Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Machine learning mediante Programmazione Genetica e induzione secondo Solomonoff, e sua applicazione alla previsione di serie temporali e alla regressione simbolica

I. De Falco, D. Maisto

RT-ICAR-NA-2005-18 **dicembre 2005**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, I-80131 Napoli, Tel: +39-0816139508, Fax: +39-0816139531, e-mail: napoli@icar.cnr.it, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Ivanoe De Falco, Domenico Maisto

Rapporto Tecnico N.: ***Data:***
RT-ICAR-NA-2005-18 **dicembre 2005**

I rapporti tecnici dell'ICAR-CNR sono pubblicati dall'Istituto di Calcolo e Reti ad Alte Prestazioni del Consiglio Nazionale delle Ricerche. Tali rapporti, approntati sotto l'esclusiva responsabilità scientifica degli autori, descrivono attività di ricerca del personale e dei collaboratori dell'ICAR, in alcuni casi in un formato preliminare prima della pubblicazione definitiva in altra sede.

Sommario

È stato svolto uno studio che affronta il problema della fondazione degli algoritmi evolutivi come un metodo di machine learning di grande potenza e generalità. Due sono i punti chiave: la teoria di Solomonoff sull'induzione probabilistica e la programmazione genetica con grammatiche context-free (CFG-GP). Quanto discusso in via teorica vedrà una sua realizzazione concreta con un programma evolutivo con grammatiche context-free (CFG) e avrà una fase di sperimentazione su alcuni problemi di inferenza induttiva. Nel paragrafo introduttivo, il primo, ci sarà una prima presentazione del problema e saranno delineati gli obiettivi di questo paper. Il paragrafo 2 contiene alcuni cenni alla teoria dell'induzione di Solomonoff per insiemi non ordinati e avanzeremo delle ipotesi applicative per il cosiddetto operatore Induzione. Il programma sviluppato verrà invece presentato nel paragrafo 3. Saranno tracciate le linee guida della sua implementazione in linguaggio Java. Nel paragrafo 4 sono riportati i Risultati Sperimentali. Viene discusso il metodo seguito per le prove empirico-sintetiche. Saranno mostrate le esecuzioni del programma per un'istanza di un particolare problema di regressione simbolica: la determinazione empirica di una legge scientifica. Successivamente saranno esaminati problemi di induzione su insiemi ordinati iniziando dalla serie di Hènon per poi passare alla serie temporale della concentrazione di ozono registrata nella città di Arosa. Per terminare presenterò le conclusioni del lavoro svolto e alcuni importanti spunti per studi futuri.

1 Introduzione

Nell'immensa varietà di sottocampi compresi attualmente nell'Intelligenza Artificiale, il settore dell'apprendimento automatico o *machine learning* riveste un ruolo fondamentale. In questi ultimi anni sono state sviluppate molte applicazioni di successo come, sistemi di filtraggio delle informazioni, procedure di *data mining*, riconoscimento del linguaggio parlato, fino ad arrivare a sistemi bioinformatici fondamentali, ad esempio, per il progetto Genoma. Allo stesso tempo ci sono stati importanti progressi nella teoria e negli algoritmi che formano le basi di questa disciplina. Una caratteristica dell'apprendimento automatico è quella di spaziare in molti campi tra cui la statistica, la teoria dell'informazione, la complessità computazionale, la teoria del controllo, le scienze cognitive e la biologia.

Il tipo di machine learning può essere classificato secondo la rappresentazione usata per un problema, gli operatori che modificano tale rappresentazione e l'informazione presentata al sistema per realizzare l'apprendimento automatico. Sebbene gli approcci più usati siano da sempre quello analitico-deduttivo e quello connessionista, negli ultimi anni gli algoritmi evolutivi si stanno imponendo come nuovo metodo generale di machine learning, anziché come una consolidata tecnica di ottimizzazione euristica. Tra questi, l'avvento recente di programmi genetici fortemente tipati e l'uso di grammatiche formali hanno permesso di introdurre un linguaggio esplicito e dei bias, necessari per restringere lo spazio di ricerca e per risolvere problemi di alta complessità.

I sistemi di programmazione genetica, in particolare, cercano di apprendere semplici programmi per attendere a qualche compito specifico. Il problema è presentato come una serie di esempi dai quali il sistema deve ricostruire una teoria generale, una relazione funzionale per l'appunto. Tale approccio è un

classico esempio di ragionamento induttivo [Mic84], classificabile in machine learning tra i casi di *apprendimento per esempi*.

Il ragionamento induttivo ha per scopo, dato un insieme di esempi riguardanti un certo concetto, l'astrazione per induzione, cioè inferendo regole o teorie generali a partire da da tali istanze iniziali, di un modello, potremmo dire una caratterizzazione del concetto stesso, in base al quale cercare di prevedere successive descrizioni. Gli esempi menzionati possono essere positivi, cioè rappresentare istanze del concetto, o anche negativi, cioè rappresentare istanze non appartenenti al concetto.

Lo studio del ragionamento induttivo coinvolge molteplici campi che vanno dalla probabilità classica alla teoria della calcolabilità. Il suo sviluppo storico inizia con la teoria bayesiana sull'inferenza induttiva.

Secondo la teoria di Bayes, occorre assegnare a ogni possibile ipotesi una *probabilità a priori* rappresentante il grado iniziale di verità delle informazioni in loro racchiuse. La tesi bayesiana è stata ampiamente applicata ma, allo stesso tempo, continuamente criticata, proprio in relazione alla natura delle probabilità a priori.

Nella prima metà del secolo scorso, lo studio della nozione di algoritmo e la nascita della teoria della calcolabilità, offrirono nuovi strumenti di indagine e rivoluzionarie prospettive di analisi. In base a queste, l'obiettivo dell'inferenza induttiva può essere considerato come l'inferenza di un processo effettivo sottostante — una macchina di Turing, secondo la tesi di Church-Turing — che ha generato un insieme iniziale di oggetti, su di un nuovo oggetto.

Sfruttando questa idea, Solomonoff, negli anni '60 [Sol64], trovò che la complessità di Kolmogorov — la cui teoria si delineava in quegli stessi anni — costituiva una nozione ausiliare per risolvere l'annosa disputa intorno alle probabilità a priori. Egli usò i risultati degli studi di Kolmogorov per determinare una singola *distribuzione di probabilità universale* $\mathbf{M}$ capace di approssimare una qualunque distribuzione di probabilità *a priori* μ.

Facendo uso della teoria di Solomonoff, è possibile mostrare — ed è il fine di questo lavoro — che un algoritmo che simula i meccanismi dell'evoluzione naturale può essere annoverato tra i più generali quanto efficaci strumenti informatici di inferenza induttiva. Infatti, le nuove popolazioni sono ottenute da quelle precedenti per mutazione e/o crossover. Tali operatori possono essere precipuamente generalizzati: la mutazione può essere considerata come un'induzione da un campione composto da un elemento, e il crossover come un'induzione da un campione di due elementi.

La categoria di algoritmi evolutivi a cui mi sono interessato è la *programmazione genetica*(GP). La sua scelta è dovuta al fatto che, in tale procedura, le possibili soluzioni sono codificate sotto forma di alberi che rappresentano dei programmi. Il processo evolutivo, dunque, agisce modificando queste strutture ed eseguendo i programmi corrispondenti per stimarne il fenotipo. In questo contesto è stato adoperato un approccio alla programmazione genetica, proposto da Whigham, che utilizza una grammatica context-free per descrivere il linguaggio dei programmi da generare.

Come qualsiasi algoritmo evolutivo, anche la GP lavora generando una popolazione di individui, ognuno dei quali rappresenta una possibile soluzione del problema in esame, e facendola evolvere affinché gli individui che la compongono si avvicinino di generazione in generazione alla soluzione cercata. È noto che per concentrare l'evoluzione dall'esplorazione dell'intero spazio delle solu-

zioni verso quelle più vantaggiose occorre una *funzione di fitness* che ne valuti la 'bontà' in relazione al problema esaminato.

