Andrew Li-Yang Liu (ALL2209)
Professor Verma
Machine Learning COMS 4771
Columbia University
December 12, 2021

<u>Machine Learning Homework 4</u>

Problem 1:

Nature creates the label values via:

$$y_i = \sum_{i=1}^{d} w_j x_{ij} + \epsilon_i$$

Where:

$$\epsilon_i \sim N(0, \sigma^2)$$
$$w_j \sim N(0, \tau^2)$$

So:

$$y_i | x_i, w \sim N\left(\sum_{i=1}^{d} w_j x_{ij}, \sigma^2 \right)$$

$$p(y_i | x_i, w) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(y_i - \sum_{j=1}^{d} w_j x_{ij}\right)^2}{2\sigma^2} \right)$$

We note that, using Bayes rule:

$$p(w | x_i, y_i) \propto p(y_i | x_i, w) p(w)$$

So, using the fact that each sample is independently drawn from the underlying distribution, the likelihood function can be computed as:

$$p(w | (x_i, y_i)_{i=1}^{n}) \propto \prod_{i=1}^{n} p(y_i | x_i, w) p(w)$$
$$= C \exp\left(-\frac{\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{d} w_j x_{ij}\right)^2}{2\sigma^2} \right) \exp\left(-\frac{n \sum_{j=1}^{d} w_j^2}{2\tau^2} \right)$$
$$= C \exp\left(-\frac{\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{d} w_j x_{ij}\right)^2}{2\sigma^2} - \frac{n \sum_{j=1}^{d} w_j^2}{2\tau^2} \right)$$

The objective of maximizing the likelihood function is expressed as:

$$\max_w C \exp\left(-\frac{\sum_{i=1}^n \left(y_i - \sum_{j=1}^d w_j x_{ij}\right)^2}{2\sigma^2} - \frac{n\sum_{j=1}^d w_j^2}{2\tau^2}\right)$$

Which reduces to:

$$\max_w -\frac{\sum_{i=1}^n \left(y_i - \sum_{j=1}^d w_j x_{ij}\right)^2}{2\sigma^2} - \frac{n\sum_{j=1}^d w_j^2}{2\tau^2}$$

Which reduces to:

$$\min_w \frac{\sum_{i=1}^n \left(y_i - \sum_{j=1}^d w_j x_{ij}\right)^2}{2\sigma^2} + \frac{n\sum_{j=1}^d w_j^2}{2\tau^2}$$

Equivalent to:

$$\min_w \sum_{i=1}^n \left(y_i - \sum_{j=1}^d w_j x_{ij}\right)^2 + \frac{n\sigma^2}{\tau^2}\sum_{j=1}^d w_j^2$$

This is equivalent to the ridge regression minimization problem with $\lambda = \frac{n\sigma^2}{\tau^2}$.

Problem 2:

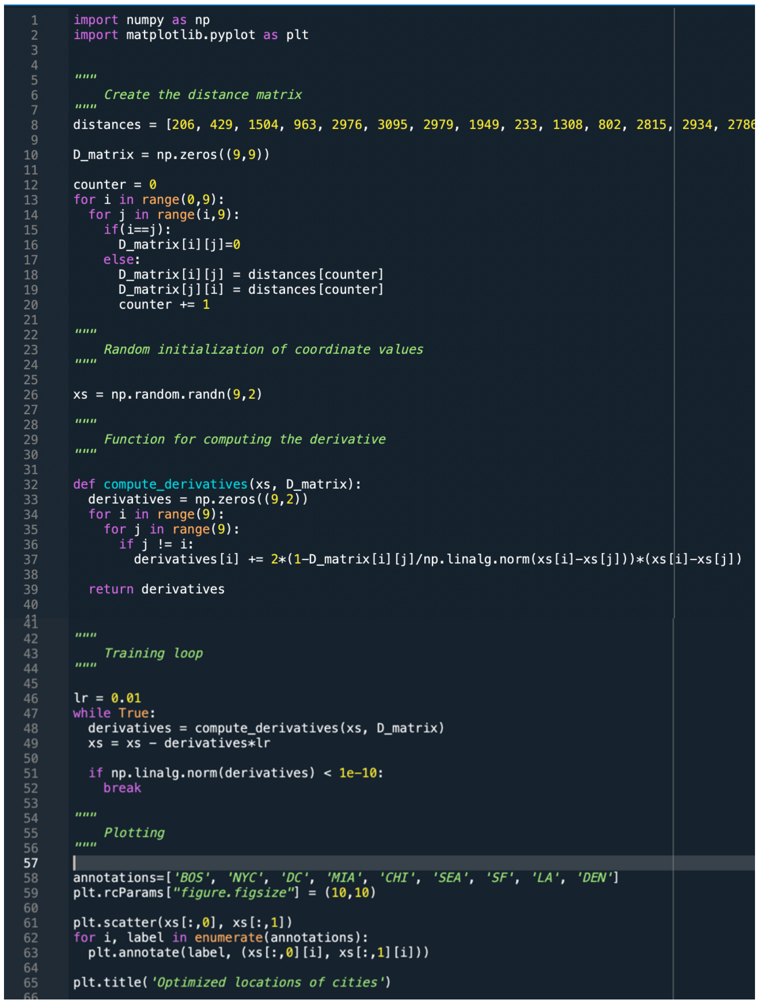
2a.

$$\frac{\partial}{\partial x_i}\sum_{i,j}\left(\left\|x_i - x_j\right\| - D_{ij}\right)^2$$

$$= \sum_j 2\left(\left\|x_i - x_j\right\| - D_{ij}\right)\frac{\partial}{\partial x_i}\left\|x_i - x_j\right\|$$

$$= \sum_j \frac{2\left(\left\|x_i - x_j\right\| - D_{ij}\right)}{\left\|x_i - x_j\right\|}(x_i - x_j)$$

