PC-2016/17 Course Project
Implementazione in CUDA dell'algoritmo KMean

Tommaso Ceccarini
E-mail address
tommaso.ceccarini1@stud.unifi.it

Federico Schipani
E-mail address
federico.schipani@stud.unifi.it

Abstract

L'algoritmo di KMeans ÃÍ uno dei piÃź popolari metodi per la clusterizzazione. Nel nostro lavoro forniamo una implementazione in CUDA che fa uso di GPU Nvidia per l'esecuzione dell'algoritmo. Inoltre forniamo un'analisi delle preformance con lo scopo di comparare la nostra implementazione parallela con una implementazione sequenziale scritta in C.

1. Introduzione

L'algoritmo KMeans ÃÍ uno degli algoritmi di clustering piÃź famosi. Lo scopo del clustering ÃÍ di dividere dati in gruppi, chiamati cluster. I cluster risultanti dall'esecuzione dell'algoritmo cattureranno la struttura dei dati.

I metodi di KMeans riescono ad eseguire l'operazione di clustering valutando una condizione di similaritÃă **KMeans methods attempt to do this by evaluating a similarity measure according to the mean value of the data that are cointained in the clusters.** Quindi, dato un insieme di dati $(\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_N})$ dove ogni dato ÃÍ un vettore reale di dimensione P, lo scopo di KMeans ÃÍ di partizionare N dati in $K(\leq N)$ insiemi $\boldsymbol{S} = \{S_1, S_2, \ldots, S_K\}$ per minimizzare la minima distanza dentro ad i singoli cluster. In altre parole il suo obiettivo ÃÍ trovare:

$$\arg\min_{\boldsymbol{S}} \sum_{i=1}^{k} \sum_{x \in S_i} ||\boldsymbol{x} - \boldsymbol{\mu}_i||^2 \tag{1}$$

dove μ_i ÃÍ la media dei punti in S_i.[1]

L'Algoritmo 1 rappresenta lo pseudocodice di un'implementazione iterativa dell'algoritmo KMeans.

I principali passi dell'algoritmo sono:

1. Selezionare casualmente K dati e renderli la media iniziale dei K cluster. Questo passo ÃÍ usualmente chiamato *initialization*.

Algoritmo 1 KMeans

```
1:  procedure KMEANS(data[n][p])
2:      mean[k][p], oldMean[k][p]
3:      assignment[n]
4:      (mean, assignment) ← initAss(data)
5:      while !stop do
6:          assignment ← calcMin(mean, data)
7:          oldMean ← mean
8:          mean ← calcMean(assignment, data)
9:          stop ← stopCrit(mean, oldMean)
10:     end while
11:     return mean
12: end procedure
```

2. Assign each data to the cluster whose mean yields the least within-cluster sum of squares. This step is usually called *assignment*.

3. Compute the new means to be the centroids of the data in the new clusters. This step is usually called *update*.

4. Repeat step 2 and 3 until assignment no longer change or only few data change during the last iteration.

1.1. KMeans sequenziale

Prima del ciclo *while* c'ÃÍ un passo di assengazione iniziale. In questo step i primi K dati sono assegnati ai primi K cluster. Uno pseudocdice di questo assegnamento iniziale ÃÍ mostrato in Algoritmo 2

Dopo questo step iniziale di assegnamento parte il vero e proprio algoritmo. Nella prima parte del ciclo *while* viene calcolata la distanza euclidea minima tra ognuno dei dati e tutti i cluster. Successivamente si assegna ogni dato al cluster piÃź vicino. Il passo successivo consiste nel ricalcolare le nuove medie dei cluster, secondo la (2).

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \tag{2}$$

Algoritmo 2 Initial Assignment

```
1:  procedure INITASS(data[n][p])
2:      mean[k][p]
3:      assignment[n]
4:      for i = 0; i < k; i++ do
5:          assignment[i] = i
6:          for j = 0; j < p : j++ do
7:              mean[i][j] = data[i][j]
8:          end for
9:      end for
10:     return (mean, assignment)
11: end procedure
```

L'ultimo step ÃÍ il criterio d'arresto. In questo semplice algoritmo, mostrato in 3, si iterano l'attuale matrice dei cluster e la matrice dei cluster calcolata al passo $i-1$. Quando $ActualValue - OldValue$ eccede una prefissata tolleranza ϵ il metodo ritorna il valore *false*. Questo valore di ritorno porterÃă ad un'altra iterazione del ciclo *while* esterno.

