Kernelized Kmeans

November 4, 2018

1 Kmeans

Given N unlabeled examples and a number of desired partitions K, the goal in kmeans is to group the examples into K homogeneous partitions The most common algorithm uses an iterative refinement technique, given C an initial set of k centroids, the algorithm proceeds by alternating between two steps: - Assignment step: Assign each observation to the cluster whose mean has the least distance, this is intuitively the nearest centroid - Update step: Calculate the new means to be the centroids of the observations in the new clusters.

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.cluster import SpectralClustering

        fig = 1

        def kmeans(X, K, maxIters = 10, plot_progress = None):
                global fig
                size = len(X)
                centroids_indexes = np.random.choice(np.arange(size), K)
                centroids = X[centroids_indexes,:]

                for i in range(maxIters):
                        # cluster Assignment step
                        C = np.array([np.argmin([np.dot(x_i-y_k, x_i-y_k) for y_k in centroids]
                        # Move centroids
                        centroids = [X[C == k].mean(axis = 0) for k in range(K)]
                        if plot_progress:
                                fig += 1
                                plt.figure(fig, figsize=(8,6))
                                plt.clf()
                                plt.scatter(X[:,0], X[:,1], c=C, s=25, edgecolor='k')
                                plt.show()
                return np.array(centroids) , C
```

1.0.1 Kernel Kmeans

Kmeans is kernelized by replace the distance/similarity computations between an example x_n and a centroid $_k$ by the kernelized versions. for example:

$d(x_n, {}_k) = ||(x_n)({}_k)||$
replaced by:
$||(x_n) - ({}_k)||^2 = ||(x_n)||^2 + ||({}_k)||^2 - 2(x_n)^T({}_k) = K(x_n, x_n) + K({}_k, {}_k) - 2K(x_n, {}_k)$
where $K(a, b)$ denotes the kernel function and is its (implicit) feature map

1.0.2 Implementation

```
In [2]: from math import sqrt

        K2_SIGMA = sqrt( 1.0/ (2.0*0.3) )
        K3_alpha = 2
        K3_constatnt = 1
        def polynomial_kernel(X, Y):
                return (X.T.dot(Y) + 1) ** 2

        def gaussian_kernel(X, Y):
                return np.exp( (-1 * (np.linalg.norm(X - Y) ** 2)) / (2 * (K2_SIGMA ** 2)) )

        def rbf_kernel(v1, v2, sigma=1.0):
            return np.exp((-1 * np.linalg.norm(v1 - v2) ** 2) / (2 * sigma ** 2))

        def sigmoid_kernel(X, Y):
                return np.tanh(K3_alpha*X.T.dot(Y) + K3_constatnt)

        def kkmeans_kernel(x,y,kernel):
                return kernel(x,x) + kernel(y,y) - 2*kernel(x,y)

        def kkmeans(X, K,kernel, maxIters=10 ,plot_progress = None):
                global fig
                size = len(X)
                centroids_indexes = np.random.choice(np.arange(size), K)
                centroids = X[centroids_indexes,:]

                for i in range(maxIters):
                        # cluster Assignment step
                        C = np.zeros(len(X))
                        for x_i in range(len(X)):
                                idx = 0
                                current_kernel_distance = kkmeans_kernel(X[x_i],centroids[0],ke
                                for y_k in range(len(centroids)):
                                        temp = kkmeans_kernel(X[x_i],centroids[y_k],kernel)
                                        if temp < current_kernel_distance:
                                                current_kernel_distance = temp
                                                idx = y_k
                                C[x_i] = idx

