Progetto K-Means

Hadoop

Per implementare l’algoritmo K-Means utilizzando Hadoop abbiamo definito le funzioni di Map e Reduce secondo il seguente pseudo-codice:

class Mapper

method Map(offset key, dataset\_line dl)

point P = parse\_point(dl)

distance dist <- MAX\_VALUE

for all centers C in cluster\_centers do

distance D <- distance(P, C)

if (D < dist) do

D <- dist

clusterIndex CI <- i

Emit(clusterIndex CI, point P)

class Reducer

method Reduce(clusterIndex CI, Points [point P (Point\_Sum, count), point P (Point\_Sum, count), point P (Point\_Sum, count) ....])

point Sum\_Coodinates ← 0

total ← 0

for all point P ∈ Points [....] do

Sum\_Coodinates ← Sum\_Coodinates + Point\_Sum

total ← total + count

point Avg ← Sum\_Coodinates/ total

Emit(clusterIndex CI, point Avg)

Una volta definite queste due funzioni è stato deciso di implementare anche una funzione Combiner per riassumere gli output del mapper con la stessa chiave prima di passarli al Reducer per ridurre la quantità dei dati e di conseguenza il tempo di trasferimento e tra il Mapper e il Reducer.

class Combiner

method Combine(clusterIndex CI, Points [point P, point P, point P ....])

point Point\_Sum ← 0

count ← 0

for all point P ∈ Points [....] do

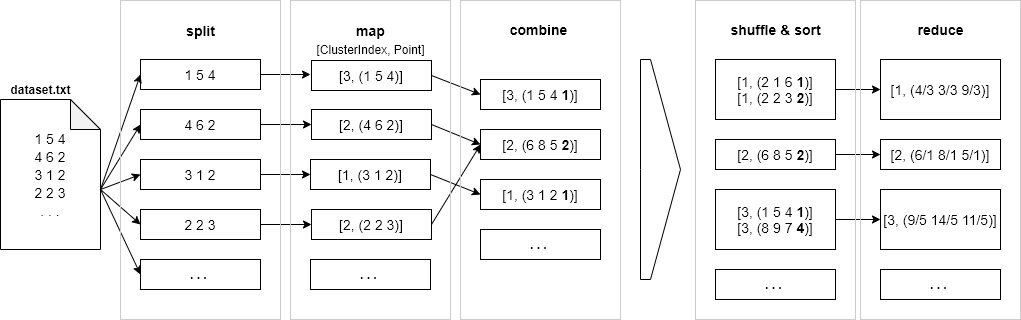
Point\_Sum ← Point\_Sum + P

count ← count + 1

point New\_Point ← [Point\_Sum, count]

Emit(clusterIndex CI, point New\_Point)

Le fasi che si susseguono ad ogni iterazione dell’algoritmo sono rappresentate nel seguente schema:



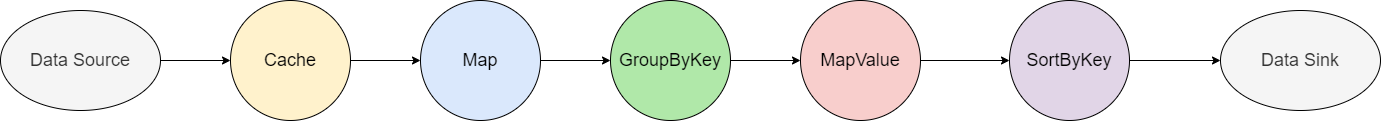
Una volta che il dataset, contenete le coordinate di vari punti, viene diviso tra i vari mapper, quest’ultimi si occuperanno di assegnare ad ogni punto l’indice del cluster più vicino considerando i cluster precedenti. Alla prima iterazione i cluster vengono estratti casualmente dal dataset. Una volta terminata questa fase entra in esecuzione il Combiner che si occuperà localmente di aggregare i punti appartenenti allo stesso cluster semplicemente effettuando la somma dei vari punti e tenendo traccia di quanti ne siano stati sommati. A questo punto i dati attraversano la fase di shuffle & sort durante la quale gli elementi con la stessa chiave vengono raggruppati in un'unica lista di punti che verranno assegnati ai vari Reducer. Infine, i Reducer si occuperanno di effettuare la somma di tutti i punti appetenti allo stesso cluster e a calcolare il nuovo centro tramite la media delle distanze dei vari punti. Tutti questi passaggi verranno eseguiti iterativamente aggiornando i vari centri dei cluster fino a raggiungere la condizione di stop o un numero massimo di iterazioni prestabilite.

