Risoluzione dei giochi dell'8 e del 15 con algoritmo IDA* e funzione euristica calcolata mediante MLP

Gianluca Pollastri
Facoltà di Ingegneria - Università di Firenze
pollastr@biancaneve.ing.unifi.it
http://www.dsi.unifi.it/aiia97/pollastri.html

1.Introduzione

Il gioco dell'8 è stato utilizzato in più occasioni ([1],[4]) come un modello per la misurazione delle prestazioni di algoritmi di ricerca a causa della dimensione relativamente contenuta dello spazio del problema (181440 configurazioni totali). Un elemento particolare di interesse nei confronti di questo gioco è il fatto di disporre di una funzione euristica ammissibile e relativamente efficiente come la distanza Manhattan (v. sotto) per stimare la distanza di una configurazione dalla meta della ricerca, da utilizzare nell'algoritmo IDA*.

D'altro canto, tale distanza, per quanto consenta la soluzione in tempi estremamente contenuti di una qualsiasi delle istanze del gioco, richiede comunque la visita media dell'1.4% del numero totale delle configurazioni per portare alla prima soluzione e mostra chiaramente i suoi limiti non appena viene applicata ad una delle varianti più estese del problema in questione.

Già il gioco del 15 si rivela praticamente intrattabile con l'algoritmo IDA* e la distanza Manhattan. D'altra parte, l'utilizzo di una funzione euristica non ammissibile (ammesso che se ne possa trovare una efficiente) avrebbe il difetto ovvio di trovare la soluzione in un numero di mosse certamente molto superiore rispetto al caso minimale.

Si è scelto quindi, al fine di ridurre la complessità media della ricerca rimanendo il più possibile prossimi alla ammissibilità, di tentare un approccio al gioco dell'8 attraverso Reti Neurali (precisamente Multilayer Perceptrons) ovvero di utilizzare una di esse come stimatore di distanza.

L'idea era quella di implementare regressori/classificatori MLP, quindi, attraverso un algoritmo di learning con supervisione, di eseguire una sessione di training su uno di essi con lo scopo di ottenere uno stimatore di distanza affidabile che potesse sostituirsi alla distanza Manhattan migliorando le prestazioni dell'algoritmo IDA*.

L'approccio applicato al gioco dell'8 è stato quindi esteso con alcuni accorgimenti al gioco del 15. In questo caso i miglioramenti computazionali rispetto alle tecniche classiche sono stati anche più vistosi, considerato il fatto che l'addestramento della rete è stato condotto (a causa della citata enorme complessità di risoluzione minimale del gioco) con un set di campioni estremamente limitato ed estratto da una porzione ridottissima dello spazio del problema.

2 Algoritmo IDA*

L'IDA* (Iterative Deepening A*,[4]) è un algoritmo di ricerca euristica. Questo algoritmo effettua una successione di visite depth first con soglie progressive di costo. Ciò significa che ad ogni iterazione vengono presi in considerazione soltanto quei nodi il cui costo è inferiore alla soglia corrente. In caso di insuccesso viene aumentata la soglia ed effettuata una nuova visita DF.

Il costo di un nodo N, indicato con f(N), è dato dalla somma della distanza (nota) d(N) del nodo N dal nodo iniziale della ricerca e della distanza (stimata) h(N) del nodo N dal goal della ricerca (*funzione euristica*).

Laddove si ponga h(N)=0, l'algoritmo si trasforma in un ID semplice il quale, come è noto, è *ammissibile*, ovvero trova sempre una soluzione minimale, ed ottiene questo risultato con costi (relativamente alla profondità della ricerca) esponenziali in tempo ma lineari in spazio.

Come noto l'algoritmo IDA* conserva la proprietà di ammissibilità a condizione che sia:

$$h(N) \leq h^*(N) \qquad \forall N$$

dove $h^*(N)$ è la distanza (minima) effettiva del nodo N dal goal.

2.1 Note sull' implementazione dell' algoritmo

Si è utilizzato l'algoritmo IDA* di Korf [4] nella sua versione standard.

Per il gioco dell'8 è stata implementata una transposition table per la gestione del controllo dei cicli. Ogni locazione della tabella è destinata a contenere il valore del livello più prossimo alla soluzione al quale la configurazione individuata dalla funzione hash è stata trovata. All'interno dell'algoritmo IDA* la tabella viene aggiornata nel

caso che la configurazione in analisi non sia ancora stata visitata, oppure che sia stata visitata in precedenza lungo un percorso più profondo di quello corrente. In caso contrario anche il figlio viene scartato. Si instaura così di un controllo completo sui cicli che riduce del 35% circa lo sforzo computazionale medio [1].

