Algoritmo Mean Shift Clustering

Adalberto Lombardi

Firenze 20/04/2023

Obiettivo

- Il mean shift clustering è algoritmo di clusterizzazione utilizzato nell'apprendimento non supervisionato;
- E’ detto non parametrico;
- Assegna i punti ai cluster in modo iterativo spostando i centrioidi verso la più alta densità di punti dati;
- Non è necessario specificare il numero di gruppi in anticipo;
- L’obiettivo è di trovare i picchi o le zone di densità nei dati che corrispondono ai cluster;
- E’ un algoritmo di *hill climbing,* ossia si muove continuamente nella direzione di maggiore elevazione terminando al raggiungimento del picco massimo.

Struttura

Il suo funzionamento può essere sintetizzato nei seguenti passi:

- Inizializzazione del centroide per ogni punto nel set di dati;
- Per ogni centroide, calcolare la media ponderata di tutti i punti nel dataset, pesando ogni punto in base alla sua distanza dal centroide;
- Spostare il centroide alla media ponderata calcolata;
- Ripetere i passi precedenti fino a quando il centroide non si sposta più di una certa soglia predefinita chiamata “picco”;
- Assegnare ogni punto al centroide più vicino.

Implementazioni

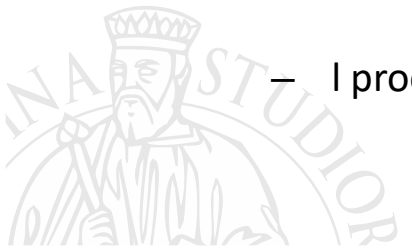
Flusso

- Sono state implementate due versioni dell'algoritmo: una sequenziale ed una parallela;
- La funzione *euclidean_distance* per calcolare la distanza euclidea tra due punti;
- La funzione *mean_shift* per eseguire il clustering;
- La variabile *start_time* contiene il momento di avvio dell'algoritmo, mentre la variabile *end_time* quello in cui termina;
- Nel codice principale vengono letti i dati in input da un file esterno;
- Viene effettuata la chiamata alla funzione *mean_shift;*
- Il tempo di elaborazione viene calcolato tramite differenza tra i valori *end_time* e *start_time e* mostrato a video.

Implementazioni

Algoritmo Sequenziale

- La variabile *radius* imposta la larghezza di banda;
- La lista *labels* per tenere traccia delle etichette di clustering;
- La lista *new_centroids* per tracciare i nuovi centroidi;
- La lista *within_radius* che contiene i punti all'interno del raggio di banda *radius;*
- Il ciclo *while* in cui viene calcolata la distanza tra il punto osservato ed il centroide;
  - Se il punto è all'interno del raggio di radius lo aggiunge alla lista *within_radius;*
  - Vengono calcolati i nuovi valori dei centroidi;
  - l processo itera fintantoché il calcolo dei nuovi centroidi non subisce più variazioni.



Implementazioni

Algoritmo Parallelo

- E’ stata creata una funzione *mean_shift_async* che esegue il mean-shift clustering su un singolo blocco di dati in modo asincrono;
- La funzione *run_mean_shift_parallel* permette la suddivisione dei dati in n_jobs blocchi e viene creato una task per ogni blocco;
- Per ogni blocco viene chiamata la funzione *MeanShift;*
- Infine con la funzione *asyncio.gather* è possibile unire i risultati dei singoli blocchi.

Risultati

- Gli algoritmi sono stati testati variando il dataset in input 1.000, 10.000 e 20.000 osservazioni;

- All'interno dell'esecuzione parallela sono state fatte differenti prove variando il numero di processi da 2, 4, 6, 8, 10 e 12;

	n. thread	1.000	10.000	20.000
Sequenziale		16.93860	2259.1824	9114.0300
Parallelo	2	13.7363	292.8279	915.0976
	4	8.2244	190.4728	577.9066
	6	6.1631	203.8658	548.5062
	8	5.1105	162.5257	378.0969
	10	4.2964	144.2527	439.6670
	12	4.3059	140.9795	399.7839

Tabella 1: Tabella dei risultati

Conclusioni

- Il programma con AsyncIO è più efficiente in termini di risorse computazionali, poiché sfrutta la parallelizzazione per eseguire il clustering in modo asincrono su più worker;
- Ha una maggiore efficienza nell'elaborazione di insiemi di dati di grandi dimensioni;
- L'utilizzo di AsyncIO permette di eseguire altre attività durante l'attesa delle operazioni di I/O;
- La velocità di esecuzione parallela dipende dalla disponibilità di risorse del sistema e dalla dimensione dei dati da elaborare;
- La scelta del numero di processi paralleli da utilizzare dipende dal singolo caso;

Conclusioni

- Il programma parallelo dovrebbe essere preferito per eseguire il clustering su insiemi di dati di grandi dimensioni e per una maggiore efficienza computazionale che di fatto né migliorano lo speedup;

- Il programma sequenziale può essere una scelta migliore per insiemi di dati più piccoli o per situazioni in cui la semplicità e la facilità di utilizzo sono prioritari.