

Paralel KMeans, Hadoop

K-Means algoritmasını nasıl paralel şekilde işletiriz? Özellikle Hadoop gibi bir Esle-İndirge (Map-Reduce) ortamını düşünelim. Veri çok büyük ölçekte olabilir ve bu veriler birden fazla makineye bölünecektir. Esle-İndirge kavramında esleme safhasında “anahtar üretiriz”, ve sonra indirgeme safhasında Hadoop sistemi böyle kurmuştur ki aynı anahtarlar tek bir makineye gönderilir, ve bu nihai aşamada artık anahtar bazında indirgeme (özetleme) yapılır.

Paralel K-Means için anahtar nedir? Anahtar, mesela küme olabilir. Yani küme 1, küme 2 gibi küme işaretleri / sayıları anahtar olarak kullanılabilirler.

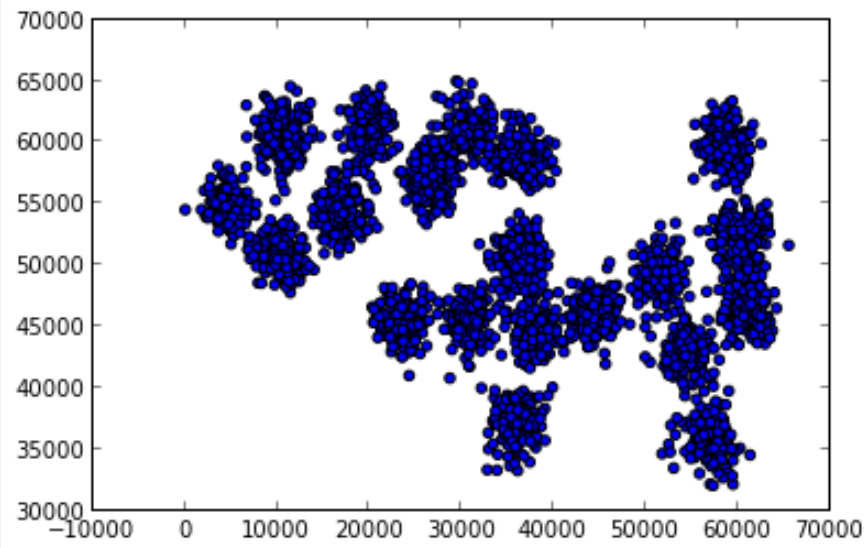
Peki anahtar ile eslenecek “değer” nedir?

Öyle bir değer arıyoruz ki üst üste konulabilecek bir şey olmalı, EI sisteminin kuvveti burada, anahtarlar farklı noktalarda üretilebiliyor, sonra tek noktada üst üste konuyor, o zaman değerler böyle üretilmeli ki bu üst üste koyma, özetleme işlemi yapılabilir.

Üst üste konabilecek şey kümeye (anahtar) ait olan veri noktası olabilir, yani basbасыagi veri noktasının kendisi değer olabilir. Normal K-Means’i hatırlarsak, her nokta için o noktaya en yakın kümeyi buluyordu ve sonra, atama işlemi bitince, her kümenin altındaki noktaların toparlayıp, onların ortalamasını alarak yeni küme merkezini hesaplıyordu. Bu ortalama işlemi üst üste konabilecek bir şey, çünkü toplama böyle bir işlem, ve / yani farklı makinalarda küme-nokta, eslemelerini üretirsek, indirgeme aşamasında o anahtar için tüm değerleri toplayıp, nokta sayısına böleriz ve yeni küme merkezini elde ederiz.

```
figure(figsize=(9,9))
im=imread('kmeans-diag.png'); imshow(im)
```

```
<matplotlib.image.AxesImage at 0x2b25d90>
```

```
print open("kmeans.py").read()
```

```
from mrjob.job import MRJob
from mrjob.protocol import PickleProtocol
import numpy as np, sys
import pandas as pd
import os, random

def euc_to_clusters(x,y):
    return np.sqrt(np.sum((x-y)**2, axis=1))

class MRKMeans(MRJob):
    INTERNAL_PROTOCOL = PickleProtocol

    def __init__(self, *args, **kwargs):
        super(MRKMeans, self).__init__(*args, **kwargs)
        self.centers_ = pd.read_csv("/tmp/centers.csv",header=None,sep=" ")
        self.k = 15

    def mapper(self, key, line):
        point = np.array(map(np.float,line.split(' ')))
        c = np.argmin(euc_to_clusters(np.array(self.centers_), point))
        yield(c, point)

    def reducer(self, key, tokens):
        new_centers = np.zeros((1,2))
        counts = 0
        for val in tokens:
            new_centers += val
            counts += 1
        yield('final', (key, new_centers[0] / counts))
```

```

def reduce_all_centers(self, key, values):
    new_centers = np.zeros((self.k,2))
    self.f=open("/tmp/centers.csv","w")
    for (cluster,val) in values:
        print cluster, val
        new_centers[cluster] = val
    for row in new_centers:
        self.f.write(" ".join(map(str,row)))
        self.f.write("\n")
    self.f.close()

def steps(self):
    return [self.mr(mapper=self.mapper,reducer=self.reducer),
            self.mr(reducer=self.reduce_all_centers)]

if __name__ == '__main__':
    for i in range(15): MRKMeans.run()

```

reduce_all_centers cagrisi tum indirgeyiciler her kume icin yeni orta noktayi hesaplayip onu yayinladiktan (emit) sonra, tum yeni merkezlerin gelecegi yer.

