



UNIVERSITÀ
degli STUDI
di CATANIA

INTERNET OF THINGS BASED SMART SYSTEMS

a.a: 2019/2020

Raiti Mario O55000434

Nardo Gabriele Salvatore O55000430

Indice

- ▶ Intro
- ▶ K-Means Algorithm
- ▶ Implementazione in Python
- ▶ Sviluppi Futuri : Implementazione Parallela

Intro

La clusterizzazione dei dati e l'utilizzo nelle DNN

Clusterizzazione dei dati e utilizzo nelle DNN

- ▶ Una deep CNN in genere coinvolge molti livelli con milioni di parametri, rendendo l'archiviazione del modello di rete estremamente grande.
- ▶ Ciò impedisce l'utilizzo di deep CNN su hardware con risorse limitate. La semplice applicazione di clustering k-means ai pesi o lo svolgimento della quantizzazione del prodotto può portare a un ottimo equilibrio tra le dimensioni del modello e la precisione della recognition.
- ▶ Tramite ciò si può ottenere un alto grado di compressione della rete mantenendo il tasso di imprecisione molto basso.

K-Means Algorithm

Overview su K-Means

Overview su K-Means (1 / 2)

- ▶ L'algoritmo K-means è un algoritmo di clustering partizionale che permette di suddividere un insieme di elementi in K gruppi sulla base dei loro attributi. Si assume che gli attributi degli elementi possano essere rappresentati come vettori, e che quindi formino uno spazio vettoriale.
- ▶ L'obiettivo che l'algoritmo si prepone è di minimizzare la varianza totale intra-cluster. Ogni cluster viene identificato mediante un centroide o punto medio. L'algoritmo segue una procedura iterativa. Inizialmente crea K partizioni assegnandogli un centroide. Ogni partizione conterrà i punti più vicini al suo centroide. In seguito, ricalcola le coordinate di ogni centroide come punto medio fra i punti della sua partizione e ricostruisce le nuove partizioni assegnando i punti più vicini al nuovo centroide. Tale procedura avviene finché l'algoritmo non converge.

Overview su K-Means (2/2)


- ▶ Per questo approccio, è sufficiente archiviare gli indici dei cluster assegnati a punti e il codebook come parametri. Dati i k centri, abbiamo solo bisogno di $\log_2(k)$ bit per codificare il dataset.
- ▶ Pertanto, il fattore di compressione è $32/\log_2(k)$, supponendo di utilizzare dei float per rappresentare ogni punto del dataset e supponendo che il codebook stesso sia trascurabile.
- ▶ Nonostante la semplicità, questo approccio offre prestazioni sorprendentemente buone per la compressione dei parametri.

Implementazione

Implementazione proposta dell'algoritmo K-Means realizzata in



Discussione sull'elaborato

- ▶ L'elaborato è reperibile al seguente link :
https://github.com/RaiMar96/loT_project_2k19-20
- ▶ Scopo dell'elaborato è stato fornire un'implementazione approssimata e precisa dell'algoritmo K-Means.
- ▶ Il linguaggio scelto per l'implementazione è Python (Python3) 
- ▶ Il grado di approssimazione viene introdotto tramite i parametri forniti in ingresso allo script (`approx_factor` , `max_iterations`). Questo permette di implementare la tecnica 'loop perforation' di Approximate Computing.
- ▶ Una descrizione più dettagliata dell'implementazione è presente nel file README.md della directory linkata in alto.

Sviluppi Futuri : Implementazione Parallela

Discussione su soluzioni alternative per il miglioramento delle performance, parallelizzando l'esecuzione.

Soluzione Alternativa (1 / 2)

- ▶ K-means è utilizzato in molti contesti che necessitano clusterizzazione, grazie alla sua semplicità.
- ▶ L'incremento dei volumi di informazioni emergenti dal progresso tecnologico, rende il clustering di dataset molto ampi un processo impegnativo. Per far fronte al problema, si cerca di andare verso algoritmi di clustering parallelo efficienti.
- ▶ La versione parallela è implementata in base al parallelismo intrinseco durante le fasi di calcolo della distanza e di aggiornamento dei centroidi.
- ▶ Il risultato della parallelizzazione dell'algoritmo è che esso funziona relativamente bene su dataset di grandi dimensioni.

Soluzione Alternativa (2/2)

- ▶ In questa discussione ci focalizzeremo sulla realizzazione di PK-means (P = parallel) usando la tecnica di MapReduce.
- ▶ I risultati sperimentali su questa tecnica dimostrano che si riesce ad ottenere una computazione efficiente anche su hardware a basso costo e basse prestazioni e che tale soluzione scala bene su dataset di grandi dimensioni.
- ▶ Questa tecnica richiede l'utilizzo di piattaforme per il processing di grandi moli di dati che utilizzano tecniche di MapReduce, come Hadoop.
- ▶ L'implementazione del PK-means si basa sull'utilizzo di una funzione di mapping , una di combine e infine una funzione di reduce.

riferimenti:

<https://arxiv.org/ftp/arxiv/papers/1608/1608.06347.pdf>

https://sites.cs.ucsb.edu/~veronika/MAE/parallelkmeansmapreduce_zhao.pdf

PK-Means

- ▶ **Map-function:** Il set di dati di input (record) è archiviato su HDFS .Tale set è suddiviso e trasmesso globalmente a tutti i mappers. Di conseguenza, i calcoli delle distanze vengono eseguiti in parallelo. Per ogni map task, PKMeans costruisce un variant centers globale che è un array contenente le informazioni sui centroidi dei cluster. Data l'informazione, un mapper può calcolare il centroide più vicino per ciascun campione del dataset. I record intermedi sono quindi composti da due parti: una chiave k, cioè l'indice del centroide più vicino e le informazioni del campione del dataset.
- ▶ **Combine-function:** dopo ogni fase di mapping si realizza una combinazione dei risultati intermedi prodotti. Nella funzione di combinazione sommiamo parzialmente i valori dei punti assegnati allo stesso cluster (parzialmente poiché consideriamo solo i punti di quel particolare host). Per calcolare il valore medio dei campioni per ogni cluster, utilizziamo la chiave k; ciò ci garantisce di computare le somme parziali dei campioni relativi allo stesso cluster. L'output di questa fase è rappresentato da record composti da due valori: la stessa chiave k che identifica il cluster e un valore contenente sia le somme parziali dei punti dello stesso cluster, sia il numero di campioni della somma.
- ▶ **Reduce-function:** prende in input i dati ottenuti dalla funzione dalla combine function di ciascun host. In questo step, sommiamo tutti i campioni e calcoliamo il numero totale di campioni assegnati allo stesso cluster. Pertanto, possiamo ottenere le coordinate dei nuovi centroidi utilizzati per la successiva iterazione. L'output di questa funzione è rappresentato da due valori: la stessa chiave k che identifica il cluster e un valore che rappresenta il nuovo centroide.

riferimenti: https://sites.cs.ucsb.edu/~veronika/MAE/parallelkmeansmapreduce_zhao.pdf