Report risultati scalable K-means algorithm

Damiano Bellucci

1 Introduzione

L'obiettivo di questo progetto è eseguire uno studio di data analysis per studiare l'efficacia dell'algoritmo K-means, implementato per essere eseguito su risorse scalabili, per problemi di transport mode detection e human activity recognition, su dataset fatti da dati provenienti da sensori di smartphone. In particolare, si è cercato di capire se a partire da questi dati è possibile fare clusterizzazione per individuare i cluster di dati relativi ai tipi di trasporto oppure alle attività umane. Possibili risvolti di questo studio possono essere il riconoscimento di tipi diversi modalità di trasporto/attività umane a partire da dati di sensori, senza però conoscerne la denominazione.

Lo studio è stato svolto adottando tecniche di analisi dei dati scalabili, tramite Spark, una libreria che permette di eseguire computazione parallela scrivendo codice sequenziale, con relativa gestione delle risorse di calcolo. Infatti lo stesso codice è stato mandato in esecuzione su diverse risorse di calcolo, dalla macchina locale ai cluster con più nodi di calcolo del servizio di cloud computing Google Cloud Platform. L'esecuzione su risorse scalabili permette di capire se lo studio può essere effettuato, anche in futuro con diversi datasets, più efficacemente, avendo a disposizione più capacità di calcolo.

Il linguaggio di programmazione utilizzato è Scala, un linguaggio tipato staticamente la cui compilazione del codice produce Java Bytecode eseguibile sulla Java Virtual Machine. Supporta la programmazione object oriented e la programmazione funzionale.

Per quanto riguarda la potenza di calcolo adottata, per l'esecuzione locale è stata utilizzata una macchina con processore 2,2 GHz Intel Core i7 quad-core e 16GB RAM e 256GB SSD, mentre per l'esecuzione in cloud per il master Intel Skylake 4vCPU e 15 GB di RAM e 65GB SSD, così come per i nodi worker.

2 Risultati di clusterizzazione

In questa sezione i risultati di clusterizzazione ottenuti per D1 (transport mode detection dataset) e D2 (human activity recognition dataset). I numero massimo di clusters considerato è stato di 20.

La messa in relazione del numero di cluster e wcss (within clusters sum of squares, un indice di misura della dispersione dei punti nei relativi cluster) è detta elbow method e serve per individuare una giusta clusterizzazione del dataset. Infatti, nel momento in cui si stabilizza il valore della wcss all'aumentare del numero dei cluster, significa che una buona compattezza dei cluster è stata raggiunta. Ne consegue che il numero giusto di cluster da considerare è quello dove la wcss inizia a stabilizzazirsi.

Uno degli obiettivi di questo studio è quello di capire se con l'applicazione dell'algoritmo K-means si riesce ad ottenere un numero di cluster che è quello originale del dataset.

Per quanto riguarda D1, il numero di cluster è 5. Come si può osservare in figura 1 i risultati ottenuti in questo studio con la tecnica dell'elbow ci suggeriscono che il numero di cluster più giusto è 9, in contrapposizione ai 5 della realtà.

Il Calinski Harabasz index è un indice che permette di fare studi sul rapporto tra la "in between sum of squares" (dispersione tra i clusters) e la wcss, cioè la varianza tra i clusters e la varianza all'interno dei cluster. Un alto numero è frutto di una clusterizzazione con clusters lontani e compatti, per questo si può usare analogamente alla wcss per fare studi sulla qualità della clusterizzazione. In figura 2 si mostra l'andamento del CH index all'aumentare del numero di cluster e si ha un massimo per numero di cluster che è 3, dove c'è la maggiore separazione tra clusters rapportata alla loro compatezza. Anche in questo caso, contro le aspettative.

Un altro indice usato per lo studio è la media wcss, cioè la somma dei singoli wcss per ogni cluster, normalizzati prima con il loro numero di samples, il tutto sommato e poi diviso per il numero dei clusters. E' un indice per confrontare la varianza totale dei cluster al variare di K. Un valore che

scende con il numero di clusters indica che man mano i clusters sono sempre più compatti. Come si può vedere in figura 3 si smette di migliorare significativamente la compattezza dei clusters dopo un numero di cluster uguale ad 8, contro l'aspettativa di 5.