Dimostrare che gli algoritmi evolutivi sono dei sistemi induttivi generali richiede l'introduzione di una opportuna funzione di fitness in grado di guidare la procedura di ricerca di una funzione verso la massimizzazione della probabilità $\mathbf{M}$ e dunque anche della probabilità reale μ. Ebbene, farò vedere che dall'espressione di una distribuzione di probabilità su un qualsiasi insieme campione, si ricava, direttamente e in maniera non arbitraria, una funzione di fitness la cui espressione è composta da due termini, dei quali l'uno calcola l'errore quadratico medio commesso dall'individuo rispetto alla funzione obiettivo, l'altro valuta la probabilità a priori dell'individuo stesso. In questa ipotesi, al minimo globale della funzione di fitness corrisponde la probabilità massima che il nuovo esempio appartenga all'insieme iniziale.

2 Il ragionamento induttivo secondo Solomonoff

L'Inferenza Induttiva (II) è il processo logico più adatto per trattare problemi di modellizzazione e previsione. Un esempio chiarirà le sue potenzialità e le problematiche legate alla sua applicazione.

Consideriamo una stringa simbolica iniziale x, composta dai simboli di un alfabeto $\mathcal{A}$, che descrive la sequenza di coppie (q_i, a_i), $i = 1, \ldots, n$, con gli elementi q_i e a_i che rappresentano la descrizione di un problema e la sua soluzione; assumiamo una distribuzione di probabilità μ sullo spazio fondamentale costituito da tutte le possibili sequenze infinite che iniziano con x, e chiamiamo $\mu(x)$ la probabilità che x sia la stringa iniziale di una qualche sequenza. Allora la probabilità che, dopo la stringa x, il simbolo successivo sia a è chiaramente $\mu(a|x) = \mu(xa)/\mu(x)$.

Il problema principale dei metodi induttivi è, però, a monte. Ed è, come già accennato, un problema cruciale e controverso per l'applicazione della regola di Bayes: la determinazione della probabilità *a priori*. Infatti, in generale non si conosce la distribuzione μ, e ciò impedisce di assegnare una probabilità alla stringa xa. È necessario quindi tentarne una stima.

Per risolvere tale problema, Solomonoff, adoperando una variante, detta a prefisso[1] e che indicheremo con il simbolo K_p, della nozione di complessità algoritmica di Kolmogorov, ha proposto quella che egli ha chiamato *distribuzione universale a priori* $\mathbf{M}$. È possibile dimostrare che $\mathbf{M}(x)$ vale all'incirca $|\mathcal{A}|^{-K_p}$, dove $|\mathcal{A}|$ è la cardinalità dell'alfabeto $\mathcal{A}$. La distribuzione universale di probabilità a priori permette di approssimare con estrema accuratezza qualsiasi, purché calcolabile, distribuzione a priori di probabilità, come sancisce la sua più importante proprietà discussa nel sottoparagrafo 2.1.2: la somma degli errori quadratici commessi stimando con $\mathbf{M}$ la probabilità μ, per ogni elemento di x, è limitata superiormente da una costante che dipende esclusivamente da μ.

2.1 L'operatore Induzione

Nell'arco di cinque decadi di ricerca, Solomonoff ritorna più volte sull'opportunità di sviluppare un metodo di previsione probabilistica che sia il più generale

[1] La complessità *a prefisso* di una stringa x è la lunghezza del programma p più breve tale che inserito in un macchina di Turing universale U, esiste una stringa ω per cui $U(p) = x\omega$.

possibile [Sol57, Sol75, Sol86].

Tra i problemi affrontati c'è anche l'estrapolazione su un insieme non ordinato di coppie ordinate di elementi, che possono essere stringhe e/o numeri. Ossia, dato il primo elemento di una nuova coppia, trovare una distribuzione di probabilità per un possibile secondo elemento della coppia.

La natura delle coppie ordinate, denotate d'ora in poi con (Q, A), può essere la più varia. Le Q possono essere dei quesiti in qualche linguaggio formale o naturale e le A possono essere associate alle risposte. Le Q possono essere gli input di un processo stocastico sconosciuto e le A possono essere gli output (Problema di Identificazione). E ancora, le Q possono essere le descrizioni di figure geometriche piane e le A possono dirci se sono convesse o concave (Problema di Categorizzazione). Le Q possono essere dei numeri, le A i valori di qualche funzione per quei numeri (Problema di *Curve Fitting*).

Il problema della determinazione di un operatore Induzione — introdotto da Solomonoff, sulla base delle soluzioni da lui trovate nell'ambito della teoria della probabilità algoritmica, per problemi di induzione su insiemi non ordinati — si risolve nella seguente richiesta: sia $[Q_i, A_i]$ un insieme non ordinato di n coppie di stringhe e/o numeri. Dato un nuovo elemento Q_{n+1}, qual'è la distribuzione di probabilità su tutti i possibili A_{n+1}? Si risponderà a questa domanda per gradi.

Per iniziare, accennerò all'impostazione formale per la determinazione di una distribuzione di probabilità algoritmica universale per insiemi non ordinati e ne mostrerò una forma adatta all'applicazione effettiva[2]. Concluderò questo sottoparagrafo esaminando l'accuratezza delle previsioni ottenute con la distribuzione di probabilità trovata[3].

2.1.1 Applicazione dettagliata dell'operatore Induzione

Consideriamo un insieme, in generale non ordinato, di coppie (Q_i, A_i) con $i = 1 \dots n$. Assegnato un nuovo punto Q_{n+1}, si vuole conoscere la distribuzione di probabilità del punto A_{n+1} corrispondente.

Dal teorema di Bayes, sostituendo l'espressione della probabilità a posteriori per gli A_{n+1},

$$\mathrm{P}(C_j|A_{n+1}) = \frac{\mathrm{P}(C_j)\,\mathrm{P}(A_{n+1}|C_j)}{\mathrm{P}(A_{n+1})},$$

nella condizione di normalizzazione

$$\sum_j \mathrm{P}(C_j|A_{n+1}) = 1,$$

si ottiene la distribuzione di probabilità a priori di A_{n+1}:

$$\mathrm{P}(A_{n+1}) = \sum_j \mathrm{P}(C_j)\mathrm{P}(A_{n+1}|C_j) \tag{1}$$

Definendo la classe C_j come l'insieme dei punti A_i che, fissata la funzione F^j, soddisfano la relazione $F^j(Q_i) = A_i$, e supponendo che ciascuna coppia

[2] Si faccia riferimento al report [Sol02].

[3] Si veda l'appendice "A" del report [Sol02].

(Q_i, A_i) sia indipendente dalle altre, posso riscrivere la (1) come:

$$\begin{aligned} P(A_{n+1}) &= \sum_j a_0^j \ \mathrm{O}^j(A_1 \dots A_{n+1}|Q_1 \dots Q_{n+1}) \\ &= \sum_j a_0^j \prod_{i=1}^{n+1} \mathrm{O}^j(A_i|Q_i) \end{aligned} \tag{2}$$

dove $\mathrm{O}^j(\cdot|\cdot)$ è la distribuzione di probabilità condizionale rispetto alla funzione F^j e a_0^j è la probabilità a priori associata a F^j.

Quest'ultima vale, approssimativamente, $|\mathcal{A}|^{-\,l(F^j)}$ dove $|\mathcal{A}|$ è la cardinalità dell'alfabeto $\mathcal{A}$ usato per descrivere le F^j, e $l(F^j)$ è la lunghezza del più breve programma *self-delimiting*[4] di F^j.

Si può dare la (2) nella seguente forma equivalente:

$$\mathrm{P}(A_{n+1}) = \sum_j a_n^j \ \mathrm{O}^j(A_{n+1}|Q_{n+1}), \tag{3}$$

con

$$a_n^j = a_0^j \prod_{i=1}^{n} \mathrm{O}^j(A_i|Q_i). \tag{4}$$

Chiameremo la (3) **operatore Induzione**.

