservando a causare questi movimenti casuali. Il "rumore" che influenza la particella browniana descrive in modo sintetico l'effetto delle variabili ignorate, e può essere trattato in modo matematicamente rigoroso (non si ottiene l'equazione del moto, ma si ottiene la proprietà delle famiglie di traiettorie). Per quanto riguarda la reversibilità, mentre le leggi danno l'idea di un modo reversibile, nella realtà le cose non stanno così, e il tempo ha una direzione ben precisa: perché se le leggi sono reversibili i fenomeni che si osservano non lo sono? Come sono le leggi che regolano questi fenomeni?

URNA DI EHRENFEST

Immaginiamo di avere due urne, che contengono N palline, e ad ogni istante estraggo una pallina a caso da destra a sinistra (o viceversa, quindi è un'operazione reversibile per la meccanica microscopica del sistema). La descrizione microscopica è equivalente ad elencare la posizione di ogni pallina, quella macroscopica si limita invece a dire quante palline sono a sinistra (e di conseguenza, sapendo quante ne abbiamo in totale, quante ne abbiamo a destra e la differenza tra le due urne $n = N1 - N2$). Se partiamo con un'urna piena e una vuota, inizialmente N1=N, N2=0, n=N spostiamo una pallina da 1 a 2, N1=N-1, N2=1, n=N-1. Andando avanti, può succedere che venga scelta la pallina inserita per ultima è bassa, e quindi ne viene scelta una da 1. Procedendo, ci si rende conto di come il tutto tende a equalizzare i valori nelle due urne, con il numero di palline che tende ad essere N/2 per entrambe le urne con fluttuazioni casuali. Una dinamica reversibile può dare origine ad un moto irreversibile per le variazioni microscopiche, e più le fluttuazioni sono piccole e trascurabili, si ha un'equazione approssimativamente deterministica (usando molte palline, le fluttuazioni spariscono e ottengo la formula $n(t) = Ne^{-kt}$). Osserviamo 3 livelli: il microscopico (dove osserviamo in questo caso una dinamica reversibile e stocastica), il mesoscopico (si guardano solo i numeri con una risoluzione che ci fa vedere le fluttuazioni e osserviamo una dinamica irreversibile e stocastica) e il macroscopico (dinamica irreversibile e deterministica, spariscono le fluttuazioni).
Lo stesso sistema può esser quindi diverso a diversi livelli, e l'irreversibilità dipende dal livello che si sta osservando. L'irreversibilità dell'urna macroscopica è legata al fatto di aver posto una descrizione probabilistica (sebbene possibile, è improbabile che si torni indietro), anche se sappiamo che anche la dinamica microscopica è retta da leggi con componente stocastica.

SISTEMI DINAMICI

I parametri nei sistemi dinamici possono causare cambiamenti quantitativi ma anche qualitativi (ad esempio, in $y = e^{a-1}x$, se a è tra 0 e 1 la funzione è decrescente, se è 1 è costante e se è >1 è crescente).

Definizione: Attrattore
Tornando all'esempio dell'urna, la variabile n=N1-N2 può avere valori tra -N e N, e a tempi lunghi tende a 0 (trascurando fluttuazioni casuali), e lo stato n=0 è detto attrattore, in quanto attrae a sé diverse condizioni iniziali. Cambiando gli attrattori, si riesce a controllare il sistema in quanto si cambia lo schema finale. Nei mondi non lineari ci possono essere più attrattori.

Definizione: Spazio delle fasi
L'insieme dei punti che raccontano la traiettoria del proprio sistema tende da una condizione iniziale ad una condizione finale (l'attrattore). Lo spazio in cui il sistema si muove è detto spazio delle fasi, e ogni suo asse rappresenta una delle variabili. Lo spazio delle fasi può essere partizionato in zone (bacini di attrazione, ovvero l'insieme delle condizioni iniziali che portano ad un determinato attrattore)

I punti fissi della dinamica sono stati che non cambiano nel tempo, e questo li rende importanti. Questi possono essere stabili o instabili, se soggetti ad una piccola perturbazione, anche se i punti fissi instabili non sono fisicamente realizzabili

I sistemi lineari sono sistemi il cui cambiamento avviene in modo lineare, ad esempio $\frac{dx}{dy} = kx$.
Una funzione che ha la stessa forma della sua derivata è l'esponenziale, ad esempio $x(t) = e^{kt}$.
La soluzione generale deriva da quella esponenziale, che è $x(t) = Ae^{kt}$, che è definita per ogni valore di t, e ha il segno di A. Se k è positivo, con t che cresce, x tende ad infinito, mentre se è negativo x tende a 0, e 0 è un punto fisso. Il problema di Cauchy ci chiede di trovare la soluzione di un'equazione differenziale con condizione iniziale $x(0) = x_0$ e vediamo che per la soluzione generale vediamo che $x(0) = A$, quindi $x(t) = x_0e^{kt}$.

E' importante distinguere i sistemi dinamici lineari da quelli non lineari, con i sistemi dinamici lineari che sono quelli in cui le derivate delle variabili dipendono solo da combinazioni lineari delle variabili stesse (ovvero, $\frac{dx}{dt} = A\boldsymbol{x} + \boldsymbol{b} \Rightarrow \frac{dx_1}{dt} = a_{11}\,x_1 + \cdots a_{1N}x_N + b_1, \ldots, \frac{dx_N}{dt} = a_{N1}x_1 + \cdots + a_{NN}x_N + b_N$). Nei sistemi lineari, se x(t) è una soluzione, anche bx(t) lo è, e se anche y(t) lo è allora lo è pure $x(t) \pm y(t)$. In generale, se x e y sono soluzioni, allora ogni loro combinazione lineare $ax(t) + by(t)$ lo è.

Definizione: Principio di sovrapposizione
Prendiamo per esempio un conto in banca, dove due capitali distinti investiti al medesimo tasso producono dopo un certo tempo, due interessi che teoricamente sommati sono uguali all'interesse prodotto da un singolo capitale iniziale pari alla somma dei due capitali distinti (la legge può essere ad esempio $\dot{x}(t) = kx(t)$, dove k è il numero di anni. Il puntino indica $\frac{dx}{dt}$). Questo è vero/realistico? In un mondo in cui non ci sono spese legate al conto, avremmo $x_1(t) = x_1(0)e^{kt}$ e $x_2(t) = x_2(0)e^{kt}$ quindi il totale diventa $x_3(t) = x_1(t) + x_2(t) = e^{kt}\big(x_1(0) + x_2(0)\big)$, quindi sarebbe vero, ma non realistico (il conto ha delle spese, e se il banchiere è bravo offre interessi più alti maggiore è la cifra offerta, e quindi non abbiamo un mondo lineare).

MODELLO LINEARE DI DUE POPOLAZIONI INTERAGENTI

L'ipotesi più semplice è che le due popolazioni interagiscono in modi esprimibili in forme lineari (irrealistico). Le due popolazioni sono $P_1 = n_1(t), P_2 = n_1(t)$ e l'interazione è $n_i = \alpha n_i(t) + \beta n_j(t)$. Quindi si ha
$\begin{cases} n_1(t) = k_{11}n_1(t) + k_{12}n_2(t) \\ n_2(t) = k_{21}n_1(t) + k_{22}n_2(t) \end{cases} \Rightarrow \begin{pmatrix} n_1(t) \\ n_2(t) \end{pmatrix} = \begin{pmatrix} k_{11}\ k_{12} \\ k_{21}\ k_{22} \end{pmatrix} * \begin{pmatrix} n_1(t) \\ n_2(t) \end{pmatrix}, se\ k_{ij} > 0$ apporto costruttivo (se negativo, distruttivo), k_{11}, k_{22} azione delle popolazione su di loro stesse, k_{12}, k_{21} azione della popolazione i sulla j e viceversa (se maggiori di 0, cooperazione, se minori di 0, competizione, se una è maggiore di 0 e l'altra è minore allora si è in

una situazione di parassitismo). I sistemi tendono ad allontanarsi dai punti di equilibrio instabile, e vanno verso uno stabile, ed è quindi il caso di capire quali sono tali punti in questo esempio. Dato che i punti di equilibrio sono punti in cui il sistema è fermo, la derivata deve essere 0. Il complesso sistema di equazioni differenziali diventa quindi algebrico: $\begin{cases} 0 = k_{11}n_1 + k_{12}n_2 \\ 0 = k_{21}n_1 + k_{22}n_2 \end{cases}$. Un possibile equilibrio è quello in cui entrambe le popolazioni siano morte, e la soluzione viene trovata risolvendo il sistema lineare di due funzioni incognite, e questa soluzione è data dalla combinazione lineare di due esponenziali $\begin{cases} n_1 = \alpha_1 e^{\beta_1 t} + \alpha_2 e^{\beta_2 t} \\ n_2 = \alpha_3 e^{\beta_1 t} + \alpha_3 e^{\beta_2 t} \end{cases}$ e si può trasformare in $\beta^2 - (k_{11} + k_{22})\beta + (k_{11}k_{22} - k_{12}k_{21}) = 0$, con i beta che possono essere reali crescenti (nodo instabile, se lo sono entrambi crescita illimitata, attrattore), reali decrescenti (nodo stabile, se lo sono entrambi morte, attrattore) o complessi (spirali stabili che tendono verso il punto di accumulo oppure verso l'infinito a seconda che la parte reale sia negativa o meno, attrattore. Se la parte reale è uguale a 0, il raggio non cambia mai ed è un'ellisse). Se uno dei beta è positivo e l'altro è negativo, si ha un punto a sella (attrattivo in una direzione, repulsivo nell'altra, attrattore). Nel caso dell'ellisse, non esiste un attrattore (che è un concetto valido per sistemi dissipativi, a differenza delle traiettorie che van bene anche in quelli conservativi). Un ulteriore esempio (solo in sistemi non lineari) è il ciclo limite (dove il sistema viene ribilanciato artificialmente, bloccato ad un certo raggio dell'ellisse, finché l'aiuto esterno smette di agire, può essere un attrattore) e nel caso in cui il moto sia molto simile a quello casuale si parla di sistema caotico (dipendenza sensibile delle condizioni iniziali, servono almeno 3 variabili nel caso di sistemi continui nel tempo, attrattore).

SISTEMI UNIDIMENSIONALI DISCRETI NEL TEMPO

La distinzione fatta tra lineare e non lineare viene fatta anche ai sistemi discreti nel tempo, e ci limiteremo al momento a sistemi unidmensionali discreti nel tempo del primo ordine $x_{t+1} = f(x_t)$ e con i sistemi lineari che sono quelli in cui, con a e b costanti, si ha $f(x_t) = a + bx_t$ (ad esempio, $x_{t+1} = kx_t$, quindi $x_1 = kx_0, x_2 = kx_1 = k^2x_0 \rightarrow x_t = k^t x_0$ che risolto in esponenziale si trova $x_0 e^{t*\ln(k)}$). Spesso è difficile risolvere l'equazione per sistemi non lineari, ma può esser facile determinare il loro comportamento al limite $t \rightarrow \infty$, con gli eventuali punti fissi che si trovano risolvendo l'equazione $x = f(x)$.

ESEMPIO DI CRESCITA DELLA POPOLAZIONE

Il numero di individui all'istante t+1 è il numero di individui nell'istante t + nati + morti, ovvero $x_{t+1} = x_t + bx_t - dx_t) = (1 + b - d)\, x_t$, che è un sistema lineare a singolo attrattore (crescita se b>d, declino se b<d, se b=d costante) di tipo esponenziale, ma questo è inesatto in quanto dovrebbe crescere all'infinito (cosa che è irrealistico). Bisogna ipotizzare che gli individui competano per le risorse, dove il termine negativo cresce in modo proporzionale agli incontri tra individui, trasformando l'espressione in $x_{t+1} = (1 + b - d^* x_t)x_t = (1 + b)x_t - d^* x_t^2$, e siamo quindi usciti dal mondo non lineare (l'asterisco indica solo che $d^* \neq d$)), che evolve in $x_{t+1} = Bx_t - Ax_t^2 = Bx_t\left(1 - \frac{A}{B}x_t\right) = (1 + b)x_t * \left(1 - \frac{d^*}{1+b}x_t^2\right)$, e con questo nuovo modello si ha una saturazione del sistema (crescita non più infinita).

Questo modello è chiamato modello di Verhulst in versione discreta $x_{t+1} = Ax_t(1 - x_t)$, ed è un modello fenomenologico, che descrive la crescita limitata di qualcosa. Il grafico, che prende il nome di mappa logistica, ha una caratteristica forma ad S. Si richiede che x sia sempre compreso tra 0 e 1, e questa condizione implica che A sia maggiore uguale di 0. $x * (1 - x)$ è una parabola, che ha il suo massimo nella sua metà, che si annulla in 0 e 1, quindi per imporre $x_{t+1} \leq 1$ è necessario che $A \leq 4$. Cerchiamo i punti fissi, ovvero quando $Ax_t(1 - x_t) = x_t$, e quindi sono intersezioni della retta $y = x$ con la parabola $y = Ax(1 - x)$. Se A è compreso tra 0 e 1, l'unico punto stabile è 0, e questo si fa calcolandosi la derivata di $y = Ax - Ax^2 \rightarrow y' = A - 2Ax$ per ricavarne la pendenza. Se invece A>1, $x = Ax - Ax^2 \rightarrow x' = x(A - 1 - Ax)$ e, cercando per quali valori è uguale a zero si ottengono due valori.

