Intelligenza Artificiale e Machine Learning

Appunti delle Lezioni di Intelligenza Artificiale e Machine Learning
Anno Accademico: 2024/25

Giacomo Sturm

Dipartimento di Ingegneria Civile, Informatica e delle Tecnologie Aeronautiche
Università degli Studi "Roma Tre"

Sorgente del file LaTeX disponibile al seguente link:
https://github.com/00Darxk/Intelligenza-Artificiale-e-Machine-Learning

Indice

1 Algoritmi

1.1 Problema di Ricerca Globale

Nei problemi precedenti quando l'algoritmo risolutivo raggiungeva uno stato obiettivo, il cammino verso quello stato rappresenta una soluzione del problema. Tuttavia in alcuni problemi lo stato obiettivo contiene tutte l informazioni rilevanti per la soluzione, dove il cammino è irrilevante. Come esempio si consideri il problema dell otto regine, è indifferente il cammino attraverso gli stadi intermedi, solamente lo stato finale, la disposizione delle regine nello stato finale.

Algoritmi di ricerca locale si utilizzano per risolvere questo tipi di problemi. In questi problemi è sempre presente uno spazio degli stati ed uno spazio degli stati aventi ciascuno una sua valutazione. Si può immaginare questi stadi su una superficie del territorio, uno spazio dove l'altezza di questo stato rappresenta la sua valutazione. L'algoritmo quindi itera su ognuno di questi stati per cercare quello di altezza maggiore, o minore, identificando quindi la soluzione al problema, indipendentemente dal cammino preso per raggiungerla. Questi punti di massimo rappresentano dei picchi, i cui punti adiacenti sono strettamente minori dello stato di massimo. Quindi l'algoritmo che parte da uno stato iniziale deve cercare un massimo globale in questo spazio, determinando quale sia tra i vari massimi locali ed i massimi locali piatto, e le "spalle" massimi locali "piatti", prima di un massimo globale.

Questi algoritmi chiamati anche di miglioramento iterativo, si muovono sulla superficie cercando questi picchi, senza tenere traccia del cammino effettuato, tenendo solamente traccia dello stato attuale e dei suoi vicini o successori, gli stati immediatamente adiacenti. Bisogna formulare il problema in modo che l'algoritmo non rimanga bloccato tra due massimi locali.

1.1.1 Algoritmo di Hill-Climbing

Questo algoritmo segue sempre le colline più ripide, si muove sempre verso l'alto nella direzione dei valori crescenti, e termina quando raggiunge uno stato per il quale si ha un picco che non ha vicino stati di valore maggiore. Tuttavia questo algoritmo può rimanere intrappolato su massimi globali.

Non viene memorizzato lo stato corrente, solamente il valore attraverso nodi che contengono lo stato ed il suo valore: `NODO=<STATO, VALORE`.

Esistono diversi tipi di algoritmi di questo genere per evitare di rimanere bloccati su picchi locali, utilizzando diverse tecniche.

Dallo stato corrente si ricava il suo valore, in seguito comincia un ciclo che prende in considerazione tutti i successori e si sceglie come `next` il nodo di valore più alto. Se tutti i nodi adiacenti hanno un valore minore di `next` allora questo rappresenta la soluzione dell'algoritmo e l'algoritmo termina, tuttavia questo stato potrebbe corrispondere ad un massimo locale, invece se esiste uno stato adiacente di valore maggiore, questo diventa `next` e si passa alla nuova iterazione.

L'algoritmo contiene una componente di ripartenza casuale, questo infatti conduce una serie di ricerche di Hill-Climbing partendo da stati generati casualmente. Questo algoritmo da un punto di vista teorico è completo, poiché con una serie infinita di ripartenze, sicuramente l'algoritmo visita tutti gli stati del sistema, trovando sicuramente la soluzione ottima del problema.

Il ciclo interno ad ogni iterazione genera un ottimo locale, e prova ad evitare ottimi locali effettuando una nuova ricerca da un nuovo stato scelto casualmente.

L'algoritmo Stochastic Hill-Climbing si ottiene modificando la procedura normale dell'algoritmo. Invece di valutare tutti i vicini dallo stato corrente, l'algoritmo sceglie casualmente uno solo dei suoi successori da valutare per determinare se si tratta il successore, ed in caso diventa il nuovo stato corrente `next`, questo viene accetta con una probabilità che dipende dalla differenza della valutazione tra i due punti: $\Delta E =$`VALUE(current)-VALUE(next)`.

Il nuovo stato viene scelto con una probabilità p, calcolata come:

$$p = \frac{1}{1 + e^{\Delta E/T}} \tag{1.1.1}$$

In seguito dopo una serie di iterazioni l'algoritmo restituisce uno stato ottimo. L'algoritmo ha quindi un solo ciclo, e può scegliere un nuovo punto con una probabilità p, quindi anche di valore minore. Questa probabilità dipende da un parametro T costante durante l'esecuzione dell'algoritmo. Se vale 1, la probabilità di accettazione è sostanzialmente pari al 100%.

All'aumentare del valore i T la probabilità di accettazione tende al 50%, diventa quindi sempre meno importante la differenza della valutazione tra i due punti, effettivamente comporta una ricerca casuale, mentre al diminuire di T, la procedura rappresenta un semplice algoritmo Hill-Climbing.

In caso di stati di valore uguale, la probabilità è del 50%, se il valore dello stato `next` è minore, la probabilità diminuisce, mentre se il valore di `next` è maggiore dello stato corrente, la probabilità aumenta.

Bisogna trovare una "link function" tra l'intervallo $\Delta E/T$ e la probabilità p.