Dall'operatore Induzione (3) e dalla (4), si deduce che la distribuzione di probabilità $\mathrm{P}(A_{n+1})$ è la somma di tutte le distribuzioni di probabilità condizionali $\mathrm{O}^j(A_{n+1}|Q_{n+1})$ associate alle funzioni F^j, pesate con il prodotto delle rispettive probabilità a priori e delle probabilità dei dati osservati in ragione di F^j.

Naturalmente è impossibile calcolare la somma infinita (3) con risorse finite, ma si può approssimarla usando un numero finito di termini che hanno un valore di a_n^j elevato. L'ideale sarebbe includere i termini di massimo peso a_n^j, ma la determinazione dei massimi pesi a_n^j è legato al problema della ricerca delle probabilià a priori a_0^j di valore massimo, cioè alla ricerca dei programmi u *self-delimiting* la cui lunghezza $l(u)$ è la più piccola possibile.

Il punto è che non esiste una procedura effettiva che espleta questa ricerca, in quanto tra i programmi u vi possono essere alcuni che non terminano, senza poterlo stabilire a priori. Il problema dunque, ha un grado di indecidibilità superiore o uguale a quello del Problema della Fermata.

Per quanto detto, l'applicazione dell'operatore Induzione si risolve nel cercare di trovare, in un tempo prefissato e arbitrario, un insieme di funzioni F^j tali che la somma:

$$\sum_j a_n^j \tag{5}$$

sia la maggiore possibile. Con un ulteriore approssimazione, potremmo spingerci a cercare, in un tempo finito, tra le a_n^j, il termine che domina la somma (5).

Si noti che l'approccio seguito è quello del metodo di inferenza bayesiana, per cui, migliore è la stima a priori, più consistenti sono le previsioni. In questo

[4]Un programma di una certa stringa è detto *self-delimiting* se non è prefisso di nessun altro programma della stessa stringa. Il più breve programma *self-delimiting* di una stringa è il suo programma canonico rispetto a K_p.

modo, utilizzando la (3), riusciamo a esprimere la probabilità di un'arbitraria A_{n+1} direttamente come funzione dell'insieme $[Q_i, A_i]$.

Affinché si possa usare l'operatore Induzione, non rimane che stabilire quali espressioni usare per le $\mathrm{O}^j(\cdot|\cdot)$ e per le a_0^j. Affronteremo le due questioni separatamente, rispettivamente nel sottoparagrafo 2.2 e nel paragrafo 3, sottoparagrafo 3.2.1.

2.1.2 Teorema della convergenza per l'operatore Induzione

Faremo vedere che, per una adeguata sequenza di coppie (Q_i, A_i), le previsioni ottenute dalla distribuzione di probabilità dell'equazione (2) sono estremamente soddisfacenti.

Ipotizziamo che la sequenza delle risposte osservate A_i siano state generate da un algoritmo probabilistico $\mu(A_i, Q_i)$, e che la μ possa essere descritta da una stringa binaria dell'alfabeto $\{0, 1\}$, di lunghezza k, utilizzando una macchina di riferimento che assegna una probabilità a priori a ogni funzione parziale ricorsiva.

Qualunque distribuzione di probabilità che assegni una probabilità a ogni possibile A_i, deve anche assegnare delle probabilità a ciascuno dei simboli della stringa che descrive A_i. Supponiamo che a_r sia la stringa composta dai primi r simboli di A_i. Allora la probabilità, rispetto a μ, che l'$(r+1)$-esimo simbolo di A_i sia 0 è

$$\sum_j \mu(a_r 0 x^j | Q_i) \Big/ \sum_j \mu(a_r x^j | Q_i)$$

con x^j che varia su tutte le stringhe finite.

Allo stesso modo, la probabilità P della formula (2) può essere usata per assegnare una probabilità a ogni simbolo di ciascuna A_i.

Rappresenteremo la sequenza delle A_i con una stringa Z, composta dalle stringhe che descrivono le A_i separate dal simbolo delimitatore b. Z, quindi, è una stringa di simboli appartenenti a un alfabeto ternario $\{0, 1, b\}$. È chiaro che, sia μ sia P, possono assegnare — come per 0 e 1 — una probabilità al simbolo b.

Abbiamo, dunque, due distribuzioni di probabilità sulla stringa ternaria Z. La prima distribuzione, μ, ha generato realmente la sequenza osservata, mentre la seconda cerca di prevedere i simboli che costituiscono Z.

Richiamiamo ora un teorema fondamentale, dovuto a Solomonoff, della teoria della Probabilità Algoritmica: il **teorema della convergenza**:

Definizione 2.1 *Sia S_n il valore atteso, rispetto a μ, del quadrato della differenza tra la probabilità μ e la probabilità* $\mathbf{M}$ *che un simbolo a, appartenente ad un qualunque alfabeto finito $\mathcal{A}$ [Hut01a, Hut01b], occorra all'n-sima previsione*

$$S_n = \sum_{l(x)=n-1} \mu(x)(\mathbf{M}(a|x) - \mu(a|x))^2.$$

Chiameremo S_n ***errore quadratico atteso all'n-sima previsione****.*

Teorema 2.1 *(**Teorema della convergenza**) Sia μ una distribuzione ricorsiva. Si ha che $\sum_n S_n \leq k/2$ con $k = K_p(\mu) \ln 2$ e $K_p(\mu)$ è la complessità a prefisso della distribuzione di probabilità μ.*

Cioè S_n converge a 0 più velocemente di $1/n$. Ciò comporta che, per $n \to \infty$,

$$\frac{\mathbf{M}(y|x)}{\mu(y|x)} \longrightarrow 1.$$

Allora, dal teorema appena proposto è possibile ricavare che:

$$\sum_l \mu(Z_l) \sum_{i=1}^{n} \sum_{j=0}^{h_i^l} \sum_{t \in \{0,1,b\}} \left(\mathrm{P}_{i,j}^l(t) - \mu_{i,j}^l(t)\right)^2 < k \ln 2, \tag{6}$$

dove l somma su tutte le stringhe Z_l composte da n stringhe binarie e finite separate dai simboli b; A_i^l è la stringa binaria che descrive A_i in Z_l; $\mathrm{P}_{i,j}^l(t)$ è la probabilità rispetto a P che il j-esimo simbolo di A_i^l sia t, condizionata dal corrispondente Q e dalle precedenti A_i^l della sequenza Z_l; $\mu_{i,j}^l(t)$ è definito in modo analogo a $\mathrm{P}_{i,j}^l(t)$, ma è indipendente dalle precedenti A_i^l; h_i^l è il numero di simboli in A_i^l (l'$(h_i^l + 1)$-esimo simbolo di A_i^l è sempre b); $\mu(Z_l)$ è la probabilità che μ assegna a Z_l in ragione della sequenza data delle Q_i; k è la lunghezza della più breve descrizione di μ.

L'espressione (6) implica che il valore atteso, rispetto a μ, dell'"errore quadratico" tra P e μ, sommato su tutti i simboli di tutte le A_i, è strettamente minore di $k \ln 2$.

Poiché il numero totale di simboli in tutte le A_i può essere molto grande, anche per un numero piuttosto piccolo di Q_i, è chiaro che l'errore commesso su un singolo simbolo decresce tanto rapidamente quanto il numero, n, di coppie (Q_i, A_i) cresce.

L'equazione (6) è una misura dell'accuratezza dell'operatore Induzione espresso nell'equazione (3). L'unica incertezza 'necessaria' è nel valore di k.

Infatti il valore della costante k dipende, in maniera critica, dal metodo utilizzato per assegnare le probabilità a priori alle funzioni F^j e a μ. La scelta del metodo migliore dipende dalle informazioni a priori che si hanno sul problema indagato e che condizionano gli esiti del processo di induzione.

2.2 Ipotesi sul modello teorico e soluzioni applicative

Avevamo subordinato l'applicazione dell'operatore Induzione (3), alla determinazione dell'espressione della probabilità condizionale $\mathrm{O}^j(\cdot|\cdot)$ e alla definizione di un algoritmo per il calcolo delle probabilità a priori a_0^j.

