CSE 382M: Found. Data Science & ML **Spring 2025**

Problem Set 3

Student Name: Noah Reef

Problem 1

Part a

The formula for $\Phi_X(P)$ is given by:

$$\Phi_X(P) = \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_i||_2^2$$

note that the inside term $\sum_{x_j \in C_i} ||x_j - \mu_i||_2^2$ can be written as

$$\begin{aligned}
\sum_{x_j \in C_i} ||x_j - \mu_i||_2^2 &= \sum_{x_j \in C_i} \left(||x_j||_2^2 + ||\mu_i||_2^2 - 2x_j \cdot \mu_i\right) \\
&= \sum_{x_j \in C_i} ||x_j||_2^2 + |C_i| \, ||\mu_i||_2^2 - 2|C_i|\mu_i \cdot \mu_i \\
&= \sum_{x_j \in C_i} ||x_j||_2^2 + |C_i| \, ||\mu_i||_2^2 - 2|C_i| \, ||\mu_i||_2^2 \\
&= \sum_{x_j \in C_i} ||x_j||_2^2 - |C_i| \, ||\mu_i||_2^2
\end{aligned}$$

and we note that

$$\begin{aligned}
\sum_{j,\ell \in C_i} ||x_j - x_\ell||_2^2 &= \sum_{j,\ell \in C_i} \left(||x_j||_2^2 + ||x_\ell||_2^2 - 2x_j \cdot x_\ell\right) \\
&= |C_i| \sum_{j} ||x_j||_2^2 + |C_i| \sum_{\ell} ||x_\ell||_2^2 - 2|C_i|^2 \, ||\mu_i||_2^2 \\
&= 2|C_i| \left(\sum_{j} ||x_j||_2^2\right) - 2|C_i|^2 \, ||\mu_i||_2^2
\end{aligned}$$

and so we see that

$$\sum_{x_j \in C_i} ||x_j - \mu_i||_2^2 = \frac{1}{2|C_i|} \sum_{j,\ell \in C_i} ||x_j - x_\ell||_2^2$$

so we can rewrite $\Phi_X(P)$ as

$$\Phi_X(P) = \sum_{i=1}^{k} \frac{1}{2|C_i|} \sum_{j,\ell \in C_i} ||x_j - x_\ell||_2^2$$

Now note that for each $x_i, x_j \in X$, we have that

$$\sqrt{r}(1-\epsilon)\left|\left|x_i - x_j\right|\right|_2 \le \left|\left|Gx_i - Gx_j\right|\right|_2 \le \sqrt{r}(1+\epsilon)\left|\left|x_i - x_j\right|\right|_2$$

by JLT and hence summing over all appropiate indicies, yields

$$\sqrt{r}(1-\epsilon)\Phi_X(P) \le \Phi_Y(P) \le \sqrt{r}(1+\epsilon)\Phi_X(P)$$

for all partitions P. Then clearly

$$\sqrt{r}(1-\epsilon)\Phi_X(\tilde{P}) \le \Phi_Y(\tilde{P}) \le \Phi_Y(P^*) \le \sqrt{r}(1+\epsilon)\Phi_X(P^*)$$

and get

$$\Phi_X(\tilde{P}) \le \frac{1+\epsilon}{1-\epsilon}\Phi_Y(\tilde{P}) \approx O(1+\epsilon)\Phi_X(P^*)$$

Part b

Let $M \in \mathbb{R}^{n\times k}$ matrix defined by

$$M_{ij} = \begin{cases} \frac{1}{\sqrt{|C_j|}} & \text{if } i \in C_j \\ 0 & \text{otherwise} \end{cases}$$

Then we see that

$$(MM^T)_{\ell j} = \begin{cases} \frac{1}{|C_j|} & \text{if } \ell \in C_j \\ 0 & \text{otherwise} \end{cases}$$

Therefore

$$(XMM^T)_i = \sum_{\ell=1}^{n} x_\ell (MM^T)_{\ell i} = \sum_{\ell \in C_j} \frac{x_\ell}{|C_j|} = \mu_i$$

then we see that

$$\left|\left|X - XMM^T\right|\right|_F^2 = \sum_{i=1}^{d}\sum_{j=1}^{n}(x_j - (XMM^T)_j)^2 = \sum_{i=1}^{d}\sum_{x_j \in C_i}\left|\left|x_j - \mu_i\right|\right|_2^2 = \Phi_X(P)$$

note that since M has at most rank k, we have that the rank of XMM^T is at most k, therefore if $X_k = U_k S_k V_k^T$ is the best rank k approximation of X, then we have that

$$\left|\left|X - X_k\right|\right|_2^2 \le \left|\left|X - X_k\right|\right|_F^2 \le \left|\left|X - XMM^T\right|\right|_F^2 = \Phi_X(P)$$

