



UNIVERSITÀ
DEGLI STUDI
FIRENZE



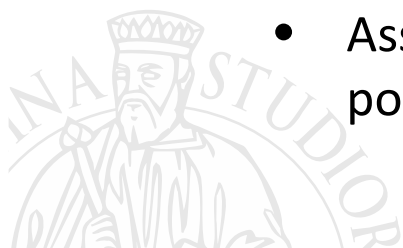
Algoritmo K-means Clustering

Adalberto Lombardi

Firenze 20/04/2023

Obiettivo

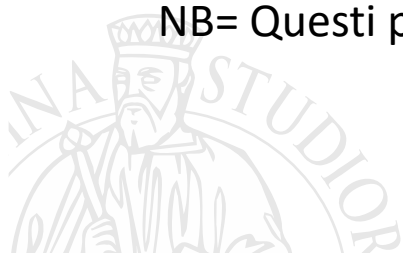
- Il k-means clustering è un algoritmo non supervisionato utile per la segmentazione dei dati.
- L'obiettivo è quello di raggruppare le osservazioni simili per scoprire pattern nascosti dividendo i dati in cluster.
- Creazione di un punto di riferimento denominato centroide
- Assegnare i dati in input ai cluster in base al punto di riferimento più vicino
- Questo permette di:
 - Assicurati che tutti i punti dati in un cluster siano il più simili possibile tra loro;
 - Assicurati che tutti i punti dati nei diversi cluster siano il più possibile dissimili tra loro.



Fasi

1. Posizionare tutte le istanze in sottoinsiemi, dove il numero di sottoinsiemi è uguale a K ;
2. Trovare il punto/centroide medio delle partizioni del cluster appena create;
3. Sulla base di questi centroidi, assegna ogni punto a un cluster specifico;
4. Calcolare le distanze da ogni punto ai centroidi e assegnare punti ai cluster in cui la distanza dal centroide è la minima;
5. Dopo che i punti sono stati assegnati ai cluster, trovare il nuovo centroide dei cluster.

NB= Questi passaggi vengono ripetuti fino al termine del processo di formazione.



Alternativa

- Dopo che i centroidi sono stati posizionati
- Vi è scambio avanti e indietro tra due diverse fasi:
 - Etichettatura del punto dati, ad ogni osservazione viene assegnata un'etichetta che lo colloca nel cluster appartenente al centroide più vicino;
 - Aggiornamento del centroide, questo viene calcolato trovando la distanza media tra tutti i punti dati attualmente contenuti in un cluster.

Scelta del numero di classi

- Un decisione iniziale molto importante riguarda la scelta del numero di classi poiché non è noto a priori;
- Vi sono diverse tecniche che permettono una sua determinazione;
- Una delle più note è la tecnica del gomito che consiste in:
 - Esecuzione di un algoritmo di clustering K-means per un intervallo di diversi valori K;
 - Utilizzo di una metrica di precisione come la somma dell'errore quadratico, per determinare quali valori di K danno i risultati migliori;



Casi d'Uso

- Utilizzato in qualsiasi situazione in cui i punti dati possono essere segmentati in gruppi/classi distinti;
- Classificazione dei documenti: raggruppando i documenti in base a funzionalità come argomenti, tag, utilizzo delle parole, metadati;
- Classificare gli utenti: in base a modelli di attività come post e commenti;
- Raggruppare le persone durante il monitoraggio della loro salute: in base a caratteristiche come comorbidità, età, anamnesi del paziente;
- Creazione di sistemi di “raccomandazione”: nello streaming video dove gli utenti possono essere raggruppati in base a modelli di visualizzazione e contenuti simili consigliati;
- Rilevamento di anomalie: identificando potenziali casi di frode o articoli difettosi.

Implementazioni

Definizioni variabili e metodi

- Sono state implementate due versioni dell'algoritmo: una sequenziale ed una parallela
- Il numero di cluster è impostato preventivamente tramite la variabile k
- Il vettore *data* contiene i punti da clusterizzare.
- Il tempo di elaborazione è misurato utilizzando la funzione *clock* della libreria *ctime*
- Il vettore *centroids* contiene i centroidi
- Il vettore *clusters* contiene l'assegnazione di ogni punto ad un cluster
- La funzione *kMeansClustering* viene chiamata per eseguire l'algoritmo K-means clustering

Implementazioni

Flusso

- Inizialmente i centroidi sono assegnati ai primi k punti del vettore *data*
- La funzione *distance* calcola la distanza euclidea tra due punti
- Il ciclo *while* esegue l'algoritmo K-means finché i centroidi non convergono o raggiungono un numero massimo di iterazioni
- Nel primo ciclo avviene l'assegnazione dei cluster ai punti
- Nel secondo ciclo avviene l'aggiornamento dei centroidi



Implementazioni

Algoritmo Parallelo

- L'esecuzione parallela inizia con la creazione dei centroidi in modo casuale e l'assegnazione di ogni punto al cluster più vicino
- Nel ciclo principale, anche questo parallelo, vengono calcolati i nuovi centroidi e assegnati i punti ai cluster più vicini fino a convergenza
- Il vettore di contatori per mantenere il numero di punti assegnati a ciascun cluster richiede una sezione critica in cui ogni thread deve eseguire l'incremento del contatore in modo atomico
- Anche l'aggiornamento dei centroidi deve essere atomico



Risultati

- Gli algoritmi sono stati testati variando il numero di cluster impostati con 2, 5 e 10 gruppi
- Il dataset in ingresso partendo da 10.000, quindi 100.000 ed infine 1.000.000 di osservazioni

| classi \ osservazioni | 10.000 | | 100.000 | | 1.000.000 | |
|--------------------------|-------------|-----------|-------------|-----------|-------------|-----------|
| | Sequenziale | Parallelo | Sequenziale | Parallelo | Sequenziale | Parallelo |
| 2 | 0,11478 | 0,02606 | 1,64101 | 0,72723 | 33,38240 | 26,79750 |
| 5 | 0,14901 | 0,05453 | 4,10492 | 3,55278 | 72,75070 | 61,51660 |
| 10 | 0,46028 | 0,10521 | 13,85390 | 5,36158 | 138,81700 | 107,15100 |

Tabella 1: Tabella dei risultati

Conclusioni

- Il programma parallelo risulta avere prestazioni migliori
- OpenMP consente di eseguire il lavoro su più thread, sfruttando i core della CPU
- Fattori determinanti:
 - il numero di core disponibili sulla piattaforma hardware;
 - l'efficienza dell'implementazione parallela;
 - dimensioni del dataset
 - Numero di classi
- Se il numero di classi preimpostate è relativamente piccolo rispetto alle dimensioni del dataset non sembra esserci una differenza così marcata di prestazioni
- Aumentando la dimensione del dataset l'implementazione parallela risulta avere uno speedup marcato rispetto alla versione sequenziale.