Università degli Studi di
Napoli Federico II

Dipartimento di Ingegneria Elettrica e delle
Tecnologie dell'Informazione

Corso di Intelligenza Artificiale

Elaborato Esercitativo

Work Projects

Milo Saverio
Pommella Michele
Previtera Gabriele

Prof. Neri Filippo

Anno 2016-2017

Work Project 2

2.1 Esercizio 1: Evolutionary Optimization

2.1.1 Esperimento 1: Ottimizzazione

Attraverso gli algoritmi forniti siamo stati in grado di sperimentare le differenti tecniche di *ricerca locale* su diversi problemi. Esse si basano sul principio dei miglioramenti successivi: si cerca in primo luogo una soluzione ed in seguito ci si concentra sull'ottimizzazione. Va considerato che spesso si riesce a determinare solo un massimo locale, ottenendo risultati più o meno soddisfacenti a seconda dell'applicazione.

Random Searching

Questo algoritmo prevede semplicemente la determinazione casuale di un insieme di soluzioni e la comparazione dei relativi costi per l'identificazione di quella a costo minimo.
Abbiamo effettuato differenti esperimenti variando il numero di soluzioni estratte casualmente: ad un suo eccessivo aumento non si ottengono necessariamente risultati nettamente migliori da giustificare l'evidente incremento della complessità computazionale in termini temporali; d'altro canto, un numero esiguo di soluzioni produce risultati dalla qualità incostante.

Hill Climbing

L'`Hill Climbing` inizialmente determina una soluzione ed i suoi "vicini": se uno di essi presenta un costo minore della soluzione corrente, diventerà la nuova soluzione alla prossima iterazione, attuando, in tal modo, il processo dei miglioramenti successivi; se la soluzione corrente ha costo minore dei suoi vicini, termina la serie di miglioramenti iterativi poiché si è in presenza di un massimo locale (minimo locale rispetto al costo).
Vari esperimenti ci hanno condotto a risultati decisamente migliori del `Random Searching`, che sottolineano la maggior efficienza dell'`Hill`

`Climbing`. La variabilità nella qualità della soluzione, al seguito di svariate esecuzioni, è dipendente dai massimi locali del problema specifico.

Simulation annealing

Versione rivisitata dell' L'`Hill Climbing`, permette con un certa probabilità di provare anche un passo che non massimizza la funzione di costo, tale probabilità è data da $e^{-(migliorcosto+costoattuale/temperatura)}$ dove, più ci avviciniamo alla soluzione(diminuzione dei costi) e con l'abbassamento della temperatura,ad ogni iterazione dell' algoritmo, tale probabilità decresce.Tale strategia ha senso, per evitare che si ci possa assestare su un punto di massimo locale, invece così facendo possiamo trovare anche soluzioni migliori.

2.1.2 Esperimento 2: Caso d'uso

L' ottimizzazione sulla quale ci vogliamo concentrare è quella di una funzione $F(x)$, così definita (se $x < 5.2, F(x) = 10$; se $5.2 <= x <= 20, F(x) = x^2$; se $x > 20, F(x) = x - 1$) dove la x può assumere valori tra $[-100, 100]$, la ricerca di tale valore verrà effettuato grazie ad un algoritmo genetico.Prima di tutto però dobbiamo definire la funzione di costo,così da discriminare quando la nostra soluzione tende a migliorare oppure peggiora:

```
1 def costmax(sol):
2         if sol[0]>100 or sol[0]<-100:
3                 cost=0
4         else:
5                 cost=F(sol)
6 return cost
```

molto semplicemente se il valore di cui si calcola il costo non è accettabile, perché tra il passaggio tra una generazione ad un' altra si è avuto un gene non voluto, gli si da un costo 0 così per la creazione della prossima generazione non verrà preso in considerazione.

2.2 Esercizio 2: MDS: Visually Exploring US Senator Similarity

2.2.1 Clustering

Il Clustering o analisi di raggruppamento è una tecnica di intelligenza artificiale volta alla selezione e raggruppamento di elementi omogenei in un

insieme di dati. Tutte le tecniche di clustering si basano sul concetto di distanza tra due elementi,che ci permette di definire il concetto di somiglianza tra gli elementi, infatti l'appartenenza o meno ad un insieme dipende da quanto l'elemento preso in esame è distante dall'insieme. La bontà delle analisi ottenute dagli algoritmi di clustering dipende dalla metrica, metriche diverse porteranno quasi certamente a cluster differenti.

2.2.2 MDS: Multidimensional scaling

Il MultiDimensional Scaling è una tecnica di analisi statistica usata per mostrare graficamente le differenze o somiglianze tra elementi di un insieme. È una generalizzazione del concetto di ordinamento: partendo da una matrice quadrata, contenente la "somiglianza" di ogni elemento di riga con ogni elemento di colonna, l'algoritmo di scaling multidimensionale assegna a ogni elemento una posizione in uno spazio N-dimensionale, con N stabilito a priori.In pratica questa tecnica parte con un sistema con tante dimensioni quanti gli elementi del sistema, e riduce le dimensioni fino a un certo numero N. Nel fare questo quindi c'è un'inevitabile perdita di informazione.

2.2.3 Riflessioni

Nel codice che ci e' stato fornito un abbiamo a disposizione due esempi, il primo si basa su il giudizio da parte di quattro venditori riguardo a 6 prodotti, e il secondo esempio si basa sull'analisi dei voti di 10 congressi americani. Lo scopo è quello di raggruppare i venditori e i senatori in base alle loro preferenze. Nel codice sono forniti anche i dataset, alcune funzioni di utilità che permettono di convertire i dataset in dataset più compatti. Inoltre la metrica utilizzata per la distanza è quella euclidea.
Dal Multidimensional scaling del primo esempio giungiamo alla seguente conclusione :" i recensori A e D hanno valori di metrica abbastanza vicini tra loro mentre per i recensori B e C no. Analizzando i dati che abbiamo a disposizione non rimaniamo sorpresi di questi risultati,l'algoritmo classifica A e D come simili infatti hanno espresso quattro pareri concordanti su sei, mentre B e C hanno espresso solo due concordanti e non è un buon risultato per classificarli come simili. Ovviamente per considerazioni più approfondite abbiamo bisogno di un numero maggiore di dati."
Per il secondo esempio costituito da un dataset reale e molto ampio giungere a delle conclusioni è stato più difficile. Una prima immagine che ci fornisce il risultato dell'elaborazione consiste nello schieramento dei Repubblicani a destra e dei Democratici a sisnistra e inoltre vediamo che la maggior parte dei senatori è schierata in uno dei due partiti. Nelle immagini successive

abbiamo i nomi dei senatori e che appartengono ad un determinato partito ed effettuando delle ricerche su internet vediamo che effettivamente la classificazione ottenuta è buona, per esempio Obama viene correttamente classificato come Repubblicano mentre Jeffords, il cui ha un passato da repubblicano viene riportato a destra ma ha un colore diverso in quanto ora è un indipendente. L'ultima serie di immagini che l'algoritmo ci fornisce, possiamo vedere che le scelte di voto dei partiti sono consistenti.Cioè se un partito sceglie di elargire quel voto, tutti gli iscritti al partito si attengono, in generale, a quella scelta.

2.3 Esercizio 3

2.4 Esercizio 4