Let $\tilde{M}$ be the matrix corresponding to the partition $\tilde{P}$ and M^* be the matrix corresponding to the parition P^*, then we have that

$$\begin{aligned}
\Phi_X(\tilde{P}) = \left|\left|X - X\tilde{M}\tilde{M}^T\right|\right|_F^2 &= \left|\left|X_k - X\tilde{M}\tilde{M}^T\right|\right|_F^2 + \left|\left|X - X_k\right|\right|_F^2 \\
&\le \left|\left|X_k - XM^*(M^*)^T\right|\right|_F^2 + \left|\left|X - X_k\right|\right|_F^2 \\
&\le \Phi_X(P^*) + \Phi_X(P^*) \\
&= 2\Phi_X(P^*)
\end{aligned}$$

Problem 2

Let $\omega \sim \mathcal{N}(0, I_d)$, $\beta \sim U(0,1)$, $\epsilon = 1, c = 2$, and $w = 4\epsilon$. Suppose we have the hash function $h : \mathbb{R}^d \to \mathbb{Z}$ defined by

$$h(x) = \left\lfloor \frac{\omega \cdot x}{w} + \beta \right\rfloor$$

define $z = (x - y) \cdot \omega$, then we see that $z \sim \mathcal{N}(0, ||x - y||_2^2)$. Let

$$u = \frac{\omega \cdot x}{w} + \beta$$
$$v = \frac{\omega \cdot y}{w} + \beta$$

then we have that if $\lfloor u \rfloor = \lfloor v \rfloor$, then

$$\lfloor v \rfloor = \left\lfloor v + \frac{z}{w} \right\rfloor \implies \left| \frac{z}{w} \right| < 1$$

and hence given a fixed z, we have that

$$\Pr[h(x) = h(y)|z] = 1 - \left| \frac{z}{w} \right|$$

and we compute the probability as

$$\begin{aligned}
\Pr[h(x) = h(y)] &= \int_{-\infty}^{\infty} \Pr[h(x) = h(y)|z]p(z)dz = \int_{-\infty}^{\infty} \left(1 - \left|\frac{z}{w}\right|\right) \frac{1}{r\sqrt{2\pi}} e^{-\frac{z^2}{2r^2}} dz \\
&= 2\int_0^w \left(1 - \frac{z}{w}\right) \frac{1}{r\sqrt{2\pi}} e^{-\frac{z^2}{2r^2}} dz \\
&= 2\int_0^4 \left(1 - \frac{z}{4}\right) \frac{1}{r\sqrt{2\pi}} e^{-\frac{z^2}{2r^2}} dz
\end{aligned}$$

where $r = ||x - y||_2$. For the case of $r = 1$, we have that

$$\Pr[h(x) = h(y)] = 2\int_0^4 \left(1 - \frac{z}{4}\right) \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz \approx 0.800352 = p_1$$

and for the case of $r = 2$, we have that

$$\Pr[h(x) = h(y)] = 2\int_0^4 \left(1 - \frac{z}{4}\right) \frac{1}{2\sqrt{2\pi}} e^{-\frac{z^2}{8}} dz \approx 0.6095 = 1 - p_2$$

```
import numpy as np


def h(x):
    beta = np.random.uniform(0,1)
    omega = np.random.randn(x.shape[0],1)
    epsilon = 1
    w = 4*epsilon
    return np.floor((np.dot(x.T,omega)[0,0]/w) + beta)

def main():

    num_trials = int(1e6)
    num_collisions = 0
    dim = 3
    p1_count = 0
    p2_count = 0

    beta = np.random.uniform(0,1)
    omega = np.random.randn(x.shape[0],1)
    epsilon = 1
    w = 4*epsilon

    for i in range(num_trials):
        x = np.random.randn(dim,1)
        y = np.random.randn(dim,1)

        while np.linalg.norm(x-y) > 1:
            x = np.random.randn(3,1)
            y = np.random.randn(3,1)

        if h(x) == h(x):
            num_collisions += 1

        p1_count += 1

    print("Probability of collision for ||x-y||_2 <= 1: ", num_collisions/
p1_count)

    num_collisions = 0

    for i in range(num_trials):
        x = np.random.randn(dim,1)
        y = np.random.randn(dim,1)

        while np.linalg.norm(x-y) < 2:
            x = np.random.randn(3,1)
            y = np.random.randn(3,1)

        if h(x) == h(x):
            num_collisions += 1

        p2_count += 1

    print("Probability of collision ||x-y|| >= 2: ", num_collisions/
p2_count)



main()
```

which gives us the following results

```
Probability of collision for ||x-y||_2 ⩽ 1:  0.627369
Probability of collision ||x-y|| ⩾ 2:  0.501816
```