Spark

Per l’implementazione dell’algoritmo K-Means tramite Spark per prima cosa si è reso necessario creare una classe Point per la gestione dei punti definiti nel dataset, nella quale, oltre ai normali metodi per l’inizializzazione degli attributi, è stato definito il metodo computeDistance() per calcolare la distanza euclidea tra due punti. Oltre a questa classe, sono stati implementati anche i seguenti metodi:

* file\_len(fname): dato il nome di un file ne ritorna la lunghezza.
* extractCenters(numberOfCenters, file, dimension): passando il numero dei centri, il file contenente il dataset e la dimensione dei punti presenti nel dataset, si occupa di estrarre casualmente dal dataset un serie di punti pari al valore numberOfCenters che verranno restituiti tramite una lista. La sua implementazione si è resa necessaria per l’inizializzazione iniziale dei centri.
* closestCentroid (line, dimension, centers): tramite i parametri line e dimension, questo metodo crea un nuovo punto per poi calcolare quale sia il centro più vicino ad esso. Quest’ultima operazione viene eseguita calcolando la distanza del punto da ogni centro presente nelle lista dei centri chiamata centers e passata come parametro.
* avg (pointList): si occupa di calcolare la distanza media di un insieme di punti che gli vengono passati tramite la lista pointList.

Una volta definite queste operazioni è stato possibile implementare il programma principale che prendendo come parametri iniziali il numero dei centri, la dimensione dei punti, il file contenente il dataset e il delta per la convergenza, per prima cosa estrae casualmente i centri iniziali tramite il metodo extractCenters() definito in precedenza. Successivamente procede alla creazione dello SparkContext e del primo RDD effettuando la lettura del file contente il dataset con le coordinate di ogni punto. Questo RDD viene salvato, tramite il metodo cache(), nella memoria RAM al fine di renderlo disponibile per le iterazioni successive.



Come si può vedere dal DAG, la successiva trasformazione che viene operata è tramite la funzione map() che a partire dall’RDD precedente per ogni set di coordinate si occupa di creare una coppia chiave-valore dove il valore è un punto e la chiave è l’indice del centro più vicino. Successivamente al fine di ottenere per ogni centro la lista dei punti vicini, viene operata una trasformazione tramite la funzione groupByKey(). A questo punto si è reso necessario calcolare la distanzia media tra i vari punti assegnati ad ogni centro per calcolarne quello nuovo, e per far questo è stata operata una trasformazione utilizzando la funzione mapValue() che tramite la funzione avg(), definita precedentemente, restituisce per ogni chiave le coordinate dei nuovi centri. Infine, viene eseguita una trasformazione di sortByKey() per riordinare il tutto.

A questo punto viene verificata la convergenza dei nuovi centri semplicemente facendo uso del metodo computeDistance() per calcolare la distanza con i centri estratti casualmente all’inizio. Se la distanza tra ogni centro dello stesso indice risulta inferiore al delta, passato come argomento del programma, significa che è stata raggiunta la convergenza e quindi, si può procedere a salvare le coordinate dei centri nel file di output. Se così non fosse, le operazioni fin qui descritte vengo eseguite di nuovo a partire dalla trasformazione eseguita tramite la funzione di map() su ogni set di coordinate precedentemente salvati in memoria, ma utilizzando i nuovi centri.

Paragonando la velocità di esecuzione dell’algoritmo kmeans sviluppato in hadoop e quello sviluppato in spark abbiamo notato che il secondo completa la sua esecuzione in un tempo visibilmente inferiore, questo dovrebbe essere dovuto al fatto che spark non deve ad ogni iterazione accedere nuovamente all’HDFS per ottenere il dataset ma ha direttamente in un RDD salvato in cache i dati che gli servono per compiere le proprie elaborazioni.

Fase di test

Abbiamo testato l’algoritmo nell’ambiente Spark e in quello di Hadoop utilizzando dataset creati ad hoc con vari cluster appositamente disposti per verificare se i centri calcolati rispettassero i cluster

Hadoop tests

n = 1000 d = 3 k = 7

Cluster: 1 Value Range Start: 0 Vale Range End 200

Cluster: 2 Value Range Start: 250 Vale Range End 450

Cluster: 3 Value Range Start: 500 Vale Range End 700

Cluster: 4 Value Range Start: 750 Vale Range End 950

Cluster: 5 Value Range Start: 1000 Vale Range End 1200

Cluster: 6 Value Range Start: 1250 Vale Range End 1450

Cluster: 7 Value Range Start: 1500 Vale Range End 1700

RESLUTS:

OUTPUT CENTER: 220.599 217.813 229.599

OUTPUT CENTER: 590.872 590.214 599.408

OUTPUT CENTER: 803.784 856.817 857.276

OUTPUT CENTER: 901.862 839.856 853.297

OUTPUT CENTER: 1099.614 1098.250 1098.103

OUTPUT CENTER: 1353.205 1349.617 1350.006

OUTPUT CENTER: 1594.230 1601.612 1608.838

n = 1000 d = 3 k = 13

Cluster: 1 Value Range Start: 0 Vale Range End 200

Cluster: 2 Value Range Start: 250 Vale Range End 450

Cluster: 3 Value Range Start: 500 Vale Range End 700

Cluster: 4 Value Range Start: 750 Vale Range End 950

Cluster: 5 Value Range Start: 1000 Vale Range End 1200

Cluster: 6 Value Range Start: 1250 Vale Range End 1450

Cluster: 7 Value Range Start: 1500 Vale Range End 1700

Cluster: 8 Value Range Start: 1750 Vale Range End 1950

Cluster: 9 Value Range Start: 2000 Vale Range End 2200

Cluster: 10 Value Range Start: 2250 Vale Range End 2450

Cluster: 11 Value Range Start: 2500 Vale Range End 2700

Cluster: 12 Value Range Start: 2750 Vale Range End 2950

Cluster: 13 Value Range Start: 3000 Vale Range End 3200

RESULTS:

OUTPUT CENTER: 95.70716666666665 108.00565384615388 102.36758974358979

OUTPUT CENTER: 468.72222068965516 460.73287586206914 463.652427586207

OUTPUT CENTER: 831.0898235294117 839.1276823529415 831.5286235294118

OUTPUT CENTER: 1076.4383448275862 1107.4235172413794 1153.5359655172413

OUTPUT CENTER: 1120.382157894737 1160.3818947368422 1061.5710526315788

OUTPUT CENTER: 1119.9980689655172 1048.656448275862 1052.2016206896549

OUTPUT CENTER: 1602.9214230769228 1598.8203589743594 1603.0518333333328

OUTPUT CENTER: 1301.9211489361699 1348.3457234042548 1350.977744680851

OUTPUT CENTER: 1397.7813333333334 1351.618 1325.8819333333336

OUTPUT CENTER: 1976.2530718954235 1977.6252745098047 1975.9225620915029

OUTPUT CENTER: 2465.62625 2467.288848684211 2467.9836447368425

OUTPUT CENTER: 3095.588986842106 3097.3975789473693 3107.675736842105

OUTPUT CENTER: 2846.118443037974 2848.902443037973 2855.8403544303806

n = 10000 d = 7 k = 7

Cluster: 1 Value Range Start: 0 Vale Range End 100

Cluster: 2 Value Range Start: 150 Vale Range End 250

Cluster: 3 Value Range Start: 300 Vale Range End 400

Cluster: 4 Value Range Start: 450 Vale Range End 550

Cluster: 5 Value Range Start: 600 Vale Range End 700

Cluster: 6 Value Range Start: 750 Vale Range End 850

Cluster: 7 Value Range Start: 900 Vale Range End 1000

RESULTS:

OUTPUT CENTER: 199.56555213435993 199.2997361791465 199.54258222533218 199.30360321903424 199.7395486354093 198.35386773967795 199.89951224632622

OUTPUT CENTER: 49.17160041987401 49.42639328201536 50.04221203638907 49.93988523442976 50.68723303009098 49.97071028691387 51.09196221133655

OUTPUT CENTER: 424.1969380685794 425.2683250524836 425.2353247025886 425.52092967109894 424.78445591322605 425.54508292512185 426.23694961511546

OUTPUT CENTER: 725.2056114805737 724.6358424921247 724.4322628631427 724.7260805040252 725.515073503676 725.1260910045497 724.5730665033251

OUTPUT CENTER: 954.3804872727267 947.3079381818187 947.094303636364 952.803043636363 951.237952727272 980.7523999999997 950.4491127272721

OUTPUT CENTER: 949.3225053078554 952.7557473460716 974.5547515923571 955.8072781316346 945.8114777070053 932.604244161359 956.7682356687895

