CS 335
Assignment 4

Aakriti

Due 29th October 2021

Contents

1 Clustering

1.1 CS 335: KMeans Implementation

(i)

Refer to `assignment_4.ipynb` for the code.

(ii)

K-means clustering algorithm is implemented which stops if which takes a maximum of **1000 iterations** and stops early if distance between each of the old and new cluster centers is less than epsilon $\epsilon = \mathbf{10^{-8}}$.

DATASET 1

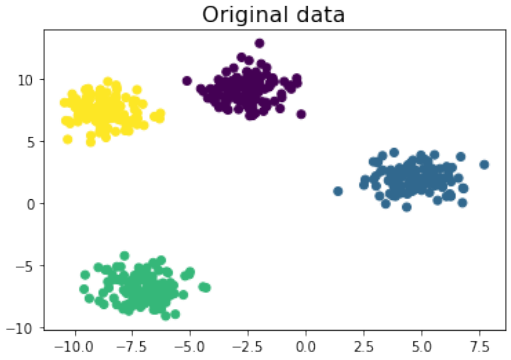

Figure 1: Original Data

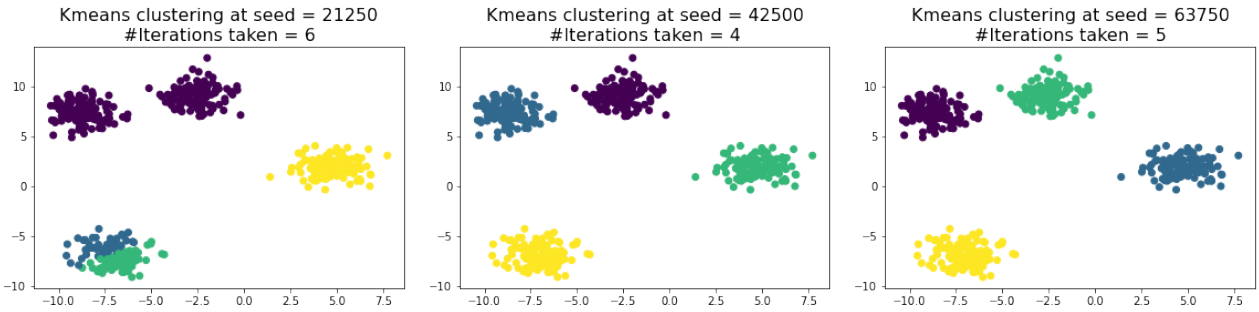

Figure 2: K-means clustered data with maximum iterations = 1000, epsilon $\epsilon = \mathbf{10^{-8}}$

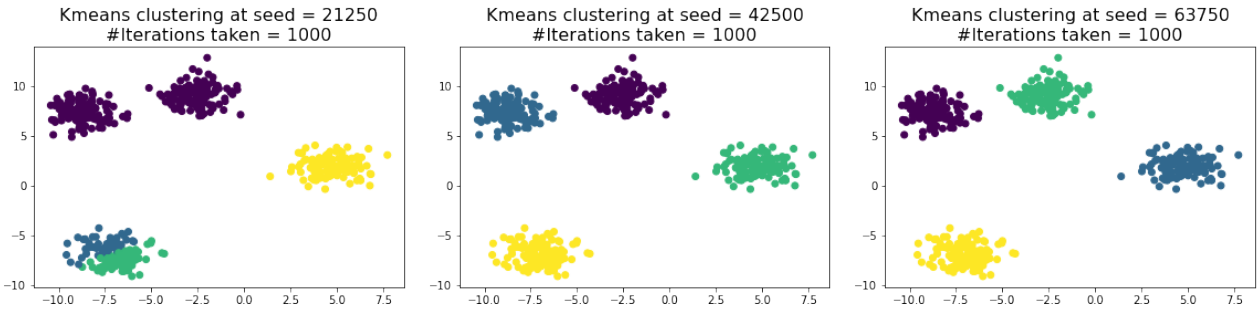

Figure 3: K-means clustered data with maximum iterations = 1000, epsilon $\epsilon = \mathbf{0}$

