STATISTICA MULTIVARIATA

Corso di laurea in matematica | Anno accademico 2004/05 | Numero 9

Ottimizzazione genetica

Gli algoritmi genetici sono una famiglia di tecniche di ottimizzazione che si ispirano all'evoluzione naturale. I sistemi biologici sono il risultato di processi evolutivi basati sulla riproduzione selettiva degli individui migliori di una popolazione sottoposta a mutazioni e ricombinazione genetica. L'ambiente svolge un ruolo determinante nella selezione naturale in quanto solo gli individui più adatti tendono a riprodursi, mentre quelli le cui caratteristiche sono meno compatibili con l'ambiente tendono a scomparire.

L'ottimizzazione genetica può essere applicata a problemi le cui soluzioni sono descrivibili mediante parametri codificabili capaci di rappresentarne le caratteristiche essenziali. Il ruolo dell'ambiente viene assunto dalla funzione obiettivo che deve essere ottimizzata.

Questo metodo presenta due grandi vantaggi: non dipende da particolari proprietà matematiche e soprattutto la complessità è in generale praticamente lineare. Negli algoritmi genetici, dopo la generazione iniziale di un insieme di possibili soluzioni (individui), alcuni individui sono sottoposti a mutazioni e a scambi di materiale genetico. La funzione di valutazione determina quali dei nuovi individui possono sostituire quelli originali.

Questa tecnica viene applicata con successo a problemi di ricerca operativa, al raggruppamento automatico (un campo della statistica che si occupa di problemi di raggruppamento e classificazione di dati), al problema del commesso viaggiatore, all'approssimazione di serie temporali, alla previsione della conformazione spaziale di proteine a partire dalla sequenza degli aminoacidi, all'ottimizzazione di reti neuronali e di sistemi di Lindenmayer, a modelli di vita artificiale (sociologi tentano invece di simulare l'evoluzione di comportamenti, ad esempio tra gruppi sociali o nazioni).

Nell'applicazione di questi metodi il matematico può intervenire in vari modi: nello sviluppo e nel controllo degli algoritmi (generazione di numeri casuali per la ricerca di conformazioni ottimali in uno spazio multidimensionale di conformazioni, grafica al calcolatore), nella codifica dei dati, nell'organizzazione delle informazioni.

Un campo di ricerca piuttosto attivo è l'ottimizzazione genetica di programmi al calcolatore (il linguaggio più adatto è, per la semplicità della sua sintassi fondamentale, il Lisp), una tecnica che viene detta *programmazione genetica* (in inglese *genetic programming*) e rientra nell'ambito dell'apprendimento di macchine (in inglese *machine learning*).

La teoria matematica degli algoritmi genetici è difficile; cfr. Schmitt e Vose.

L'algoritmo di base

Come vedremo, nell'ottimizzazione genetica è molto importante studiare bene la struttura interna del problema e adattare l'algoritmo utilizzato alle caratteristiche del problema. Nonostante ciò presentiamo qui un algoritmo di base che può essere utilizzato in un primo momento e che ci servirà anche per le applicazioni al raggruppamento automatico.

Siano dati un insieme X e una funzione $f : X \longrightarrow \mathbb{R}$. Vogliamo minimizzare f su X (nell'ottimizzazione genetica i vincoli devono in genere essere descritti dalla funzione f stessa e quindi l'insieme ammissibile A coincide con X).

Fissiamo una grandezza n della popolazione, non troppo grande, ad esempio un numero tra 40 e 100. L'algoritmo consiste dei seguenti passi:

(1) Viene generata in modo casuale una popolazione P di n elementi di X.
(2) Per ciascun elemento x di P viene calcolato il valore $f(x)$ (detto *rendimento* di x).
(3) Gli elementi di P vengono ordinati in ordine crescente secondo il rendimento (in ordine crescente perché vogliamo minimizzare il rendimento, quindi gli elementi migliori sono quelli con rendimento minore).
(4) Gli elementi migliori vengono visualizzati sullo schermo oppure il programma controlla automaticamente se i valori raggiunti sono soddisfacenti.
In questo punto l'algoritmo può essere interrotto dall'osservatore o dal programma.
(5) Gli elementi peggiori (ad esempio gli ultimi 10) vengono sostituiti da nuovi elementi generati in modo casuale.
(6) Incroci.
(7) Mutazioni.
(8) Si torna al punto 2.