3.La rete neurale

I test sono stati eseguiti su MLP feedforward con due o tre livelli di unità nascoste. La scelta della struttura della rete è stata semplicemente fatta con un approccio di tipo 'trial and error'.

3.1 Codifiche degli ingressi e delle uscite

Per la scelta della rete neurale da utilizzare si è dovuto tenere conto delle possibili codifiche per gli ingressi e per l'uscita.

Si è fatto ricorso a codifiche dell'ingresso a massima entropia, molto ridondanti (si sono utilizzate 81 unità nel gioco dell'8 e 256 in quello del 15). Nel gioco dell'8 ad esempio l'ingresso numero (n*9+k) è alto (1) se la tessera numero n si trova nella k-esima casella, basso (0) altrimenti.

Per quanto riguarda l'uscita, per il gioco dell'8 si è usata una sola unità con l'unico accorgimento di normalizzare (per un fattore 1/32) il valore della stima. Nel gioco del 15 invece è stata adottata una struttura mista, con una unità per la stima (normalizzata per 1/30) e 256 unità per la costruzione di un *autoassociatore*, struttura cui si tornerà in dettaglio più avanti.

3.1 Algoritmi di training

Si sono utilizzati algoritmi di training sia di tipo online che di tipo batch. Nel primo caso l'aggiornamento dei parametri della rete avviene per ciascuno dei campioni usati nel training, nel secondo esso avviene una sola volta per ogni passaggio dell'intero set.

L'algoritmo batch ha portato, a fronte di una maggior lentezza nella convergenza, a stime più precise del caso online. Alla descritta lentezza dell'algoritmo batch si è tentato di ovviare con un criterio euristico di incremento del passo di aggiornamento dei parametri della rete. In particolare si è scelto di aumentare gradualmente il peso degli aggiornamenti quando i vettori delle variazioni dei parametri si conservavano approssimativamente paralleli per un certo numero di passi, salvo poi immediatamente tornare ad un passo 'breve' non appena il parallelismo cessasse.

I risultati descritti sono stati ottenuti con un utilizzo 'puro' questa seconda classe di metodi.

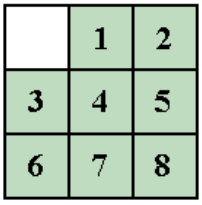
4. Il gioco dell'8

Le configurazioni complessive (legali) del gioco dell'8 sono:

9!/2 = 181440.

Si è considerata come configurazione goal quella in fig.1.

Fig.1 Il gioco dell'8



La distanza Manhattan associa ad ogni tessera la distanza in righe più la distanza in colonne dalla sua posizione giusta. Essa è evidentemente ammissibile.

Con un totale di circa 470 milioni di visite, si è eseguito un test su tutte le 181440 configurazioni del problema da utilizzare come raffronto per i test successivi. I risultati principali sono riportati nella tabella 1.

4.1 Soluzione del gioco dell'8 con MLP

La rete usata per la soluzione del gioco dell'8 ha due livelli nascosti composti da 6 e 19 unità (dall'ingresso verso l'uscita).

Le combinazioni del gioco dell'8 si addensano in massima parte ad una distanza dalla soluzione superiore a 15 mosse (il 97.37% dei casi). Ciò nonostante si è scelto un training set con una distribuzione uniforme delle configurazioni partendo dall'assunto ideale che con uno stimatore completamente informato viene visitata una ed una sola configurazione per ogni livello del grafo. Il training set usato è composto da 9300 campioni (il 5% del totale). Questo semplice approccio ha permesso di risolvere il gioco con i risparmi di tempo e nodi visitati descritti in tabella 1. Si noti che il numero medio di mosse alla prima soluzione è cresciuto soltanto da 21.97 a 22.84.

Fig.2

Gioco dell'8

Numero di conf. vs numero di mosse nel caso minimale (dotted) e con MLP

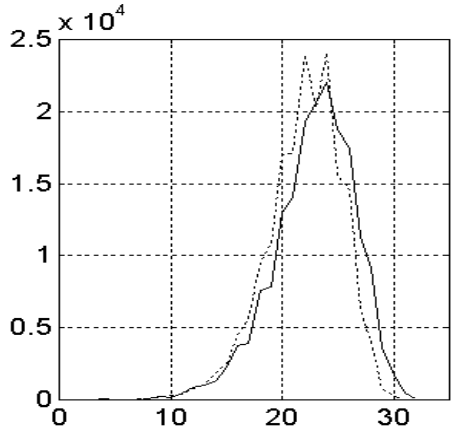


Tabella 1
Test sul gioco dell'8

	Manhattan	MLP
Configurazioni	181440	5000
Mosse medie	21.97	22.84
Visite medie	2604.07	337.24
Tempo medio (ms)	352.87	282.14
Br. factor(eur.)	3.92	5.83

5. Il gioco del 15

Le configurazioni complessive del gioco del 15 sono:

16!/2 = 10,461,394,944,000.