                        centroids = [X[C == k].mean(axis = 0) for k in range(K)]
                print(centroids)
                return np.array(centroids) , C
```

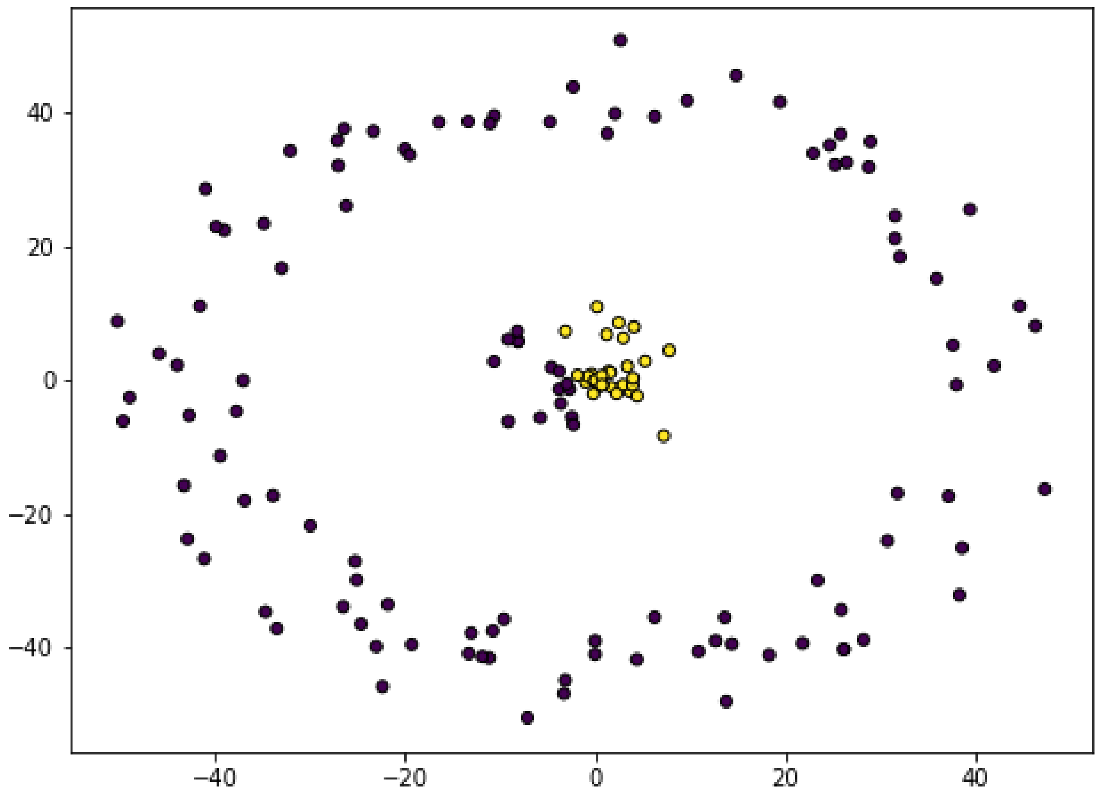
1.0.3 Tests

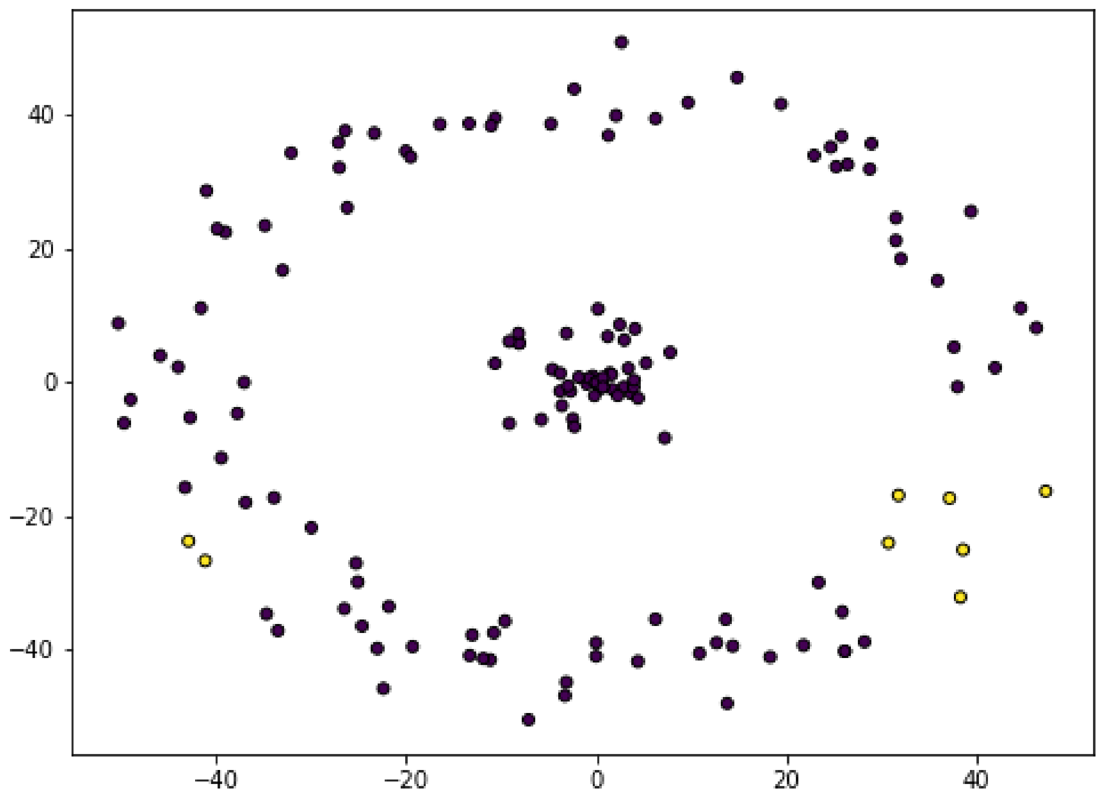
comparing our implementation of kernel kmeans with the one in sklearn

```
In [3]: if __name__ == "__main__":
                data = np.loadtxt("../data_k2.data")
                centroids, C = kkmeans(data, 2, gaussian_kernel ,maxIters=50, plot_progress=Tru
                fig += 1
                plt.figure(fig, figsize=(8,6))
                plt.clf()
                plt.scatter(data[:,0], data[:,1], c=C, s=25, edgecolor='k')
                plt.show()

                clustering = SpectralClustering(n_clusters=2 ,affinity = "rbf", gamma=0.3, coe
                fig += 1
                plt.figure(fig, figsize=(8,6))
                plt.clf()
                plt.scatter(data[:,0], data[:,1], c=clustering.labels_, s=25, edgecolor='k')
                plt.show()
```

```
[array([-3.90218148, -2.8254477 ]), array([1.59390657, 1.21866111])]
```





40
20
0
−20
−40
−40
−20
0
20
40