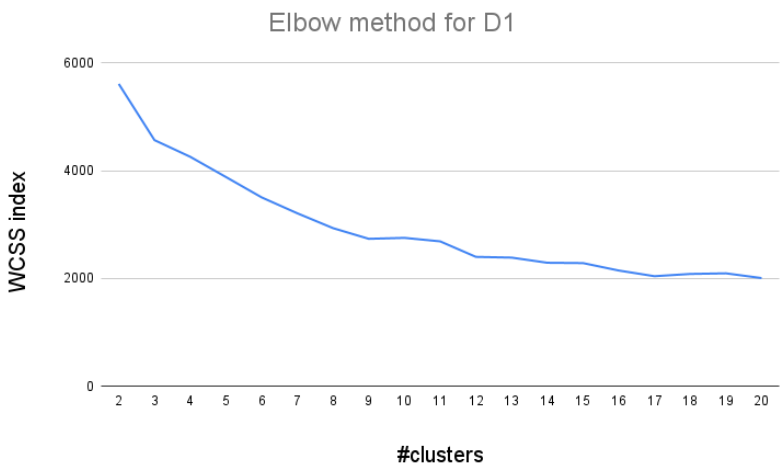

Figura 1: Elbow method per D1

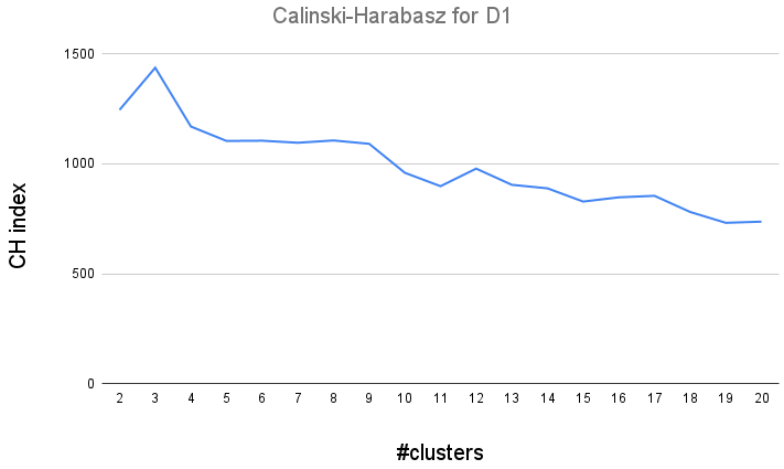

Figura 2: Calinski-Harabasz index per D1

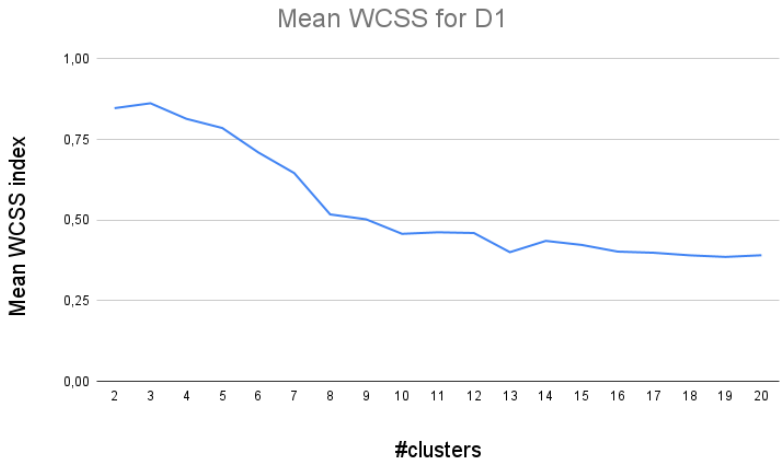

Figura 3: Mean WCSS per D1

Per quanto riguarda D2, come si può notare in figura 4 che per wcss si ha un elbow intorno ad un numero di cluster uguale a 8, così come suggerisce la media wcss in figura 6. Come si può notare per l'indice Calinski-Harabasz invece, l'ottimo in questo caso si raggiunge già con due cluster, con un valore che va a scendere progressivamente, ad indicare l'ottenimento di cluster via via più sparsi internamente e vicini tra loro, cioè non una buona clusterizzazione. Questo sta ad indicare che non si può fare clusterizzazione efficace del dataset.