Questo numero, unito alla ridottissima densità delle soluzioni, è evidentemente un grave ostacolo a qualsiasi tentativo di condurre una ricerca efficiente.

L' utilizzo *tout court* dell'algoritmo IDA* non ha mostrato speranze di applicabilità almeno facendo uso delle euristiche tradizionali. La distanza Manhattan, ad esempio, per quanto conservi la proprietà di ammissibilità anche nel gioco del 15, richiede un numero medio di circa 363 milioni di visite per pervenire alla prima soluzione ed un numero massimo superiore a $6x10^9$ ([1],[4]).

Fig.3
Il gioco del 15

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

5.1 Training set per il gioco del 15.

L'applicabilità del metodo descritto riguardo al gioco dell'8 è resa assai più difficoltosa dai problemi di reperimento di un training set consistente. Il test più coraggioso mai eseguito con IDA* sul gioco del 15 ([1],[4]) non supera infatti i 100 campioni, che sono un totale insufficiente per addestrare una rete (v.[2]).

D'altra parte non esiste altra euristica nota, eccetto la distanza Manhattan, che permetta di risolvere il gioco del 15 in modo minimale e la minimalità è un requisito indispensabile perché il la rete apprenda a stimare nel modo desiderato.

Si noti però che la ricerca con IDA* si svolge in gran parte tra le configurazioni meno distanti dalla soluzione purché si disponga di uno stimatore anche molto approssimativo che consenta di raggiungerle. Il grafo di ricerca infatti si espande esponenzialmente col progredire dei livelli visitati e se è vero che nel gioco del 15 probabilmente soltanto lo 0,01% delle configurazioni dista meno di 30 mosse dalla soluzione, già con l'uso di una stima disperata come la distanza Manhattan si spende in queste la maggior parte della ricerca.

Allo stesso tempo la soluzione del gioco del 15 con un cutoff di 30 non è molto più dispendiosa rispetto al semplice caso del gioco dell'8. In conseguenza non è particolarmente oneroso risolvere un numero di istanze sufficientemente ampio da costituire un training set per addestrare un MLP a stimarne la distanza.

Sono state quindi individuate e risolte 12301 istanze del gioco distribuite in modo lineare (v. allegati) in funzione della distanza dal goal usando un cutoff di 30 sui livelli del grafo esaminati.

5.2 La rete e l'apprendimento

Per ovviare poi al problema di fornire alla rete uno strumento per individuare se la sua stima sia attendibile, ovvero se la configurazione esaminata disti effettivamente meno di trenta mosse oppure se la sua distanza debba essere stimata diversamente (ad esempio con la distanza Manhattan) si sono affiancate alla unità di uscita deputata alla valutazione della stima tante unità di uscita quante sono quelle di ingresso per realizzare un auto-associatore [5]. Tali uscite sono cioè state istruite a riprodurre i segnali di ingresso per tutte le configurazioni a distanza minore di 30 dal goal contenute nel training set. Durante la stima, quelle configurazioni per cui il coseno dell'angolo tra il vettore di ingresso e quello delle uscite di auto-associazione sia inferiore ad una soglia di 0.5 sono considerate come non riconosciute, quindi più distanti di 30 mosse dalla meta.

Per rendere più efficiente la rete a rigettare queste configurazioni si sono aggiunti al training set 2000 esempi negativi, ovvero istanze scelte casualmente per le quali l'algoritmo IDA* minimale non avesse

trovato soluzione entro il cutoff, e quindi provatamente più distanti di questo. In tal caso durante l'apprendimento si è passato alle uscite auto-associanti della rete un vettore con entrate tutte nulle. Per quanto riguarda l'uscita di stima, nel caso degli esempi negativi ci si è limitati ad imporre che fosse superiore a 30. Si è cioè passato 31 (salvo la citata normalizzazione) quando la stima fosse inferiore a tale valore, la stima stessa altrimenti.

La rete adottata, che consta di un solo livello nascosto composto da 23 unità, durante l'apprendimento ha inaspettatamente mostrato una certa capacità di estrapolazione. Al termine del training, su un set scelto casualmente di esempi negativi la rete ha rivelato una stima media prossima alle 45 mosse, molto superiore quindi al valore minimo imposto di 31.

Si è dunque deciso di utilizzare comunque lo stimatore anche in quei casi in cui l'istanza fosse rigettata dall'auto-associatore, introducendo una funzione di 'appesantimento'.

