K Means Cluster Algorithm

I have implemented 3 initialization method, namely random initialization, using gonzales' algorithm, and k means++. Using maximal iteration and predefined convergence threshold to control for stopping criterion. Also a similarity function was implemented for evaluation.

Firstly I will introduce three initialization methods.

Then I will design some experiments to see the properties of each method, i.e. stabilization, convergence speed and so on.

---

1. Initialize the cluster center:

input:

k: the number of clusters

indices: pixel value. For example: I(x,y)

indices_length: length*width. For example: for a image 200*200, indices_length = 400

output:

centers: centers[index]

center_length = k

random_shuffle(indices_length)
for(index:k)
get the elements of random_shuffle sequently rnd
centers[index] = indices[rnd]

Attention: avoiding choosing two indices with same pixels

2. Intialize centers in the k-means using gonzales' algorithm

input:

k: the number of clusters

indices: pixel value. For example: I(x,y)

indices_length: length*width. For example: for a image 200*200, indices_length = 400

output:

centers: centers[index]

center_length = k

1. Choose the first cluster center randomly.
2. for (index : k)
    for (j : indices_length)
        calculate the minimum length(difference of RGB/Gray Value) among all the centers and pixels in the image, so that the centers we choose are spaced apart from each other

3. Initialize centers in the k_means using kmeans++

input:

k: the number of clusters

indices: pixel value. For example: I(x,y)

indices_length: length*width. For example: for a image 200*200, indices_length = 400

output:

centers: centers[index]

center_length = k

1. Choose the first cluster center randomly. Centers[0] = indices[index] which index = rand_int(n)
2. calculate the for each point i closestDistSq[i] = d(center[0],indices[i]), and the sum of distance
3. for (i:k)

choose a number randomly from 0 to sum

find a suitable candidate for next cluster center w.r.t the random number we choosed above

computer the new potential, ensure that the minimal distance with all cluster centers will be maximized!!

(example candidate(c), d(c,1) = 100, d(c,2) = 100, d(c,3) = 1, even though the first 2d is big, but it is near to the third center! So that it is not a good candidate!)

update the new closetDistSq[i] for each point.

Now I will do some test of this algorithm.

1. For three initialization method, let iteration=1,cluster=2 (most simple case) to see the cluster result. The result of method 1 would have big differences.
2. For three initialization method, observe the difference convergence speed.
3. See the cluster result of different images. Choose different k.

---

1. Iteration = 1;
   cluster = 2;
   Run 5 times.

Method 1:

Method 2:

Method 3:

It seems that method 2 is more stable than the other two methods. Then here comes the question, which is stable between method 1 and 3. Let's run more cases and have a look.

Method 1:

Method 3:

As we can see, Method 3 has more better clustering results. So method 3 is more stable than 1 in this case. It is the same as the analysis. In addition, it use the probability to choose the initialized point, so it is also not 'hard' like method 2. It means, it has also some instable cases.

---

2. k =10;
   iteration = 10;

Method 1:

```
The convergence rate is: 1312.69
The convergence rate is: 9.35873
The convergence rate is: 0.572199
The convergence rate is: 0.045158
The convergence rate is: 0.00371609
The convergence rate is: 0.000308584
The number of iteration is: 5 until it convergence
Kmeans Using self implementation costs: 901 milliseconds
  The similarity is: 0.976126
```

```
The convergence rate is: 1496.41
The convergence rate is: 1.48797
The convergence rate is: 0.00248104
The convergence rate is: 5.13972e-06
The number of iteration is: 3 until it convergence
Kmeans Using self implementation costs: 648 milliseconds
The similarity is: 0.976358
```

```
The convergence rate is: 1347.58
The convergence rate is: 40.5834
The convergence rate is: 8.12188
The convergence rate is: 1.76874
The convergence rate is: 0.39855
The convergence rate is: 0.0923127
The convergence rate is: 0.0218432
The convergence rate is: 0.00525127
The convergence rate is: 0.00127695
The convergence rate is: 0.000313017
The number of iteration is: 9 until it convergence
Kmeans Using self implementation costs: 1449 milliseconds
The similarity is: 0.954845
```

Method 2:

```
The convergence rate is: 1537.1
The convergence rate is: 341.866
The convergence rate is: 334.584
The convergence rate is: 331.244
The convergence rate is: 327.965
The convergence rate is: 324.717
The convergence rate is: 321.502
The convergence rate is: 318.319
The convergence rate is: 315.168
The convergence rate is: 312.047
It did not convergence if the number of iteration is: 10
Kmeans Using self implementation costs: 1831 milliseconds
The similarity is: 0.978117
```

It did not convergence in 10 iterations.
Now change iteration to 10000 and see.

```
The number of iteration is: 1281 until it convergence
Kmeans Using self implementation costs: 168597 milliseconds
The similarity is: 0.978117
```

Even though method 2 is well performed but convergence is very slow compared with method1 and method 3. Let’s see the next.

Method 3:

```
The convergence rate is: 1944.65
The convergence rate is: 0.640126
The convergence rate is: 0.000822383
The number of iteration is: 2 until it convergence
Kmeans Using self implementation costs: 746 milliseconds
The similarity is: 0.973968
```

```
The convergence rate is: 2242.39
The convergence rate is: 0.0830039
The convergence rate is: 1.29788e-05
The number of iteration is: 2 until it convergence
Kmeans Using self implementation costs: 729 milliseconds
The similarity is: 0.947206
```

```
The convergence rate is: 2168.84
The convergence rate is: 0.160515
The convergence rate is: 3.9772e-05
The number of iteration is: 2 until it convergence
Kmeans Using self implementation costs: 730 milliseconds
The similarity is: 0.965373
```

---

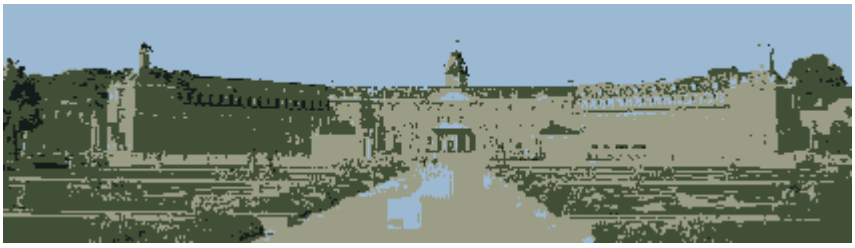
3. Use method 3 to see cluster results from different k.

Original image

k = 2

k=3

k=4



k=5

k=10

k =100

Original image

k = 10;

Conclusion:

1. Initialization plays an important roll in the k means algorithm.
2. Three methods can segment image with good performance.
    - Method 1 and 3 convergent faster than method 2.
    - Method 2 and method 3 performs better than method 1. It means more stable.