DATASET 2

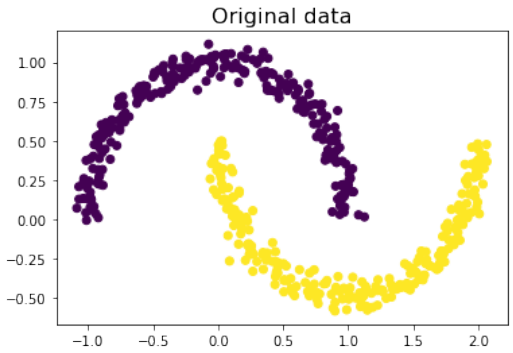

Figure 4: Original Data

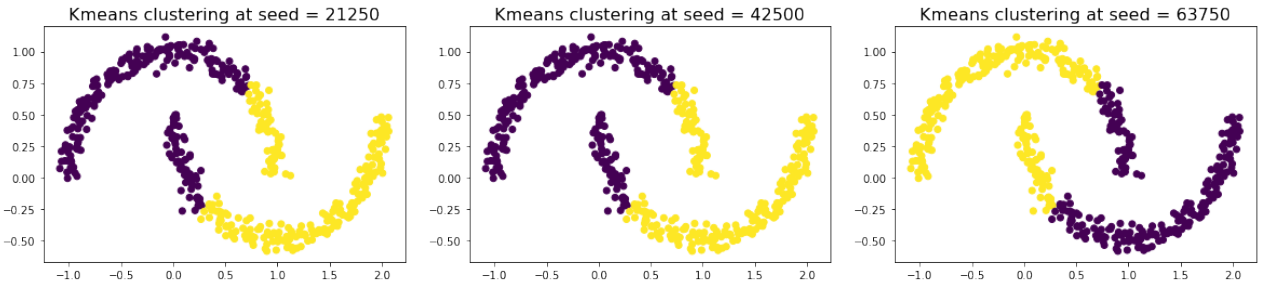

Figure 5: K-means clustered data with maximum iterations = 1000, epsilon $\epsilon = 10^{-8}$

DATASET 3

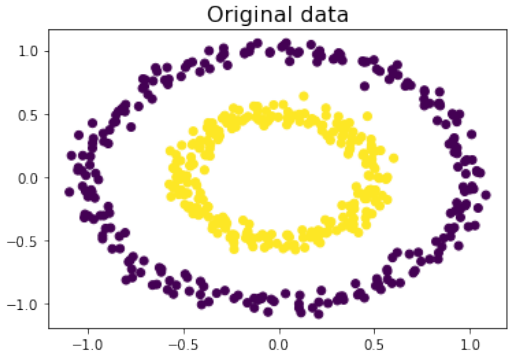

Figure 6: Original Data

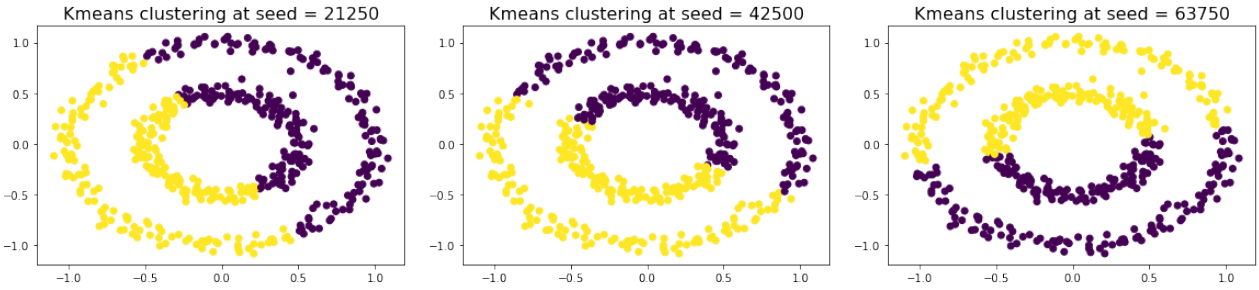

Figure 7: K-means clustered data with maximum iterations = 1000, epsilon $\epsilon = 10^{-8}$

Comments -

- For linearly separable dataset with spherical clusters, if initialization is good, K-means can find can find the optimal clusters and converge in a reasonable amount of time.
- Since it uses euclidean distance for assigning points to centroids, K-means clustering is not suited to all datasets. K-means may NOT perform well in the following cases -
  1. Clusters are non-spherical or non-convex shaped as is seen in case of Dataset 2
  2. Clusters are non-linearly separable as is seen in case of Dataset 2 and 3
  3. Cluster centroids have poor initialization as is seen in the case of Dataset 1 with seed = 21250
  4. Data has outliers. We do not see a clear example of this, but it can be inferred from some runs on Dataset 1 that outliers add unnecessary noise and negatively affect the position of centroids
  5. Clusters are of unequal size or data points of one cluster are much more spread apart more than others. Being based on distance, K-means algorithm does not let data points that are far-away from each other share the same cluster as seen in Dataset 3.
- Clusters formed upon convergence are heavily dependent on the initialization of centroids. So, if the initialization is good, algorithm converges quickly to the optimal clusters, but if initialization is bad, even after taking 1000 iterations of the K-means algorithm, the clusters remain almost the same in the above datasets.
- For the given datasets, if the condition for convergence is that the distance between each of the old and new cluster centers is less than epsilon $\epsilon = 10^{-8}$, algorithm converges in less than 100 iterations.