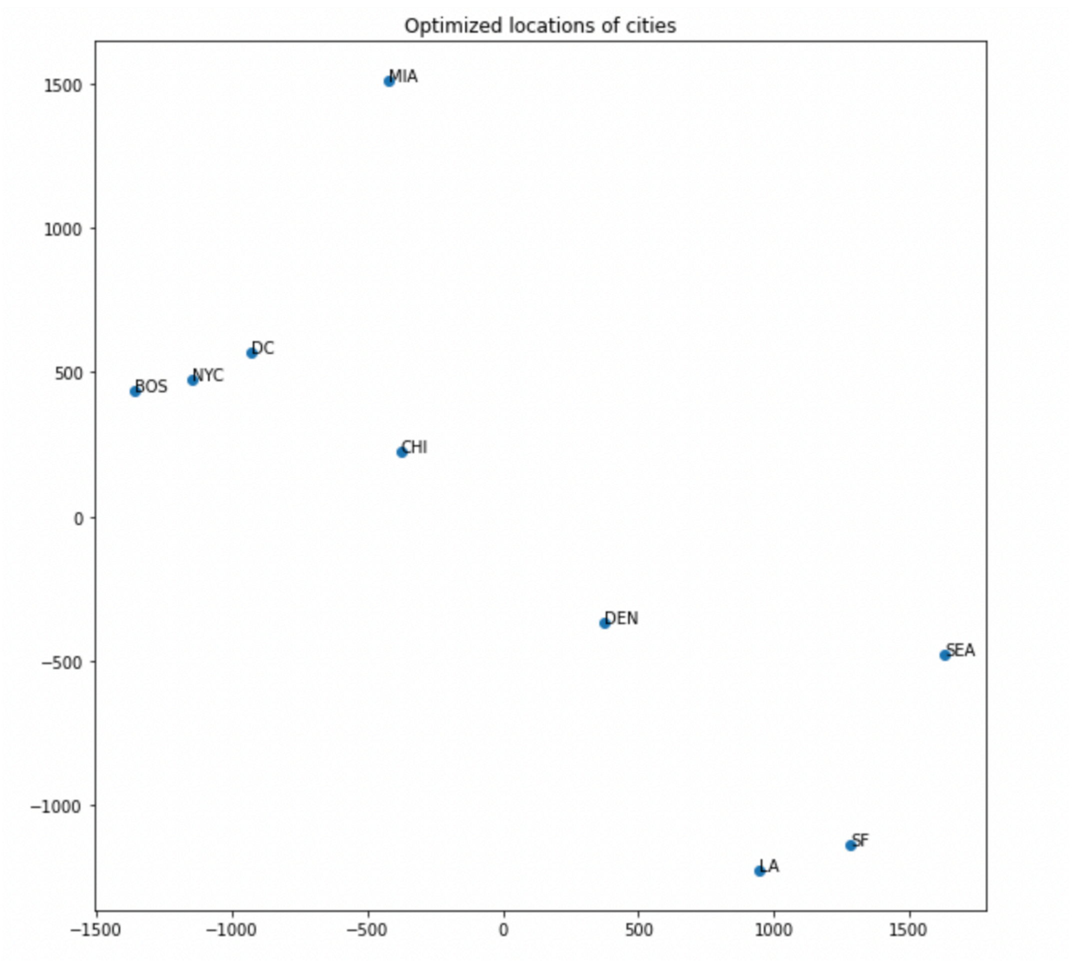
2b. (CODE SUBMITTED TO PROGRAMMING SECTION ON GRADESCOPE)



```python
import numpy as np
import matplotlib.pyplot as plt


"""
    Create the distance matrix
"""
distances = [206, 429, 1504, 963, 2976, 3095, 2979, 1949, 233, 1308, 802, 2815, 2934, 2786

D_matrix = np.zeros((9,9))

counter = 0
for i in range(0,9):
  for j in range(i,9):
    if(i==j):
      D_matrix[i][j]=0
    else:
      D_matrix[i][j] = distances[counter]
      D_matrix[j][i] = distances[counter]
      counter += 1

"""
    Random initialization of coordinate values
"""

xs = np.random.randn(9,2)

"""
    Function for computing the derivative
"""

def compute_derivatives(xs, D_matrix):
  derivatives = np.zeros((9,2))
  for i in range(9):
    for j in range(9):
      if j != i:
        derivatives[i] += 2*(1-D_matrix[i][j]/np.linalg.norm(xs[i]-xs[j]))*(xs[i]-xs[j])

  return derivatives


"""
    Training loop
"""

lr = 0.01
while True:
  derivatives = compute_derivatives(xs, D_matrix)
  xs = xs - derivatives*lr

  if np.linalg.norm(derivatives) < 1e-10:
    break

"""
    Plotting
"""

annotations=['BOS', 'NYC', 'DC', 'MIA', 'CHI', 'SEA', 'SF', 'LA', 'DEN']
plt.rcParams["figure.figsize"] = (10,10)

plt.scatter(xs[:,0], xs[:,1])
for i, label in enumerate(annotations):
  plt.annotate(label, (xs[:,0][i], xs[:,1][i]))

plt.title('Optimized locations of cities')
```

2c.

From the generated plot, it is clear that the estimated locations and the actual geographic locations are rather similar. In particular, we expected BOS, NYC, and DC to be close (as they are on the East coast) and LA, SF, and SEA to be on the other end. It is also reasonable that CHI and DEN appeared somewhere in the middle. The relative distances thus all make sense. However, the relative positions of the plot are somewhat different compared to the real map. For example, in the plot, it is suggested that MIA is North of LA, SF, SEA, DEN, and CHI, whereas MIA is supposed to be South of those locations.



Problem 3:

3.1.