Figure 3.2. Output of Python code

Problem 3

Let $X \in \mathbb{R}^{d\times n}$ and define the kernel matrix as $K_{ij} = k(x_i, x_j)$ for kernel function k. We define the diagonal matrix D as $D_{ii} = \sum_{j=1}^n K_{ij}$ and get the Laplacian of the graph as $L = D - K$. Then the normalized Laplacian is computed as

$$D^{-1/2}LD^{-1/2} = I - D^{-1/2}KD^{-1/2} \implies D^{-1/2}KD^{-1/2} = I - D^{-1/2}LD^{-1/2}$$

then if $y = D^{1/2}x$, where x is an eigenvector of L, then we have that

$$D^{-1/2}LD^{-1/2}y = D^{-1/2}LD^{-1/2}D^{1/2}x = D^{-1/2}Lx = D^{-1/2}(\lambda x) = \lambda y$$

and

$$D^{-1/2}KD^{-1/2}y = (1-\lambda)y$$

thus the eigenvectors of $D^{-1/2}KD^{-1/2}$ are the same as the eigenvectors of $D^{-1/2}LD^{-1/2}$, but are reversed in order. Now since the row sum of L is always zero, we have that the lowest eigenvalue is 0 with corresponding eigenvector of all 1's and hence we take the first 2 eigenvectors of L, corresponding to the two lowest singular vectors of L. We then use thses singular vectors to construct a projection matrix that maps into the space spanned by these two vectors, and then apply k-means to this projected data to form our clusters of $k = 2$, thus we are choosing the second largest eigenvector of $D^{-1/2}KD^{-1/2}$.

Problem 4

Let $X \subseteq \mathbb{R}^3$, and suppose that k_1 and k_2 are kernel functions on X, then we have that if $k(x, y) = k_1(x, y)k_2(x, y)$, then we have that

$$k(x, y) = k_1(x, y)k_2(x, y) = k_1(y, x)k_2(y, x) = k(y, x)$$

and

$$\sum_{i,j} k(x_i, x_j)c_ic_j = \sum_{i,j} k_1(x_i, x_j)k_2(x_i, x_j)c_ic_j = \sum_{i,j} k_1(x_i, x_j)\phi_2(x_i)c_i\phi_2(x_j)c_j = \sum_{i,j} k_1(x_i, x_j)d_id_j \geq 0$$

Which is true since k_1 is a kernel function. Now if $k(x, y) = k_1(x, y) + k_2(x, y)$, then we have that

$$k(x, y) = k_1(x, y) + k_2(x, y) = k_1(y, x) + k_2(y, x) = k(y, x)$$

and

$$\sum_{i,j} k(x_i, x_j)c_i c_j = \sum_{k_1}(x_i, x_j)c_i c_j + \sum_{k_2}(x_i, x_j)c_i c_j \geq 0$$

since k_1 and k_2 are kernel functions. Lastly, suppose that $k(x, y) = f(x)f(y)$, for $f : X \to \mathbb{R}$, then we have that

$$k(x, y) = f(x)f(y) = f(y)f(x) = k(y, x)$$

and

$$\sum_{i,j} k(x_i, x_j)c_i c_j = \sum_{i,j} f(x_i)f(x_j)c_i c_j = \left(\sum_i f(x_i)c_i\right)^2 \geq 0$$

thus k is a kernel function.

Problem 5

First recall for kernel k-means we have that

$$||\phi(x_i) - \mu_C||_2^2 = k(x_i, x_i) + \frac{1}{n_C^2}\sum_{\ell,j\in C} k(x_\ell, x_j) - \frac{2}{n_C}\sum_{j\in C} k(x_i, x_j)$$

then for a given iteration and single data point x_i, we have that worst-case complexity is given by $O(dn^2)$, since if we consider the case where all points are allocated to a single cluster, then computing $k(\mu_C, \mu_C)$ would require n^2 computations of k with complexity of $O(d)$. Then if we sum over all x_i in our data set, we get that our overall complexity for a single iteration is $O(dn^3)$. For the case where K is a Nystrom approximation of rank k, we have that the complexity of computing K is $O(dnk)$, and hence the complexity of a single iteration is $O(dnk^2)$.