In sostanza, quando l'istanza viene accettata si preleva la pura stima, quando essa è rigettata, se ne valuta la distanza dal goal come:

$(1+\text{stima}/\alpha)*\text{stima}$

dove α è una costante di normalizzazione determinata in modo da spostare la stima media intorno alle 53 mosse predette da Rheinefeld [1].

Tabella 2

Test sul gioco del 15[1].

	Manhattan	MLP
onfigurazioni	100	103
osse per onfig.	53	56.2
isite per conf.	$363x10^6$	$408x10^3$
empo per onf. (s)	$30.7x10^3$	$1.01x10^3$

5.3 Risultati

A fronte di una crescita complessiva del peso computazionale di una singola visita (dovuta all'aggravio di metalivello) di un fattore 29.3 rispetto al caso della semplice distanza Manhattan, l'applicazione dello stimatore così costruito all'algoritmo IDA* ha portato ad una assai più rilevante diminuzione del numero di visite per istanza risolta. In particolare, su 104 configurazioni scelte casualmente il numero medio di visite è sceso dai circa 363 milioni stimati ([1],[4]) per il caso della Manhattan a circa 408000, con una riduzione di un fattore 889. Il tempo speso mediamente nella soluzione di una istanza si riduce quindi di circa 30 volte rispetto all'algoritmo classico.

Il numero di mosse alla prima soluzione è stato mediamente di 56.2, soltanto 3 in più rispetto al caso (stimato) della Manhattan.

Per un riassunto dei dati si veda la tabella 2.

Per maggiori dettagli si rimanda agli allegati.

6. Conclusioni

L'utilizzo di stimatori di distanza costituiti da reti neurali nell'algoritmo IDA* ha dimostrato una notevole efficienza, come risulta chiaramente dal confronto effettuato con la classica distanza Manhattan su un problema molto esteso (si ricordi: 10^{13} istanze) e con una bassissima densità di soluzioni come il gioco del 15. Gli stimatori ottenuti permettono inoltre, sui problemi testati, di mantenere un comportamento assai prossimo alla minimalità. Tale comportamento è stato verificato in media, mentre non esiste allo stato attuale un criterio sicuro per delimitare la sovrastima della lunghezza del cammino risolutivo (dovuta alla occasionale mancanza di ammissibilità dell'euristica) che si può presentare.

Nel gioco del 15 in particolare la rete usata è stata addestrata con un set di esempi che rappresenta poco più di $1/10^9$ del totale delle istanze e che inoltre è stato estratto da un sottoinsieme di queste limitatissimo e contraddistinto da una estrema 'facilità' computazionale rispetto alla media.

Nonostante ciò la rete ottenuta è stata utilizzata, sia pure con un fattore di correzione per le stime meno affidabili, senza il bisogno di ricorrere ad altri strumenti per la valutazione della distanza.

Se il successo dell'approccio adottato sui problemi in questione dipenda dalla natura dei problemi stessi (ad esempio dal fatto che il branching factor è piuttosto contenuto), o se invece esso possa essere proficuamente utilizzato su una più vasta classe di problemi di ricerca, non è domanda cui si possa dar risposta senza studi più approfonditi. Una traccia in tal proposito può essere fornita da una recente ricerca [10], in cui reti in BP che utilizzano metodologie più sofisticate rispetto a quelle qui trattate sono state applicate come euristiche in un problema di ricerca ampio come la deduzione automatica, con risultati molto concreti.

[1] I valori relativi alla distanza Manhattan sono quelli riportati da [1] eccetto il tempo che è stato calcolato (per esigenze di confronto) moltiplicando per il numero medio di nodi visitati il tempo medio di una visita ottenuto sostituendo la Manhattan allo stimatore neurale nella nostra implementazione.

Bibliografia

[1] A.Reinefeld - Complete Solution of the Eight Puzzle and the benefit of node ordering in IDA*.
[2] M.I.Jordan/C.M.Bishop - Neural Networks.
[3] E.Trentin et alii - Neural Networks for speech recognition.
[4] R.E.Korf - Depth-first iterative-deepening: an optimal admissible tree search.
[5] M.Gori/L.Lastrucci/G.Soda - Autoassociator-based models for speaker verification.
[6] S.V.Chenoweth/H.W.Davis - High performance A* search using rapidly growing heuristics.
[7] M. Ginsberg - Essentials of Artificial Intelligence.
[8] G.Ausiello et alii - Teoria e progetto di algoritmi fondamentali.
[9] B.Stroustrup - Il linguaggio C++ *(seconda edizione)*.
[10] C. Goller - A Connectionist Approach for Learning Search-Control Heuristics for Automated Deduction Systems (*Ph.D.thesis- Tech.University München,1997*)