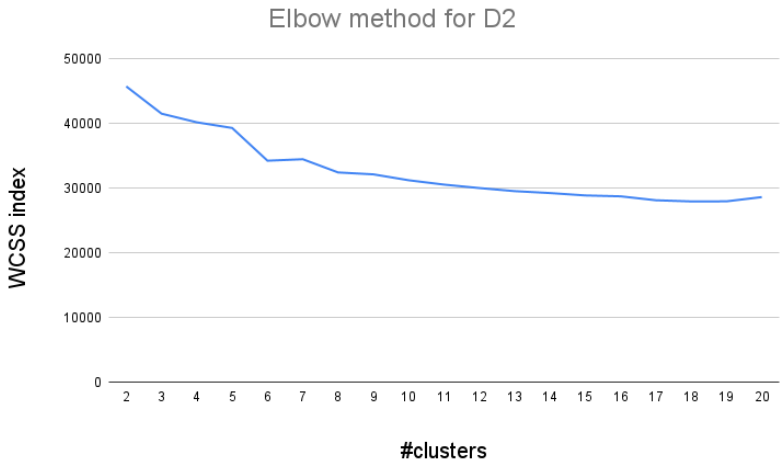

Figura 4: Elbow method per D2

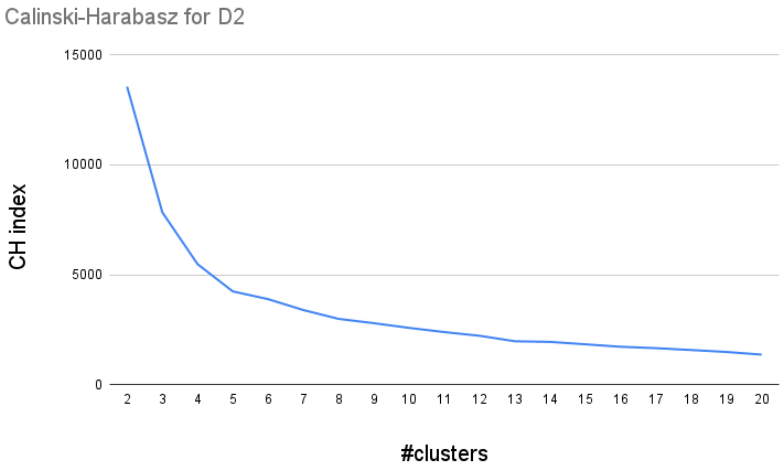

Figura 5: Calinski-Harabasz index per D2

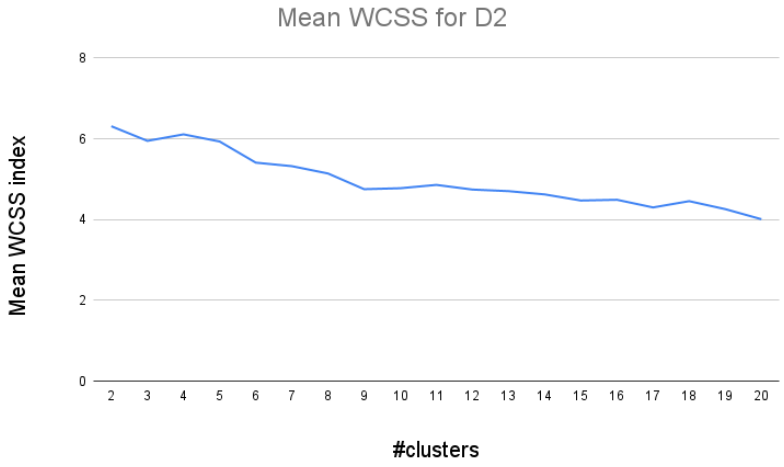

Figura 6: Mean WCSS per D2

Come benchmark per l'agoritmo k-means implementato per questo studio è stato adottato l'algoritmo k-means della libreria scikit-learn, i risultati nelle figure 7 e 8 per l'elbow method con wcss. Come si può vedere, i risultati sono paragonabili a quelli ottenuti in questo studio, con un elbow intorno k=8 per D1 mentre per D2 intorno ai 6. I risultati dei due algoritmi sono quindi paragonabili.

