# elk-KMeans算法及其Spark实现

## 背景

## 产生场景

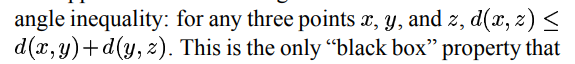
在进行k均值的过程中，特别是数据量n>>k时，每次迭代的过程中对应的距离中心往往变动很小，每次迭代的过程中有大量的中心点的距离是不需要计算的。Elk三角不等式将通过一些三角不等式法则节省一些不必要的欧几里得距离下的运算。

## K均值的基本概念和本文的符号约定

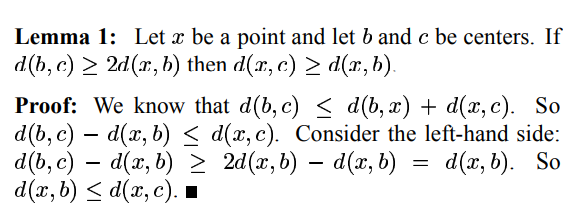
c(x) = c

## 三角不等式规则即应用

### 基本三角不等式



### 引理1



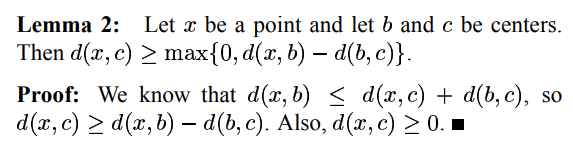
### 引理1在k均值中的应用

If d(x, c(x)) < 1/2\*d(c(x), c’) 任意c’ then x 必然依旧属于c(x)，不需要再计算

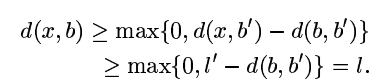
也等价于

If d(x, c(x)) < 1/2\*Min d(c(x), c’) base on c’ then x 必然依旧属于c(x)，不需要再计算

### 引理2



### 引理2在k均值中的应用



所以，d(x, b)的下界 >= max{0, 上次迭代的d(x,b)的下界 – d(b, b’)}

### 引理1和2产生的约简规则总结

规则1：If d(x, c(x)) < 1/2\*Min d(c(x), c’) base on c’ then x 必然依旧属于c(x)，不需要再计算

规则2：任意c’，任意x，l(x, c’)为d(x, c’)的下界，若l(x, c’)>= d(x, c(x))则x 必然依旧属于c(x)，不需要再计算

规则3：任意c’，任意x，l(x, c’)为d(x, c’)的下界，则l’(x, c’)为上次迭代的下界，c’’为c’上次对应的中心点。这里两次迭代的中心点的对应时指点更新后c’’对应的点取均值为c’，由于更新中点奔去的方向是多向的，因此c’和c’’谈不上是真正的一一对应，只是迭代产生的效果，某种程度上是我们为了使用三角不等式而撮合的。

则l(x, c’)>= max{0, l’(x, c’) – d(c, c’’)}

注意：这里所说的下界和上届，只是为了节省计算的情况下，在没有计算该距离时对该距离的估计的上下界，一旦该距离计算了，那么上届 = 下界 = 该距离。

## elk-KMeans算法

step1初始化:

初始化c

任意x，任意c，l(x, c) = 0

任意c(x) = argmin{d(x, c)} base on c

u(x) = d(x, c(x))

r(x) = true，记录x的边界是否更新过

step2迭代直至收敛条件

While (i <= maxIterations or converge){

step2-1计算任意两中心之间的距离d(c, c’)，并计算d(c, c’)的最小值的二分之一赋值为sc

step2-2 if(u(x) >= sc) continue to step2-3 否则跳至step2-4

step2-3

for(x <- X){

for(c <- C if c != c(x)){

if(u(x) > l(x, c)) && u(x) > 1/2\*d(c, c(x)){

if(r(x)){

计算d(x, c(x))

u(x) = d(x, c(x))

l(x, c(x)) = d(x, c(x))

r(x) = false

}else{

d(x, c(x)) = u(x)

}

If(d(x, c(x)) > l(x, c) || d(x, c(x)) > 1/2\*d(c, c(x))){

计算d(x, c)

l(x, c) = d(x, c)

If(d(x, c) < d(x, c(x))){

c(x) = c

}

}

}

}

}

step2-4更新中心和上下界以及r(x)

对同一个中心c对应的点求均值m(c)

更新l(x, c) = max{0, l’(x, c’) – d(c, m(c))}

更新u(x) = u(x) + d(m(c(x)), c(x))

r(x) = true

赋值c = c(x)

}

## elk-KMeans算法的本地实现

## elk-KMeans分布式算法

## elk-KMeans算法的分布式实现

## 小结

## 收获和参考文献