Cominciamo con lo stimare la distribuzione di probabilità condizionale $\mathrm{O}^j(\cdot|\cdot)$ associata alle funzioni F^j. A tale scopo, introduciamo nel nostro modello l'ipotesi che ogni elemento A_i sia dato da una funzione deterministica F^j di Q_i, più un errore, indicato con ε_j, che dipende direttamente dalla funzione F^j:

$$A_i = F^j(Q_i) + \varepsilon_j \tag{7}$$

Si assuma ora che le ε_j abbiano una distribuzione normale con valore medio zero e con deviazione standard σ indipendente da Q_i e dall'indice j. La distribuzione di ε_j è data da

$$\mathrm{p}(\varepsilon_j) = (2\pi\sigma^2)^{-1/2} \exp\left\{ -\frac{\varepsilon_j^2}{2\sigma^2} \right\} \tag{8}$$

Sostituendo la (7) nella (8), si ricava che la distribuzione di probabilità, rispetto a F^j, delle A_i, condizionate dalle Q_i, si scrive come

$$\mathrm{O}^j(A_i|Q_i) = (2\pi\sigma^2)^{-1/2} \exp\left\{ -\frac{[F^j(Q_i) - A_i]^2}{2\sigma^2} \right\} \tag{9}$$

Pertanto, sotto le ipotesi (7) e (8), le $\mathrm{O}^j(\cdot|\cdot)$ sono distribuzioni di probabilità gaussiane con deviazione standard σ.

Operando la sostituzione della (9) nella (4) si ha:

$$a_n^j = a_0^j \ (2\pi\sigma^2)^{-1/2} \exp\left\{ -\sum_{i=1}^{n} \frac{[F^j(Q_i) - A_i]^2}{2\sigma^2} \right\} \tag{10}$$

Questa nuova espressione per le a_n^j, suggerisce una via differente da quella proposta da Solomonoff [Sol02].

Anziché massimizzare la (5), procediamo minimizzando, rispetto alle funzioni F^j, il negativo del logaritmo naturale dei suoi termini:

$$-\ln a_n^j = \frac{1}{2\sigma^2} \sum_{i=1}^{n} \Big[F^j(Q_i) - A_i \Big]^2 - \ln a_0^j + \ln \sigma + \frac{\ln(2\pi)}{2} \tag{11}$$

Si osservi che per la ricerca del minimo, il terzo e il quarto termine della (11) sono ininfluenti e quindi possono essere omessi:

$$-\ln a_n^j \approx \frac{1}{2\sigma^2} \sum_{i=1}^{n} \Big[F^j(Q_i) - A_i \Big]^2 - \ln a_0^j. \tag{12}$$

In questo modo, non solo avremo a disposizione funzioni lineari delle a_n^j, che dipendono esplicitamente dalle F^j e dalle loro probabilità a priori a_0^j, ma potremo anche scrivere una procedura meno dispendiosa, in termini di risorse impiegate, che le calcoli.

Ribadiamo che la somma nella (3) dovrebbe essere calcolata su tutte le funzioni *ricorsive*, ma siccome l'insieme di tali funzioni non è effettivamente numerabile, ci 'limiteremo' a considerare solo le funzioni *parziali* ricorsive.

Il meglio che si possa fare è usare una procedura in generale non terminante, ma che fornisca come risultati intermedi un'approssimazione sempre migliore delle a_n^j, e quindi trovare un termine, il cui peso ha il maggior valore possibile, in un tempo arbitrario.

L'uso degli Algoritmi Evolutivi sembra essere un approccio ragionevole per risolvere questo problema [Sol02].

In generale gli algoritmi evolutivi, lavorano su popolazioni di individui piuttosto che su singole soluzioni e quindi effettuano una ricerca parallela nello spazio del problema. Un algoritmo evolutivo non converge necessariamente a una soluzione ottimale in un tempo finito ma, a secondo della sua efficienza, ne può fornire una buona approssimazione.

Tra gli algoritmi evolutivi, la Programmazione Genetica è certamente l'algoritmo da preferire per trovare delle espressioni funzionali, ad esempio le F^j, in grado di interpolare tutte le coppie (Q_i, A_i) massimizzando contemporaneamente la loro probabilità a priori. In merito vedremo come l'uso delle CFG

permetta di introdurre le informazioni a priori circa lo specifico problema trattato, mediante il calcolo delle a_0^j. A questo scopo, è stato implementato un algoritmo (si veda il sottoparagrafo 3.2.1 del paragrafo 3) che, previa definizione della grammatica adoperata, valuta la probabilità a priori a_0^j di ogni possibile soluzione F^j.

3 Il sistema induttivo CFG-GP

Per verificare sperimentalmente la teoria sviluppata, è stato realizzato un sistema che implementa in linguaggio Java un algoritmo di programmazione genetica ricorrendo all'approccio proposto da Whigham [Whi96] che fa uso di CFG. Il metodo fu ideato nel tentativo di superare le difficoltà riscontrate nell'applicazione della programmazione genetica a problemi spazialmente complessi. Tali problemi richiedono un linguaggio in grado di racchiudere importanti informazioni sullo spazio di ricerca, ma che, allo stesso tempo, permetta di avere prestazioni accettabili, da un punto di vista delle risorse, anche con popolazioni numerose. Da qui la necessità di adoperare una grammatica formale per generare gli alberi di derivazione che sia una rappresentazione *a priori* dello spazio di ricerca. Ne consegue che la popolazione evoluta dall'algoritmo è costituita non più dai programmi, ma dai loro alberi di derivazione associati alla grammatica specificata. Le ragioni della scelta della grammatica context-free per rappresentare la struttura dei programmi, nonché in maniera dichiarativa i biases, sono molteplici:

1. Le grammatiche context-free sono semplici e facilmente comprensibili.
2. Una grammatica context-free può essere usata per rappresentare a priori una conoscenza incompleta di un dato problema
3. È abbastanza facile usare una grammatica context-free per definire nel modo più generale possibile, un linguaggio opportuno a descrivere un problema dato. Una volta descritto il linguaggio, esso è modificabile a piacimento per rappresentare un certo bias.
4. Gli operatori genetici possono essere definiti in maniera semplice per operare sugli alberi di derivazione associati ai programmi.
5. Tutti i problemi tipicamente trattati con la programmazione genetica sono esprimibili in un linguaggio context-free.

Il suddetto sistema è stato sviluppato in maniera da supportare una certa flessibilità applicativa, al fine di poter studiare diversi tipi di problemi e di consentire la sperimentazione di diverse impostazioni dell'algoritmo evolutivo. Forniti il problema da studiare, la grammatica da usare e i parametri che regolano il processo, definiti gli operatori genetici, il programma genera una popolazione di individui e ne simula l'evoluzione per un numero fissato di generazioni. Il generatore di espressioni fornisce, come popolazione iniziale, un insieme di programmi differenti per taglia, forma e funzionalità. La sua implementazione, per mezzo di un automa che costruisce formule ben formate in una particolare CFG, assicura la correttezza sintattica dei programmi generati. Diversamente da Whigham, la scelta delle regole di produzione che formano l'albero di derivazione di un individuo è effettuata tramite una distribuzione uniforme, così da evitare di favorire

una particolare regola alle altre. L'automa principia richiamando la regola di inizio, per poi continuare applicando, una alla volta, altre produzioni permesse in quella situazione. Qualora non esistano regole applicabili o qualora la taglia del programma ecceda un valore prefissato, l'automa si arresta. Nel secondo caso, l'individuo viene scartato e rimpiazzato da uno nuovo. Il risultato della procedura è un albero in cui i nodi sono simboli non terminali, i rami sono passi di derivazione e il frutto è il programma rappresentato.

La specifica descrizione dei programmi — il fenotipo — come alberi di derivazione della CFG adottata — il genotipo — consente di effettuare i cambiamenti inferti alla struttura genotipica dagli operatori genetici con delle semplici operazioni sugli alberi. Inoltre, risulta assai facile rispettare la condizione di chiusura. È, infatti, sufficiente agire sui nodi interni dell'albero e sostituire una produzione con un'altra avente nella parte sinistra lo stesso simbolo non terminale.

