Hadoop版本之
K-Means Algorithm Implementation

中興資工碩一

7111056211 康智絜

Outline

- Algorithm
- Mapper做什麼?
- Reducer做什麼?
- 是否需要Setup Function?
- Pseudo Code

Algorithm

Do

Map

每讀取一筆data就和每個center的centroid做對比,
計算出k個center中和data距離最近的centroid,
再將這個centroid作為新的key,
該筆data作為value

Reduce

MapReduce會將相同key的value歸併在一起變成一個iterable,
再求出這個iterable中的data的平均值,
作為新的centroid

Until reduce求出的新的平均值和原先的centroid相同

Mapper做什麼?

- 題目令k = 3，所以會將dataset分成3個群，並隨機設定3個centroid
- 分別為
  - centroid 1
  - centroid 2
  - centroid 3
- 利用Euclidean distance來計算點和點之間的距離
- 分別計算每個point離哪個centroid距離最近
- 將距離該point最近的centroid設為key，該point設為value

Key	Value
[centroid1,	point1]
[centroid2,	point3]
[centroid3,	point7]
[centroid2,	point6]
.....	
[centroid1,	point12]

Reducer做什麼?

- 利用reducer的功能，將相同key的point蒐集在一起
- 計算出各個key的points的平均值，當作新的centroid
- 將舊的centroid設為key，新的centroid設為value

Key	Values
[centroid1,	(point1+point12+...)/size]
[centroid2,	(point3+point6+...)/size]
[centroid3,	(point7+point15+...)/size]

Key	Value
[centroid1,	centroid1']
[centroid2,	centroid2']
[centroid3,	centroid3']

是否需要Setup Function?

- Setup Function：task一開始時執行一次
- 程式執行一開始時需要讀取centroid以及有幾個center，以該題為例，center為k = 3

Pseudo Code - Setup

```
void setup(Context context){
    k <- center.size()  // k為center的個數
    initially randomly set k cluster centers
}
```

Pseudo Code - Map

```
void Map(LongWritable key, Text value, Context context){
    for i ( 1 to 3 ) do  // 題目假定k = 3，為3個center

        /* 計算地區每小時的PM2.5和centroid的Euclidean distance */
        for j ( 0 to sizeOfDataField - 1 ) do // sizeOfDataField為一筆data有幾個field
                                              // 在PM2.5的例子中一個地區有24個data field
            dis += centroid – field  // 計算data field和centroid的距離
        end for

        /* 找出距離最小的centroid */
        if dis < min then
            min = dis
            index = i  // i是第i個center
        end if

    end for

    context.write( i, value) // key為哪個center，value為PM2.5的data
}
```

Pseudo Code - Reduce

```
void Reduce(IntWritable key, Iterable<Text> value, Context context){

    /* 讀取相同key的所有不同地區的PM2.5的值 */
    Iterator it <- value.iterator()
    while it.hasNext() do              // 如果value還有值就將值加入fieldList中
         fieldList.add( it.next() )
    end while

    /* 計算新的centroid */
    for i ( 1 to 3 ) do
         sum <- 0

         for j ( 1 to fieldSize ) do
             sum += fieldList.get(j).get(i)
         end for

         average[i] <- sum / filedList.size()
    end for

    /*舊的centroid設為key，新的centroid設為value */
    context.write( centroid_old , average)
}
```