ALGORITMO DI LEARNING PER ALBERI DI DECISIONE E PRUNING

I file all'interno del file 'archivio.tar' descrivono la procedura di apprendimento per alberi di decisione come descritto in R & N 2009 §18.3, con l'utilizzo dell'entropia come misura di impurità, aggiungendo infine un algoritmo di prunning per la potatura dell'albero creato precedentemente.
I file utilizzati a tale scopo sono sostanzialmente tre:
BuildingDecisionTree: questo file mi contiene l'algoritmo DT_Learn che descrive la procedura di apprendimento dell'albero di ricerca.
Pruning : contiene l'algoritmo per eseguire il post-pruning
Analisi : contiene le funzione atte a fare la classificazione e, per l'appunto, l'analisi dell'errore commesso definendo l'accuratezza dell'albero nel classificare i dati di test.
I file restanti sono:
BuildDataset: all'interno ci sono delle funzioni che servono semplicemente a costruire il training set e il test set. Inoltre troviamo la funzione per creare il validation set usato nella fase di pruning.
Read_MNIST: è l'unico file preso da parti esterne che serve soltanto a leggere i file t10k-images-idx3-ubyte, t10k-labels-idx1-ubyte, train-images-idx3-ubyte, train-labels-idx1-ubyte e riportare il contenuto in matrici di 60000 righe e 784 colonne per il training set e 10000 righe e 784 colonne per il test set, con l'aggiunta di due vettori rappresentanti uno le classi del training set e l'altro le classi del test set.
Main: file che serve semplicemente per far partire tutte le funzioni

Il Dataset utilizzato è il MNIST che rappresenta un Dataset di immagini scritte a mano. Questo è rappresentato, come detto precedentemente , come una matrice . Le righe mi rappresentano le immagini scritte a mano, le colone i pixel accesi per l'immagine. Il database viene quindi preparato dal file BuildDataset in cui ci sono tre funzioni **:** makeTrainingDataset(),makeTestDataset(),makeValidationSet().
La prima funzione citata, utilizzando il metodo load(), definito nel file Read_MNIST, legge i file riguardanti il training set e appende , per ogni immagine , la sua classe come ultimo parametro ottenendo una matrice con 785 colonne. In questo modo è più semplice l'utilizzo di quest'ultimo in DT_Learn.
Le restanti funzioni fanno la stessa cosa ma la differenza è che restituiscono una matrice contenente l'immagini e un vettore contenente le classi delle immagini. Questo serve a semplificare l'operazione di pruning e di analisi.
Vediamo adesso come viene fatto l'apprendimento dell'albero di ricerca.
La funzione che viene utilizzata si trova nel file BuildingDecisionTree ed è DT_Learn. Questa prende in ingresso il training set, gli attributi(che in questo caso non sono altro che gli indici delle colonne) e il target attribute (anche questo è un indice che mi rappresenta l'ultima colonna della matrice in cui si trovano le classi).
Essendo una funziona che lavora con chiamate ricorsive all'inizio si trovano due controlli: uno controlla se ci sono ancora attributi , l'altro invece controlla se le classi rimanenti del dataset sono tutte uguali. In ambo i casi viene creata una foglia rappresentata dalla classe trasformata in stringa, in modo tale da riuscire a distinguerle dai nodi nella fase di classificazione. Se i due controlli risultano falsi si procede alla creazione dell'albero. Prima di tutto viene scelto il best attribute: questo è l'attributo che ha la

maggior information gain. Questa scelta viene fatta chiamando la funzione takeBestAttribute a cui si passa dataset, attributi e target attribute. Dopo averlo selezionato, questo viene messo come radice di un albero, rappresentato da un dizionario. Successivamente, per ogni valore che quel attributo può assumere, viene creato un sub-dataset, attraverso la funzione makeSubDataset, in cui ogni immagine ha nell'attributo scelto il valore corrente del ciclo. Questo viene usato per creare un sottoalbero chiamando ricorsivamente la funzione DT_Learn, il quale verrà aggiunto al nostro albero di partenza. Infine viene ritornato dalla funzione tutto l'albero.

Dopo la costruzione dell'albero si può fare la classificazione. Questa viene fatta attraverso la funzione che si trova nel file Analisi. Il metodo classify che, per ogni immagine, prende l'attributo definito dal nodo radice, controlla quale è il valore relativo a quell'attributo nell'immagine e successivamente continua lo studio nel sottoalbero che ha come radice il nodo a cui arrivo in base al valore dell'attributo studiato. Questo processo è eseguito fino a che non si arriva ad una foglia.

Infine si controlla l'accuratezza comparando le classi trovate con l'albero con le classi reali del test set.

Il pruning dell'albero si fa attraverso la funzione postpruningTree(), contenuta nel file Pruning. Questo metodo ispeziona ogni nodo dell'albero. Preso un nodo lo trasforma in foglia , mettendoci la classe più comune rispetto al training set riferita a quel nodo. Successivamente va a verificare, sul validation set, se l'accuratezza del nuovo albero è maggiore rispetto a quello precedente. Se si lascia la foglia, altrimenti elimina la potatura e , in entrambi i casi, ripete la procedura per un altro nodo.

Il file main è il file che fa partire tutto quello che è stato descritto precedentemente, scegliendo il numero di esempi da utilizza. Si

può notare che l'accuratezza varia in base alla grandezza del training test e del test set : più aumento le immagini contenute nel training test più , in seguito, riconoscerò le immagini contenute nel test set. Si può vedere infatti che, considerando un training con 10000 immagini, l'accuratezza è :

- Con 50 immagini nel test set intorno al 30 percento
- Con 1000 immagini nel test set intorno al 31 percento
- Con 10000 immagini nel test set intorno al 32,5 percento

I risultati dopo il pruning non cambiano molto

NB : il computer utilizzato per fare i test non permetteva di usare database molto grandi . Il numero massimo di immagini utilizzate per il training sono 10000. Anche con la diminuzione della grandezza delle immagini, invece che immagini 28x28 si sono ridotte a immagini 14x14, il problema sussisteva. Per questo motivo non è stato possibile effettuare dei test più accurati.

Bazzanti Giulio