COMS 4701 Artificial Intelligence Sec 01

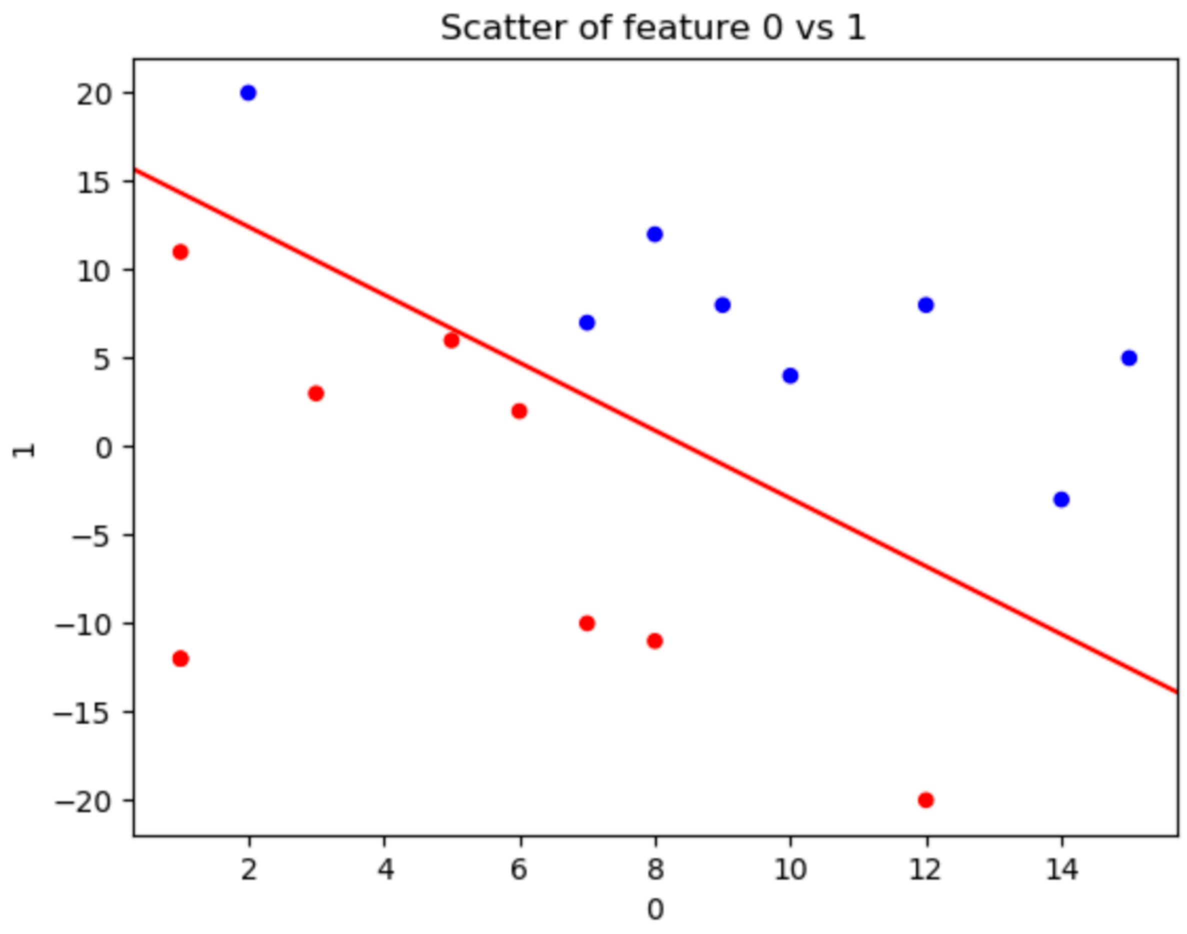
HW 3 readme

Name:	Yu Wang
Uni:	yw3025
Date:	April. 15, 2018

Question 1: Perceptron

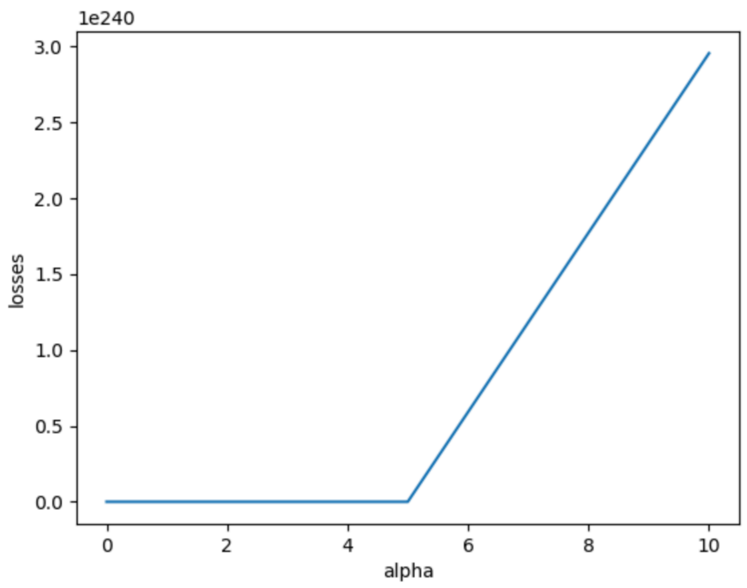
Final weights: weight1 = -5.0 weight2 = -2.0   b = 39.0

Decision boundary:



Question 2: Linear Regression

Firstly, I get the loss (risk) for each of the α(first nine) and plot as follows:



The loss is as follows:

α	loss
0.001	0.50066568900492847
0.005	0.22505856097398774
0.01	0.083755584470633596
0.05	0.0024586753174161064
0.1	0.002368882181992573
0.5	0.0023640503440444274
1	0.0023640503440440336
5	5.9782421818154526e+174
10	2.9566947121550958e+240

So, we can see that the optimal α lies roughly between [0.05, 2]

Then, I do a Grid search(evenly spaced) over parameter α and iteration(loop count) and find the optimal solution:

α → [0.05, 2]

Loop count → [50, 1000]