Per osservare le oscillazioni in avvicinamento ad un attrattore, si può usare il metodo della scala.
Si inizia da un certo punto, e si guarda tirando una retta perpendicolare all'asse x per vedere, partendo dalla bisettrice, dove va sulla parabola, e una volta intersecati alla parabola si ripete sull'asse y, per poi ripetere finchè non si giunge all'attrattore.

Al crescere di A la parabola diventa sempre più ripida, e che l'avvicinamento si ha solo se l'oscillazione si smorza, si può dimostrare che un punto fisso x^* di una mappa $x_{t+1} = f(x(t))$ è stabile se in quel punto $\left|\frac{df}{dx}\right| < 1$. Nel nostro caso, la derivata vale $A - 2Ax$ e nel punto $\frac{A-1}{A}$ vale $-A + 2$ il cui modulo supera 1 se A>3, valore oltre il quale il punto non è più stabile. Otteniamo quindi un ciclo limite di periodo 2 (ed è anche attrattore), caratterizzato da $x_{t+2} = x_t$, $x(t+1) = f(x(t)), x(t+2) = f(x(t+1)) = f\left(f(x(t))\right) = f^2(x(t))$ (funzione applicata due volte, non al quadrato). Andando a calcolarmi la derivata di f^2, e ponendola <1, si nota che nemmeno questa è stabile (e che ha 2 attrattori) e, lasciando crescere A, questa diventa ancora più instabile.

Come evidenziato dalla Cascata di Feigenbaum, fino a 3 A ha un singolo punto fisso, mentre superato questo i punti fissi diventano 2, poi 4, poi 8 ecc. La distanza tra una biforcazione e quella successiva è sempre più piccola di quelle precedenti, quindi i punti di biforcazioni sono infiniti: da un'equazione banale, siamo giunti ad un punto in cui il sistema può essere presente in infiniti punti della cascata. Vi sono ad esempio zone in cui si passa da caos ad una zona in cui si oscilla solo in 4 punti, per poi riprendere (ma in modo più compresso, il che vuol dire che queste zone di regolarità immerse nel caos sono più frequenti man mano che si va avanti). Lo stesso sistema, quindi, può essere ordinato o caotico. Esiste un punto di accumulazione $A_\infty = 3{,}5699\ldots$ (superato questo si raggiunge il caos) e punti di raddoppiamento si addensano secondo la regola $\frac{A_n - A_{n-1}}{A_{n+1} - A_n} = 4{,}669\ldots$, valore molto importante nel mondo naturale (ad esempio l'instabilità fluidodinaimca).

Finora abbiamo considerato sistemi di primo ordine, dove compaiono solo derivate dx/dt, e sistemi di derivate di ordine superiore possono essere trasformati in sistemi di primo grado mediante l'aggiunta di variabili.
Consideriamo l'equazione $\frac{dx}{dt} = -ax - bx^3$, e imponiamo $b > 0$. E' un'equazione differenziale, e per definire la storia di un sistema come questo è necessario conoscere la posizione in x(0). Poniamo $\frac{dx}{dt} = 0 \rightarrow -ax - bx^3 = 0, x_1 = 0, x_2, x_3 = \pm\sqrt{-\frac{a}{b}}$ (la prima soluzione esiste sempre, l'altra solo se a<0) e i punti fissi trovati esistono indipendentemente dalla condizione iniziale. Al posto di risolvere l'equazione differenziale, cerchiamo di calcolare il potenziale per capire se questi punti sono fissi.

Definizione: Potenziale
Il potenziale è definito come $\frac{dx}{dt} = f(x) = -\frac{dV}{dx}$, dove $-V$ è primitiva del membro a destra dell'equazione $\frac{dx}{dt} = f(x)$.
Se V cresce con x, dV/dx è positiva, f(x)=-dV/dx è negativa, dx/dt è negativa e quindi x tende a diminuire (se il potenziale è crescente aumentando x, allora la x dell'oggetto tende a scendere nel senso che va a sinistra (se nel quadrante 1)).
Se V cala con x, dV/dx è negativo, f(x)=-dv/dx è positivo, dx/dt è positivo e x tende a crescere (al crescere di x scende il potenziale, e quindi si sposta verso dx).
Il sistema si muove da punti instabili a punti stabili (che è il punto più basso della valle del potenziale).

Tornando al nostro caso, abbiamo quindi $\frac{dx}{dt} = \frac{dV}{dx} \rightarrow V(x) = \frac{a}{2}x^2 + \frac{b}{4}x^4, \frac{dV}{dx} = ax + bx^3 = x(a + bx^2)$ con soluzioni identiche alle precedenti se equiparata a 0 e calcoliamo la derivata seconda $\frac{d^2V}{dx^2} = a + 3bx^2$. Se a>0 l'unica soluzione reale di dV/dx è x=0, ma se a è negativo allora abbiamo anche le altre due soluzioni, con x=0 che diventa un massimo e le altre due nuove soluzioni che diventano due minimi che creano così una doppia buca (e tutto questo per la variazione di a). C'è stato quindi uno scambio di stabilità, che ha cambiato il panorama degli attrattori e che indica una sorta di auto-organizzazione.

AGENTI

Un agente è qualcuno che agisce in nome di qualcun altro (un'agenzia) con il potere di agire su delega, senza però venir controllato dal delegante. Nel mondo reale, ogni agente ha uno scopo ben preciso, sono delegati e sanno come fare il loro lavoro.

Definizione: Agente
Un agente è una unità computazionale che può essere in grado di percepire ed agire nel suo ambiente e ha un comportamento autonomo che almeno parzialmente dipende dalla sua esperienza personale. Da un altro punto di vista, gli agenti sono delle società artificiali (persone). Ogni agente ha regole comportamentali e stati interni, alcuni dei quali sono fissi nella vita dell'agente mentre altri cambiano con l'interazione con l'ambiente o altri agenti. Gli stati degli agenti possono quindi essere costanti oppure variabili.

Gli agenti sono quindi entità computazionali (manipolano informazioni) autonome e collocate in un ambiente (reale o virtuale). Inoltre, possono interagire con altri agenti e questo può comportare l'emergenza di strutture sociali di più alto livello. Estendendo il concetto, un agente è un'entità, fisica o virtuale, che:

- E' in grado di percepire parzialmente il suo ambiente, di cui ha solo una percezione parziale e nel quale è in grado di agire
- E' guidato da una serie di tendenze (obiettivi e funzioni da ottimizzare)
- Può comunicare direttamente con altri agenti
- Possiede abilità e può fornire servizi
- E' in grado di replicarsi
- Possiede un comportamento che tende verso la realizzazione dei suoi obiettivi che, considerando abilità, risorse disponibili e situazione, elabora una strategia

Quando si realizza un agente, sono due i paradigmi che si tengono a mente.
Paradigma debole
Un agente è un sistema autonomo, reattivo, proattivo (comportamento diretto ad obiettivo) e dotato di abilità sociale (comunicazione)
Paradigma forte
Alle proprietà precedenti vengono aggiunte caratteristiche "umane" come conoscenza, senso d'obbligo, intenzioni e scopi.

Spesso alludiamo a sistemi software come se fossero essere senzienti, e questo è chiamato "posizione intenzionale". Può essere comoda, ma anche fuorviante (qual è il reale grado di autonomia di un sistema artificiale?).
Altre caratteristiche degli agenti sono la mobilità, la veridicità (correttezza nella produzione di informazioni), la beneveolenza (correttezza delle azioni) e la razionalità (coerenza tra azioni e obiettivi). Si possono considerare diverse classi di agente:

- Basate sulla logica, la parte decisionale è realizzata mediante deduzioni logiche
- Reattive, la parte decisionale è realizzata tramite forme di mappatura diretta situazione->azione (i sensori sono direttamente legati alle reazioni, ad esempio roomba)
- Belief-Desire-Intention (BDI), la parte decisionale dipende dalle manipolazioni delle strutture dati che rappresentano aspettative, desideri e intenzioni dell'agente
- Layered, la parte decisionale è realizzata attraverso vari livelli, ciascuno dei quali più o meno dotato di capacità cognitive riguardo all'ambiente

I comportamenti, ovvero "la modalità di fare qualcosa", possono essere teleonomico (diretto verso uno scopo esplicito) o riflessivo (regolato sulle proprie percezioni interne). Gli agenti possono essere cognitivi (rappresentazione esplicita del mondo) o reattivi (con rappresentazione subsimbolica del mondo, integrata nelle sue capacità sensorie e motorie).

Gli agenti cognitivi possono reagire basandosi su rappresentazioni del mondo, memorizzare situazioni da analizzare, pianificare il proprio comportamento, prevedere le conseguenze delle proprie azioni e costruire dei piani e ottimizzare il numero dei movimenti. I reattivi possono reagire ad eventi, non possiedono piani prestabiliti e possono fare emergere comportamenti complessi, anche se modellati con regole semplici.

SISTEMI MULTI-AGENTE

Vi sono numerosi esempi di modelli ad agente, anche se le più interessanti sono quelle in cui molti agenti interagiscono in modo concorrente sullo stesso sistema.

Definizione: Sistema multi-agente
In informatica un sistema multi agente è composto da numerosi agenti in grado di instaurare interazioni reciproche. L'interazione può essere sotto forma di scambio di messaggi o attraverso la modifica del loro ambiente comune. Gli agenti possono raggrupparsi in organizzazioni per portare a termine il loro lavoro.

I motivi di usare sistemi multi agente sono diversi. In primis, un solo agente è limitato, in quanto ha limiti nella quantità di conoscenza che può acquisire e processare in un certo tempo, e in secondo luogo molti problemi sono intrinsecamente distribuiti, e serve concentrare più agenti maggiormente densi di informazione (ad esempio, agenti che devono investigare i contorni all'interno dell'immagine). Inoltre, le interazioni possono avvenire tra agenti specializzati, dato che molti problemi richiedono l'interazione tra diverse competenze, e i sistemi multi agente sono simili a organizzazioni reali, il che permette la creazione di sistemi software di supporto al lavoro di organizzazioni umane, e servono quindi a minimizzare la distanza concettuale fra le organizzazioni reali e simulate.

Più in generale, per maneggiare sistemi complessi è più facile pensare in termini di sistemi multi agente dato che è il nostro stesso mondo ad essere formato da continue interazioni autonome esercitate da entità limitate il cui fine dell'interazione è perseguire un obiettivo comune, partendo da scopi personali, e avendone noi esperienza sono facilmente raccontabili tali interazioni. Molti sistemi informatici, sebbene non siano stati pensati nell'ottica multi-agente, ne hanno le caratteristiche: sono fatti da componenti distribuite, dialogano tra loro e sono situati in un ambiente fisico o simulato.

I sistemi multi agenti vengono applicati, ad esempio, in ambito di controllo di processo (agenti embedded), dove l'agente è immerso in un sistema per il controllo di, ad esempio, uno strumento e l'agente comunica con i sui simili per coordinare l'azione (ad esempio, una catena di montaggio). Questi sistemi sono applicati nell'industria, in casa/ufficio (controllo di temperatura) e nelle città, ad esempio per il controllo del traffico. Ogni agente è specializzato nel massimizzare lo sfruttamento di una certa risorsa, interagendo con altri agenti per assicurare un generale bilanciamento (ad esempio, in un sistema di telefonia mobile, reti elettriche o reti di calcolatori).
Altro esempio di impiego, è quello della gestione di gruppi di lavoro collaborativi, nei quali gli agenti possono agire autonomamente o come avatar di esseri umani, in un processo di lavoro distribuito (controllo della produzione di un documento multi-autore, gestione di coalizioni temporanee). Inoltre, i sistemi multi agente vengono impiegati per complesse simulazioni (le simulazioni possono essere analitiche, basate su sistemi di equazioni differenziali (caratterizzate da sistemi dinamici, spazio delle fasi e difficoltà a maneggiare sistemi discreti), o multiagente, che modellano sistemi multi agente reali tramite esecuzioni di componenti autonome ed interagenti e sono semplici da analizzare).
I sistemi multi agente servono anche per l'ottimizzazione multi agente che si basa su agenti che analizzano situazioni locali e interagiscono tra di loro per costruire una soluzione globale.