By conditional independence of s and x given y, we have:

$$P[s=1,x|y=1]=P[s=1|y=1]P[x|y=1]$$

So, one of the terms that we are interested in is:

$$P[s=1|y=1]=\frac{P[s=1,x|y=1]}{P[x|y=1]}$$

Apply Bayes Theorem to the denominator of the RHS:

$$=\frac{P[s=1,x|y=1]}{\left(\frac{P[y=1|x]P[x]}{P[y=1]}\right)}$$
$$=\frac{P[s=1,x|y=1]P[y=1]}{P[y=1|x]P[x]}$$

$$= \frac{P[s=1,x,y=1]}{P[y=1|x]P[x]}$$

Note that since $P[s=1|x,y=0]=0$, then: $P[s=1,x,y=0]=P[s=1|x,y=0]P[x,y=0]=0$, so we can add zero to the numerator:

$$= \frac{P[s=1,x,y=1]+P[s=1,x,y=0]}{P[y=1|x]P[x]}$$
$$= \frac{P[s=1,x]}{P[y=1|x]P[x]}$$
$$= \frac{P[s=1|x]}{P[y=1|x]}$$

Hence:

$$P[y=1|x] = \frac{P[s=1|x]}{P[s=1|y=1]}$$

As desired.

3.2.

$$P[y=1|x,s=0]$$

Use the fact that $P(A|B,C)=\frac{P(A,B|C)}{P(B|C)}$ to obtain:

$$= \frac{P[y=1,s=0|x]}{P[s=0|x]}$$
$$= \frac{P[y=1,s=0,x]}{P[x]P[s=0|x]}$$
$$= \frac{P[s=0,x|y=1]P[y=1]}{P[x]P[s=0|x]}$$

Apply conditional independence of x and s given y:

$$= \frac{P[s=0|y=1]P[x|y=1]P[y=1]}{P[x]P[s=0|x]}$$

We can apply Bayes rule on the numerator:

$$= \frac{P[s=0|y=1]P[y=1|x]}{P[s=0|x]}$$

Now apply the result from 3.1. to obtain:

$$= \frac{1-P[s=1|y=1]}{P[s=1|y=1]}\frac{P[s=1|x]}{1-P[s=1|x]}$$

As desired.

3.3.

$$E_{(x,y)\sim D}\big[1[f(x)\neq y]\big] = \int_x dx \int_s d\mu(s) \int_y d\mu(y)\, p(x,y,s)1[f(x)\neq y]$$

$$= \int_x dx \int_s d\mu(s) p(x,s,y=0)1[f(x)\neq 0] + p(x,s,y=1)1[f(x)\neq 1]$$

$$= \int_x dx\, p(x,s=0,y=0)1[f(x)\neq 0] + p(x,s=0,y=1)1[f(x)\neq 1] + p(x,s=1,y=0)1[f(x)\neq 0] + p(x,s=1,y=1)1[f(x)\neq 1]$$

Since $p(x,s=1,y=0)=0$, we must have: Since $p(x,s=1,y=1)=p(x,s=1)$. Hence:

$$= \int_x dx\, p(x,s=0,y=0)1[f(x)\neq 0] + p(x,s=0,y=1)1[f(x)\neq 1] + p(x,s=1)1[f(x)\neq 1]$$

$$= \int_x dx\, p(x,s=1)1[f(x)\neq 1] + p(x,s=0)\big(\Pr[y=1|x,s=0]\,1[f(x)\neq 1] + \Pr[y=0|x,s=0]\,1[f(x)\neq 0]\big)$$

As desired.

3.4.

For empirical error, we can simply take:

$$\frac{1}{|S|}\sum_{(x,s)\in S} 1[s=1]\,1[f(x)\neq 1] + 1[s=0]\big\{w(x)1[f(x)\neq 1] + \big(1-w(x)\big)1[f(x)\neq 0]\big\}$$

$$= \frac{1}{|S|}\sum_{\substack{(x,s)\in S\\ s=1}} 1[f(x)\neq 1] + \frac{1}{|S|}\sum_{\substack{(x,s)\in S\\ s=0}} w(x)1[f(x)\neq 1] + \big(1-w(x)\big)1[f(x)\neq 0]$$

Where:

$$w(x) = P[y=1|s=0,x]$$

And thus:

$$1-w(x) = P[y=0|s=0,x]$$

By our previous assumptions, w(x) is estimable using only (s,x). All other terms in our empirical error also involve only s and x, as desired.

Problem 4:

4.a.

If we can simultaneously optimize over k and c, we can achieve zero error by having k=n and let each cluster center be right on top of each data point ($c_i = x_i \ \forall i$). This defeats the purpose of clustering.

4.b.i.

If k=1:

$$\sum_{i=1}^{n} \min_{j\in\{1,\dots,k\}} \left|\left|x_i - c_j\right|\right|^2$$
$$= \sum_{i=1}^{n} \left|\left|x_i - c\right|\right|^2$$

Minimizing over this objective, we take the derivative with respect to c and set it to zero:

$$\frac{\partial}{\partial c}\sum_{i=1}^{n} \left|\left|x_i - c\right|\right|^2 = \sum_{i=1}^{n} -2(x_i - c) = 0$$
$$\sum_{i=1}^{n} x_i = nc$$
$$c^* = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Hence, if k=1 (we want only 1 cluster), we must assign the cluster center to be the **average** of the n points. This achieves an objective value of:

$$\sum_{i=1}^{n} \left|\left| x_i - \frac{1}{n}\sum_{j=1}^{n} x_j \right|\right|^2$$

4.b.ii:

Consider the example of d=1, k=2, where we have the data points:

$$x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 5$$

If we initialize our clusters centers to be the following:

$$c_1 = 1.5, c_2 = 4$$

Then k-means assigns the following points to each of the clusters:

$$C_1 = \{1,2\}$$
$$C_2 = \{3,5\}$$

Using these cluster assignments, we see that we get cluster centers of $c_1 = 1.5$ and $c_2 = 4$ again, so k-means has converged. These (converged) assignments give us the objective value of:

$$0.5^2 + 0.5^2 + 1^2 + 1^2 = 2.5$$

But consider the alternative cluster centers $c_1' = 2$ and $c_2' = 5$, with cluster assignments $C_1 = \{1,2,3\}$, $C_2 = \{5\}$. This gives us an objective value of:

$$1^2 + 1^2 = 2$$

Which is lower than the objective value for the previous cluster assignments and cluster centers. Hence, k-means is suboptimal (does not necessarily reach global optimum objective value) for d=1, k=2.