(iii)

A good centroid initialization should ensure that the initial cluster centers are placed as close as possible to the optimal cluster centers.

A) A simple initialisation method -
For k clusters, randomly chose k unique points from the dataset as initial centroids for clusters.

Justification of choice -

- A fast initialisation with linear time complexity and straightforward to implement

- As clusters detected through k-Means are more probable to be near the modes present in data, by randomly choosing points from data, we are making it more probable to get a point that lies close to the modes
- In cases such as Dataset 1,where clusters are spread apart and well separated, this method is more likely to find initial centroids that are spread apart too, and more likely belonging to different clusters.

B) Alternative initialisation method tested -
For k clusters, randomly assign each point in the data to a random `cluster_id` $\in$ {`1, 2, ...k`}. For each `cluster_id`, take mean of all points assigned to that `cluster_id` as the initial centroid for that cluster.

Justification for not choosing this method -

- The initial points chosen lie very close to the global mean of the data. So, initial centroids are lie close to each other and can cause confusion in clusters.
- K-means is more likely to get stuck in Local Optima if initialized using this method.

Empirical comparison of (A) and (B) -

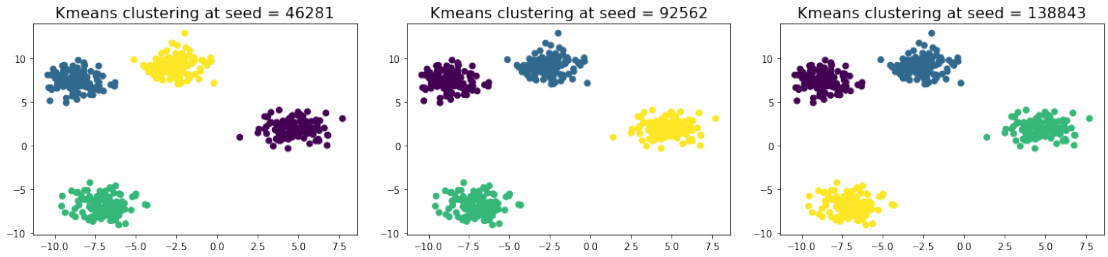

Figure 8: K-means clustered data with initialisation method of **(A)**

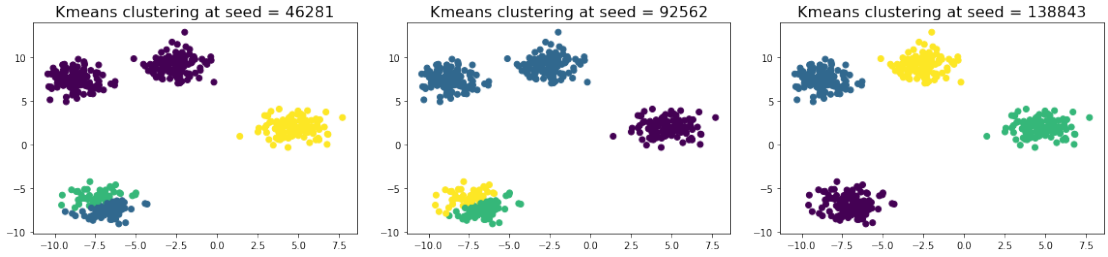

Figure 9: K-means clustered data with initialisation method of **(B)**

As we can see in runs for three different seeds, (A) performs better than (B) in most cases.

C) Best initialisation method implemented -

1. Choose a random point from the data
2. Choose the next point such that is more probable to lie at a large distance from the first point. For this, sample a point from a probability distribution that is proportional to the squared distance of a point from the first center.

3. The remaining points are generated by a probability distribution that is proportional to the squared distance of each point from its closest center. So, a point having a large distance from its closest center is more likely to be sampled.