L'EMERGERE DI ENTITÀ COLLETTIVE

La swarm intelligence è una proprietà di un sistema in cui il comportamento collettivo di agenti non sofisticati che interagiscono localmente con l'ambiente produce l'emergere di pattern funzionali globali nel sistema.
In altre parole, è un gruppo di comportamenti espressi da uno sciame di individui semplici che collettivamente porta avanti un compito apparentemente intelligente e dove nessun individuo può proclamarsi intelligente. La swarm intelligence, considerando quanti sistemi naturali la possiedono, non sembra essere qualcosa di incidentale, ma una proprietà di molti sistemi che l'hanno mantenuta durante l'evoluzione naturale. Comportamenti come quello della swarm intelligence sono detti "emergenti": essi non derivano da una scelta razionale, né da una programmazione e nessun elemento li dirige. Gli individui del sistema non hanno una visione globale, e non hanno la consapevolezza di stare eseguendo qualcosa di intelligente, e questa intelligenza è osservata in comportamenti di alto livello (su grande

scala), emersi spontaneamente nel sistema (negli sciami le interazioni sono locali, ma il comportamento osservato è globalmente organizzato), e negli sciami si parte dal disordine per poi propendere all'ordine (strutture e comportamenti organizzati). I sistemi dotati di swarm intellicence sono spesso estremamente adattabili, in quanto possono auto organizzarsi indipendentemente dalle circostanze in cui si trova l'ambiente e possono cambiare il comportamento in risposta a stimoli ambientali senza perdere il coordinamento distribuito.

Esaminando la swarm intelligence, possiamo individuare 3 elementi fondamentali:

- Individui, che percepiscono l'ambiente locale, reagiscono in base ad un semplice meccanismo reattivo (azione-reazione), influiscono sulle caratteristiche dell'ambiente e, spesso, agiscono in base a qualche meccanismo stocastico
- Interazioni, tipicamente indirette, con gli agenti che possono modificare e "annusare" l'ambiente (stigmergia) e possono vedere il comportamento degli altri agenti (interazione mediata dal comportamento)
- Ambiente, nel quale in cui gli elementi del gruppo vivono e con cui possono interagirvi, che può possedere proprietà proprie e può avere processi attivi in grado di modificare dinamicamente le proprie caratteristiche.

La capacità di auto organizzazione dei sistemi di swarm intelligence è basata sul fatto che gli elementi del gruppo possano ricevere delle retroazioni (feedback) sulle loro azioni. Le retroazioni possono essere:

- Positive (attivazione/rinforzo), il comportamento di un individuo (o i suoi effetti) possono sollecitare altri individui verso comportamenti simili così che molti individui iniziano a comportarsi in modo organizzato
- Negative (controllo/inibizione), il comportamento di un individuo (o i suoi effetti) possono evitare che altri individui convergano sul medesimo comportamento e in questo modo impedire che il sistema raggiunga un equilibrio stabile

E' quindi presente una continua tensione tra controllo e rinforzo. Anche la casualità ha una grande importanza, in quanto spesso la swarm intelligence si basa sulle scelte casuali degli individui che compongono il sistema, con gli individui che hanno una certa probabilità di compiere una determinata azione. Questo consente di raggiungere schemi di auto organizzazione da qualsiasi configurazione iniziale, di sfuggire ai minimi locali e di reagire ai cambiamenti di situazione ambientale. Di conseguenza, il sistema esibisce adattività, con nessuna configurazione che rappresenta quella finale e con quelle alternative che vengono sempre esplorate, mantenendo così la possibilità di reagire a mutazioni ambientali e con una maggiore probabilità di trovare la soluzione migliore. Per raggiungere l'auto organizzazione in modo robusto è necessario che il gruppo sia numeroso (più è numeroso, più le retroazioni sono efficaci), e questo permette di esplorare vaste aree di ricerca, tollerare malfunzionamenti locali e permettere l'adattività (mentre la maggioranza porta avanti il proprio lavoro, gli altri possono cercare soluzioni alternative).

Gli sciami sono gruppi di agenti (quindi sistemi multi agente) con componenti autonome, situati in un ambiente e interagenti, ma sono comunque entità distinte: la filosofia di base è diversa, in quanto il focus è più sul gruppo che sulle sue componenti (che possono essere irrazionali o probabilistiche) e vi è una grande enfasi sul ruolo dell'ambiente, che non è solamente un modo per accedere alle informazioni, ma un mezzo di coordinamento con processi automatici.

La swarm intelligence permette di ottenere comportamenti utili con componenti semplici e distribuite in un ambiente, che interagiscono in modo locale e caratterizzati da organizzazione globale e in grado di esibire flessibilità. Si ottengono sistemi con componenti semplici, in grado di interagire localmente e esibenti comportamenti efficienti e flessibili, al prezzo della certa definizione del risultato (non si può prevedere la configurazione finale) e della dispersione di risorse in attività apparentemente inutili.
In natura non sono solo sciami "stupidi" a esibire la swarm intelligence, ma anche sciami di creature intelligenti, il che implica che il potere delle interazioni supera quello degli individui, qualsiasi sia la ragione delle mosse del singolo o il tipo di interazioni che questo ha con altri, ed è importante anche il ruolo delle scelte stocastiche, con le componenti stupide che sembrano agire casualmente e con le componenti intelligenti che agiscono fuori dal gruppo che possono essere percepite come casuali.

ESEMPI DI SWARM INTELLIGENCE

SINCRONIZZAZIONE

Vi sono specie di lucciole che in particolari condizioni emettono luce in modo sincrono, senza accordi preventivi e senza seguire leader (questo fenomeno è visibile anche altrove, negli applausi, nelle mosse delle cellule responsabili del battito cardiaco, ecc). La casualità dei comportamenti è qui fondamentale, e più il gruppo è numeroso più la sincronizzazione è veloce. Ogni lucciola ha una propria frequenza di illuminazione, e ogni lucciola percepisce il grado di illuminazione del proprio ambiente locale. Le retroazioni sono diverse: se la somma dei lampi delle lucciole circostanti raggiunge una certa soglia (rinforzo) e la lucciola che percepisce la luminosità non è ancora eccitata a sufficienza (controllo) viene riportata l'eccitazione a 0.

AGGREGAZIONE

Esistono specie di amebe che passano buona parte del loro tempo come individui singoli, prendendo decisioni come un collettivo. In particolari condizioni ambientali, ogni cellula emette un particolare ferormone e inizia a annusare l'ambiente alla ricerca della maggiore intensità, con il risultato che vengono a formarsi dei cluster (e più il cluster è grande, maggiore è l'intensità dell'odore, rinforzo positivo) con i ferormoni che si diffondono nell'ambiente, e vanno a far "schierare" le cellule in gruppi diversi, finché non rimane un solo gruppo.
Le applicazioni dell'aggregazione sono negli algoritmi per la formazione di coalizioni (gruppi di agenti possono avere la necessità di riunirsi per risolver problemi che il singolo non sa risolvere) e comunità (coalizioni divise per interesse).

STORMI

Gli stormi sono in grado di mantenere coerenza spaziale e temporale, con bordi ben definiti e in grado di causare eventi. In natura il loro comportamento deriva dagli uccelli, nelle simulazioni dai boids, che simulano i tre comportamenti degli uccelli:

- Separazione: cambiare direzione per evitare sovraffollamenti
- Allineamenti: cambiare direzione per allinearsi alla locale direzione di volo
- Coesione: cambiare direzione per andare verso la posizione media dei compagni di volo

Ogni Boids reagisce solo ai compagni di volo presenti in un limitato vicinato, e questo può essere visto come la zona in cui i compagni di volo possono influenzare il comportamento di ogni boid. Le applicazioni dei boids sono molteplici, e si possono far tendere al rimaner vicino a zone che soddisfano particolari criteri, il che permette ad esempio di evidenziare i contorni in un'immagine.

ANT SORTING (ORDINAMENTO)

I nidi di formiche sono strutture molto organizzate, in quanto mettono in luoghi diversi tipi di oggetto specifico, senza che però nessuna formica ne conosca la mappa, e non ci sono decisioni a priori sulla collocazione dei diversi tipi di oggetto.

Le formiche ordinano gli oggetti seguendo una dettagliata procedura:

- Vagano nel nido a caso
    - Questo permette alle formiche di esplorare tutto il formicaio prima o poi, e se ci sono oggetti essi verranno presi
- Annusano gli oggetti vicini, e memorizzano temporaneamente questi oggetti
    - Più grande è il mucchio di oggetti, minore è la probabilità che la formica sottragga qualcosa, e è più probabile che vi lasci qualcosa
- Se la formica non trasporta nulla, prende un oggetto con una certa probabilità, e questa cala se ha già visto oggetti simili nel passato

- Se la formica sta trasportando qualcosa, lo lascia con una certa probabilità, e questa aumenta se la formica sta vedendo oggetti dello stesso tipo

Alla fine, tutti gli oggetti dello stesso tipo sono radunati nel medesimo mucchio, anche se in generale si hanno grandi mucchi dello stesso tipo di oggetto. I mucchi più grandi diventano crescono, con i gruppi piccoli che vengono prima o poi inseriti in quelli più grandi. L'elemento fondamentale è la memoria delle formiche, che permette loro di valutare la dimensione dei gruppi. Caratteristica importante è la flessibilità, con il sistema che opera indipendentemente dalla struttura del nido, posizione iniziale degli oggetti o numero di formiche, e che continua a funzionare anche se vengono spostati gli oggetti da un agente esterno. Un esempio di applicazione, ad esempio, è la distribuzione file in una rete peer to peer.

ANT FORAGING

Quando le formiche cercano il cibo, escono dal nido ed iniziano a camminare a caso, indipendentemente le une dalle altre e evitando ostacoli (più sono, più esplorano in fretta). Ad un certo punto una trova il cibo, torna alla base e rilascia un ferormone che attrae le formiche verso la risorsa (più l'odore è intenso maggiore è la probabilità per una formica di andare in quella direzione). Le formiche tornano al nido in modo stocastico quando hanno trovato il cibo (nel modello visto su netlogo non è così), ed il percorso per raggiungere il cibo è ottimale finché non viene rimpiazzato da uno migliore (più il percorso è lungo, minore è la concentrazione di ferormone, quindi in questo modo le formiche trovano il percorso più breve) oppure quando diventa inutile (e sparisce, perché nessuna formica lo ripercorre per prendere una risorsa che si è esaurita). Nell'ant foraging ci sono diverse retroazioni:

- Positive
    - Il ferormone attiva più formiche, le conduce al cibo e permette la formazione di percorsi stabili
- Negative
    - L'evaporazione del ferormone implica che nessun percorso duri per sempre, e questo permette l'eliminazione di percorsi inutili

L'evaporazione e la casualità permettono l'esplorazione dell'intero spazio di ricerca, permettendo di trovare il percorso più breve, quindi l'ambiente non è solo passivo.
Un meccanismo non presente in natura, ma che viene usato nei sistemi artificiali, prevede l'impiego di due ferormoni, quello del nido (che serve per guidare le formiche direttamente al nido) e il ferormone del cibo. L'ant foraging viene usato per la ricerca di percorsi ottimali in reti (problema del TSP) oppure per la creazione di nuovi percorsi in grafi (routing in networks, dove l'ambiente è il grafo, i gradienti di ferormone sono le tabelle di routing, le formiche con cibo sono i pacchetti, quelle senza sono i controllori attivi del routing cioè esplorano lo spazio per creare percorsi, la sorgente di cibo è l'emettitore di pacchetti e il nido è il ricevitore), anche se in questo caso è importante regolare bene i parametri dell'evaporazione e della velocità di diffusione del ferormone e del numero di formiche. L'Ant Foraging è una valida strategia per rintracciare informazioni in una rete p2p: quando un utente richiede un servizio, viene lanciato un certo numero di formiche alla ricerca di tale servizio, per poi rientrare una volta trovato rilasciando ferormone (questo permette di avere percorsi rinforzati per accedere ai servizi più usati).

Negli esempi visti, la stigmergia è rappresentata dai ferormoni (ant foraging, slime), dalla percezione della situazione ambientale (ant sorting) e la percezione indiretta dei lampi di luce (lucciole). La stigmergia disaccoppia le interazioni ed è consapevole del contesto. Inoltre, emerge che l'ambiente non è passivo, e che l'auto organizzazione necessita di essere attivamente supportata dall'ambiente.
E' inoltre chiaro come gli sciami non siano intelligenti, quindi possiamo astrarlo ad un sistema composto da semplici componenti elettriche reattive, connesse ad una rete ed esibenti pattern di sincronizzazione, e possiamo dire che non vi è mente, anche se… è una descrizione che si addice anche al cervello umano (sciame di neuroni). Molti sistemi naturali esibiscono comportamenti apparentemente intelligenti (flessibili, auto-organizzati, mirati al raggiungimento

di uno scopo), e almeno una parte di questi può essere trasferita a sistemi artificiali (allo scopo di ottenere applicazioni robuste).

NETLOGO

Netlogo è un linguaggio di programmazione basato sugli agenti, che possono essere dei turtle (agenti che si muovono nel mondo), patches (agenti che non si muovono, compongono il mondo, può creare nuove turtle), link (agenti che collegano due turtle) o degli observer (l'utente, può creare nuove turtle). Netlogo è stato sviluppato per studiare i fenomeni di swarm intelligence. Per avere un sistema omogeneo, il piano 2d in cui si visualizza la simulazione è toroidale, ovvero andando a sinistra si esce da destra (idem sopra e sotto) e viceversa (questo può essere cambiato).