To consider the case of k>2, simply extend the example above by adding more data points (in our example, we consider having k+2 data points):

$$x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 5, x_5, x_6, x_7, x_8, \dots, x_{k+2}$$

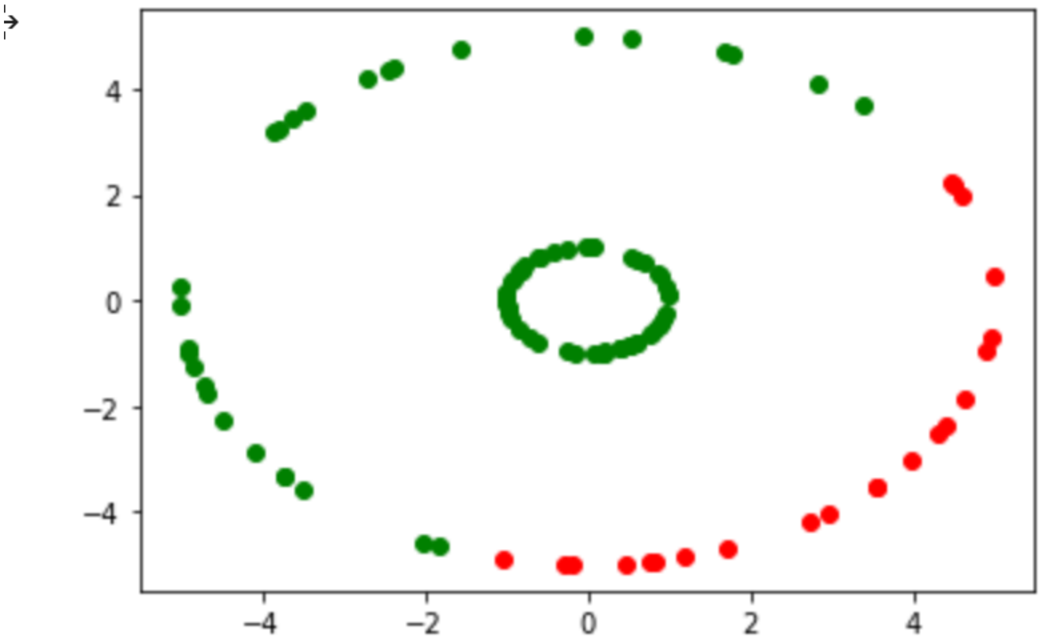
(for our example, let $x_5, \dots, x_{k+2}$ take any value)

We still let $c_1 = 1.5$, $c_2 = 4$. Let $c_3 = x_5, c_4 = x_6, c_5 = x_7, \dots, c_k = x_{k+2}$ Clearly, the clusters $c_3, \dots, c_k$ each contribute zero loss. So we can simply apply the same argument used in the case with k=2 by considering the alternative assignment $c_1' = 2, c_2' = 5, c_3{}' = x_5, c_4{}' = x_6, c_5{}' = x_7, \dots, c_k{}' = x_{k+2}$. It is then clear that k-means is suboptimal for any k>2 as well.

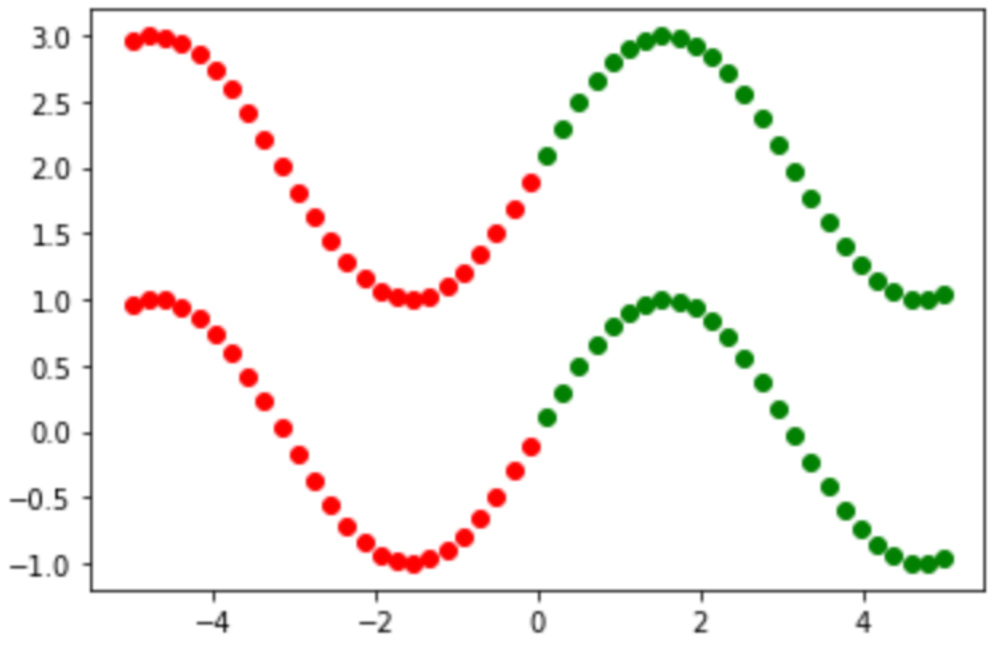
Hence, we've disproved the notion that k-means is optimal for d=1, k>=2 via a counterexample.