Gli algoritmi genetici si basano quindi su tre operazioni fondamentali: *rinnovamento* (introduzione di nuovi elementi nella popolazione), *mutazione*, *incroci*.

Il processo evolutivo è un processo lento, quindi se la funzione da ottimizzare è molto regolare (differenziabile o convessa), gli algoritmi classici aprossimano la soluzione molto più rapidamente e permettono una stima dell'errore. Ma in molti problemi pratici, in cui la funzione di valutazione è irregolare o complicata (se ad esempio dipende in modo non lineare da molti parametri) e non accessibile ai metodi tradizionali, l'ottimizzazione genetica può essere di grande aiuto.

In questo numero

Problemi di ottimizzazione

Siano dati un insieme X, un sottoinsieme A di X e una funzione $f : X \longrightarrow \mathbb{R}$. Cerchiamo il minimo di f su A, cerchiamo cioè un punto $a_0 \in A$ tale che $f(a_0) \leq f(a)$ per ogni $a \in A$. Ovviamente il massimo di f è il minimo di $-f$, quindi vediamo che non è una restrizione se in seguito in genere parliamo solo di uno dei due.

Ci si chiede a cosa serve l'insieme X, se il minimo lo cerchiamo solo in A. La ragione è che spesso la funzione è data in modo naturale su un insieme X, mentre A è una parte di X descritta da condizioni aggiuntive. Quindi i punti di X sono tutti quelli in qualche modo considerati, i punti di A quelli *ammissibili*. In alcuni casi le condizioni aggiuntive (dette anche *vincoli*) non permettono di risalire facilmente ad A, e può addirittura succedere che la parte più difficile del problema sia proprio quella di trovare almeno un punto di A.

Soprattutto però spesso X ha una struttura geometrica meno restrittiva che permette talvolta una formulazione geometrica degli algoritmi o una riformulazione analitica del problema.

Se l'insieme X non è finito, l'esistenza del minimo non è ovvia; è garantita però, come è noto, se X è un sottoinsieme compatto di $\mathbb{R}^n$ e la funzione f è continua.

Sul significato degli incroci

Le mutazioni da sole non costituiscono un vero *algoritmo*, ma devono essere considerate come un più o meno abile meccanismo di ricerca casuale. Naturalmente è importante lo stesso che anche le mutazioni vengano definite nel modo più appropriato possibile.

Sono però gli incroci che contribuiscono la caratteristica di algoritmo, essenzialmente attraverso un meccanismo di *divide et impera*. Per definirle nel modo più adatto bisogna studiare attentamente il problema, cercando di individuarne *componenti* che possono essere variati indipendentemente l'uno dagli altri, cioè in modo che migliorando il rendimento di un componente non venga diminuito il rendimento complessivo.

Ciò non è sempre facile e richiede una buona comprensione del problema.

Il metodo spartano

Il criterio di scelta adottato dalla selezione naturale predilige in ogni caso gli individui migliori, dando solo ad essi la possibilità di moltiplicarsi. Questo meccanismo tende a produrre una certa uniformità qualitativa in cui i progressi possibili diventano sempre minori e meno probabili. Il risultato finale sarà spesso una situazione apparentemente ottimale e favorevole, ma incapace di consentire altri miglioramenti, un *ottimo locale*.

Perciò non è conveniente procedere selezionando e moltiplicando in ogni passo solo gli elementi migliori, agendo esclusivamente su di essi con mutazioni e incroci. Se si fa così infatti dopo breve tempo le soluzioni migliori risultano tutte imparentate tra loro ed è molto alto il rischio che l'evoluzione stagni in un ottimo locale che interrompe il processo di adattamento senza consentire ulteriori miglioramenti essenziali.