Gli operatori genetici implementati per il processo evolutivo sono il crossover e la mutazione. Il crossover opera scegliendo in un primo individuo un nodo non terminale in maniera casuale; quindi seleziona in un secondo individuo, un nodo etichettato con lo stesso simbolo. Infine, scambiando i rispettivi sottoalberi, ottiene due nuovi individui figli. Il crossover non sortisce effetto se non riesce a trovare nel secondo genitore un nodo corrispondente al simbolo non terminale selezionato nel primo genitore. L'operatore di Mutazione, invece, agisce casualmente su un nodo non terminale di un individuo rimuovendo il relativo sottoalbero e sostituendolo con uno nuovo avente per radice il nodo selezionato. Per questo operatore sono state introdotte due varianti: una macro-mutazione che lavora su simboli terminali e non terminali, e un micro-mutazione che opera, invece, esclusivamente su simboli terminali.

3.1 La valutazione

La valutazione di un individuo si espleta in due fasi: nella prima il frutto del suo albero di derivazione, cioè il programma, è estratto ed eseguito su un insieme di dati di esempio, ricavando così una sequenza di valori in output; nella seconda fase l'output ottenuto è usato per calcolare il valore di fitness da assegnare all'individuo. Le espressioni valutate sono stringhe del linguaggio definito dalla grammatica prescelta. Il valutatore del sistema è in grado di interpretare qualunque espressione appartenete al linguaggio descritto dalla grammatica seguente:

$S \to D$
$D \to f(x) = E$
$E \to f(x - N) \mid (EOE) \mid F(E) \mid R \mid x \mid (\text{if}\, BEE)$
$B \to (ECE) \mid (\text{not}\, B) \mid (B \,\text{and}\, B) \mid (B \,\text{or}\, B)$
$C \to = \mid < \mid >$
$O \to + \mid - \mid * \mid /$
$F \to$ sqr | cube | pow4 | sqrt | cubert | root4 | log | exp | sin | cos | tan | asin | acos
$N \to i \in \mathbb{N}$
$R \to k \in \mathbb{R}$

L'intento di verificare la correttezza del sistema e la sua efficacia su problemi d'induzione generica, ossia definiti su insiemi ordinati e non, ha di conseguenza condizionato l'implementazione del processo di valutazione.

Per lo studio dei problemi induttivi su insiemi non ordinati, il valutatore opera con i programmi sull'insieme delle *questions* Q_i delle coppie (Q_i, A_i) — i dati empirici. I dati sono suddivisi casualmente in tre insiemi, uno di *training*($\mathcal{T}$), uno di *validation* ($\mathcal{V}$) e uno di *prediction* ($\mathcal{P}$). Il programma viene dapprima valutato su $\mathcal{T}$ e i valori ottenuti in output sono confrontati con le *answers* A_i dei dati, per calcolare la fitness. Al termine del processo inferenziale, viene selezionato il miglior programma sulla base dei valori ricavati su $\mathcal{V}$, insieme sul quale sono stimate le capacità induttive delle soluzioni. Infine, si verifica su $\mathcal{P}$ la qualità della migliore soluzione indotta.

Se, invece, si esamina un problema induttivo definito su un insieme ordinato $[Q_i, A_i]$, l'ordinamento di tale insieme implica che l'insieme delle Q_i è un intervallo. In questo caso, $\mathcal{T}$ e $\mathcal{V}$ sono generati in maniera random da un unico intervallo che precede $\mathcal{P}$. Inoltre è stato introdotto un ulteriore intervallo, detto di *seed* ($\mathcal{S}$). $\mathcal{S}$ è formato dai valori iniziali dei dati che possono essere forniti come seme ai programmi in esecuzione. La lunghezza dell'intervallo di *seed* può essere opzionalmente calcolata dinamicamente, in modo che la sua lunghezza coincide con il numero massimo di valori precedenti richiesti dal programma sottoposto a valutazione.

3.2 La funzione di fitness

Nel paragrafo 2 è stato dimostrato che un problema induttivo su un generico insieme $[Q_i, A_i]$ può essere risolto trovando una funzione F^j rispetto alla quale la (12) è la minore possibile. Consistentemente con le considerazioni effettuate e con le ipotesi fatte, la funzione di fitness adottata è:

$$F(p) = \omega \cdot \left(\frac{1}{n} \frac{1}{\sigma^2} \sum_{Q_i \in \mathcal{Q}} \left| p\left(Q_i\right) - A_i \right|^2 \right) - \ln(a_0(p)) \tag{13}$$

dove p è l'espressione valutata, σ è la deviazione standard dei dati empirici, $\mathcal{Q}$ è l'insieme delle *questions* Q_i dal quale l'inferenza induttiva procede e n il loro numero, $p\left(Q_i\right)$ è il valore del programma per l'i-esima *question*, A_i è la *answer* relativa a Q_i, ω è il peso associato al termine di errore. Per finire, $a_0(p)$ è la probabilità a priori del programma p.

Come si vede, la (13) è un'espressione funzionale di due termini che ricalca esattamente la (12). Infatti, il primo termine è il prodotto di un Mean Square Error (MSE) — errore quadratico medio — per un peso $\omega \in \mathbb{R}^+$, e valuta l'errore commesso dal programma p nell'approssimazione del problema in esame. La presenza del peso è giustificato in quanto, per le ipotesi (7), (8) e (9), la deviazione standard per il rumore ε è fissato mentre quella della MSE dipende dalla particolare soluzione considerata e, in generale, dai biases e dai parametri che guidano la ricerca del processo evolutivo. Il secondo termine, invece, dipende dalla probabilità *a priori* del programma p vagliato. Per valutarlo, partendo dalla CFG, un algoritmo calcola la probabilità *a priori* $a_0(p)$ dell'albero di derivazione che genera p.

È importante sottolineare che, sebbene la (13) possa sembrare una particolare forma di 'fitness con parsimonia', essa deriva espressamente e interamente dai risultati teorici conseguiti nei paragrafi precedenti. Il suo impiego non è una decisione dettata da osservazioni euristiche o considerazioni arbitrarie, ma dalla coerenza con la teoria sviluppata.

Come menzionato prima, $\mathcal{V}$ permette di selezionare tra tutte le soluzioni la migliore scoperta durante l'intero processo di inferenza. In fatti, a ogni generazione la (13) è calcolata sull'insieme $\mathcal{V}$ per ogni individuo appartenente alla popolazione. Per cui, il risultato di un run è l'individuo con il miglior valore di fitness sul *validation* registrato tra tutte le generazioni.

3.2.1 Calcolo della probabilità *a priori*

Il valore delle a_0 si assegna tramite una procedura che simula la regola di Laplace per eventi successivi e indipendenti. Una volta specificata la grammatica, la probabilità α_{i_p} che una data produzione è presente nell'i-esimo nodo dell'albero di derivazione del programma p è k/m, dove k è il numero di volte in cui la produzione è occorsa precedentemente nella definizione dell'albero e m è il numero totale di tutte le produzioni legali in quel punto, incrementate per il numero di volte in cui sono precedentemente comparse nella definizione dell'albero. Il prodotto

$$a_0(p) = \prod_{i_p=1}^{q} \alpha_{i_p} \,, \tag{14}$$

dà la probabilità *a priori* del programma p, frutto dell'albero di q produzioni, relativa alla grammatica scelta [Sol02].