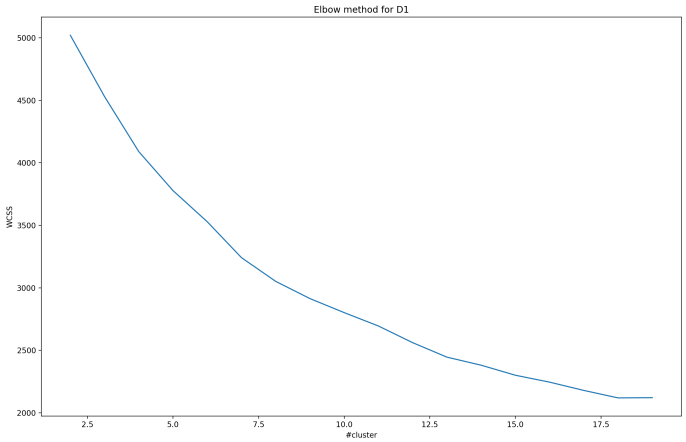

Figura 7: Benchmark elbow method per D1

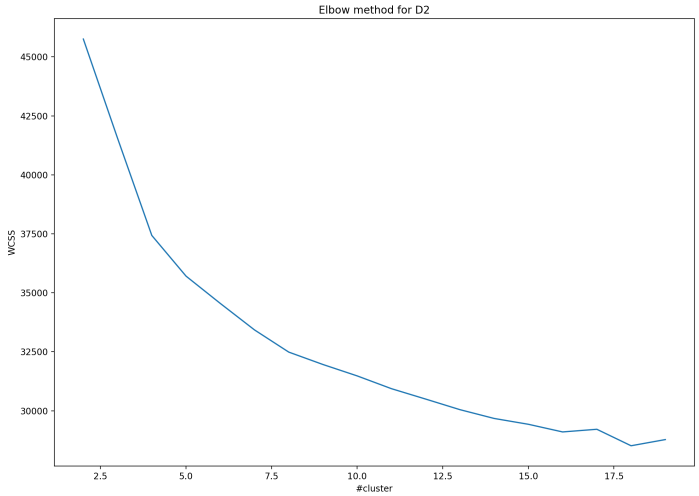

Figura 8: Benchmark elbow method per D2

3 Risultati performance

In questa sezione si riportano i risultati riguardanti il tempo totale di esecuzione di k-means da 2 a 20 clusters, con varie risorse di calcolo, dal locale con 1 e 4 threads alle risorse nel cloud con cluster con numero di nodi 2 e 4.

Come si può notare nelle figure 9 e 10, che mostrano il tempo totale impiegato per eseguire k-means da 2 a 20 cluster, si ottengono risultati migliori con l'esecuzione locale, in particolare quella con 4 threads.

Per D1 l'esecuzione peggiore è, come facilmente prevedibile, con la configurazione locale ad 1 thread, mentre la migliore con locale a 4 threads. I tempi nel cluster sono di poco migliori se vengono aumentati i nodi (da 2 a 4) ma rimangono comunque sotto il miglior risultato in locale.

Per d2 invece, l'esecuzione migliore è quella locale con 4 threads, mentre la peggiore il cluster con 4 nodi che addirittura performa peggio dell'esecuzione locale con un solo thread. Inoltre, nel cluster con numero di nodi maggiore si ha una performance peggiore.