For α I iterate over evenly spaced 20 value, for loop count, I iterate over evenly spaced 100 value and find the parameter combination with the lowest loss. The code snippet is as follows:

```
# grid search to find the best parameter
best_alpha = None
best_iter = None
best_out = None
best_loss = float("inf")
for alpha in np.linspace(0.05, 2, 20):
    for max_iter in np.linspace(50, 1001, 100):
        max_iter = int(max_iter)
        w, loss = gradient_descent(X, y, alpha, max_iter)
        if loss < best_loss:
            best_iter = max_iter
            best_alpha = alpha
            best_out = [alpha, max_iter, w[0], w[1], w[2]]
            best_loss = loss

result.append(best_out)
result_out = pd.DataFrame(result)
result_out.to_csv(out_filename, header=None, index=None)
```

The optimal parameter I find is as follows (last line in csv file):

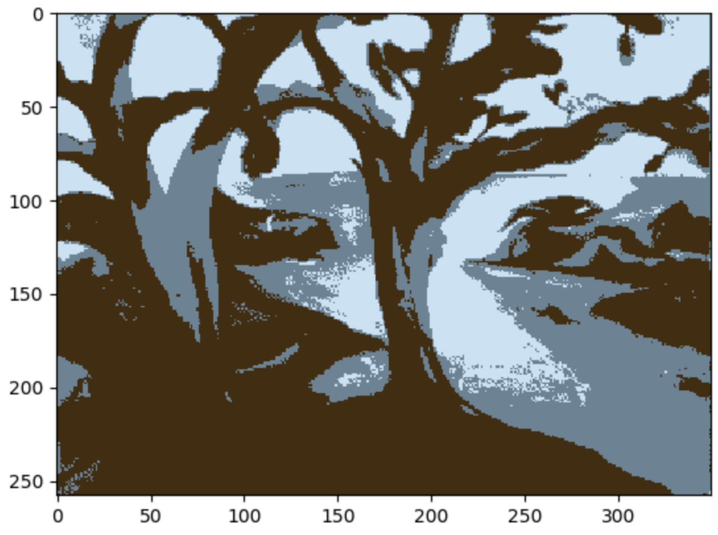
```
0.25526315789473686,386,1.0964608113924048,0.12861723232328506,0.0014024838966507712
```