Un caso come esempio. Supponiamo di avere la grammatica non contestuale:

$S \to D$
$D \to f(x) = E$
$E \to (EOE) \,|\, F(E) \,|\, x$
$O \to + \,|\, - \,|\, * \,|\, /$
$F \to sqr \,|\, exp \,|\, sin$

Prendiamo un'espressione appartenente al linguaggio generato da questa grammatica: mettiamo sia $f(x) = sin(x * x)$. L'albero di derivazione associato a tale espressione è mostrato in Figura 3.2.1. La stima della probabilità di ogni produzione dell'albero è riassunta nella tabella seguente:

Produzioni	Produzioni legali in quel nodo	Probabilità
$S \to D$	$S \to D$	1
$D \to f(x) = E$	$D \to f(x) = E$	1
$E \to F(E)$	$E \to (EOE) \,\|\, F(E) \,\|\, x$	$1/(1+1+1)$
$F \to sin$	$F \to sqr \,\|\, exp \,\|\, sin$	$1/(1+1+1)$
$E \to (EOE)$	$E \to (EOE) \,\|\, F(E) \,\|\, x$	$1/(1+2+1)$
$E \to x$	$E \to (EOE) \,\|\, F(E) \,\|\, x$	$1/(2+2+1)$
$O \to *$	$O \to + \,\|\, - \,\|\, * \,\|\, /$	$1/(1+1+1+1)$
$E \to x$	$E \to (EOE) \,\|\, F(E) \,\|\, x$	$2/(2+2+2)$

La probabilità *a priori* $a_0(p)$ dell'albero in fig. 3.2.1 è il prodotto delle probabilità riportate in tabella. Tale procedura risponde appieno ai risultati ottenuti dalla teoria della probabilità algoritmica, nel senso che, fissata la grammatica di riferimento, essa attribuisce probabilità maggiori a funzioni che hanno una com-

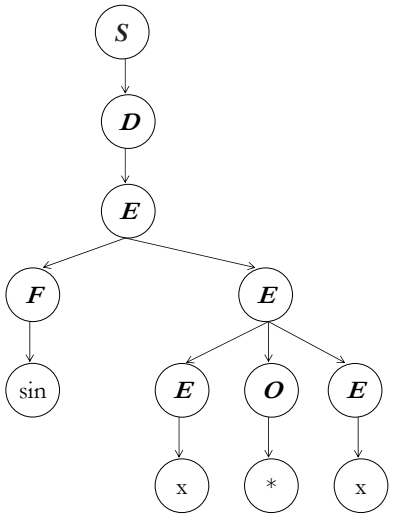

Figura 1: Albero di derivazione dell'espressione $f(x) = sin(x * x)$.

plessità descrittiva minore rispetto alla grammatica. Per cui, tra due funzioni differenti rappresentanti due possibili soluzioni a un dato problema, l'algoritmo evolutivo sceglierà quella più 'semplice', determinando un'applicazione effettiva del "rasoio di Occam".

4 Risultati sperimentali

Il sistema è stato testato su tre problemi differenti: la legge di Coulomb come esempio di inferenza induttiva su insiemi non ordinati, la serie di Hènon e la serie temporale della concentrazione di ozono nella città di Arosa (Svizzera), come esempi di inferenza induttiva su insiemi ordinati. Le esecuzioni del programma sono state guidate da un insieme di parametri, fra i quali quelli che controllano il processo evolutivo e, per le serie, quelli che specificano le lunghezze degli intervalli in cui essa è suddivisa. Altro parametro importante è il peso ω. Il suo valore deve essere scelto in maniera da favorire l'evoluzione degli individui più semplici e, allo stesso tempo, in modo da permettere la creazione di programmi sufficientemente complessi in grado di riprodurre la funzione originale.

Per tutti i problemi sono stati eseguiti 10 runs variando esclusivamente il random seed. In tutti i casi sono stati utilizzati, una popolazione composta da 200 individui, un meccanismo di selezione a torneo di 5 unità, un operatore di crossover con una probabilità di attivazione del 30%, e un operatore di mutazione (con 100% di probabilità) distinto in macro- e micro-mutazione, rispettivamente con il 30 e il 70% di attivazione. Il numero massimo di generazioni consentito per ciascun run è 1000. Dopo una fase preliminare di tuning, una per ciascuno dei problemi, ω è stato fissato nell'ordine di 10^3, e la profondità massima degli alberi è stata posta pari a 15. La grammatica adoperata in tutti e tre i casi è

mostrata in Tabella 2, con $I_R = [0, 10]$ per i primi due problemi e $I_R = [-1, 1]$ per il terzo.

Regole		
S	$\rightarrow$	$f(x) = E$
E	$\rightarrow$	$f(x - N) \mid (EOE) \mid F(E) \mid R \mid x$
O	$\rightarrow$	$+ \mid - \mid * \mid /$
F	$\rightarrow$	$sqr \mid cube \mid sqrt \mid cubert \mid log \mid exp \mid sin \mid cos$
N	$\rightarrow$	$t \in \mathcal{S} \subseteq \mathbb{N}$
R	$\rightarrow$	$c \in I_R \subseteq \mathbb{R}$

Tabella 2: La grammatica usata negli esperimenti

4.1 Legge di Coulomb

Come caso di regressione simbolica si è deciso di determinare la legge di Coulomb. La sua espressione è:

$$\vec{F}_{21} = \frac{1}{4\pi\varepsilon_0} \frac{q_1 q_2}{r^2} \hat{r}_{21} \ , \tag{15}$$

dove q_1 e q_2 sono due cariche puntiformi, r è il modulo del vettore posizione $\vec{r}_{21}$ di q_2 rispetto a q_1 e $\hat{r}_{21}$ è il suo versore, ε_0 è la costante dielettrica del vuoto. È stato calcolato il modulo di $\vec{F}$ tra un elettrone e un protone aggiugendovi un errore normale con una deviazione standard uguale a 1.0, ottenendo 60 coppie (r_i, F_i) con r_i and F_i rispettivamente in nanometri e in 10^{-8}N. Gli insiemi $\mathcal{T}$, $\mathcal{V}$ e $\mathcal{P}$ hanno le dimensioni di 40, 10 e 10. Siccome il problema è definito su un insieme non ordinato, la regola di produzione $E \rightarrow f(x - N)$ è stata disattivata.

Le statistiche sull'insieme di risultati ricavati dalle 10 prove eseguite sono mostrate nella Tabella 3.

	Miglior Individuo	**Valore Medio**	**Dev. St.**
Generazione	13	134.25	108.58
N.ro Nodi	9	13	2.42
Probabilità a priori	7.44E–6	1.45E–6	2.72E–6
Fitness	0.0193	0.0243	0.0028

Tabella 3: Statistiche dei risultati sperimentali per la determinazione della legge di Coulomb.

Il miglior individuo dà il programma $f(x) = (2.19/sqr(x))$ e il suo andamento sull'insieme $\mathcal{T} + \mathcal{V}$ è graficato in Figura 2.

Analizzando tra le esecuzioni quelle che convergono più lentamente alla soluzione ottimale, si nota che la regressione mediante il processo evolutivo si articola in due fasi distinte. La prima, dalla generazione iniziale alla 120-esima, è caratterizzata da una ricerca di una espressione che approssima al meglio la (15). In questa fase, vengono trovate soluzioni con un crescente complessità descrittiva ma in grado di fornire un errore di approssimazione sempre minore. Nella seconda fase, dalla 120-esima alla 360-esima generazione, invece, vengono valutate espressioni sempre più semplici, meno complesse e dunque con una probabilità *a priori* maggiore.

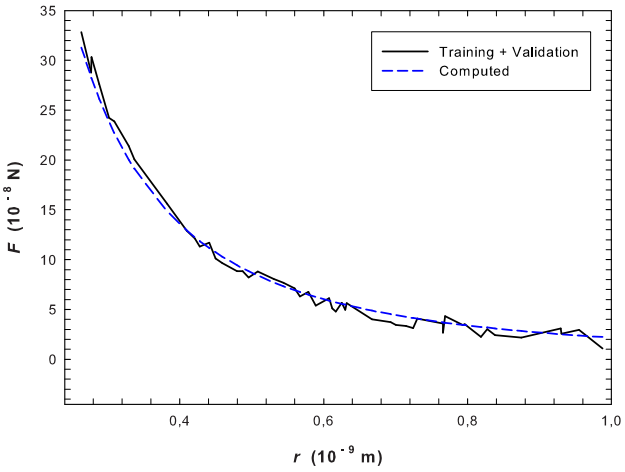

Figura 2: Grafico del miglior individuo per la legge di Coulomb sull'insieme $\mathcal{T} + \mathcal{V}$

Questo particolare comportamento della ricerca evolutiva, illustrato nel grafico 3, è molto simile a quello umano per cui trovata una prima soluzione a un problema, successivamente si tenta di migliorarla e di semplificarla [IDF05].