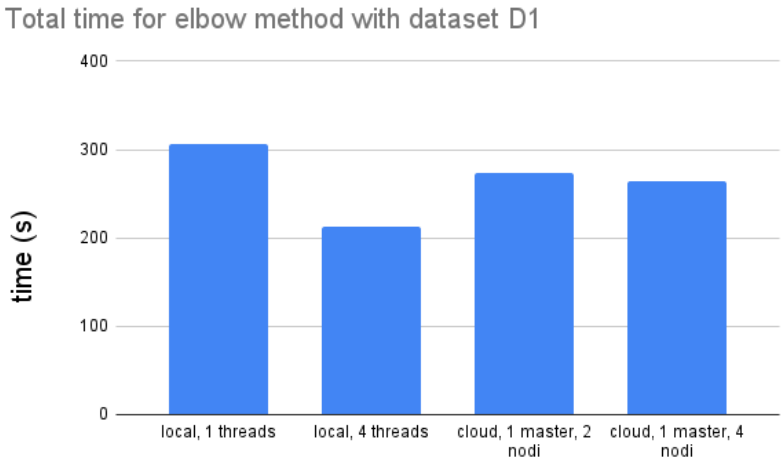

Figura 9: Totale tempi di esecuzione per k da 2 a 20 per D1

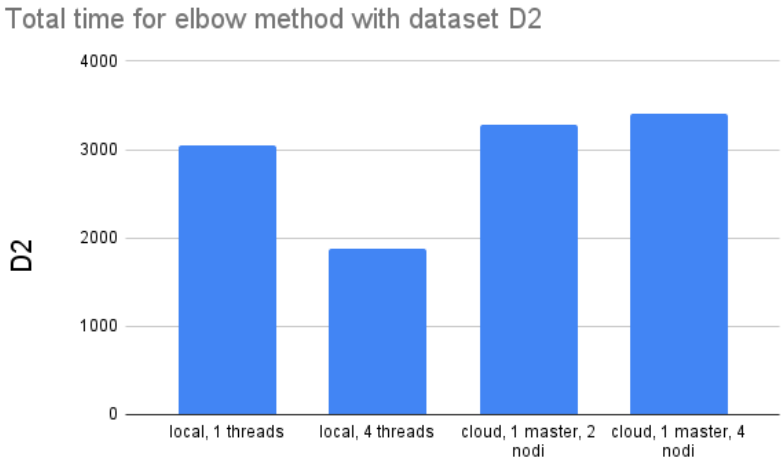

Figura 10: Totale tempi di esecuzione per k da 2 a 20 per D2

In figura 11 per D1 e figura 12 per D2 un focus sui tempi di esecuzione in base al numero di clusters, partendo dal presupposto che più il numero di clusters è alto e mediamente più tempo di calcolo è richiesto, in quanto ci saranno mediamente più iterazioni dell'algoritmo k-means per arrivare a convergenza.

Si può notare come per D1 i tempi di esecuzione nel cluster sono uguali per numero di nodi 2 e 4, mentre l'esecuzione locale con 4 threads ha il miglior risultato in quasi tutti i casi. L'esecuzione nei cluster non è mai la migliore.

Per D2 invece si può notare come da k=15 in poi i tempi peggiori riguardano l'esecuzione sui cluster. Anche in questo caso l'esecuzione locale con 4 threads è sempre la migliore.