Justification for this method -

- Here we see that, a point having a large distance from its closest center is more likely to be sampled as initial centroid.
- The effect is an attempt to push the centroids as far from one another as possible, covering as much of the occupied data space as they can from initialization. This would greatly reduce the probability that 2 points belonging to the same cluster are sampled as initial centroids of different clusters. So, in case of linearly separable datasets, this would be optimal.

By running the algorithm for the same 3 seeds, we can see that **(C)** would be the most ideal initialization method for the given datasets and almost always gives optimal clusters -

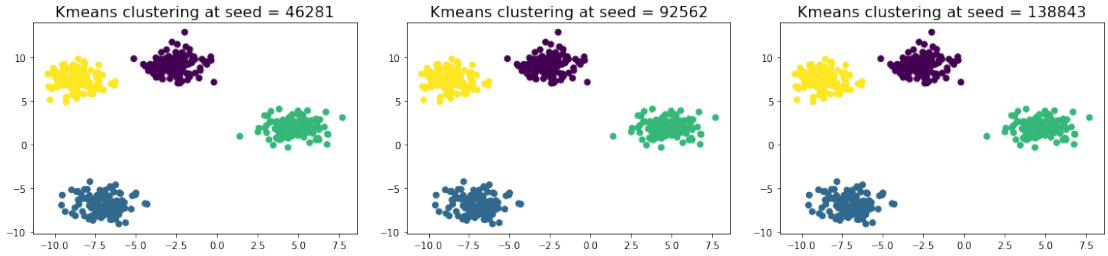

Figure 10: K-means clustered data with initialisation method of **(C)**

2 Kernel design and Kernelized clustering

2.1 CS 337: Proving kernel validity

Prove that the function $K_\sigma : \mathbb{R}^n \times \mathbb{R}$ *defined as* $K_\sigma(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(\frac{-\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}\right)$ *is a valid Kernel. You may use the properties proved in class.*

We prove using the following properties -

(a) Sum Closure: If $K_1(x, y)$ and $K_2(x, y)$ are valid kernels, then $K_1(x, y) + K_2(x, y)$ is a valid kernel [from slides]

(b) Product Closure: If $K_1(x, y)$ and $K_2(x, y)$ are valid kernels, then $K_1(x, y) \times K_2(x, y)$ is a valid kernel [from slides]

(c) Positive Scaling: If $K(x, y)$ is a valid kernel, then $\alpha K(x, y)$ is also a valid kernel for $\alpha > 0$.
Since ϕ is a feature map for $K(x, y)$, $\sqrt{\alpha}\phi$ is valid feature map for $\alpha \times K(x, y)$

(d) Any positive (or even non negative) constant is a valid kernel i.e., $K(x, y) = a$ for $a \geq 0$ is a valid kernel.
Since we can construct a feature space $\phi(x)$, such that $\phi(x) = [\sqrt{a}]$

(e) If $K(x, y)$ is avlid kernel, then $e^{K(x,y)}$ is also a valid kernel.

Since, we can expand the above by Taylor series expansion, we have -

$$e^{K(x,y)} = \sum_{i=0}^{\infty} \frac{K(x, y)^i}{i!}$$

- Now, each term of $\frac{K(x, y)^i}{i!}$ is valid kernel.
- $K(x, y)^i$ is a valid kernel as, by using Product closure of kernels (b), we multiply $K(x, y)$, a valid kernel i times to get $K(x, y)^i$.
- Since, $i!$ is just a constant > 0, we can apply Positive scaling (c) to a valid kernel $K(x, y)^i$ with constant $\frac{1}{i!} > 0$ to get valid kernel $\frac{K(x, y)^i}{i!}$.
- For $i = 0$, the constant 1 is also a vlaid kernel by reasoning of (d) shown above.
- Now, as each individual term of the summation $\sum_{i=0}^{\infty} \frac{K(x, y)^i}{i!}$ is a vlid kernel by Sum closure of kernels (a) we have that their sum is also a valid kernel.

Thus, the whole expression $e^{K(x,y)}$ is a valid kernel.

Since $e^{K(x,y)}$ is a valid kernel, there exists a feature map $\phi : \mathbb{R}^m \to H$ where, H is a Hilbert space, such that -

$$e^{K(x,y)} = \phi(x)^T \phi(y)$$

Since $x^T y$ is a valid kernel (when $\phi(x) = x$) thus, by positive scaling (c) we have that $\frac{x^T y}{\sigma^2}$ is also a valid kernel.