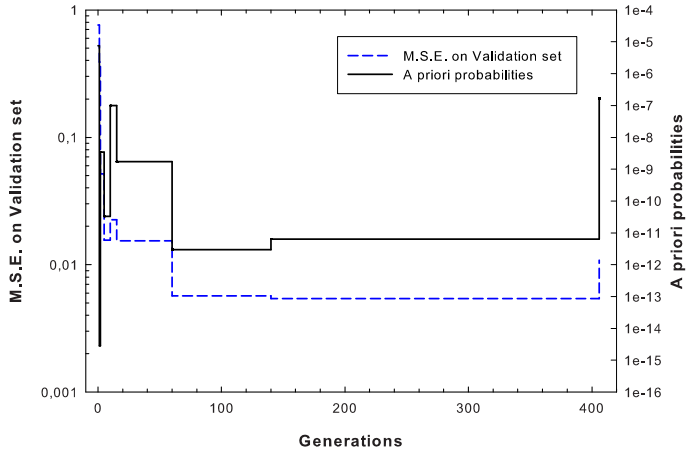

Figura 3: Confronto tra l'errore MSE su $\mathcal{V}$ e le probabilità a priori dell'individuo "most time-consuming" rispetto al numero di generazioni

4.2 La serie di Hènon

È una mappa iterativa bidimensionale con soluzioni caotiche, proposta da M. Hènon (1976) [Hèn76] come semplificazione della mappa di Poincarè per il modello di Lorenz:

$$h(t) = 1 + b \cdot h(t-2) - a \cdot h^2(t-1) \ , \text{ with } h(0) = 0.1 \text{ and } h(1) = 0 \qquad (16)$$

dove a e b sono i parametri positivi di biforcazione, con b una misura del tasso di contrazione dell'area (dissipazione). La serie di Hènon è la più generale tra le

mappe quadratiche 2–D e, per $b = 0$, si riduce alla mappa quadratica collegata alla serie logistica. La serie ammette soluzioni limitate o caotiche al variare dei valori di a e b. Un comportamento caotico è stato riscontrato per $a = 1.4$ e $b = 0.3$. Tali valori sono stati usati per gli esperimenti. Le lunghezze degli intervalli $\mathcal{S}$, $\mathcal{T}$, $\mathcal{V}$ e $\mathcal{P}$ sono rispettivamente 10, 20, 10 e 10, mentre sono state disattivate le regole di produzione della CFG con il simbolo F.

Nonostante la serie di Hènon dipenda fortemente dalle condizioni a contorno, in tutti i run effettuati è stata ottenuta la soluzione canonica (16), a meno di permutazioni dei termini. Nel caso migliore, la soluzione è stata raggiunta in 79 generazioni, su una media di 344 (si veda la Tabella 4).

	Miglior Individuo	**Valore Medio**	**Dev. St.**
Generazione	79	343.6	359.80
N.ro Nodi	23	31	8.55
Probabilità a priori	4.59E–12	1.84E–12	2.36E–12
Fitness	0.0026	0.0033	0.0007

Tabella 4: Statistiche dei risultati sperimentali per la serie di Hènon

Il migliore individuo ha espressione $f(x) = (1.0 + ((0.3 * f(x-2)) - (f(x-1) * (1.4 * f(x-1)))))$. Il suo grafico è in Figura 4.
Il processo evolutivo evidenzia le due fasi già descritte per la determinazione

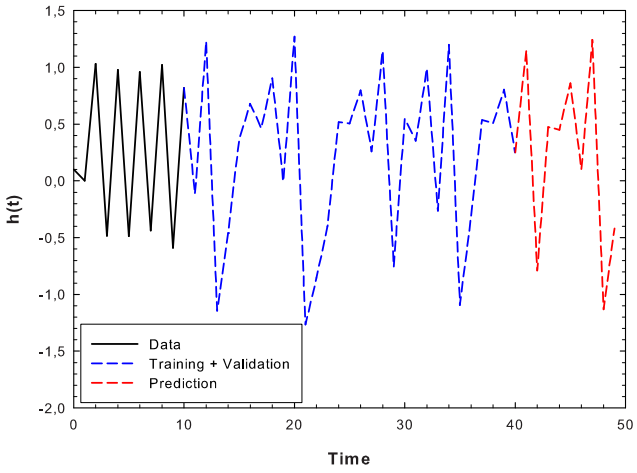

Figura 4: Grafico del miglior individuo per la serie di Hènon.

empirico-sintetica della legge di Coulomb (Figura 5).

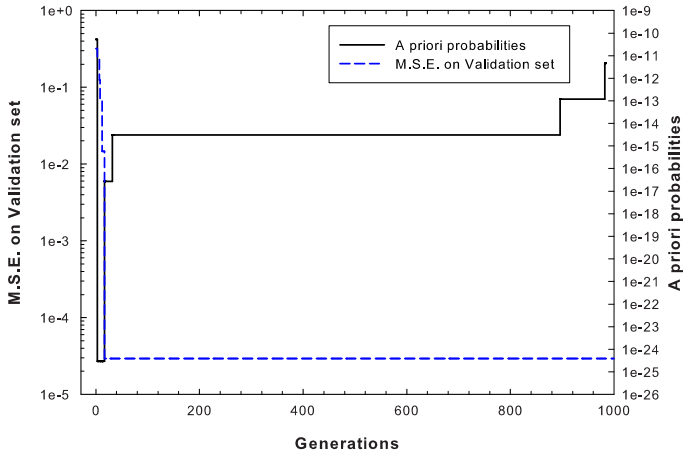

Figura 5: Confronto tra l'errore MSE su $\mathcal{V}$ e le probabilità a priori dell'individuo "most time-consuming" rispetto al numero di generazioni per la serie di Hènon.

4.3 La serie temporale di Arosa

L'ultimo problema presentato è la serie temporale che descrive la concentrazione mensile dell'ozono nella città di Arosa (Svizzera) registrate dall'inizio del 1926 al 1972 [HM94]. Le misurazioni mostrano un chiaro andamento periodico, con una escursione di circa 100 unità Dobson.
La lunghezza per $\mathcal{S}$ è 50, per $\mathcal{T} + \mathcal{V}$ è pari a 400, 30% del quale è usato come $\mathcal{V}$, e quella di $\mathcal{P}$ è uguale a 30.

Rispetto alla serie precedente sussiste qualche complicazione. I risultati sperimentali evidenziano che nessuno dei run raggiunge un MSE uguale a zero su $\mathcal{T}$. Comunque, in ogni run si sono avute espressioni con una buona approssimazione della serie. Il miglior programma registrato in 10 run rappresenta l'espressione:

$$f(t) = f(t-24) + \frac{((\cos(f(t-6) + e^{f(t-24)}))^2 - 0.41)^3}{e^{f(t-27)}} \tag{17}$$

La Tabella 5 riporta le statistiche delle prove, mentre la Figura 6 mostra il comportamento su $\mathcal{T}$, $\mathcal{V}$ e Figura 7 il comportamento su $\mathcal{P}$. Una semplice analisi della soluzione rivela che, nonostante la difficoltà del problema, il sistema GP ha saputo riscoprire il ciclo stagionale soggiacente, individuandone il periodo. Riguardo il comportamento evolutivo, è evidente dalla Fig. 8 che la dinamica del processo di ricerca segue lo schema composto di due distinte fasi esposto nei casi precedenti più volte: dalla generazione 1 alla 256, dalla 257 alla 399 e, l'ultima, dalla 400 a quella finale. Ciò lascia supporre che più il problema è complesso, più frequente è la ripetizione dello schema.

	Miglior Individuo	Valore Medio	Dev. St.
Generazione	876	651.2	281.03
N.ro Nodi	33	50.5	9.57
Probabilità a priori	2.06E-25	2.08E-26	6.49E-26
M.S.E. su $\mathcal{T}$	0.2345	0.21609	0.0158
M.S.E. su $\mathcal{V}$	0.1956	0.2325	0.0315
M.S.E. su $\mathcal{P}$	0.2011	0.2525	0.0460

Tabella 5: Statistiche della serie temporale di Arosa.