4.c.i:

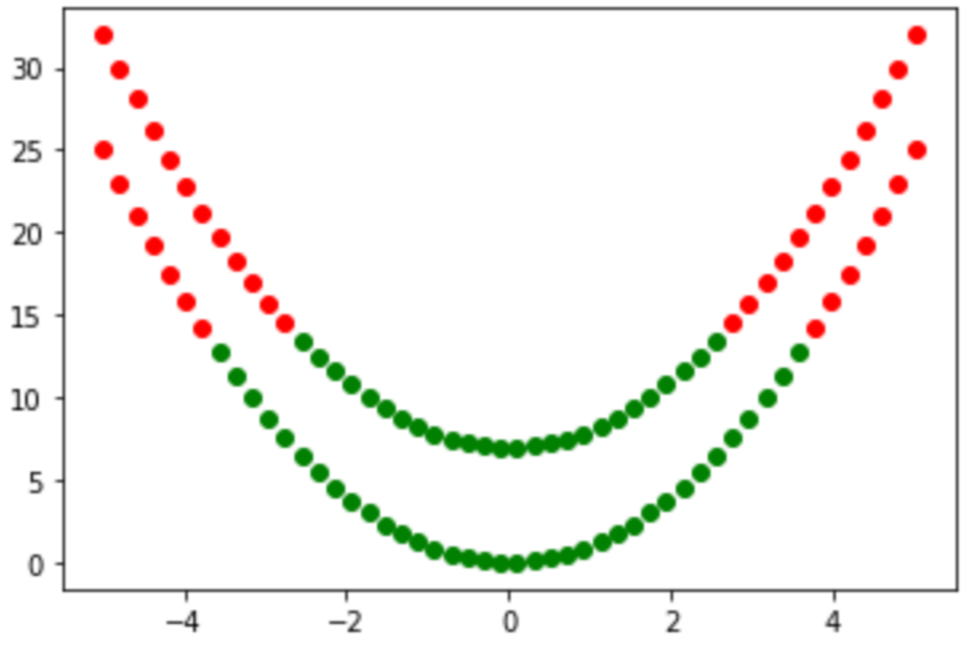
Two concentric circles:



Two sinusoids:



Two parabolas:



4.c.ii:

c_j should be the average of all the points assigned to the jth cluster. Hence, we can take:

$$\alpha_{ij} = \frac{z_{ij}}{\sum_{k=1}^{n} z_{kj}}$$

Since: $n_j = \sum_{k=1}^{n} z_{kj}$. Hence, we can write:

$$c_j = \sum_{i=1}^{n} \alpha_{ij}\phi(x_i)$$

4.c.iii:

$$\begin{aligned}
&\left|\left|\phi(x_i) - \phi(x_j)\right|\right|^2 \\
&= \langle \phi(x_i) - \phi(x_j), \phi(x_i) - \phi(x_j) \rangle \\
&= \langle \phi(x_i), \phi(x_i) \rangle + \langle \phi(x_j), \phi(x_j) \rangle - 2\langle \phi(x_i), \phi(x_j) \rangle
\end{aligned}$$

Which is a linear combination of inner products, as desired.

4.c.iv:
From ii and iii, we see:

$$\begin{aligned}
&\left|\left|\phi(x_i) - c_j\right|\right|^2 \\
&= \langle \phi(x_i), \phi(x_i) \rangle + \langle c_j, c_j \rangle - 2\langle \phi(x_i), c_j \rangle \\
&= \langle \phi(x_i), \phi(x_i) \rangle + \left\langle \sum_{l=1}^{n} \alpha_{lj}\phi(x_l), \sum_{l=1}^{n} \alpha_{lj}\phi(x_l) \right\rangle - 2\left\langle \phi(x_i), \sum_{l=1}^{n} \alpha_{lj}\phi(x_l) \right\rangle \\
&= \langle \phi(x_i), \phi(x_i) \rangle + \sum_{l,m} \alpha_{lj}\alpha_{mj}\langle \phi(x_l), \phi(x_m) \rangle - 2\sum_{l=1}^{n} \alpha_{lj}\langle \phi(x_i), \phi(x_l) \rangle
\end{aligned}$$

Which is a linear combination of inner products between data points, as desired.

4.c.v:
I split my answer into two parts – a theoretical version (that, in general, cannot be implemented) and a practical version.

In Theory:

Given k, $x_1, \dots x_n$, and some kernel ϕ. Assume we can compute inner products efficiency.

Randomly initialize centers $c_1, \dots, c_k$

```
While centers $c_1, \dots, c_k$ have not converged:
    For each $x_i$ (i=1,2,…,n):
        For each $j$ (j=1,2,…,k):
            Compute distances $\langle \phi(x_i), \phi(x_i) \rangle + \sum_{l,m} \alpha_{lj}\alpha_{mj}\langle \phi(x_l), \phi(x_m) \rangle - 2\sum_{l=1}^{n} \alpha_{lj}\langle \phi(x_i), \phi(x_l) \rangle$
        Assign $x_i$ to the cluster j with the smallest distance.

    For each j (j=1,2,…,k):
        Recompute $c_j = \sum_{i=1}^{n} \alpha_{ij}\phi(x_i)$ where $\alpha_{ij} = \frac{z_{ij}}{\sum_{k=1}^{n} z_{kj}}$
```

In practice:

Note that we generally cannot explicitly initialize the centers $c_1, \dots, c_k$ since these centers live in the same space as $\phi(x_i)$, which is potentially infinite-dimensional. So, in practice, we can do the following instead:

Given k, $x_1, \dots x_n$, and some kernel ϕ. Assume we can compute inner products efficiency.