Komut satirindan tek makina icin Hadoop'suz isletelim,

```

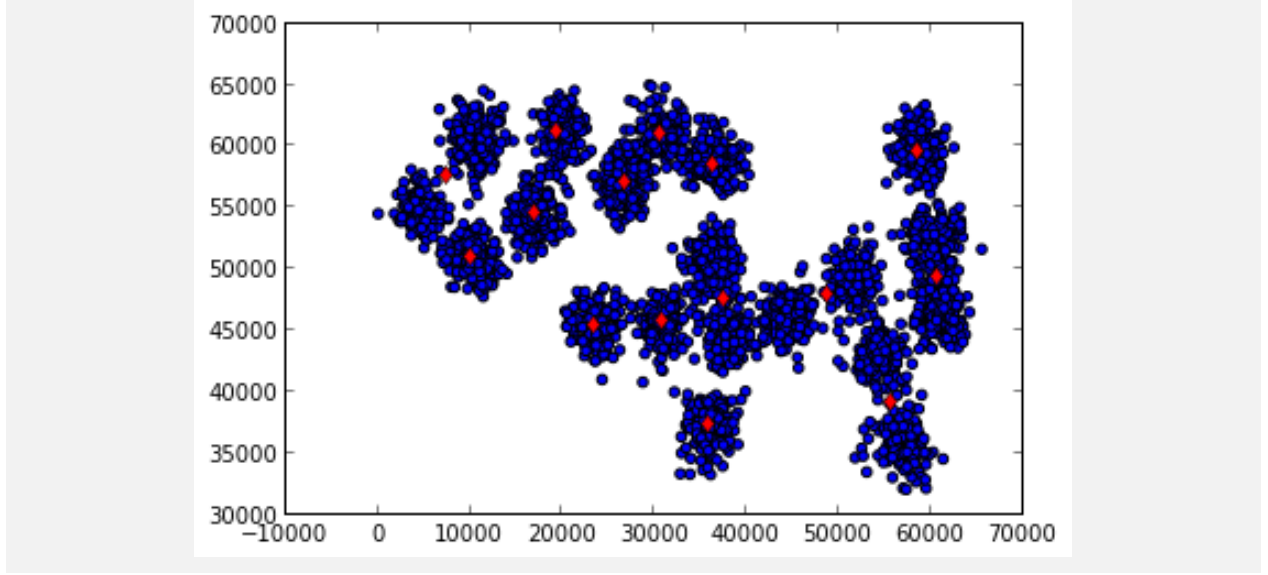
!sort --random-sort synthetic.txt > /tmp/synthetic.txt
!head -15 /tmp/synthetic.txt > /tmp/centers.csv
!python kmeans.py synthetic.txt

```

```

import pandas as pd
df1 = pd.read_csv("synthetic.txt",sep=" ",header=None)
plt.scatter(df1.ix[:,0],df1.ix[:,1])
plt.hold(True)
df2 = pd.read_csv("/tmp/centers.csv", sep=" ", header=None)
plt.plot(df2.ix[:,0],df2.ix[:,1], 'rd')
plt.show()

```



K-Means'i 20 kere islettik. Eger istenirse (hatta daha iyi olur) dongu bir while icine konur ve bitis icin “stabilite sarti” aranir. Stabilite yeni kume merkezinin eskisinden “cok fazla degisik olup olmadigi” sartidir, degisim yoksa artik sonucu bulmusuz demektir, daha fazla donguye gerek kalmayacaktır. Biz donguyu 20 kere donguyu islettik, (bu problem icin) yeterli oldu.

K-Means isini bitirdikten sonra elde edilen sonuclari okuyabiliriz. Nihai kume merkezleri /tmp/-centers.csv icinde. Bu merkezleri alip, ham veri uzerinde kirmizi nokta olarak gosteriyoruz.

Sonuclar fena degil. Iste bu metotla terabayt olceginde, devasa bir veriyi 20-30 makinaya dagitarak parca parca isleyip kumelemeniz mumkundur. Endustride son zamanlarda habire duyulan Buyuk Veri (Big Data) olayi iste bu.