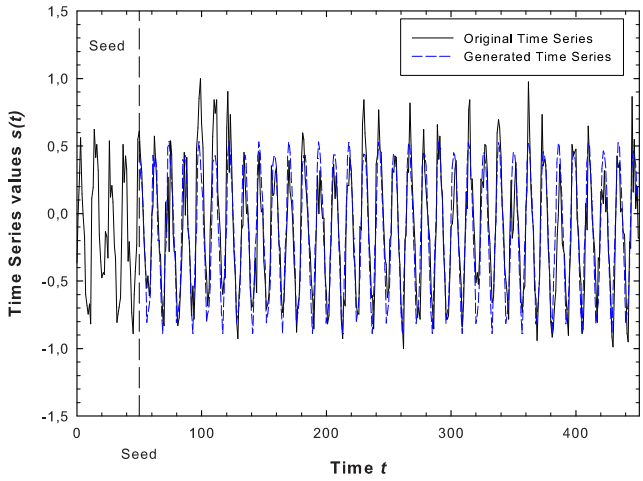

Figura 6: Grafico del miglior individuo per la serie di Arosa su $\mathcal{T} + \mathcal{V}$.

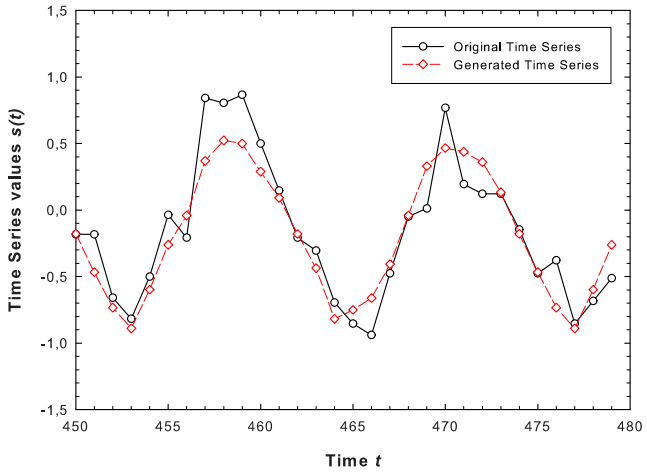

Figura 7: Grafico del miglior individuo per la serie di Arosa su $\mathcal{P}$.

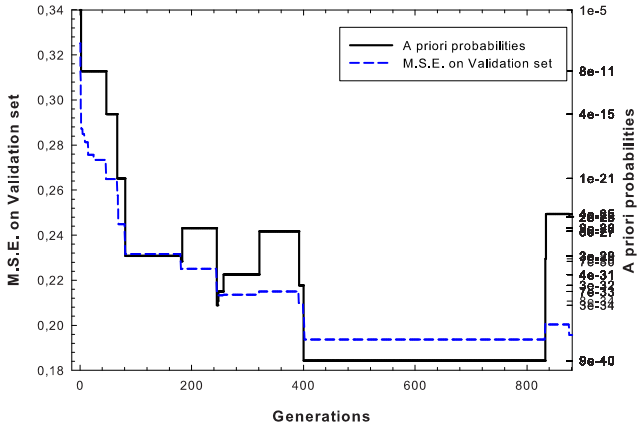

Figura 8: Confronto tra l'errore MSE su $\mathcal{V}$ e le probabilità a priori dell'individuo "most time-consuming" rispetto al numero di generazioni per la serie di Arosa.

5 Conclusioni

In questo lavoro è stato dimostrato che gli algoritmi evolutivi sono un metodo generale per risolvere problemi di inferenza induttiva in maniera biologicamente ispirata e diretta. Diretta perché è possibile con tali algoritmi esprimere concretamente, premettendo alcune considerazioni niente affatto restrittive, la teoria algoritmica di Solomonoff sull'induzione probabilistica, adottando una funzione di fitness ricavata da una distribuzione di probabilità definita sull'insieme degli esempi utili all'apprendimento.

In particolare il sistema di machine learning realizzato in CFG-GP, consente di trovare le soluzioni che meglio approssimano i dati e che hanno la maggiore probabilità a priori. La riprova è data dalla determinazione esatta della legge di Coulomb e della serie di Hènon, e dall'approssimazione della serie temporale di Arosa con un'espressione che individua con un'elevata precisione il periodo e il range delle oscillazioni dei dati reali.

Inoltre dalle statistiche registrate al variare delle generazioni durante i processi evolutivi, emerge che il sistema persegue una chiara strategia di ricerca che alterna fasi di analisi a quelle di sintesi.

Ulteriori studi avranno come obiettivo la progettazione di un sistema di inferenza induttiva CFG-GP incrementale, capace cioè di sfruttare l'esperienza derivata dall'apprendimento di alcuni concetti per la scoperta di nuovi. Questa progressione dell'apprendimento automatico comporta un updating della distribuzione di probabilità che guida la ricerca evolutiva, dovuta essenzialmente al fatto che la grammatica iniziale subisce una continua variazione.

Inoltre, metodi più accurati per il calcolo delle probabilità *a priori* da assegnare alle F^j, migliorerebbero l'abilità di apprendimento del sistema. Infatti, da alcune considerazioni bayesiane risulta che migliore è la precisione con cui sono valutate le probabilità a priori, maggiore è l'efficacia del processo di ricerca.

Ringraziamenti

Questa ricerca è stata condotta presso l'Istituto di Calcolo e Reti ad Alte Prestazione, CNR di Napoli, in collaborazione con Ivanoe De Falco, Antonio Della Cioppa ed Ernesto Tarantino. Vorrei ringraziare tutti loro per l'ospitalità e il supporto.

Riferimenti bibliografici

[HM94] K. W. Hipel and A. I. McLeod. *Time Series Modelling of Water Resources and Environmental Systems*. Elsevier, Amsterdam, NL, 1994.

[Hèn76] M. Hènon. A two–dimensional mapping with a strange attractor. *Communications of Mathematical Physics*, 50:69–77, 1976.

[Hut01a] M. Hutter. Convergence and error bounds of universal prediction for general alphabet. In *Proceedings of the 12th European Conference on Machine Learning*, pages 239–250. ECML-2001, 2001.

[Hut01b] M. Hutter. New error bounds for solomonoff prediction. *Journal of Computer and System Sciences*, 62(4):653–667, 2001.

[IDF05] A. Passaro E. Tarantino I. De Falco, A. Della Cioppa. Inductive inference of chaotic series by genetic programming: a Solomonoff-based approach. In *Proceedings of the 2005 ACM Symposium on Applied Soft Computing*, pages 966–967, Santa Fe, USA, 2005. (SAC2005).

[Mic84] Ryszard S. Michalski. *A theory and methodology of inductive learning*, volume Machine Learning: An Artificial Itelligence Approach, chapter 4, pages 83–134. Springer-Verlag, 1984.

[Sol57] R.J. Solomonoff. An inductive inference machine. IRE Convetion Record, Section on Information Theory, 1957.

[Sol64] R.J. Solomonoff. A formal theory of inductive inference. *Information and Control*, 7:1–22, 224–254, 1964.

[Sol75] R.J. Solomonoff. Inductive inference theory — a unified approach to problems in pattern recognition and artificial intelligence. In *Proceedings of the 4th International Conference on Artificial Intelligence*, pages 274–280, Tblisi, Georgia, settembre 1975.

[Sol86] R.J. Solomonoff. The application of algorithmic probability to problems in artificial intelligence. In L.N. Kanal and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 473–491. Elsieevr Science Pubblishers B.V., 1986.

[Sol02] R.J. Solomonoff. Progress in incremental machine learning. In *Workshop on Universal Learning Algorithms and Optimal Search*, Whistler, British Columbia, Canada, dicembre 2002. Neural Information Processing Systems (NIPS) Conference.

[Whi96] P. A. Whigham. *Grammatical Bias for Evolutionary Learning.* PhD thesis, School of Computer Science, University College, University of New South Wales, Australian Defence Force Academy, 1996.