Question 3: Clustering

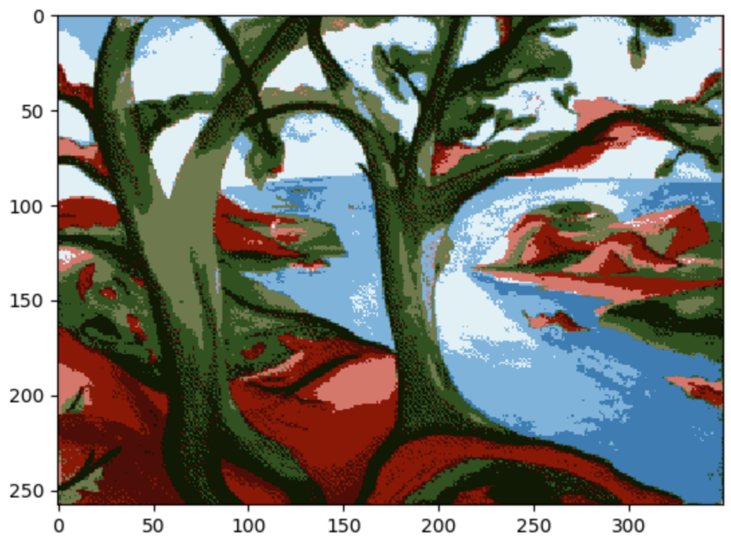
1. K-means

I choose K equals to 3, 9, 20

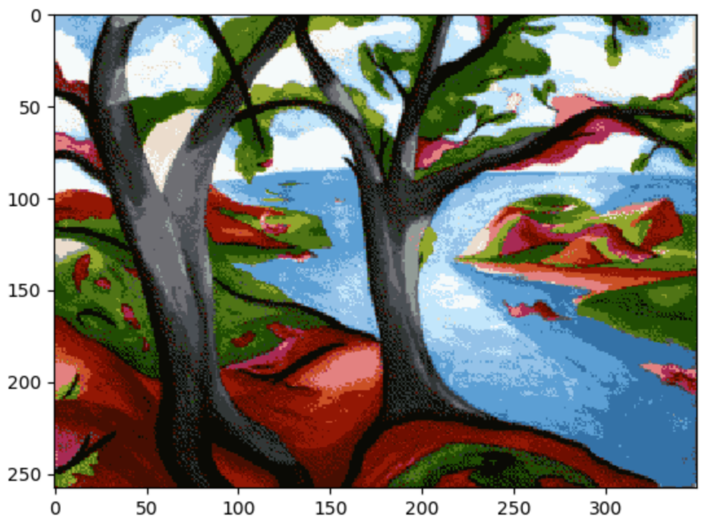
Result k = 3:



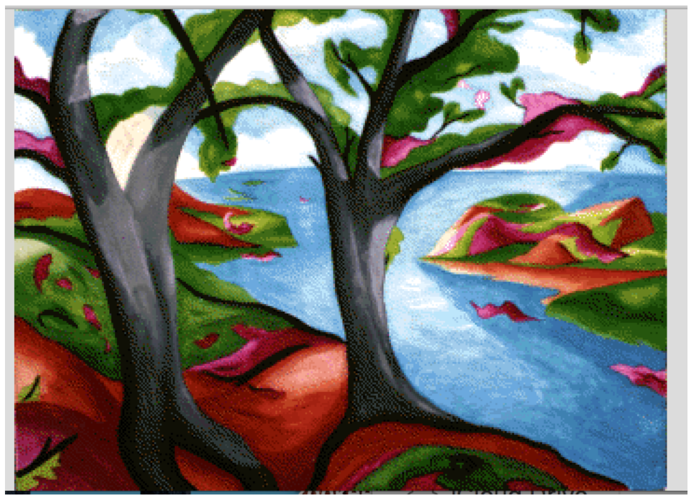
Result k = 9:



Result k = 20



Original:



Note:

As I increase the k value, the segmentation image looks more like the original image. This is because more similar pixels are group together under the same value and decrease the gradient across different group of pixels (I mean the margin looks more `soft`).

In essence, for a 350 × 258 image, if we assume all pixel rgb values are not the same, 350 * 258 clusters will in theory fully construct the original image(as each pixel has their original value).

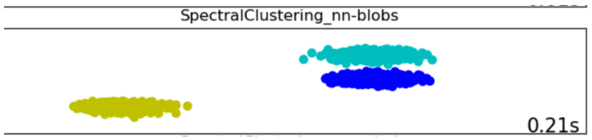
2. Bonus

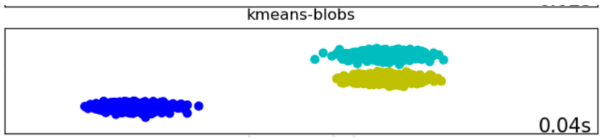
I choose the toy dataset (small dataset generated by numpy) since spectral clustering is extremely slower on an RBG image.

I choose to generate 7 toy dataset and compare it among k-means and spectral clustering.(Reference: starter code from sklearn sample code for clustering)

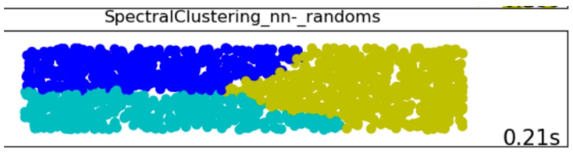
Note: Sklearn sample code provide a blobs, moons, circle data for us, I generate circle dot and spiral for more comparison.