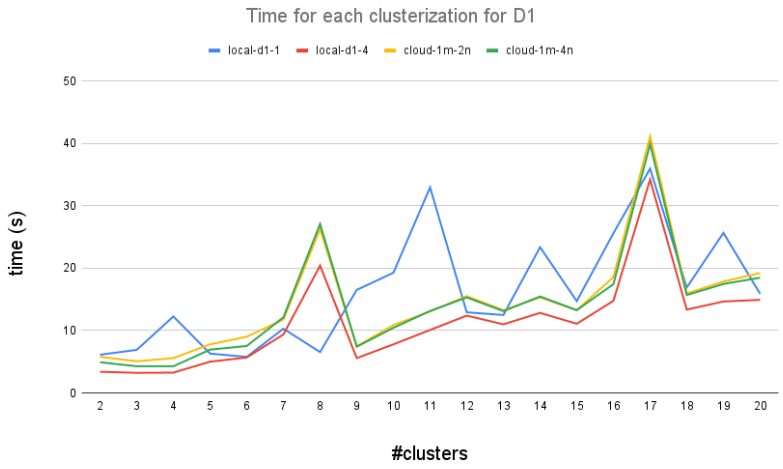

Figura 11: Tempi di esecuzione per k da 1 a 20 per D1

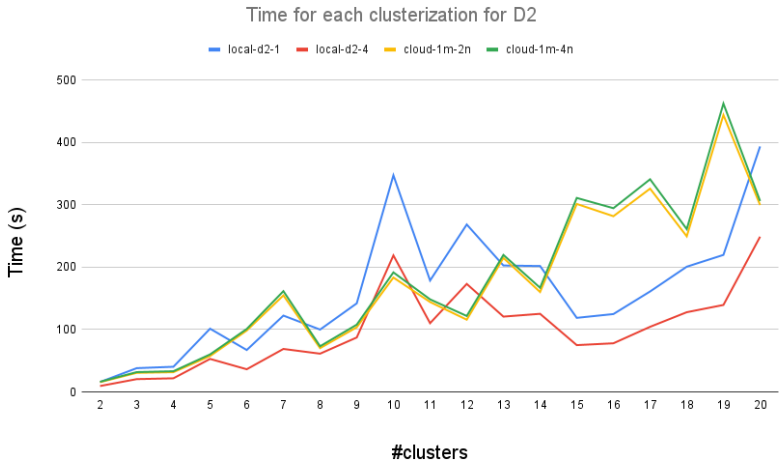

Figura 12: Tempi di esecuzione per k da 1 a 20 per D2

4 Conclusioni e sviluppi futuri

I risultati ottenuti per la clusterizzazione nei due datasets si discostano da quella che è la situazione ideale (numero di clusters individuati uguale a quello reale).

Infatti, per D1 il numero dei cluster reale è 5, mentre tramite l'algoritmo k-means implementato, con tecnica dell'elbow method con wcss è stato riscontrato che è 9, mentre con indice Calinski-Harabasz è 3 e con mean wcss è 8. Per quanto riguarda il dataset D2 invece, anche in questo caso si è avuta una discordanza tra i cluster reali, che sono 6, mentre quelli trovati tramite elbow method con i vari indici: 8 per wcss, 2 per Calisnki-Harabasz e 9 per la mean wcss. I risultati migliori quindi si sono ottenuti tramite l'indice wcss. I risultati ottenuti in questo studio si discostano poco dal benchmark effettuato tramite algoritmo k-means della libreria scikit-learn. Quest'ultimo infatti tramite tecnica dell'elbow method individua il numero di clusters ottimo per D1 a 8 e 6 per D2.

I risultati ottenuti suggeriscono quindi che con elbow method tramite indice wcss è possibile fare clusterizzazione approssimativa su dati relativi alla transport mode detection e human activity recognition provenienti da sensori di smartphone. Uno sviluppo futuro possibile potrebbe essere quello di applicare preliminarmente algoritmi per eliminare quei samples nel dataset che sono frutto del rumore dei sensori, ad esempio attraverso l'algoritmo DBSCAN, per escluderli cercando così di andare a migliorare poi risultati della clusterizzazione.

Per quanto riguarda i tempi di esecuzione, questi sono stati contro le aspettative. Infatti le prestazioni migliori sono state ottenute in locale. Su questo bisogna indagare sulla potenza delle macchine nel cloud rispetto alla risorsa di calcolo locale. Ulteriori test andrebbero fatti con cluster di dimensione più grande di quella presa in considerazione in questo studio. Inoltre, una conclusione potrebbe essere quella che i dataset non sono sufficientemente grandi da poter presupporre la diminuizione dei tempi di esecuzione su un cluster rispetto ad una esecuzione su macchina locale, tenendo in considerazione che per dataset piccoli non conviene una esecuzione su cluster, in quanto l'overhead aggiunto dalla latenza di rete porta l'esecuzione locale ad essere più veloce.