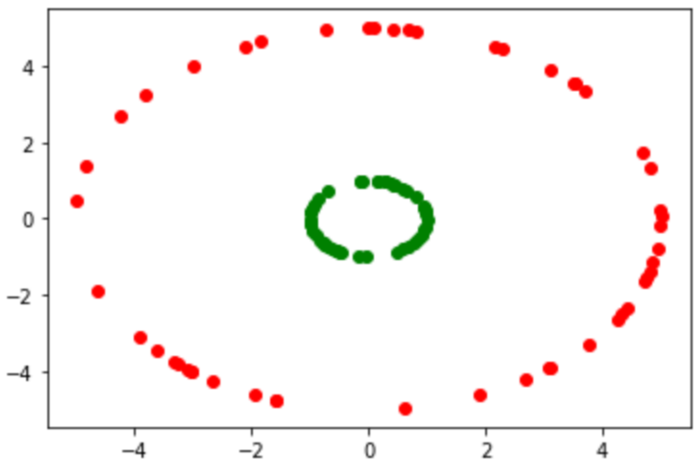
Randomly initialize cluster assignments $(\boldsymbol{x_1}, \boldsymbol{C_1}), (\boldsymbol{x_2}, \boldsymbol{C_2}), \dots, (\boldsymbol{x_n}, \boldsymbol{C_n})$.

```
While cluster assignments haven't converged:
    For each $x_i$ (i=1,2,…,n):
        For each $j$ (j=1,2,…,k):
            Compute distances $\langle\phi(x_i),\phi(x_i)\rangle + \sum_{l,m}\alpha_{lj}\alpha_{mj}\langle\phi(x_l),\phi(x_m)\rangle - 2\sum_{l=1}^{n}\alpha_{lj}\langle\phi(x_i),\phi(x_l)\rangle$
        Assign $x_i$ to the cluster j with the smallest distance.
```

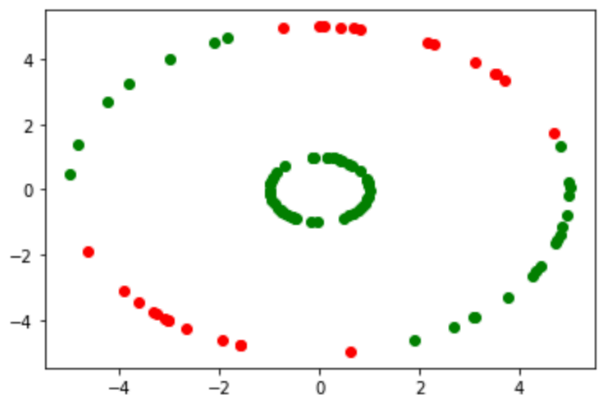
4.c.vi.
CODE SUBMITTED TO GRADESCOPE.

Concentric circles with RBF kernel. I used $2\sigma^2 = 9$:

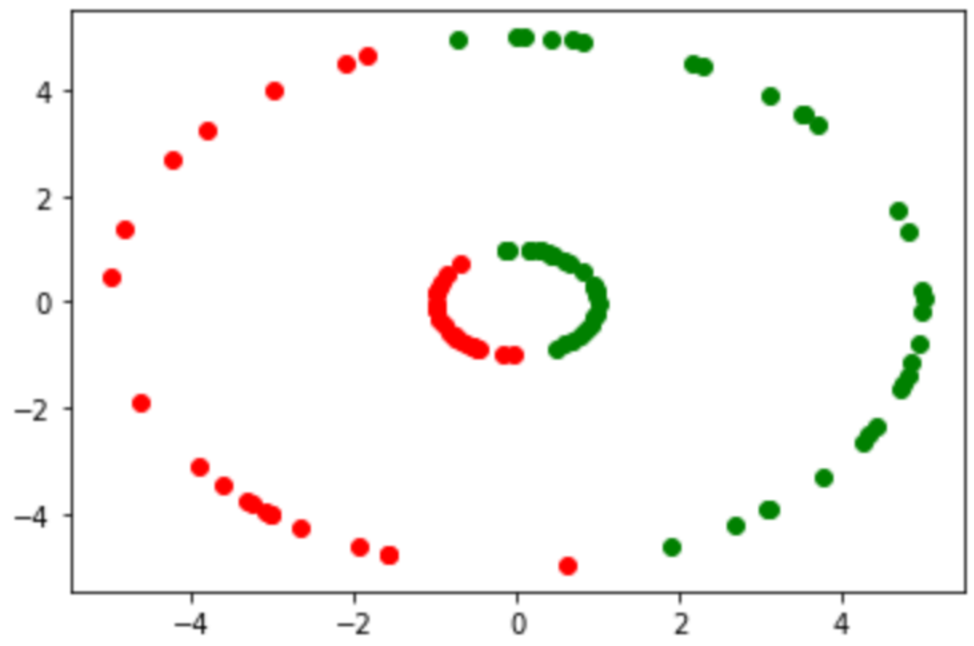
$$K(x_i, x_j) = \exp\left(-\frac{\left\|x_i - x_j\right\|^2}{2\sigma^2}\right)\exp\left(-\frac{\left\|x_i - x_j\right\|^2}{9}\right)$$



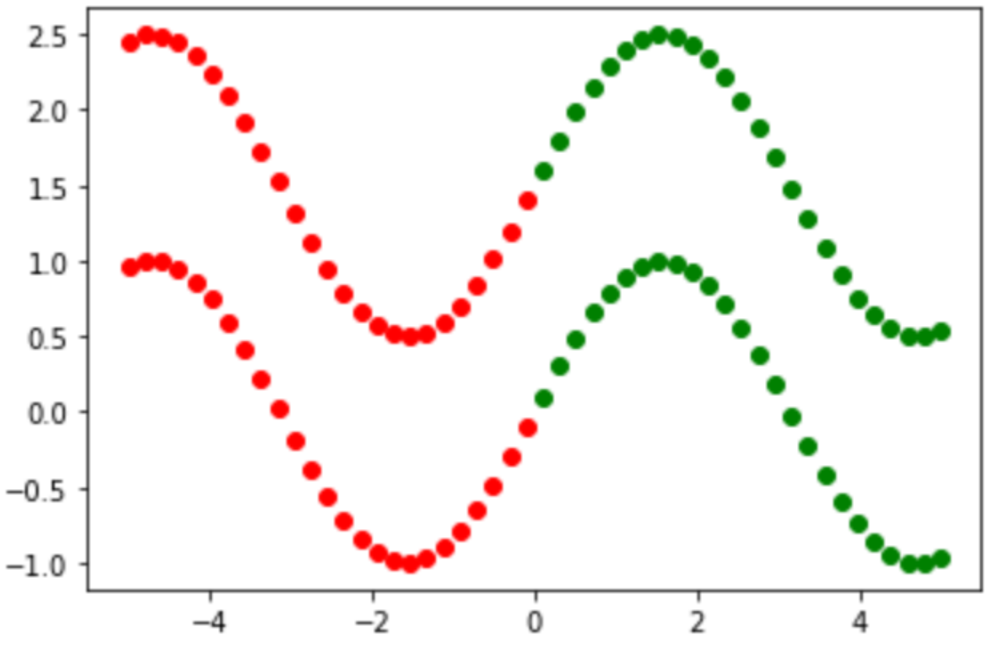
Concentric circles with quadratic kernel:



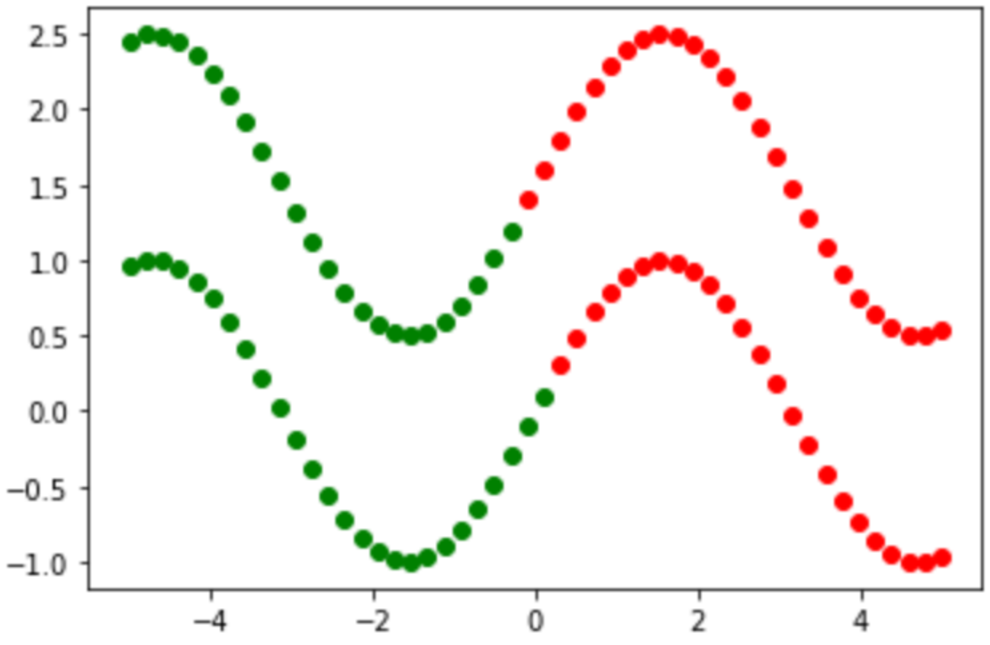
Concentric circles with linear kernel:



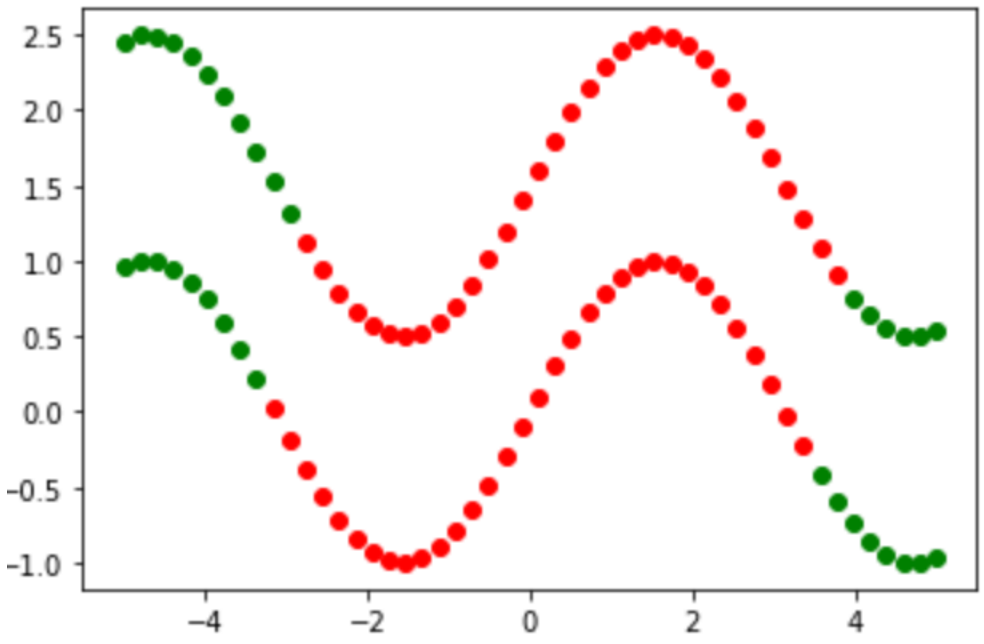
Sinusoids with linear kernel:



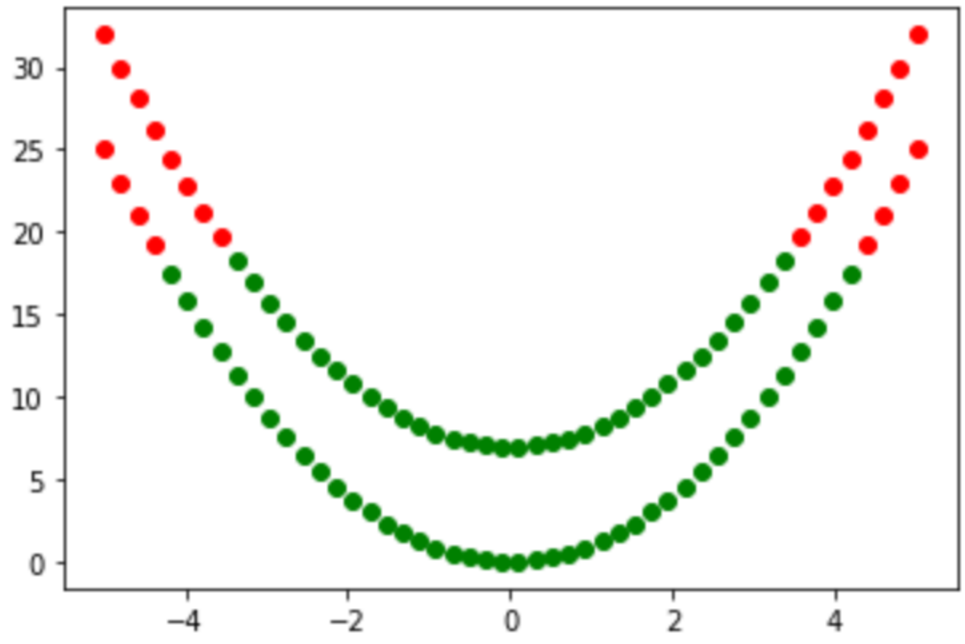
Sinusoids with RBF kernel:



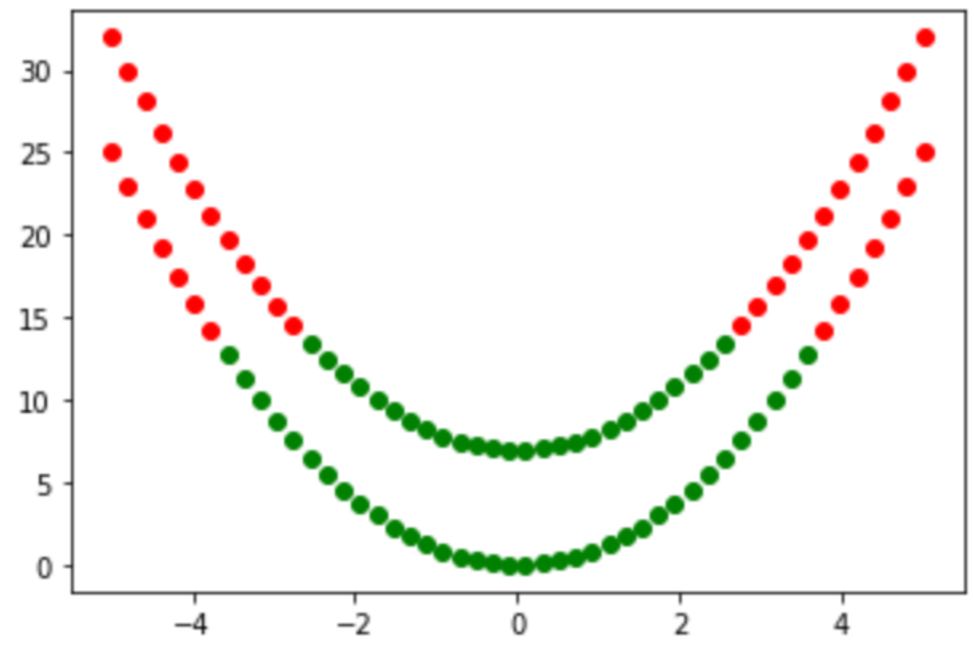
Sinusoids with quadratic kernel:



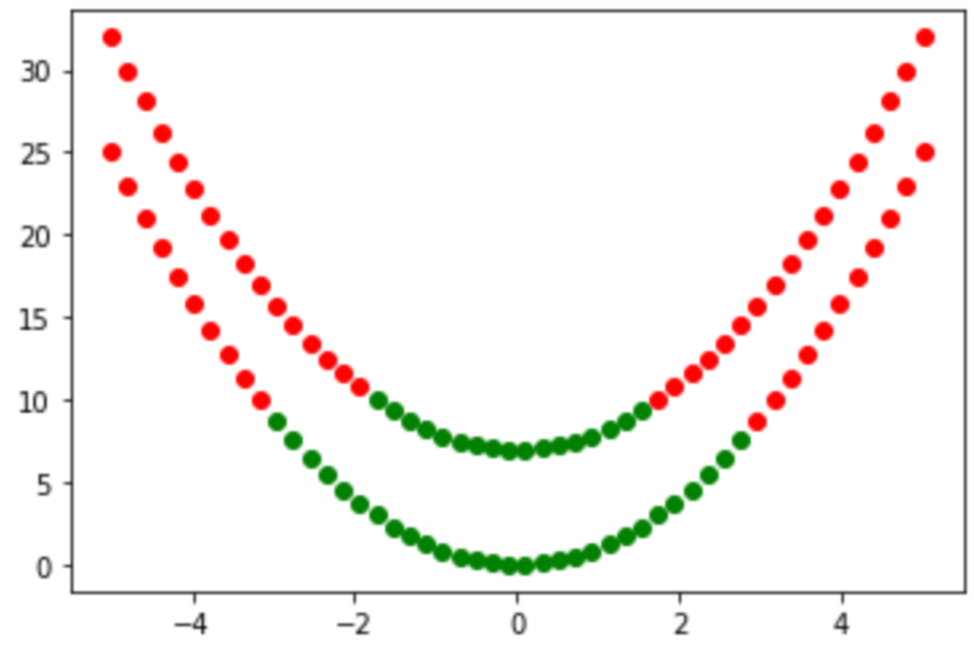
Quadratics with quadratic kernel:



Quadratics with linear kernel:



Quadratics with RBF kernel:



Conclusion:

It turns out that the RBF, quadratic, and linear kernels are simply unable to map the quadratic and sinusoidal datasets onto a space where they are linearly separable. The RBF kernel, however, does do very well for concentric circles when we set $2\sigma^2 = 9$.