Nome	Descrizione
Pcolor	Contiene il colore della patch
Plabel	La patch appare con un valore attaccato ad essa come testo
Plabel-color	Determina il colore del label
Pxcor/pycor	Le coordinate

Le patches hanno delle coordinate intere (pxcor, pycor), e quella collocata a (0,0) è chiamata origine. Il numero totale delle patches dipende dalle impostazioni min-pxcor, max-pxcor, min-pycor, max-pycor (che di default sono -16,16,-16,16). Le patch contengono le variabili riportate a lato. Le variabili del patch possono essere cambiate da qualsiasi turtle ci sia sopra

Le turtle possiedono delle coordinate decimali (xcor, ycor). Le variabili predefinite delle turtle sono riportate qui sotto.

Nome	Descrizione
Breed	Contiene il tipo della tartaruga
Color	Contiene il colore della tartaruga
Heading	Indica la direzione in gradi
Hidden?	Booleano che indica se la tartaruga è nascosta o meno
Label	Etichetta della tartaruga
Label-color	Colore del label
Pen-mode	Imposti questa variabile per disegnare linee, cancellarne o interrompere una di queste azioni
Pen-size	Dimensione penna
Shape	Stringa che contiene il nome della forma attuale del turtle
Size	Dimensione apparente
Who	Id della turtle
Xcor, ycor	Coordinate

Segue un codice di esempio di un modello in cui le turtle si muovono casualmente sul piano:

```
to setup
  clear-all; Pulisce tutto, resetta il mondo
  crt turtlenumber; Crea tartarughe in base allo slider
  ask turtles [set Pen-mode "down"]; Imposto la modalità di scrittura
end

to go
  ask turtles [
    pen-down
    rt random 10
    lt random 10
    fd random 5]; Chiedo a tutte le tartarughe di muoversi in avanti (fd) e di ruotare a
destra (rt) e a sinistra (lt) di un numero da 1 a 10
    ; turtles restituisce una lista delle tartarughe ordinata in modo casuale
    ;pen-down abbassa la penna (non è necessario vista l'istruzione del pen-mode nel
setup
end
```

Pen-mode è una variabile turtle integrata, e mantiene lo stato della penna (può essere impostata a "up", "down", "erase"), e può essere rimpiazzato da 3 comandi equivalenti (pen-down,pen-erase…)

In netlogo, l'osservatore non ha né contesto né luogo fisico, le turtle si muovono e hanno una visione del mondo limitata, le patches non possono muoversi, hanno dei vicini e costituiscono una sorta di automa cellulare, e i link sono agenti che collegano due turtle.

Per definire variabili globali si può ricorrere alla sintassi global [nome_variabile], e per impostarne il valore set nome_variabile valore/espressione. Per definire variabili legate alle entità si può usare turtles-own, patches-own, links-own [lista_variabili], e le modifiche possono essere fatte unendo il predicato "ask" con quello "set". Per mostrare una variabile di una certa turtle, si può usare show [nome_variabile] of turtle id_tartaruga (id tartaruga può anche essere una coordinata, ad esempio "1 10"). Per definire variabili locali si può ricorrere al predicato let nome_variabile. In netlogo a comunicare compiti agli agenti sono i comandi (azione da completare) o i reporter (calcola un risultato e lo riporta), quelli pre-inseriti in netlogo sono detti primitive. Una procedura (o comando) è definita con to nome_comando [param] e chiusa righe dopo con end. E' possibile definire i propri reporter, che vengono definiti con to-report nome [param] e viene chiuso dall'end, e devono reportare (con la sintassi report variabile) un valore. Per i controlli condizionali si usa ifelse condizione [se vero][se falso], oppure if condizione [se vero].
Per selezionare un subset di tutte le turtle, si possono usare o i reporter predefiniti turtles-here (solo le turtle che si trovano sulla patch corrente della tartaruga), turtle-at (tartarughe su una certa patch a coordinate relative rispetto ad una turtle) o turtles-on (turtle che stanno su una certa patch), other turtles (tutte le altre tartarughe), turtles with [condizione] (seleziona solo le turtle con una certa caratteristica), in-radius (seleziona le turtle in un certo raggio). Neighbors4 è un comando rapido per parlare delle patch a Nord, sud, ovest ed est rispetto a quella attuale (comando alternativo, patches at-points [[1 0][0 1][-1 0][0 -1]]), e può essere usato ad esempio con turtles-on. Oltre ad ask, con gli agenti si può interagire con any? (verifica se l'agenset è vuoto o meno), all? (verifica se ogni agente soddisfa una condizione), count (restituisce il numero di agenti), one-of (preleva a caso un'agente dal set). La differenza tra self e myself è che il primo è la turtle o l'entità a cui si è fatto l'ask, myself è la turtle o la patch che ha chiesto l'esecuzione del comando.

OSSERVAZIONI DI LABORATORIO

TERMITI

Dal momento che le termiti non interagiscono tra di loro (a differenza delle formiche), viene il dubbio che non si tratti effettivamente di swarm intelligence (una singola termite può fare il lavoro di tutte le altre, cosa non vera nel caso delle formiche). In particolare, il lavoro di 1000 termiti fatto in un tempo T viene fatto da 10 termiti in un tempo $T * 100$. In particolare, all'aumento delle termiti, il tempo di raggiungimento del medesimo risultato cala in modo iperbolico ($c * x^{-1}$). Al variare del gruppo delle termiti, quindi, vi è un aumento di efficacia (il gruppo c'è, ma non è coordinato e quindi non mostra la swarm intelligence). I parametri rg_libera e rg_inerzia sono legati alla memoria dello sciame, per così dire.

ALGORITMI GENETICI

ISPIRAZIONE BIOLOGICA

In natura esistono molti sistemi in grado di apprendere e, su scala temporale differente, altrettanto fanno le singole specie all'interno della storia dell'evoluzione biologica. Parlando di cambiamento delle specie, a livello storico esistono due filoni di pensiero: quello di Lamarck, dove i cambiamenti avvenuti in una vita vengono trasmessi ai discendenti (collo delle giraffe, che si allunga di generazione in generazione) e che porta ad un graduale adattamento della specie all'ambiente, e quello di Darwin e Wallace, dove i cambiamenti che avvengono durante la vita non sono ereditabili e le specie cambiano perché vi sono differenze tra gli individui fin dalla nascita (che sono casuali). Nel caso di Darwin, l'ambiente esercita una pressione selettiva che favorisce i portatori di caratteri vantaggiosi, e coloro che non li possiedono verranno eliminati, lasciando solo quelli con le caratteristiche migliori.

Darwin applica alla biologia la teoria economica di Malthus (superata una certa linea "portante" di risorse, la popolazione cala per poi stabilizzarsi sulla portante): due specie che occupano lo stesso luogo competono per le stesse risorse, e gli abitanti totali sono $N = N_1 + N_2, N_1(t+1) = N_1(t)[1 - k_1] - \beta N(t)^2, N_2(t+1) = N_2(t)[1 + k_2] - \beta N(t)^2$, dove a sopravvivere è quella che riesce a crescere più velocemente.

Darwin, pur avendo intuito le caratteristiche dell'evoluzione non ne conosce il meccanismo, che viene alla luce in seguito agli esperimenti di Mendel (geni, forme dominanti e recessive) e alle scoperte di Chargaff (il DNA trasmette l'informazione genetica) e di Watson e Crick (doppia elica). Il DNA è costituito da una sequenza di nucleotidi, ciascuno dei quali contiene una base azotata (A,C,G o T), e contiene l'informazione necessaria per sintetizzare l'm-RNA e quindi le proteine. Le basi azotate, raccolte a tre a tre, vanno a formare 21 amminoacidi, con ogni gene (insieme di amminoacidi) che va a codificare una proteina, le quali svolgono un ruolo strutturale e controllano le attività chimiche della cellula. Le fonti naturale di variabilità sono la mutazione puntiforme (modifica casuale e puntuale nella sequenza nucleotidica di un gene), la duplicazione dei geni (a cui segue una deriva genica) e la ricombinazione comosomica, e le mutazioni portano a cambiamenti estremamente graduali.

Evoluzione e apprendimento sono simili, in quanto nel corso dell'evoluzione una specie apprende implicitamente le caratteristiche stabili dell'ambiente in cui vive, e l'apprendimento può esser visto anche come un processo di generazione di diversi comportamenti alternativi e di selezione di quelli che sono maggiormente adatti.

Definizione: Fitness
La fitness di un individuo è definita come il numero di discendenti fertili. Fra gruppi di individui che popolano la stessa nicchia, un vantaggio in termini di fitness causa la colonizzazione della popolazione.

In un modello evolutivo, esistono quindi una popolazione composta da individui diversi, un sistema di valutazione del fitness di ogni individuo, un meccanismo per generare nuovi individui a partire da quelli con fitness più elevata e un meccanismo che introduce novità.

ALGORITMI GENETICI

Un insieme di soluzioni è detta popolazione, m stringhe binarie a ciascuna delle quali è associato un valore di fitness. Le stringhe binarie rappresentano cromosomi, e i geni che li compongono codificano una determinata proteina, che vanno quindi a determinare il "corpo" dell'individuo. Per risolvere i problemi imitando la selezione naturale, si valutano gli individui della popolazione corrente, si trovano i più adatti, li si ricombina, li si muta e si dà origine ad una nuova popolazione, e il processo si ripete. La scelta dei genitori proporzionale al fitness (ogni individuo ha una probabilità di essere scelto) è quella più vicina alla metafora biologica, ma presenta svantaggi. La ricombinazione (crossover) di due individui che portano due porzioni utili si possono generare individui molto migliori di entrambi, con fitness simili o relativamente scadenti, anche se questa procedura non è in grado di creare un tratto se non è presente in nessuno dei due genitori. Per questo si introduce la mutazione, che può seguire due metodi: o con una certa probabilità ogni nuovo individuo originato dal crossover viene mutato in una posizione casuale oppure prendo un genitore e lo muto sicuramente . Un possibile algoritmo risulta essere quanto riportato a lato.

```
T=0
crea popolazione P composta da N stringhe binarie lunghezza L
while t<tmax
        valuta la fitness del contenuto di P
        scegli le coppie in modo prop. al fitness
        da ogni coppia di genitori genera una coppia di figli
        poni P=P', ovvero aggiungi la nuova generazione
        incrementa t
end
```

La fitness media cresce con le iterazioni, e tende a raggiungere quella dell'individuo migliore, anche se questo non accade in natura. Quando la fitness media si avvicina sufficientemente alla fitness del migliore, l'algoritmo può essere arrestato.

INTERPRETAZIONE DELL'ALGORITMO GENETICO

Uno schema è un insieme di individui che hanno una parte in comune, e può essere definito da una stringa di lunghezza L (pari a quella degli individui) con elementi che possono essere 0,1 e # (cond. Indifferenza). Secondo Holland l'efficacia dei GA è legata al fatto che ad ogni generazione la valutazione del fitness di N individui comporta una stima della fitness media. Un esempio di schema è $01\#\#1 = \{01001, 01011, 01101, 01111\}$. Ad ogni stringa binaria, corrispondono 3^L schemi (perché ci sono 3 simboli), e ogni individuo appartiene a 2^L schemi diversi.

Esaminando una popolazione di N individui si valutano implicitamente al massimo $N * 2^L$ schemi (parallelismo implicito), il cui effettivo numero è minore a causa di ripetizioni. La riproduzione degli schemi opera mediante un meccanismo di selezione, senza crossover o mutazioni, quindi all'istante t vi è una popolazione P e al successivo P'. Ogni elemento di P' è generato dall'individuo i di P con probabilità pari a $\frac{f_i}{\sum f_k} = \frac{f_i}{N<f>}, f_i = fitness, < f > = fitness\ media$, quindi il numero di discendenti di un individuo i è pari a $\frac{Nf_i}{N<f>} = \frac{f_i}{<f>}$. Sia $f(H)$ la fitness media di uno schema, e consideriamo lo schema H composto da $m = m(H, P)$ individui in P, e la fitness media risulta quindi essere $f(H) = \frac{\sum_k f_k}{m}, k \in H$, e si rinominano gli individui in modo che quelli di H si trovino nella parte iniziale della popolazione. Il numero atteso di individui in H alla nuova generazione diventa $\langle m' \rangle = \langle m(H, P') \rangle = \frac{(f_1 + f_2 + \cdots + f_h)}{\langle f \rangle} = \frac{mf(H)}{\langle f \rangle}, \frac{\langle m' \rangle}{m} = \frac{f(H)}{\langle f \rangle}$, con la proporzione di schemi buoni che cresce con la fitness dello schema in modo $\simeq$ esponenziale.

Il crossover classico può distruggere facilmente uno schema buono: definiamo la lunghezza $d(H)$ di uno schema H come la distanza massima fra due posizioni fissate dello schema (ad esempio #1#1## ha lunghezza 2, #1###1 ha lunghezza 4), e la probabilit che il crossover rompa uno schema è pari a $\frac{d(H)}{L-1}$ se i genitori non appartengono allo stesso schema. Uno schema ha buone possibilità di conservarsi se è generale e compatto e, riassumendo, la selezione muta la frequenza degli schemi già presenti, ma non introduce nuove informazioni, mentre la mutazione può distruggerle.