La caratteristica di poter scegliere come passo uno stato peggiore questo algoritmo potrebbe evitare massimi locali.

Questo algoritmo prende il nome dall'analogia con il processo di metallurgia per temprare un materiale, questo processo infatti raggiungere uno stato di struttura cristallina ad energia minima. La differenza principale con l'algoritmo stocastico, è la possibilità di variare il valore di T gradualmente durante l'esecuzione dell'algoritmo. Il valore di T parte da un valore elevato, per poi diminuire nel tempo, come se fosse la temperatura durante un processo di temperatura.

In questo algoritmo la probabilità di accettazione di un certo nodo viene implementata in modo differente rispetto all'algoritmo stocastico normale, poiché nel simulated annealing si considera solamente un semiasse dell'ascissa, dato che in caso `next` sia migliore non viene calcolata la probabilità di accettazione. Si accetta sempre uno stato di valore migliore, mentre l'accettazione di uno stato peggiore dipende da una probabilità.

Molte implementazioni dell'algoritmo seguono la stessa sequenza di passi. Si assegna la variabile `T`, e si sceglie uno stato corrente casuale al primo passo. Si determina un successore e si ripete per un certo numero di cicli nel secondo, e come passo finale si diminuisce la temperatura e si ripete dal primo passo se la temperatura non ha raggiunto la temperatura minima. Quindi l'algoritmo termina solamente quando la temperatura ha raggiunto la temperatura minima.

Le aree di applicazione di questo algoritmo sono molto vaste nell'informatica. Proposto negli anni '80 ha avuto un enorme successo, anche solo nella sua versione base.

2 Introduzione a Python

Python è un linguaggio di programmazione vastamente utilizzato nell'area dell'intelligenza artificiale e nel machine learning. Recentemente è diventato il linguaggio di programmazione più diffuso al mondo. Python è un linguaggio general-purpose, ideato da Guido van Rossum nel 1989, a più alto livello del C, poiché gestisce automaticamente le più fondamentali operazioni.

La versione di Python utilizzata nel corso è la versione 3, nell'ambiente Anaconda. Può essere avviato tramite un interprete. Alla creazione di una variabile non è necessario definirne il tipo, il nome identificativo è arbitrario e può contenere numeri, ma non cominciare con un numero, viene consigliato di utilizzare un carattere minuscolo come primo carattere del nome. Esistono 33 parole chiave, non utilizzabili come nomi di variabili. Si può assegnare un valore ad una variabile tramite l'operatore `=`, senza specificarne il tipo.

Esistono una serie di operatori aritmetici come `+`, `-`, `*`, \, `**` e \\ (floor division). Una differenza tra Python 2 consiste nella gestione del resto, infatti in Python 2 viene considerata solo la parte intera del resto.

Si possono inserire dati tramite la funzione `input`, e si può convertire in un tipo specifico come `tipo(var)`. I commenti vengono realizzati tramite il carattere #.

In Python sono presenti tutti gli operatori booleani di C, in aggiunta sono presenti altri operatori `is` ed `is not`.

Sono presenti gli stessi operatori logici di C, e come in C un qualsiasi valore diverso da zero corrisponde al booleano `true`.

In Python per identificare funzioni o istruzioni condizionali non si usano parentesi, ma si indenta di quattro posizioni. Dopo la condizione dell' istruzione condizionale vanno inserite dei due punti `:`.

Si possono gestire le eccezioni con il costrutto `try` ed `except`:

```
try:
  # corpo del try
except:
  # corpo dell'except
```

In Python sono integrate tantissime funzioni utili, per svolgere attività comuni, utilizzabili senza doverle definire, queste sono funzioni "built-in". Per invocare funzioni presenti in un certo modulo si utilizzata la notazione puntata `nomeModulo.nomeFunzione()`.

Dati gli algoritmi analizzati precedentemente, si nota la necessita di introdurre generatori di numero casuali. La maggior parte di generatori casuali, sono deterministici, ovvero dato lo stesso input, generano gli stessi numeri casuali. Si utilizzano quindi numeri pseudo-casuali, generati da un calcolo deterministico, ma non è quasi possibile distinguerli da numeri generati casualmente. In Python esiste il modulo `random` contenente funzioni pertinenti alla generazione di numeri casuali.

Per definire nuove funzioni si utilizza la parola chiave `def`, specificando il nome, tra parentesi tonde gli argomenti ed i due punti, indentando di quattro posizioni per scrivere il corpo della funzione:

```
def nomeFunzione(nomeArgomenti):
  # corpo della funzione
# resto del codice
```

Dopo aver passato degli argomenti ad una funzione, questi vengono assegnati a delle variabili locali. Si può utilizzare anche una variabile come argomento. Inoltre tutte le aggiunte possibili alle funzioni built-in, si possono effettuare sulle funzioni definite dall'utente come `_twice`, per ripetere due volte la funzione.

Si dividono le funzioni in due tipi "fruitful function" e "void function", le prime restituiscono un valore, le secondo non restituiscono valore. Le prime quindi vengono usate per assegnare o inizializzare variabili. Se si tenta di assegnare il risultato di una void function ad una variabile, viene ottenuto un valore chiamato "None". Questo valore ha un suo proprio tipo.

Si possono realizzare cicli tramite il costrutto `while` o `for`, seguito da una condizione booleana e dai due punti `:`. Si può interrompere il ciclo con `break`, e si può saltare l'iterazione corrente con `continue`. Quando bisogna iterare su una collezione, si può realizzare un ciclo for-each:

```
for elem in Array
  # corpo del ciclo
```