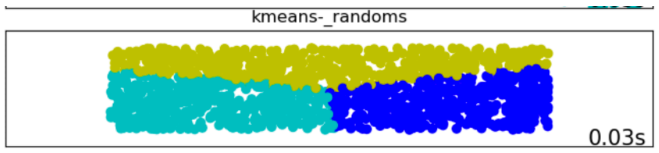
1) Blobs



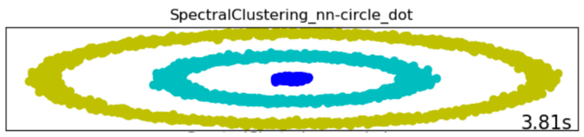


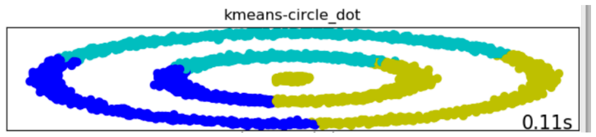
2) Random



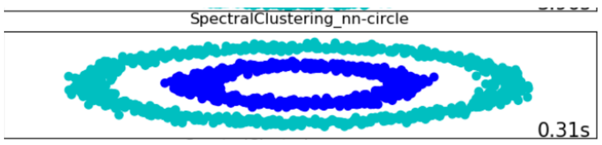


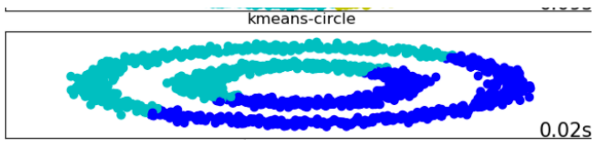
3) Circle dot



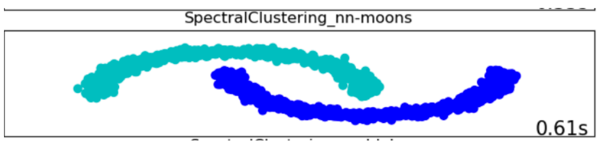


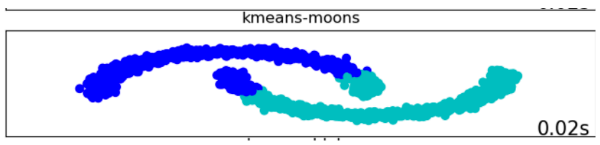
4) Circle



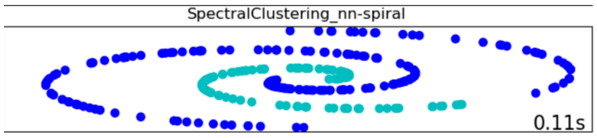


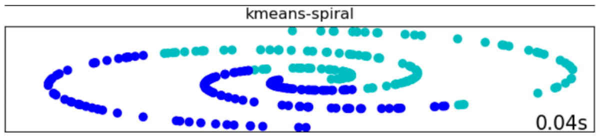
5) Moons



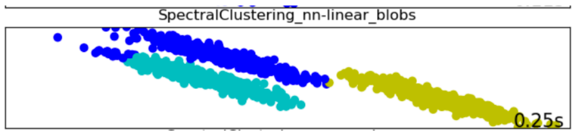


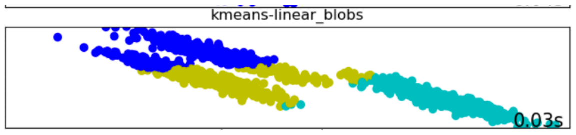
6) Spiral





7) Linear blobs





Note that

1. For dataset 1 and 2, both spectral clustering and K-means performs well. In dataset 1 we have circular shape clusters. In dataset 2, there is no clear clusters (serve as random baseline).
2. For dataset 3,4,5,6,7. When there is non-globular shape, spectral cluster perform better than k-means. K-means fail to effectively cluster circle even when the true number of clusters K is known to the algorithm.
3. For K-means, which compare the distance between two point using features(2nd norm feature vector difference), aimed for circular clusters like the glob where all members in the cluster are close to each other in Euclidean distance
4. For Spectral cluster, we cluster data not by their original space but by the similarity matrix. The matrix is generate as an N by N matrix(where N is the sample number), the (i, j)th entry is the similarity defined by your affinity function(which can be nearest neighbour or rbf kernal) which is calculate in the transform space.
5. So spectral clustering is more general which aim to capture the neighbouring information(similarity) in different transformed space. The k-means can be seen as when spectral clustering use a Euclidean measure as similarity. So, spectral clustering performs better in my sample(where I use Nearest neighbour as similarity matrix).