Indichiamo con $P(H, t) = \frac{m(H,t)}{N}$ la frazione di individui che appartengono ad H e con $\phi(H, t) = \frac{f(H,t)}{\langle f \rangle_t}$. Se $d(H) = 0$ allora $P(H, t + 1) \geq \phi(H, t)P(H, t)$, ovvero crescita esponenziale degli schemi finché H non si diffonde al punto di modificare la fitness di una buona parte della popolazione e quindi il proprio $\phi(H, t)$. Se invece $d(H) = L - 1$ allora $P(H, t + 1) \geq [\phi(H, t)P(h, t)]^2$, condizione sufficiente per avere $P(H, t + 1) \geq P(H, t)$ è $\phi(H, t)^2 P(H, t) \geq 1$, uno schema con fitness doppia della media cresce se è presente in ¼ della popolazione.
Dimostrazione
Indichiamo con $P(H, t) = \frac{m(H,t)}{N}$ la frazione di individui che appartengono a H e con $\phi(H, t) = \frac{f(H,t)}{\langle f \rangle_t}$ il fitness ratio. Se $d(H) = 0$ allora ogni volta che si applica il crossover si ha un figlio in H, e la probabilità di selezionare un genitore in H è $\phi(H, t)P(H, t)$ quindi $P(H, t + 1) = \phi(H, t)P(H, t) +$ nuovi individui in $H \geq \phi(H, t)P(H, t)$. Se $d(H) = L - 1$ allora solo i figli dell'incrocio di due individui che sono entrambi in H saranno in H, $P(H, t + 1) = [\phi(H, t)P(H, t)]^2 +$ nuovi individui in $H \geq [\phi(H, t)P(H, t)]^2$.
La condizione $P(H, t\text{è}1) \geq P(H, t)$ escludendo i nuovi nati, è $\phi(H, t)^2 P(H, t) \geq 1$.

Se non ci fosse crossover, gli schemi migliori si replicherebbero, ma non ci sarebbe la componente che crea novità. In generale, la popolazione tende a convergere verso una situazione in cui tutti si somigliano, e questo porta a fenomeni di convergenza prematura (una popolazione non ottimale colonizza tutto), il che arresta il progresso. Per controllare questo fenomeno, è necessario poter valutare e controllare i tempi di convergenza.

TECNICHE PER IL RALLENTAMENTO DELLA CONVERGENZA

Strategia delle nicchie
Una strategia è quella di immaginare un sistema composto da tanti sottosistemi (celle) collocati nello spazio, e che la genetica lavori dentro ad ogni cella. In natura, i compagni per la riproduzione vengono scelti tra quelli appartenenti all'ambiente locale e prop. al loro fitness. I figli rimangono vicini ai genitori (le mutazioni avvengono solo localmente), con ambienti favorevoli che favoriscono la crescita di individui forti e i conflitti che vengono risolti dipendentemente alla forza relativa dei contendenti (all'individuo più forte servirà maggior tempo prima di invadere altre celle). Un genoma con alta fitness, prima di confrontarsi con genomi distanti, deve invadere la zona locale con i suoi cloni, il che porta un tempo maggiore per invadere la popolazione totale, e permette alle altre celle di evolversi in modo indipendente, la cui diversità viene mantenuta a lungo (nicchie ecologiche).

Strategia della scelta dei genitori: Elitismo e scaling
La scelta dei genitori proporzionale alla fitness è quella più vicina alla realtà, ma ha lo svantaggio di poter far sparire

individui molto performanti e possibile convergenza prematura se un individuo è molto superiore alla media.
Le alternative a questo sistema sono l'elitismo (si garantisce la sopravvivenza degli individui con fitness più alta) e lo scaling (selezione dipendente in modo non lineare del fitness, con la pressione selettiva che aumenta o si riduce in base alle loro performance).
Nello scaling, la probabilità non è proporzionale alla fitness, ma è funzione di essa: $p_i = \frac{F(f_i)}{\sum_j F(f_j)}$.

Strategie di selezione
Nella selezione per rango, si ordinano gli individui per il fitness in modo decrescente, e si fissa una distribuzione di punteggi a probabilità decrescente con la posizione occupata, indipendente dai valori del fitness. Il vantaggio è che non si ha convergenza prematura (quello che ha la fitness più alta, anche se molto distante dal secondo, sarà comunque vicino ad esso con questa selezione) e non c'è selezione, ma è pesante come procedura e non è biologicamente plausibile.

Nella selezione a torneo, vengono scelti s individui (in modo uniforme, gli individui possono essere scelti più volte) in una popolazione di N individui, con l'individuo scelto per la riproduzione che è quello con fitness più alto tra i contendenti, e si prosegue fino a generare N nuovi contendenti. Supponendo che esista un solo individuo con fitness superiori a tutti, che chiamiamo i, alla generazione successiva il numero di copie di i è s (questo impedisce che tutta la popolazione venga colonizzata in un passo, al passo successivo se s<<N ce ne saranno circa s^2 e aumenta man mano la probabilità che due individui superiori partecipino allo stesso torneo.

Strategie sugli operatori genetici
Il crossover tende a distruggere combinazioni di building blocks lontani, e vi sono due alternative: o il crossover a due punti (come accade in natura, una selezione di blocchi viene scambiata) o il crossover uniforme (vengono scambiati gruppi di blocchi).

Una possibile variante dei GA è usare una codifica basata su numeri reali o su valori discreti: il crossover rimane lo stesso, ma l'operatore di mutazione deve essere modificato (in caso discreto, si sceglie a caso una posizione e si sostiuisce il simbolo con uno dei restanti k-1 valori dell'alfabeto scelto a caso (dist. Uniforme), in caso continuo si sceglie una posizione a aggiungi al valore presente un numero casuale (dist. Gaussiana a media nulla)).

OTTIMIZZAZIONE

I GA possono essere usati per cercare il valore massimo di una data funzione, e il sistema è stocastico (non vincolato a cadere in estremo locale) e non sono richieste proprietà particolari da imporre alla funzione da ottimizzare. I GA possono essere usati anche quando non è nota la funzione da ottimizzare (basta disporre di un metodo per valutare la fitness di ogni soluzione). Selezione, crossover e mutazione sono operatori genetici che consentono di affrontare molti problemi di ottimizzazione (ottimizzazione non lineare, soluzione ottimale vs soluzione soddisfacente) e il metodo è particolarmente efficace quando il crossover non genera soluzioni illegali e la struttura dello spazio è tale per cui le posizioni dei massimi sono correlate. Se la funzione è piatta, l'unica strategia efficace è la ricerca esaustiva (che diventa presto impraticabile). Il caso più semplice è quello delle monotone con un solo massimo, e diversi metodi ci permettono di localizzare il massimo (sebbene il GA sia lento a convergere ad esso). I GA sono particolarmente efficaci quando vi sono diversi massimi locali e quando la posizione dei massimi locali contiene informazioni utilizzabili dall'algoritmo per cercare soluzioni ancora migliori

CONOSCENZA SUL DOMINIO

Gli operatori genetici ad hoc incorporano conoscenze sul dominio, e evitano la proliferazione di soluzioni illegali. Nel problema del commesso viaggiatore, il cromosoma è una lista delle città visitate, e si indica con $d(i,j)$ la distanza tra due città. Avendo quindi $x = [x_1 \dots x_L], d = d(x_1, x_2) + \cdots + d(x_{L-1}, x_L)$ e la funzione fitness è una funzione decrescente della lunghezza complessiva del cammino ($f = 1/d$). Per evitare di avere ripetizioni, si può convenire che tutti i cromosomi partano dalla stessa città, con il crossover che deve essere modificato per non creare combinazioni illegali: il crossover a mappa parziale opera definendo una lezione di matching, che è la parte destra del punto di

taglio, e ogni elemento del primo genitore viene verificato e scambiato se appartiene alla mappa di scambio (si fa la stessa cosa con il secondo), e questo genera sempre tour legali.

APPLICAZIONI

Utilizzando diversi tipi di GA e operatori sono nate applicazioni in vari settori, tra cui problemi classici, problemi di scheduling, ottimizzazione funzioni, image processing, bin packing, machine learning…

SISTEMI A CLASSIFICATORI

Rimanendo nell'ambito dei sistemi ad ispirazione biologica e in grado di apprendere, si affronta il tema dei sistemi a classificatori, che nascono nell'ambito delle ricerche sulle intelligenze artificiali, di cui si era fortemente enfatizzata la capacità di risolvere problemi generali in fase iniziale, per poi concentrare la ricerca verso una sempre maggiore conoscenza specifica sul dominio (ci si rende conto della conoscenza specifica rispetto ai metodi generali di ragionamento).

DAI SISTEMI ESPERTI AI CLASSIFICATORI

Nei sistemi esperti, la conoscenza è rappresentata da regole del tipo "SE condizione ALLORA azione" (se canta e vola è un uccello: quando la base dei fatti contiene informazioni che soddisfano la condizione, la regola viene applicata (canta e vola) e l'azione può aggiungere nuova conoscenza alla base dei fatti (è un uccello), su cui si potrà continuare a lavorare (se è un uccello e ha le piume nere, è un corvo)). La conoscenza è quindi composta da regole e da fatti specifici relativi al caso in questione, con il motore inferenziale (quello descritto precedentemente) che decide quale regola applicare, e il motore inferenziale e la base di conoscenza rappresentano il fulcro del sistema. Vi è separazione tra la conoscenza e il suo uso: la conoscenza può venir fornita, ma è la macchina inferenziale a concatenare le regole (o inferenze) tra di loro. Un esempio di sistema esperto è Mycin, ma ci sono anche Prospector (ricerca giacimenti), Dendral (analisi composti mediante spettrometria) e altri ancora.

Mycin
Usato per diagnosi di malattie infettive. Se l'organismo è gram-positivo, se la morfologia dell'organismo è cocco e la conformazione di crescita è a blocchi, allora potrebbe essere uno stafilococco. Mycin si basa sui dati di culture di batteri e informazioni sul paziente, e lavora ipotizzando una diagnosi e, mediante un backward chaining in profondità, cerca dati a favore della sua ipotesi. Usa inoltre un meccanismo basato su fattori di certezza per dare un peso diverso alle catene di inferenze, e confronta i pesi complessivi e propone la diagnosi più supportata.

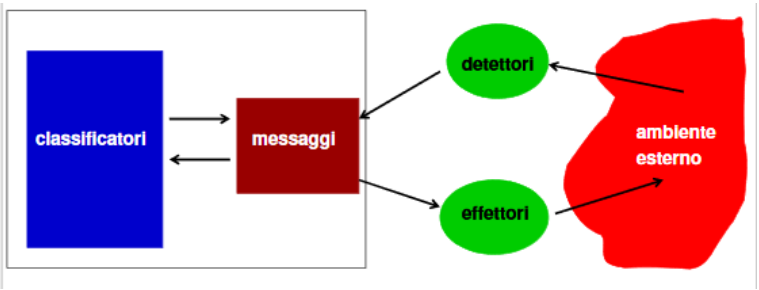
I programmi tradizionali sono rigidi, mentre questi sistemi esperti appaiono più efficaci e naturali da comprendere (fornendo prestazioni umane), e questo attirò un grande interesse applicativo, e i successi iniziali crearono grandi aspettative ad inizio anni 80, diffondendo la convinzione che sviluppare sistemi esperti fosse semplice e veloce (si sosteneva che sarebbe sufficiente riempire una base di conoscenze mediante un un inserimento di regole ottenute attraverso un dialogo con un esperto umano). Dalla fine degli anni 80 a metà degli anni 90, la sottovalutazione dei problemi legati all'acquisizione e alla rappresentazione formale delle conoscenze (formalismi innaturali, codifica dei fatti non precisa), il tempo elevato per sviluppare un sistema esperto, la difficoltà di aggiornare delle conoscenze, assenza di prestazioni fuori da questo mondo, la necessaria integrazione con il resto del sistema informativo, improvvisazione da parte dei programmatori e overselling da parte dei venditori determinarono l'Inverno delle IA. Dalla metà degli anni 90, le aziende che avevano investito saggiamente su applicazioni ben meditate, avevano ottenuto risultati, a dimostrazione che i sistemi esperti possono funzionare egregiamente, ma non ci sono scorciatoie, e l'aumento dell'efficienza permette una riduzione di personale e un basso time-to-market per un determinato prodotto, e si tende a sviluppare sistemi per la gestione delle conoscenze: un sistema esperto non è una IA generalista, ma specializzata.