Let us assume -

$$\exp\left(\frac{x^T y}{\sigma^2}\right) = \phi(x)^T \phi(y)$$

So we have -

$$\begin{aligned}
\exp\left(\frac{-\|x-y\|^2}{2\sigma^2}\right) &= \exp\left(\frac{-\|x\|^2}{2\sigma^2} + \frac{-\|y\|^2}{2\sigma^2} + \frac{2x^T y}{2\sigma^2}\right) \\
&= \exp\left(\frac{-\|x\|^2}{2\sigma^2}\right) \times \exp\left(\frac{x^T y}{\sigma^2}\right) \times \exp\left(\frac{-\|y\|^2}{2\sigma^2}\right) \quad [Rearranging] \\
&= \exp\left(\frac{-\|x\|^2}{2\sigma^2}\right) \phi(x)^T \phi(y) \exp\left(\frac{-\|y\|^2}{2\sigma^2}\right) \quad [Substituting] \\
&= \phi'(x)^T \phi'(y)
\end{aligned}$$

where, $\phi'(x) = \phi(x) \exp\left(\frac{-\|x\|^2}{2\sigma^2}\right)$

$\therefore$ Since a feature map ϕ' exists, $\exp\left(\frac{-\|x-y\|^2}{2\sigma^2}\right)$ is a valid kernel.
Hence, proved!

2.2 CS 337: Simple Kernel Design

(i)

Is there a condition on r_1, r_2 such that the vanilla KMeans (vanilla means we need to run the algorithm as is on the given data without transformations of any kind) algorithm gives us the correct clusters? Explain with sound arguments.

For any pair of values r_1 and $r_2 (> r_1)$ the points cannot be clustered using (vanilla) K means algorithm. This is due to the fact that K-means cannot clusterize clusters which are not linearly separable or non-convex. We can prove this specifically for the given radial dataset using contradiction. Suppose there exist 2 distinct cluster centers c_1 and c_2 such that K-means can perfectly cluster the dataset. c_1 corresponds to radius r_1 and c_2 to radius r_2. Now, consider the following cases:

Case: (a) Suppose c_1 and c_2 both inside the circle of radius r_2. If we take the line joining c_1 and c_2, it will intersect the outer circle at points p_1 and p_2. As, p_1 and p_2 belong to the cluster corresponding to c_2, their distances from c_2 should be less than their corresponding distances from c_1. So, we have -

$$\|p_2 - c_2\| \leq \|p_2 - c_1\| \tag{1}$$

and

$$\|p_1 - c_2\| \leq \|p_1 - c_1\| \tag{2}$$

However, since p_1 and p_2 lie on the line joining c_1 and c_2, one of them would have to be closer to c_1 and the other one to c_2. So, we have -
Either

$$\|p_1 - c_2\| < \|p_1 - c_1\| \ \ \& \ \ \|p_2 - c_1\| < \|p_2 - c_2\| \tag{3}$$

Or

$$\|p_2 - c_2\| < \|p_2 - c_1\| \ \ \& \ \ \|p_1 - c_1\| < \|p_1 - c_2\| \tag{4}$$

If (3) is true, (3) and (1) give contradiction and if (4) is true, (4) and (2) give a contradiction!

Case: (b) Suppose c_1 lies inside the outer circle, and c_2 lies outside. Let the line joining c_1 and c_2 intersect the outer circle at p_1 and p_2 such that p_1 lies between c_1 and c_2 and p_2 lies away from both. So, we have

$$\|p_2 - c_1\| < \|p_2 - c_2\| \tag{5}$$

Since, p_1 and p_2 lie on the outer circle, they belong to cluster corresponding to c_2 so, equations (1) and (2) are valid. But now again, we have a contradiction with equation (1) and (5)!

Case: (c) Similarly, we suppose c_2 lies inside the outer circle, and c_1 lies outside.
Let the line joining c_1 and c_2 intersect the inner circle at p_1 and p_2 such that p_1 lies between c_1 and c_2 and p_2 lies away from both. So, we have

$$\|p_2 - c_2\| < \|p_2 - c_1\| \tag{6}$$

Now, p_2 belongs to cluster corresponding to c_1 so,

$$\|p_2 - c_1\| \leq \|p_2 - c_2\| \tag{7}$$

The above two equations, (6) and (7) contradict each other so, here too we arrive at a contradiction!

Case: (d) Finally, consider the case when both c_1 and c_2 lie outside the outer circle. Then consider the point on the outer circle that lies on the line joining the center and c_1. This point is closer to c_1 than c_2, again giving a contradiction!