Per questa ragione, per impedire il proliferare di soluzioni tutte imparentate tra di loro, a differenza dalla selezione naturale non permettiamo la proliferazione identica. Nelle *mutazioni* il peggiore tra l'originale e il mutante viene sempre eliminato, e negli *incroci* i due nuovi elementi sostituiscono entrambi i vecchi, anche se solo uno dei due nuovi è migliore dei vecchi.

Precisiamo quest'ultimo punto. Supponiamo di voler incrociare due individui A e B della popolazione, rappresentati come coppie di componenti che possono essere scambiati: $A = (a_1, a_2), B = (b_1, b_2)$. Gli incroci ottenuti siano per esempio $A' = (a_1, b_2), B' = (b_1, a_2)$. Calcoliamo i rendimenti e assumiamo che i migliori due dei quattro elementi siano A' e B. Se però scegliamo questi due, nelle componenti abbiamo (a_1, b_2) e (b_1, b_2) e vediamo che il vecchio B è presente in 3 componenti su 4 e ciò comporterebbe quella propagazione di parentele che vogliamo evitare.

Negli incroci seguiamo quindi il seguente principio: Se nessuno dei due nuovi elementi è migliore di entrambi gli elementi vecchi, manteniamo i vecchi e scartiamo gli incroci; altrimenti scartiamo entrambi gli elementi vecchi e manteniamo solo gli incroci.

Numeri casuali

Successioni di numeri (o vettori) casuali (anche in forme di tabelle) vengono usate da molto tempo in problemi di simulazione, statistica, integrazione numerica e crittografia. Attualmente esiste un grande bisogno di tecniche affidabili per la generazione di numeri casuali, come mostra l'intensa ricerca in questo campo.

Il termine numero casuale ha tre significati. Esso, nel calcolo delle probabilità, denota una *variabile casuale* a valori numerici (reali o interi), cioè un'entità che non è un numero ma, nell'assiomatica di Kolmogorov, una funzione misurabile nel senso di Borel a valori reali (o a valori in $\mathbb{R}^n$ quando si tratta di vettori casuali) definita su uno spazio di probabilità, mentre le successioni generate da metodi matematici, le quali sono per la loro natura non casuali ma deterministiche, vengono tecnicamente denominate successioni di numeri *pseudocasuali*.

Il terzo significato è quello del linguaggio comune, che può essere applicato a numeri ottenuti con metodi *analogici* (dadi, dispositivi meccanici o elettronici ecc.), la cui casualità però non è sempre affidabile (ad esempio per quanto riguarda il comportamento a lungo termine) e le cui proprietà statistiche sono spesso non facilmente descrivibili (di un dado forse ci possiamo fidare, ma un dispositivo più complesso può essere difficile da giudicare). Soprattutto per applicazioni veramente importanti è spesso necessario creare una quantità molto grande di numeri casuali, e a questo scopo non sono sufficienti i metodi analogici. Oltre a ciò normalmente bisogna conoscere a priori le proprietà statistiche delle successioni che si utilizzano.

Siccome solo le successioni ottenute con un algoritmo deterministico si prestano ad analisi di tipo teorico, useremo spesso il termine „numero casuale“ come abbreviazione di „numero pseudocasuale“.

Una differenza importante anche nelle applicazioni è che per le successioni veramente casuali sono possibili soltanto stime probabilistiche, mentre per le successioni di numeri pseudocasuali si possono ottenere, anche se usualmente con grandi difficoltà matematiche, delle stime precise.