Le reti neurali, come si vedrà in seguito, rappresentano un approccio alternativo (o complementare) a quello simbolico, e la crescita delle risorse hw permette di ottenere ottime prestazioni in molti settori, anche se rimangono comunque oscure (non hanno sistemi a catene di inferenze, e per questo non possono spiegare i motivi dietro alle

loro conclusioni). Esistono anche metodi di apprendimento che ragionano sugli esempi, i quali sono vulnerabili a difetti, rumore e contraddizioni, e risulta interessante combinare le proprietà di auto-organizzazione con le capacità di spiegazione e concatenazione di inferenze, dando origine ai Sistemi a classificatori.

DESCRIZIONE

I sistemi classificatori (CS) sono sistemi che imparano come svolgere un compito interagendo con un ambiente in parte ignoto, usando meccanismi di feedback per guidare un'evoluzione interna che modifica il proprio modello del mondo (basato su regole), e combinano gli aspetti simbolici (rappresentazione esplicita della conoscenza basata su regole) con i meccanismi di auto organizzazione dinamica (in altre parole, è un sistema esperto con abilità di apprendimento). Gli aspetti sub-simbolici dei CS sono che le regole esistenti vengono valutate in funzione del loro utilizzo, le nuove regole vengono create mediante variazioni di regole esistenti.
I CS vengono introdotti negli anni 80 da Holland, nell'ambito della simulazione dei processi di apprendimento induttivo mediante metodi ispirati dalle scienze biologiche, facendo ricorso al meccanismo di valutazione delle regole basato su una metafora economica e che hanno riscosso numerosi successi nell'ambito della classificazione, della verifica, nel movimento in ambiente incognito, ambito militare e in particolare nell'ambito dei sistemi multi agente, in cui gli agenti sono dei CS e riescono ad adattarsi autonomamente al sistema. Se un messaggio presente nella lista dei messaggi soddisfa la condizione, allora cerca di impostare il messaggio azione, che può consistere in nuova conoscenza che viene aggiunta alla lista dei messaggi oppure un messaggio per gli effettori, con i classificatori che possono agire simultaneamente. Esistono due tipi di dinamiche: quella rapida, in cui un sistema di apprendimento con rinforzo assegna un valore alle diverse regole sulla base del loro contributo per il funzionamento del sistema, e quella lenta, in cui un sistema di generazione di nuove regole basato su algoritmi genetici elimina le regole meno utili e le sostituisce con variazioni o combinazioni di regole esistenti.



I CS seguono un determinato ciclo di funzionamento:

- Alla lista dei messaggi (del passo precedente) si aggiungono messaggi in input da parte dei detettori
- Si confrontano i messaggi con le condizioni di tutti i classificatori
- I classificatori soddisfatti competono per impostare i messaggi
- I vittoriosi impostano e ripagano quelli che hanno impostato i messaggi che hanno consentito loro di attivarsi
- La vecchia lista dei messaggi viene cancellata e sostituita dai messaggi impostati dai classificatori esistenti
- Gli effettori verificano se ci sono messaggi di output sulla lista, nel caso eseguono l'azione corrispondente
- I classificatori che hanno impostato i messaggi di output vengono ripagati dall'esterno (premio o punizione)

I messaggi altro non sono che stringhe binarie, composte da un numero L (fisso) di elementi {0,1}. I classificatori sono definiti dalla condizione, stringa L di simboli {0,1,#}, e una parte azione {0,1,*} (*=passthrough) e da una variabile reale s, che indica la forza. Un classificatore fa match con messaggio se la sua condizione corrisponde con in messaggio in tutti i punti diversi dai #, e nei punti dell'azione con * viene fatta passare come parte di azione la parte di azione del messaggio in entrata (ad esempio, se il messaggio in entrata ha 10011 e l'azione della regola è 101**, diventa 10111).

Definizione: Specificità del classificatore
La specificità è definita come la frazione di elementi della condizione diversi da #. Indicati con @ il numero di # nella parte di conzione, è uguale a γ =(L-@)/L.

La competizione è basata su una funzione (bid) $b_i = \gamma_i s_i$, e la scelta dei vincitori viene eseguita in modo probabilistica. La forza del classificatore è misura della sua utilità dimostrata (forza elevata->bid elevato->elevata possibilità di impostare), e più si prendono manovre giuste, maggiore è la "popolarità" del classificatore, ma se inizia a sbagliare perde popolarità e quindi si riduce la frequenza con la quale imposta. E' necessario ricompensare le regole a monte di quella che ha fornito l'output corretto, e si introduce un meccanismo locale che coinvolge solo i classificatori che si attivano in due istanti successivi. L'evoluzione delle forze viene definita dall'algoritmo "bucket brigade", il cui pseudocodice è riportato a lato. E' utile far coesistere regole generali e specifiche (gerarchie di default), e in assenza di dati specifiche quelle generali offrono utili indicazioni: la dipendenza del bid dalla specificità consente di privilegiare i classificatori più specifici, conservando regole generali anche se contraddette in casi particolari.

```
Aggiornamento_forza(s[i]){
        p[i]=alpha*s[i];
        new_s[i]=s[i]*(1-beta);
        if (i ha impostato Q messaggi) new_s[i]=new_s[i]-Q*p[i];
        if (C_i ha impostato un messaggio all'istante precedente AND
i classificatori i[1]…i[n] hanno impostato usando quel messaggio)
                new_s[i]=new_s[i]+p[i[i]]…p[i[n]];
        if (C_i ha impostato un messaggio di output)
                new_s[i]=new_s[i]+reward;
        s[i]=new_s[i];
}
```

Una popolazione è composta da individui differenti (ognuno possiede un genotipo ereditabile) ed esiste un meccanismo per creare nuovi genomi a partire da quelli di uno o alcuni individui e una competizione tra individui simili per riprodursi. Nella popolazione di un GA, ogni individuo può rappresentare una soluzione al problema in esame, e gli operatori genetici consentono di introdurre novità casuali e con i figli che assomigliano molto ai genitori, è presente un sistema di valutazione della fitness e ad ogni passo si crea una nuova popolazione. Questo può essere integrato nel nostro discorso per l'apprendimento a lungo termine (quello a breve termine viene gestito dal bucket brigade): si scelgono i genitori delle nuove regole sulla popolazione C di classificatori in base alla forza, si applicano gli operatori genetici per ottenere un insieme G<C di regole (in cui è presente l'elitismo), si rimuovono da G i classificatori in base alla forza (quelli a bassa forza vengono eliminati, gli altri vengon mantenuti) e i nuovi classificatori vengono aggiunti alla base di regole.

Numero_generazioni = 1
WHILE (Numero_generazioni <= Max_gen){
- Valuta la forza di tutti i classificatori della popolazione attuale P , col ciclo a regole fisse;
- valutane la fitness;
- P' = Ø;
- Scegli fra i classificatori G/2 coppie di genitori, in maniera proporzionale alla fitness;
- Incrociali con crossover a un punto, ottenendo G figli, e inseriscili in P' ;
- Ad ognuno degli elementi di P' applica l'operatore di mutazione puntuale (con una piccola probabilità);
- Aggiungi a P' le C-G regole con la fitness più alta ;
- P = P' ;
- Numero_generazioni = Numero_generazioni + 1;

}

Dal momento che il sistema è stocastico, non cadrà in un estremo locale (il crossover permette di esplorare regioni distanti nello spazio degli stati), e non vi sono richieste di alcun genere da imporre a priori alla funzione da ottimizzare (funzionano bene quando c'è una struttura nell'insieme dei valor estremi, i massimi locali forniscono indicazioni sulla localizzazione del massimo assoluto) e rappresentano una buona alternativa quando non si hanno informazioni a priori sulla funzione fitness. Nelle classiche applicazioni di GA questa esiste, ma nei sistemi "alla Michigan" è impossibile assegnare il fitness ad un individuo su cui agisce il GA: il fitness, o forza, dipende dalle interazioni ed è quindi collettiva, con i classificatori che co-evolvono (il cambiamento di un classificatore può influenzare la fitness degli altri). Partendo da una popolazione limitata con classificatori casuali, è probabile che i messaggi provenienti non facciano match con nessun classificator, e non vi sia messaggio di output. Per questo esistono operatori genetici specifici come il cover detector (se nessun classificatore fa match, crea un classificatore che fa match con un'azione casuale) e il cover effector (se non ci sono messaggi di output, genera un classificatore che matcha la situazione attuale e manda un output casuale).

ASPETTI SUGGESTIVI

Il sistema elabora in parallelo, e l'apprendimento combina una metafora economica (breve termine) e biologica (lungo termine), basandosi sulla interazione fra metodi genetici e sistemi dinamici, con le regole che sono "agenti in un mercato", e sono semplici. E' presente una nascita spontanea di catene di regole,e da qui si formano anche le gerarchie di default. Altro fattore importante è la co-evoluzione, ovvero il valore del classificatore dipende dagli altri

classificatori (e il valore di una regola dipende dalla presenza di altre regole, le quali formano tra loro relazioni in corso dell'evoluzione).

I problemi che sorgono sono quelli della convergenza prematura, individui di ugual lunghezza (i CS devono aver la stessa dimensione), mancanza di previsioni, limitata efficacia nella formazione di catene lunghe e da una possibile struttura piatta.

RETI NEURALI

Un programma è dotato di solito di input e di output, e così anche un sistema dinamico che elabora informazione. Il caso più studiato è quello in cui l'input è la condizione iniziale del sistema e l'output è la condizione finale.
Il calcolatore incorpora un modello di cosa sia la computazione, basato sulla manipolazione di simboli fisici (proposto anche per descrivere le attività umane): l'idea dietro al cognitivismo è che l'hardware è marginale, e ciò che conta è l'abilità di manipolare tali simboli (è possibile un'intelligenza artificiale, questo concetto si è poi evoluto). Dal momento che il calcolatore è in grado di simulare sistemi, possiamo cercare di simularvi anche altre architetture per l'intelligenza, anche ispirate dall'organizzazione fisica del cervello: nascono così i primi modelli neurali.

L'intelligenza artificiale nasce nel 56 a Dartmouth, ed è composto da sistemi generalisti (sistemi esperti) che manipolano simboli (approccio simbolico). I sistemi simbolici sono basati sulla logica, sono fragili rispetto alla presenza di contraddizione e costruir sistemi esperti è complesso. Lo stimolo verso sistemi robusti rispetto alle contraddizioni è ciò che riaccende l'interesse verso le reti neurali negli anni 80.

Le reti neurali sono ispirate alla biologia, in particolare nel campo degli studi sul cervello, all'interno del quale l'attività di elaborazione viene svolta dai neuroni (il cervello non è una macchina di Von Neumann), e dopo numerose ricerche a riguardo si comprende che il ragionamento 1 neurone-1 cervello è inadeguato, e si pone l'enfasi sulle rappresentazioni distribuite a cui si devono anche le proprietà di robustezza del cervello, sebbene in tempi moderni si è convinti che ci siano neuroni speciali, anche se la rappresentazione distribuita rimane prevalente. Il neurone trasmette un impulso lungo l'assone se la somma dei suoi ingressi supera una certa soglia, $I(t) = \sum s(t_i)\phi(t - t_i)$, dove ϕ è una funzione decrescente al crescere dell'intervallo e misura la durata del ricordo di un impulso arrivato in precedenza: se arrivano molti impulsi in un tempo breve, il neurone spara. I neuroni non sono tutti uguali, inoltre: alcuni sono più spessi, alcuni sono meno spessi, quindi bisogna associarvi un peso. McCulloch e Pitts dimostrano come i neuroni possano essere usati come porte logiche, il che rappresenta un risultato incredibile, pur non avendo definito il metodo di costruzione (risulterebbe comodo avere un sistema in grado di regolare automaticamente pesi e soglie).

MODELLO DI HOPFIELD

Nel modello di Hopfield, che descrive una rete neurale usando un modello booleano con aggiornamento asincrono (si sceglie un neurone alla volta), la rete è composta di N neuroni booleani a soglia (i quali sono numerati) con x_i che indica il valore di attivazione del neurone i-esimo con il vettore X che descrive lo stato della rete. Il tempo evolve in passi discreti, quindi passa per lo stato X(0), X(1)… e per ogni coppia di neuroni è definita un'intensità di connessione (peso sinaptico), con w_{ik} che indica l'intensità di connessione dal neurone k al neurone i, $w_{ik} \in R, w_{ii} = 0$ (ovvero l'autoeccitazione/inibizione è proibita), con ogni neurone collegato con gli altri. Ad ogni neurone è assegnata una soglia $\theta_i \in R$, e l'output del neurone i-esimo all'istante t è definito come $I_i\big(X(t)\big) = \sum_k w_{ik}x_k(t)$: se la sommatoria degli input è maggiore della soglia, sparo il segnale, se è minore si spegne e se è uguale il nodo rimane com'era. La rete neurale si comporta come materiali magnetici disordinati, in cui lo spin su e lo spin giù corrispondono ai due stati dei neuroni: nei materiali ferromagnetici le interazioni sono cooperative (tutti positivi), mentre in quelli antiferromagnetici sono anticooperative (tutti negativi).
Nei vetri di spin sono presenti sia interazioni eccitatorie che inibitorie, e c'è corrispondenza fra i pesi delle reti neurali e i vetri di spin. La funzione energia, ricavata con i parallelismi al magnetismo e alla fisica, diventa $E(X) = \frac{1}{2}\sum_{i,j=1,i\neq j}^{N} w_{ij}x_ix_j + \sum_{i=1}^{N}\theta_i x_i$. Hopfield ipotizza che questa legge si applichi anche alle reti neurali, il che implica che il sistema neurale propende ad andare in determinati punti (i "pozzi" degli attrattori) e smetterà di oscillare, e si può dimostrare che se W è una matrice simmetrica l'energia è una funzione non crescente nella

dinamica asincrona, $E\big(X(t+1)\big) \leq E\big(X(t)\big)$ e che $E\big(X(t+1)\big) = E\big(X(t)\big) \rightarrow X(t+1) = X(t)$: l'energia è una funzione di Lyapunov per il modello di Hopfield simmetrico, e tende a un punto fisso.