OUTPUT CENTER: 945.7751674876852 947.9367660098519 924.0611428571436 942.1217487684734 952.832657635468 928.9028694581275 943.7515418719212

n = 100000 d = 7 k = 13

Cluster: 1 Value Range Start: 0 Vale Range End 300

Cluster: 2 Value Range Start: 350 Vale Range End 650

Cluster: 3 Value Range Start: 700 Vale Range End 1000

Cluster: 4 Value Range Start: 1050 Vale Range End 1350

Cluster: 5 Value Range Start: 1400 Vale Range End 1700

Cluster: 6 Value Range Start: 1750 Vale Range End 2050

Cluster: 7 Value Range Start: 2100 Vale Range End 2400

Cluster: 8 Value Range Start: 2450 Vale Range End 2750

Cluster: 9 Value Range Start: 2800 Vale Range End 3100

Cluster: 10 Value Range Start: 3150 Vale Range End 3450

Cluster: 11 Value Range Start: 3500 Vale Range End 3800

Cluster: 12 Value Range Start: 3850 Vale Range End 4150

Cluster: 13 Value Range Start: 4200 Vale Range End 4500

RESULTS:

OUTPUT CENTER:148.23544256978256, 153.45998765425875, 150.06715975348652, 152.86101234266842, 148.81115328673699, 147.31195375142062, 142.69776334712962

OUTPUT CENTER:493.12810236547896, 521.56835712698472, 514.36696385274143, 483.95324689648635, 493.09517532684566, 472.19745536598636, 519.07418529631472

OUTPUT CENTER:498.41158522118462, 465.84564894561542, 484.43758925649585, 546.22858792319719, 507.60216841018483, 544.95404019815965, 483.89406548648488

OUTPUT CENTER:782.28230216854165, 834.76065464989846, 846.43932123186084, 858.41905304198468, 875.61405498414894, 877.32405640984864, 867.31705419418866

OUTPUT CENTER:897.15635214098458, 859.80254164988454, 851.13154987484152, 844.09352848944356, 836.04423154984862, 832.97356498789985, 839.89252310940894

OUTPUT CENTER:1214.6852170780669, 1217.1380201640648, 1196.6715648456504, 1156.1293218798135, 1251.7915498784665, 1192.8350354810945, 1218.0762165498401

OUTPUT CENTER:1183.8891496721713, 1197.7161795195734, 1212.7841410570078, 1239.5042725075206, 1153.9952788220587, 1206.2545572272789, 1182.9137272727852

OUTPUT CENTER:1624.4858473849135, 1624.3449789573424, 1617.6269500773564, 1625.3645678565676, 1623.0773641123862, 1624.8957869048578, 1618.4258544657865

OUTPUT CENTER:2186.3838865450985, 2179.6552345737457, 2189.3735467245989, 2183.5934563456113, 2187.0346345678997, 2185.2381235634326, 2197.1176674566432

OUTPUT CENTER:3125.2448793048955, 3124.8940028179598, 3125.5432478568713, 3125.4432346734346, 3124.0264526741134, 3125.4087345621245, 3125.4596245647112

OUTPUT CENTER:3997.7467608126385, 3999.9653084561986, 3998.5871983250363, 4002.2477816587614, 4000.0191203650096, 3999.1613145876105, 3999.3023245266234

OUTPUT CENTER:3857.9421309845609, 3867.8443894560115, 3856.6541203478097, 3852.3567458698769, 3853.4833459612344, 3857.9943425662345, 3855.4692134256894

OUTPUT CENTER:4248.2943249850986, 4249.6042348576908, 4255.2464698659824, 4252.7982390845087, 4259.1022348756092, 4256.8234523485343, 4257.0837624855879

Spark Tests

n = 1000 d = 3 k = 7

Cluster: 1 Value Range Start: 0 Vale Range End 200

Cluster: 2 Value Range Start: 250 Vale Range End 450

Cluster: 3 Value Range Start: 500 Vale Range End 700

Cluster: 4 Value Range Start: 750 Vale Range End 950

Cluster: 5 Value Range Start: 1000 Vale Range End 1200

Cluster: 6 Value Range Start: 1250 Vale Range End 1450

Cluster: 7 Value Range Start: 1500 Vale Range End 1700

RESLUTS:

121.511, 120.084, 133.82

487.213, 484.702, 493.972

849.545, 846.429, 853.037

1099.615, 1098.251, 1098.103

1300.319, 1364.33, 1384.257

1379.927, 1342.184, 1332.7

1594.231, 1601.613, 1608.839

n = 1000 d = 3 k = 13

Cluster: 1 Value Range Start: 0 Vale Range End 200

Cluster: 2 Value Range Start: 250 Vale Range End 450

Cluster: 3 Value Range Start: 500 Vale Range End 700

Cluster: 4 Value Range Start: 750 Vale Range End 950

Cluster: 5 Value Range Start: 1000 Vale Range End 1200

Cluster: 6 Value Range Start: 1250 Vale Range End 1450

Cluster: 7 Value Range Start: 1500 Vale Range End 1700

Cluster: 8 Value Range Start: 1750 Vale Range End 1950

Cluster: 9 Value Range Start: 2000 Vale Range End 2200

Cluster: 10 Value Range Start: 2250 Vale Range End 2450

Cluster: 11 Value Range Start: 2500 Vale Range End 2700

Cluster: 12 Value Range Start: 2750 Vale Range End 2950

Cluster: 13 Value Range Start: 3000 Vale Range End 3200

RESULTS:

196.86, 203.383, 197.835

555.392, 543.493, 553.012

976.92, 979.337, 970.93

1464.432, 1468.903, 1465.311

1871.41, 1848.424, 1866.172

1791.655, 1830.257, 1790.789

2094.327, 2098.725, 2098.112

2340.677, 2348.402, 2349.783

2596.04, 2590.565, 2591.947

2836.722, 2782.862, 2826.74

2888.383, 2870.717, 2902.717

2803.735, 2903.02, 2826.239

3095.589, 3097.398, 3107.676

n = 10000 d = 7 k = 7

Cluster: 1 Value Range Start: 0 Vale Range End 100

Cluster: 2 Value Range Start: 150 Vale Range End 250

Cluster: 3 Value Range Start: 300 Vale Range End 400

Cluster: 4 Value Range Start: 450 Vale Range End 550

Cluster: 5 Value Range Start: 600 Vale Range End 700

Cluster: 6 Value Range Start: 750 Vale Range End 850

Cluster: 7 Value Range Start: 900 Vale Range End 1000

RESULTS:

49.172, 49.426, 50.042, 49.94, 50.687, 49.971, 51.092

199.566, 199.3, 199.543, 199.304, 199.74, 198.354, 199.9

348.9, 350.654, 349.949, 350.749, 348.338, 350.588, 350.39

499.494, 499.882, 500.521, 500.292, 501.231, 500.502, 502.083

649.995, 649.653, 649.117, 649.568, 650.958, 649.671, 649.152

800.499, 799.704, 799.832, 799.97, 800.138, 800.652, 800.072

950.338, 949.357, 949.678, 950.828, 949.992, 950.195, 950.71

n = 100000 d = 7 k = 13

Cluster: 1 Value Range Start: 0 Vale Range End 300

Cluster: 2 Value Range Start: 350 Vale Range End 650

Cluster: 3 Value Range Start: 700 Vale Range End 1000

Cluster: 4 Value Range Start: 1050 Vale Range End 1350

Cluster: 5 Value Range Start: 1400 Vale Range End 1700

Cluster: 6 Value Range Start: 1750 Vale Range End 2050

Cluster: 7 Value Range Start: 2100 Vale Range End 2400

Cluster: 8 Value Range Start: 2450 Vale Range End 2750

Cluster: 9 Value Range Start: 2800 Vale Range End 3100

Cluster: 10 Value Range Start: 3150 Vale Range End 3450

Cluster: 11 Value Range Start: 3500 Vale Range End 3800

Cluster: 12 Value Range Start: 3850 Vale Range End 4150

Cluster: 13 Value Range Start: 4200 Vale Range End 4500

RESULTS:

149.2, 151.459, 150.067, 151.861, 149.811, 149.311, 147.697

497.128, 526.562, 514.365, 462.953, 493.095, 462.197, 516.0

497.411, 465.84, 484.437, 546.228, 507.6, 544.954, 483.894

772.282, 834.76, 846.439, 858.419, 875.614, 877.324, 867.317

898.156, 859.802, 851.131, 844.093, 836.044, 832.973, 839.892

1217.685, 1207.138, 1186.67, 1156.129, 1251.791, 1192.835, 1218.076

1185.889, 1197.713, 1212.78, 1239.504, 1153.995, 1206.254, 1182.913

1724.42, 1725.497, 1725.626, 1726.366, 1725.062, 1724.895, 1726.425

2426.383, 2423.655, 2424.373, 2424.59, 2426.034, 2426.238, 2425.117

3125.244, 3124.894, 3125.54, 3125.443, 3124.02, 3125.408, 3125.459

3997.746, 3999.965, 3998.587, 4002.247, 4000.019, 3999.161, 3999.302

3649.942, 3649.844, 3649.654, 3649.356, 3649.483, 3649.994, 3651.469

4349.294, 4348.604, 4350.246, 4350.798, 4349.102, 4350.83, 4349.08