Since we arrive at a contradiction in all cases, our assumption is wrong!

Hence, we have showed that vanilla K-means cannot clusterize this dataset.

(ii)

For the configurations of r_1, r_2 that are not clusterable, can you suggest a kernel that will help KMeans identify the correct clusters? Specify both the transformation $\phi(x)$ and the kernel function $k(x, x')$. Further show that the kernel function you propose is a valid kernel.

The transformation $\phi(x)$ is defined as -

$$\phi(x) = \|x\|_2^2$$

Consider the kernel as follows -

$$K(x, y) = \langle \phi(x), \phi(y) \rangle = \|x\|_2^2 \|y\|_2^2$$

To show that this is a valid Mercer Kernal, consider the square integrable function $g(x)$. Then, we have -

$$\int_x \int_y K(x, y) g(x) g(y) dx dy = \int_x \int_y \|x\|^2 \|y\|^2 g(x) g(y) dx dy$$

Using Fubini's theorem, the RHS can be written as -

$$\begin{aligned} \int_x \int_y \|x\|^2 \|y\|^2 g(x) g(y) dx dy &= \left(\int_x \|x\|^2 g(x) dx \right) \left(\int_y \|y\|^2 g(y) dy \right) \\ &= \left(\int_x \|x\|^2 g(x) dx \right)^2 \geq 0 \end{aligned}$$

Therefore, the defined kernel is a valid Mercer Kernel!

Justification for choice -
Kernels operate in a high-dimensional, implicit feature space without necessarily computing the coordinates of the data in that space, but rather by simply computing the Kernel function. Kernel trick allows us to project our data into a higher dimensional space to achieve linear separability and solve the K-Means problem in a more efficient way.

We need to project the circles dataset into a space such that it is linearly separable based on radius. After zero centering of data, the 2-norm of any point $\|x\|_2$ is equal to length of its radius. So, points

in Circles dataset at distance r_1 from origin will have lower values of Kernel function than points at distance r_2 hence, the dataset can be clustered.

The results for the above defined kernel are presented below -

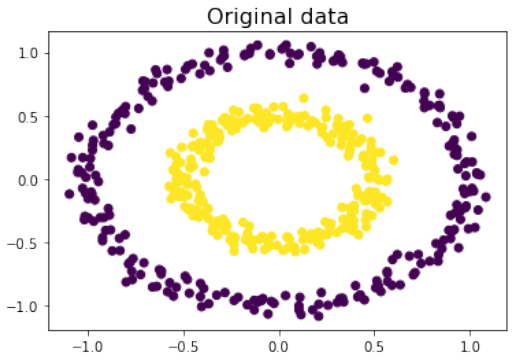

Figure 11: Original Data

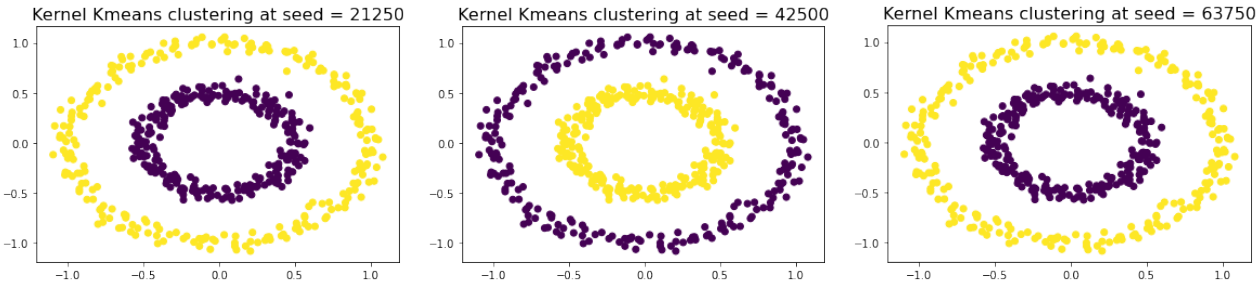

Figure 12: Kernel K-means clustered data

Refer to `assignment_4.ipynb` for the implementation of Kernel K-means.

The algorithm runs for **1000** iterations with an early stopping condition that the fraction of points for `cluster_id` does not change should be less than $\mathbf{10^{-8}}$.
For initialisation, the `cluster_id` of each point in the data is randomly chosen from {1, 2, ...k} for k clusters (a bit similar to what is described in 1.1 (iii) - (B)).