Dimostrazione: energia funzione non crescente (FACOLTATIVA)

Si supponga che all'istante t venga aggiornato il neurone k-esimo, e sia $\delta x_k(t)$ la sua variazione, e si definisca il termine di Kroneker come δ_{ij} tale che sia 1 solo quando i = j. La variazione di E è:

$$\delta E(t) = -\frac{1}{2}\sum_{i,j=1,i\neq j}^{N} w_{ij}\Big(\big(x_i(t+1)x_j(t+1) - x_i(t)x_j(t)\big)\big(\delta_{ik}+\delta_{jk}\big)\Big) + \sum_{i=1}^{N}\theta_i\big(x_i(t+1)-x_i(t)\big)\delta_{ik} =$$

$$-\frac{1}{2}\sum_{i,j=1,i\neq j}^{N} w_{kj}\Big(\big(x_k(t+1)x_j(t+1) - x_k(t)x_j(t)\big)\Big)$$

$$-\frac{1}{2}\sum_{i=1,i\neq k}^{N} w_{ik}\big(x_i(t+1)x_k(t+1) - x_i(t)x_k(t)\big) + \theta_k\big(x_k(t+1)-x_k(t)\big) =$$

$$-\frac{1}{2}\sum_{j=1,j\neq k}^{N}\big(w_{kj}+w_{jk}\big)\big[x_j(t+1)x_k(t+1) - x_j(t)x_k(t)\big] + \theta_k\big(x_k(t+1)-x_k(t)\big)$$

Dato che $j \neq k, x_j$ non cambia, quindi

$$\big[x_j(t+1)x_k(t+1) - x_j(t)x_k(t)\big] = x_i(t)[x_k(t+1)-x_k(t)]$$

Dato che solo il k-esimo neurone è cambiato, si ha

$$\delta E(t) = \frac{1}{2}\sum_{j=1,j\neq k}^{N}\big(w_{kj}+w_{jk}\big)[x_k(t+1)-x_k(t)]x_j(t) + \theta_k\big(x_k(t+1)-x_k(t)\big)$$

$$= -\frac{1}{2}\delta x_k(t)\sum_{j=1,j\neq k}^{N}\big(w_{kj}+w_{jk}\big)x_j(t) + \theta_k\delta x_k(t)$$

Se $w_{kj} = w_{jk}$ allora $\delta E(t) = -\delta x_k(t)\sum_{j=1,j\neq k}^{N}\big(w_{kj}x_j(t)-\theta_k\big) = -\delta x_k(t) = \big[I_k\big(X(t)\big)-\theta_k\big] \leq 0$.

Per usare la rete, bisogna far si che l'evoluzione dinamica corrisponda a operazioni significative, che permette alla rete di riconoscere e classificare ed è per questo fondamentale che gli attrattori coincidano con schemi aventi significato. Le fasi di apprendimento e riconoscimento sono separate: nella prima, si modificano i pesi, in cui pattern di connessione alterano le connessioni, mentre nella seconda l'evoluzione avviene senza modificare le connessioni sinaptiche. L'apprendimento deve indirizzare verso un senso utile l'evoluzione, ergo far coincidere gli attrattori a pattern con significato e con bacini di attrazione adeguati.

APPRENDIMENTO HEBBIANO

Hebb osserva che quando l'assone della cellula A è sufficientemente vicino da eccitare una cellula B e ripetutamente e in modo persistente partecipa alla sua attivazione, si verifica qualche processo di crescita o cambiamento metabolico che aumenta l'efficienza di A nell'eccitare B. Per portare questa osservazione nel modello si rafforzano le connessioni tra neuroni che compaiono nello stesso stato, e si indeboliscono quelle tra neuroni in stati diversi: dobbiamo fare in modo che, insegnando un pattern $A = (a_1, \dots, a_n)$, le sinapsi dei neuroni aumentino di intensità, una possibilità è $\Delta w_{ik} = (2A_i - 1)(2A_k - 1)$ (per apprendere uno schema, se due neuroni sono entrambi accesi vale +1, se sono entrambi spenti vale +1, altrimenti -1). I neuroni "accesi" forniscono un contributo che tende ad accendere quello erroneamente spento (anche i neuroni spenti danno lo stesso contributo), e sebbene crei disturbo finché non supera un certo livello il rumore è tollerabile. Per insegnare un pattern A, quindi, le sinapsi che sono nello stesso stato aumentano di intensità, e la regola per l'apprendimento di M pattern è $\Delta w_{ik} = \sum_m (2A_i^m - 1)(2A_k^m - 1)$.

Definizione: Sovrapposizione di due pattern

$Q(X,Y) = \sum X_k Y_k$, coincide con il numero di posizioni in cui X e Y hanno il valore 1

Definizione: Completamento di pattern X
Il completamento del pattern X risulta essere $X' = \{x_i' | x_i' = 1 - x_i, i = 1 \dots N\}$, $Q(X,Y) + Q(X',Y) = N_y$

RELAZIONE TRA INPUT NETTO E OVERLEAP

$I_i(X) = \sum_{j\neq i}\sum_m (2A_i^m - 1)\big(2A_j^m - 1\big)X_j$, ma $2A_i^m - 1 = A_i^m + (1 - A_i^m) - 1 = A_i^m - A_i^m$ quindi $I_i(X) = \sum_m (2A_i^m - 1)\sum_{j\neq i}\big(A_j^m - A_j^m\big)X_j = \sum_m (2A_i^m - 1)(Q(A^m,X) - Q(A^m,X) - X_i\sum_m (A_i^m - A_i^m)^2 = \sum_m 2(A_i^m - 1)\big(Q(A^m,X) - Q(A^m,X)\big) - MX_i$

UN SOLO PATTERN A, SOGLIA NULLA

$w_{ik} = (2A_i - 1)(2A_k - 1)$, si nota che $w_{ik} = (2A_i - 1)(2A_k - 1) = (2(1 - A_i') - 1)(2(1 - A_k') - 1) = (2A_i' - 1)(2A_k' - 1)$, e insegnare un pattern o un suo complementare è la medesima cosa.

$$I_i(X) = (2A_i - 1)[Q(A,X) - Q(A',X)] - X_i$$

Un termine del tipo $(2A_i - 1)$ moltiplicato per un numero positivo abbastanza grande tende a permettere $X_i \to A_i$, se $A_i = 1$ l'input netto è positivo quindi, quando si aggiorna l'i-esimo neurone, esso si porta a 1 indipendentemente da quale fosse il suo stato precedente, mentre se $A_i = 0$ l'input netto è negativo, quindi il neurone i-esimo si porta allo stato 0. Data l'espressione precedentemente riportata, si verifica facilmente che A e A' sono punti fissi, e se $A, A' \neq 0$ lo stato (0…0) è comunque un punto fisso.
Considerando il caso $X(0) \neq A, A', 0$ si ha $X(0) \to A$ se $Q\big(A, X(0)\big) > Q\big(A', X(0)\big)$ e $X(0) \to A'$ se $Q\big(A, X(0)\big) < Q\big(A', X(0)\big)$.
Dimostrazione
$I_i(X) = (2A_i - 1)[Q(A,X) - Q(A',X)] - X_i$, se $Q(A,X) > Q(A',X) + 1$ allora $I_i(X)$ ha lo stesso segno di $2A_i - 1$, quindi ogni neurone tende al valore corrispondente in A_i. Se $Q(A,X) = Q(A',X) + 1$ allora se $X_i = 0$ $I_i(X)$ ha lo stesso segno di $2A_i - 1$ e il neurone tende al valore che ha in A, se $X_1 = 1$ e $A_i = 1$ $I_i(X) = 0$ e il neurone resta a 1, se $X_i = 1$ *e* $A_i = 0$ $I_i(X)$ ha valore negativo e il neurone tende a 0.

La convergenza verso lo stato cui X assomiglia maggiormente (nel senso specificato dalla misura dell'overleap Q) è veloce, e si realizza in un numero di interazioni non superiore al numero R di passi necessari per aggiornare almeno una volta ogni elemento che non si trova nello stesso stato dell'attrattore. Inoltre, lo stato $X = (0,0,\dots)$ è un punto fisso, anche se non può essere raggiunto a partire da nessuno stato iniziale (a meno che A e A' non coincidano con esso), con lo stato (0…0) acquista un bacino di attrazione se la soglia è diversa da 0. Se $Q(A,X) = Q(A',X)$ la rete può evolvere verso uno o l'altro dei due attrattori in funzione della sequenza di aggiornamento, quello che ha uno 0 in corrispondenza della prima posizione che viene aggiornata in cui X(0) abbia valore 1 (il sistema non è deterministico). Se $Q(A,X) = Q(A',X)$. Allora $I_i(X) = (2A_i - 1)[Q(A,X) - Q(A',X)] - X_i = -X_i$: se viene aggiornato un neurone che vale 0, esso rimane invariato e rimane $Q(A,X) = Q(A',X)$ e la prima volta che viene aggiornato un neurone che vale 1 esso diventa 0, e a quel punto i due overlap non sono più uguali; è maggiore l'overlap col pattern (A o A') che ha il valore 0 in quella posizione, e da quel momento in poi il sistema tende verso quel pattern.

DOPPIO PATTERN A E B, SOGLIA NULLA

Dimostrazione
Dato $I_i(X) = (2A_i - 1)[Q(A,X) - Q(A',X)] + (2B_i - 1)[Q(B,X) - Q(B',X)] - 2X_i, P \subset \{A, B, A', B'\}$, dimostriamo che i pattern base siano punti fissi:
$I_i(A) = (2A_i - 1)Q(A,A) + 2(B_i - 1)[Q(B,A) - Q(B',A)] - 2A_i$, ma $Q(B',A) = Q(A,A) - Q(B,A)$ quindi $I_i(A) = (2A_i - 1)Q(A,A) + 2(B_i - 1)[2Q(B,A) - Q(A,A)] - 2A_i = 2(A_i - B_i)Q(A,A) + 2(2B_i - 1)Q(B,A) - 2A_i$. Le varie combinazioni sono: $A_i = B_i = 1 \to I_i = 2Q(B,A) - 2 \geq 0$, $A_i = 1, B_i = 0 \to I_i = 2Q(A,A) - 2Q(B,A) - 2 \geq 0, A_i = 0, B_i = 1 \to I_i = -2Q(A,A) + 2Q(B,A) \leq 0, A_i = B_i = 0 \to I_i = -2Q(B,A) - 2 \leq 0$, con il nuovo valore che combacia in ogni caso con A_i. Quindi, se $Q(X(0),P)$ è maggiore dell'overlap con gli altri 3 pattern di base, allora $X \to P$.

FORMULE UTILI: EVOLUZIONE DA UNO STATO ARBITRARIO

Sia Q(A,X) maggiore dell'overlap con gli altri pattern base, e sia $Q(B,X) \geq Q(B',X)$, per definizione si ha $Q(A',X) = Q(X,X) - Q(A,X)$ (stessa cosa per B), quindi $Q(B',X) = Q(X,X) - Q(B,X) > Q(X,X) - Q(A,X) = A(A',X)$ e per finire $Q(A,X) - Q(A',X) < Q(B,X) - Q(B',X)$. Dato che si ha a che fare con valori interi di Q, $Q(A,X) \geq Q(B,X) + 1, Q(A',X) \leq Q(B',X) + 1$ quindi $Q(A,X) - Q(A',X) \geq Q(B,X) - Q(B',X) + 2$.

X tende al pattern base con cui ha overlap maggiore: sia $I_i(X) = (2A_i - 1)(Q_{AX} - Q_{A'X}) + (2B_i - 1)(Q_{BX} - Q_{B'X}) - 2X_i$, supponiamo che X abbia overleap con A maggiore. I_i è la somma di un termine $(2A_i - 1)$ moltiplicato per un valore positivo, che tende a far assumere al neurone i il valore A_i, e la somma di altri due termini (il cui valore nel caso peggiore non basta a togliere A_i) e questo porta il tutto a A_i: X tende a A.