Algoritmo 3 Stop Criterion

```
1:  procedure STOPCRIT(mean[k][p], oldMean[k][p])
2:      for all value, oldValue in mean, oldMean do
3:          if abs(value − oldValue) > ε then
4:              return false
5:          end if
6:      end for
7:      return true
8:  end procedure
```

2. Implementazione

2.1. Come vengono rappresentati i dati ed i cluster

Le due differenti implementazioni dell'algoritmo di KMeans, riportate in sezione 2.2 e 2.3, risolvono il problema del clustering per un valore arbitrario del parametro P. Tuttavia le analisi delle performance, mostrate in Sezione 3, sono state effettuate nel caso in cui $P = 2$, ovvero il caso in cui i dati sono vettori bidimensionali.

Le due maggiori strutture su cui il codice lavora sono la matrice dei dati e la matrice dei centroidi, rispettivamente di dimensione $N * P$ e $K * P$. In entrambe le implementazione viene usato anche un vettore chiamato *assignment* che, per tutti gli N dati, memorizza l'indice del cluster al quale appartengono.

mettere il codice delle dichiarazioni delle tre strutture?

2.1.1 Come viene generato il dataset

Dopo che sono state dichiarate, e successivamente istanziate, le strutture che sono necessarie per rappresentare i dati e i centroidi viene effettuata una genrazione pseudocasuale dei dati. Questa generazione viene effettuata in maniera parallela attraverso le API CUDA, in particolare ÃÍ stata usata la libreria cuRAND. Il metodo che ÃÍ stato sviluppato per questo scopo ÃÍ `generateRandomDataOnDevice()`. Quesot metodo usa un seed prefissato e, con l'aiuto di un altro metodo che fa parte della libreria cuRAND, genera i valori del dataset. Nella Sezione 3 sono mostrate le analisi delle performance nel caso in cui i dati sono generati con una distribuzione normale o uniforme. **mettere il codice del metodo che generate i dati**

2.2. Una implementazione sequenziale in C

Lo step di inizializzazione

Dopo aver generato casualmente il dataset nella memoria globale del device ÃÍ necessario istanziare la memoria richiesta per memorizzare la matrice dei dati e dei centrodi nella memoria dell'host. Inoltre, per come funziona l'algoritmo, ÃÍ necessaria un'altra matrice che memorizza il valore della media durante la precedente iterazione. Questa matrice ÃÍ usata per implementare il criterio d'arresto. L'inizializzazione ÃÍ effettuata da un semplice ciclo for che assegna `data[i][j]` a `mean[i][j]` per `i` tra 0 e $K-1$. Questa strategia d'assegnamento ÃÍ motivata dal fatto che i dati sono generati casualmente. **codice dell'istanziazione delle matrice e ciclo for che effettua l'inizializzazione**

Lo step di assegnamento

Lo step di aggiornamento

Il criterio d'arresto

ESEMPIO ESEMPIO ESEMPIO CODICE 1

Codice 1: blablabla

```
1  while (!stopCriterion) {
2    for (int i = 0; i < N; i++) {
3      float minDistance = 999999.9;
4      short minIndex = -1;
5      float distance = 0;
6      for (int z = 0; z < K; z++) {
7        distance = 0;
8        for (int j = 0; j < COMPONENTS; j++) {
9          distance +=pow(
10             (hostData[i * COMPONENTS + j]
11                 - mean[z * COMPONENTS + j]),2);
12
13       }
14       if (distance < minDistance) {
15         minIndex = z;
16         minDistance = distance;
17       }
```

```
18
19        }
20        assignment[i] = minIndex;
21      }
```

2.3. Parallel CUDA Implementation

3. Performance analysis

Riferimenti bibliografici

[1] Wikipedia. K-means clustering — wikipedia, the free encyclopedia, 2017. [Online; accessed 23-January-2017]. 1