Spieghiamo l'importanza di questo fatto assumendo che il comportamento di un dispositivo importante (che ad esempio governi un treno o un missile) dipenda dal calcolo di un complicato integrale multidimensionale che si è costretti ad eseguire mediante un metodo di Monte Carlo. Se i numeri casuali utilizzati sono analogici, cioè veramente casuali, allora si possono dare soltanto stime per la probabilità che l'errore non superi una certa quantità permessa, ad esempio si può soltanto arrivare a poter dire che in non più di 15 casi su 100000 l'errore del calcolo sia tale da compromettere le funzioni del dispositivo. Con successioni pseudocasuali (cioè generate da metodi matematici), le stime di errore valgono invece in tutti i casi, e quindi si può garantire che l'errore nel calcolo dell'integrale sia sempre minore di una quantità fissa, assicurando così che il funzionamento del dispositivo non venga mai compromesso.

runif

Una successione di n numeri casuali reali (uniformemente distribuiti) in $[a, b]$ si ottiene con

```
runif(n,min=a,max=b)
```

Nell'ottimizzazione genetica spesso vogliamo anche usare numeri casuali interi; a questo scopo definiamo la seguente funzione:

```
Snc.interi = function (n,min=1,max=6)
floor(runif(n,min=min,max=max+0.999))
```

R usa, nell'impostazione iniziale, come generatore di numeri casuali un algoritmo detto *Mersenne twister*, dovuto a Matsumoto e Nishimura, considerato uno dei migliori generatori conosciuti. Per le altre scelte possibili consultare `?Random`.

Numeri casuali in crittografia

Si dice che Cesare abbia talvolta trasmesso messaggi segreti in forma crittata, facendo sostituire ogni lettera dalla terza lettera successiva (quindi la a dalla d, la b dalla e, ..., la z dalla c), cosicché *crascastramovebo* diventava *fudvfdvwudpryher* (usando il nostro alfabeto di 26 lettere). È chiaro che un tale codice è facile da decifrare. Se invece $(x_1, \ldots, x_N)$ è una successione casuale di interi tra 0 e 25 e il testo $a_1 a_2 \ldots a_N$ viene sostituito da $a_1+x_1, \ldots, a_N+x_N$, questo è un metodo sicuro. Naturalmente sia il mittente che il destinatario devono essere in possesso della stessa lista di numeri casuali.

La scoperta di farmaci

È forse sorprendente che in un recente testo di disegno dei farmaci si trovi il seguente brano che abbiamo tradotto: *„I matematici negli ultimi decenni hanno aggiunto i principi dell'evoluzione al loro strumentario. Utilizzando replicazioni, mutazioni e incroci essi hanno sviluppato algoritmi genetici. Chi ha mai potuto ammirare come un tale algoritmo risolve i più complessi problemi di ottimizzazione in tempo incredibilmente breve, non avrà più dubbi che anche l'evoluzione delle specie biologiche si è svolta in modo analogo.“* (Böhm/Klebe/Kubinyi, 231)

Il futuro dell'industria farmaceutica sarà fortemente influenzato dai progressi nella comprensione dettagliata delle informazioni contenute nel genoma e della struttura e funzione delle molecole biologiche per i processi normali e patologici della vita, e quindi anche una sempre migliore comprensione molecolare delle malattie che permetterà una progettazione razionale e mirata di molecole farmaceutiche. Nuove tecniche permettono di fornire in tempi brevi numerosi composti da sottoporre a test e da classificare; il matematico, nel suo ruolo di semplificatore della complessità, può nella ricerca sviluppare nuovi metodi di classificazione o nuovi test statistici.

Bibliografia

12701 **H. Böhm/G. Klebe/H. Kubinyi:** Wirkstoffdesign. Spektrum 1996.

9797 **R. Centrella:** Numeri casuali - teoria e generatori dicotomici. Tesi Ferrara 1997.

2789 **D. Goldberg:** Genetic algorithms in search, optimization, and machine learning. Addison-Wesley 1989.

12718 **A. Joereßen/H. Sebastian:** Problemlösung mit Modellen und Algorithmen. Teubner 1998.

12910 **V. Nissen:** Einführung in evolutionäre Algorithmen. Vieweg 1997.

17090 **I. Rechenberg:** Evolutionsstrategie '94. Frommann-Holzboog 1994.

15003 **L. Schmitt:** Theory of genetic algorithms.
Theor. Computer Sci. 259 (2001), 1-61.

17033 **M. Vose:** The simple genetic algorithm. MIT Press 1999.