EQUISOVRAPPOSIZIONE

Si esamina ora il caso di equisovrapposizione di due pattern: sia $Q(A,X) = Q(B,X) > Q(A',X), Q(B',X), I_I(X) = (2A_i - 1)\big(Q(A,X) - Q(A',X)\big) + (2B_i - 1)\big(Q(B,X) - Q(B',X)\big) - 2X_i$ ma $Q(A',X) = Q(X,X) - Q(A,X)$ quindi anche $Q(A',X) = Q(B',X)$ per cui $I_i(X) = \big((2A_i - 1) + (2B_i - 1)\big)\big(Q(A,X) - Q(A',X)\big) - 2X_i \rightarrow I_I(X) = 2\Big((A_i + B_i - 1)\big(Q(A,X) - Q(A',X)\big) - X_i\Big)$, e vi sono quindi 8 casi possibili per A_i, B_i, X_i. In particolare, ci sono due casi in cui X_i senza modificare l'uguaglianza di overleap, e in due casi modifica l'uguaglianza (sono i casi in cui $X_i = 1$ e gli altri hanno valori diversi): se non esiste nessuna cella in cui $X_i = 1, A_i \neq B_i$ allora il pattern X tende all'AND dei pattern A e B. Questo comportamento genera un nuovo attrattore.

PIÙ PATTERN

Se i pattern sono più di due, non è possibile garantire che essi siano punti fissi. Il primo termine stabilizza un attrattore, ma il contributo degli altri può superarlo: caso abbastanza estremo, richiede che i pattern base siano molto simili. Se invece i pattern sono scorrelati, è possibile memorizzare diversi stati con $Q(A,A)$ che domina la somma (A è quindi l'attrattore dominante). La funzione di input è una somma, su tutti i pattern del training set, di termini come $(2A_i - 1)\big(Q(A,X) - Q(A',X)\big)$. Se lo stato iniziale X ha una forte sovrapposizione con A e una modesta con gli altri pattern, allora il termine $(2A_i - 1)Q(A,X)$ domina la somma e X tende a A (il pattern a cui assomiglia di più). Se l'input netto è dominato dal termine $(2A_i - 1)Z, Z > 0$, il sistema tenderà a A_i.

Pattern simili tendono a dare origine a fenomeni di interferenza, la cui entità dipende dai pattern: se tutti i pattern sono casuali e scorrelati, $pr\{A_i^m = 1\} = pr\{A_i^m = 0\} = 1/2$ allora si può apprendere un numero di pattern $M_{\max} \approx \alpha N$ $(\alpha = 0.14)$.

Il modello possiede quindi la capacità di eseguire riconoscimenti robusti in presenza di rumore o malfunzionamento di qualche neurone: gli stati spuri sono perturbazioni (dal punto di vista del controllo) e rappresentano elaborazioni che il sistema astrae sulla base di esempi (punto di vista dell'emergenza).

Se vengono forniti molti esempi di U, non perfetti, emerge sempre un U preciso. Da tanti esempi errati, viene sempre ricostruito perfettamente (il sistema è in grado di far emergere gli archetipi). Il modello di Hopfield ha riacceso l'interesse per le reti neurali, e mette in evidenza la robustezza di un sistema di questo tipo. Caratterizzato da elaborazione basata su attrattori, evidenzia in modo chiaro la relazione fra proprietà dinamiche e prestazioni della rete, e consente una approfondita analisi della dinamica.

PERCETTRONI

Un percettrone è un tipo di classificatore binario che mappa i suoi ingressi in un valore di output.

PERCETTRONI BOOLEANI

Il modello di Hopfield possiede diverse limitazioni, ovvero un numero di pattern memorizzabili che scala linearmente con il numero di neuroni, ma il costo computazionale è N^2, interferenze tra pattern, nessuna distinzione tra input/interni/output, elevato numero di connessioni e simili.

Una struttura alternativa è quella a strati: vi è uno strato di neuroni di ingresso, diversi strati interni dal significato non predefinito e uno strato di output che esegue la classificazione finale. La rete più semplice è quella a 2 strati, uno di input e uno di output, e si può usare un neurone booleano per distinguere gli input e realizza una regola di decisione che corrisponde alla suddivisione dello spazio delle features mediante un iperpiano. Nel caso in cui l'input sia binario, eseguire una classificazione equivale a determinare una funzione booleana che vale 1 in corrispondenza dei casi sopra la soglia e 0 in caso diverso, e ha un'interpretazione geometrica $\sum w_k x_k - T = 0$ (T è la soglia, formalmente un nuovo neurone di input con x=1 e w=-T) è l'equazione di un iperpiano di dimensioni N-1 in uno spazio a N dimensioni, con la regola del neurone a soglia che associa l'uscita 11 (le possibili uscite sono 00, 01, 10, 11, parlando di funzioni logiche come AND, OR, ecc) a tutti i punti del semipiano superiore, a destra. Per lo XOR, sono necessarie invece due rette in quanto la funzione e identità non sono linearmente separabili, ovvero si incrementa la complessità del percettrone aumentandone gli strati. I percettroni vengono usati in molti campi, ad esempio nell'analisi dei segnali (i vari input del percettrone sono tutti neuroni legati a vari punti di lettura), oppure nella diagnosi di malattie (ogni percettrone ha come input vari neuroni che contengono l'intensità dei vari sintomi). Per l'apprendimento, inizialmente i pesi w assumono valori casuali, e si hanno a disposizione alcuni esempi per l'addestramento, e in corrispondenza dei casi la rete fornisce la sua classificazione: se questa è corretta non si fa nulla, se è sbagliata si modificano i pesi in modo da migliorare le prestazioni (come in Hopfield, ma è graduale e dipendente dalle risposte della rete). Per trovare un valore di pesi che consenta la realizzazione di una funzione (nel caso in cui il percettrone sia a N ingressi, singola uscita booleana e senza strati intermedi), diventa $y = H(\sum w_k x_k - T)$.
Sia y l'output del percettrone, y_d quello desiderato e $\epsilon > 0$, e si scelga un vettore iniziale di pesi $W(0)$, e finchè W non resta costante per un certo numero di volte consecutive, si ripete, per ogni pattern di input X, l'applicazione del percettrone ($y = H(\sum w_k x_k - T)$), se $y = y_d \rightarrow\ W(k+1) = W(k), y < y_d \rightarrow\ W(k+1) = W(k) + \epsilon x, y > y_d \rightarrow W(k+1) = W(k) - \epsilon x$.
Se l'output ottenuto y è inferiore a quello voluto y_d, la modifica fa si che la volta dopo $W(k+1)\ * X = W(k) * X + \epsilon |X|^2 > W(K) * X$ per cui l'input è maggiore, ed è possibile che superi la soglia e fornisca quanto desiderato (y=1, se $y > y_d$ allora si sottrae epsilon).

Se la funzione booleana è una funzione lineare a soglia (separabile linearmente) allora la regola locale di apprendimento del percettrone può trovare un insieme di pesi che la realizza in un numero finito di passi con ϵ costante o $\approx 1/k$. L'apprendimento globale segue bene o male i passi presentati prima (si presentano tutti i pattern, e si cambia poi alla fine), ma su tutta la rete, ed è diviso nella fase di addestramento, in cui si aggiornano i pesi sinaptici usando training set (usati per adattare i pesi della rete) e test set (insieme su cui viene valutata la qualità dell'apprendimento), e in fase di riconoscimento, dove lavora a pesi costanti.

PERCETTRONI CONTINUI

Esiste un efficace algoritmo di apprendimento anche per le reti a più strati che si applica a neuroni continui, con i neuroni booleani possono venire considerati tali con una funzione di attivazione sigmoide ripida (squasching functions). Nel caso dei nodi continui, l'apprendimento è sempre composto da una presentazione del pattern di un training set, dalla verifica della risposta e della eventuale correzione. Il training algorithm prevede che venga presentato uno stato di ingresso x_1, calcoliamo l'output e lo confrontiamo con valore desiderato. Si modificano i pesi, in modo da ridurre l'errore $E_i = \left(y - y_i^d\right)^2, y_i^d$ è il valore desiderato in corrispondenza di quel pattern, con l'errore globale che diventa la somma di tutti gli errori, e diventa quindi un problema di ottimizzazione.
Regola: Delta rule
La regola $\Delta W = -\frac{\epsilon dE}{dw}$ implica che se y maggiore di y_d allora Δw è proporzionale a -x, diversamente a x, come il percettore booleano, diventando quindi l'analogo continuo della regola booleana del percettrone.

La funzione costo di E dipende da tutti i pesi della rete, e per cercare il minimo una tecnica è quella del gradiente, dove si calcola lungo quale direzione la funzione diminuisce più rapidamente e si sposta il valore delle variabili di una certa quantità in quella direzione. E' possibile avere anche diversi neuroni di output, definendo una distanza tra il pattern di uscita ottenuto e quello desiderato, tentando di minimizzare la distanza, e si usa il quadrato della distanza euclidea per determinare l'errore su uno degli esempi del training set di R neuroni in output $E(W) = \sum_{r=1}^{R}(y_r - d_r)^2$.

Le funzioni booleane sono $f(n) = 2^{2n}$, quelle separabili sono $c(n) = 2 * \frac{2^{n^2}}{n!}$ (dove n è il numero di neuroni), e $\lim_{n\to\infty} \frac{c(n)}{f(n)} = 0$. Il percettrone multistrato ha quindi dei limiti, in particolare può imparare solo un numero limitato di casi, anche se si possono mettere in parallelo più percettroni e si rende necessario inserire neuroni intermedi che realizzano una rappresentazione interna dell'input.

MULTI-LAYER PERCEPTION

Si consideri un percettrone con strati nascosti e neuroni interni e di output continui, mentre quelli di ingresso sono booleani. I percettroni a più strati vengono addestrati mediante un procedimento analogo a quello di quelli semplici: come è stato detto, l'apprendimento diventa un problema di ottimizzazione $E(W) = \sum_{r=1}^{R}(y_r - d_r)^2$, dove E è la funzione dell'insieme dei parametri W. Un metodo interessante si basa sul metodo della discesa del secondo gradiente, dove le correzioni ai pesi che arrivano allo stato di output si calcolano con la delta rule, e sul metodo di retropropagazione del gradiente, che consente di calcolare le modifiche ai pesi del primo strato note quelle dell'ultimo strato. Il meccanismo permette di calcolare le modifiche ai pesi dei neuroni nascosti: dopo aver presentato un pattern, ne viene presentato un altro, e la validità delle presentazioni della rete viene poi valutata su un insieme di esempi nuovi che la rete deve ancora incontrare. L'apprendimento globale è una discesa secondo gradiente (può essere intrappolato in minimi locali, usa tutte le informazioni prima di modificare i pesi), e quello locale sovrappone del rumore alla discesa (la forma della funzione energia può essere diversa da quella complessiva, e risulta possibile muoversi contro il gradiente e permette di fuggire dai minimi locali).

Ogni funzione continua a valori reali può essere approssimata uniformemente da un percettrone con uno strato intemedio, un neurone di output e funzioni di trasferimento sigmoidali.

Quando si chiede alla rete di inferire una relazione partendo da una sua parziale specificazione, si può non avere una soluzione univoca, ovvero possono esserci diverse risposte ugualmente corrette: il numero di esempi deve essere adeguato per garantire una soluzione univoca (tale numero è ignoto) e devono essere rappresentativi.

Definizione: Overfitting/Underfitting
Utilizzando un numero elevato di neuroni è possibile prevenire ad una classificazione molto precisa dell'insieme di addestramento, con i confini di separazione fra le classi che possono essere molto frastagliati e tali da adattarsi all'insieme dei casi, anche se una rete simile può avere scarse capacità di generalizzazione. L'undefitting è una rete che non ha imparato abbastanza, mentre l'overfitting è una rete che ha imparato troppo e ha acquisito variazioni non importanti dei dati (il che porta ad avere previsioni erronee).

Esistono molti modelli di reti oltre a quelli esaminati, come reti a strati con feedback, reti auto-organizzate di Kohonen, modelli ART, e negli ultimi anni l'interesse si è riacceso sul Deep Learning, grazie a miglioramenti tecnici (aumento dei layer) e alla alta disponibilità di dati su cui fare gli addestramenti.

VITA ARTIFICIALE

La vita artificiale cerca di capire i fenomeni del mondo vivente mediante la loro riproduzione in sistemi artificiali, ovvero simulazioni o sistemi come i robot. La vita artificiale guarda in modo unificato a tutti i fenomeni del mondo vivente, come la biologia, le scienze cognitive e sociali.
Il rivolgersi all'evoluzione naturale è legato al fatto che sia molto efficace, con regole di "selezione naturale" e "fitness", e permette adattabilità e robustezza (la selezione permette di "guidare" la popolazione in zone in cui questa

può sopravvivere) ed è permessa da una codifica delle caratteristiche degli individui, con i geni che sono i blocchi costruttivi.

Utilizzando dei mattoni base (dotati di peso), delle strutture connettive e dei connettori che collegano tra di loro dei motori, si può lasciare il GA creare delle strutture fisiche più o meno funzionali. Se i mattoni sono i neuroni, le strutture connettive le sinapsi, il GA può fare la stessa cosa facendo però evolvere delle menti.