Name: Pragya Shukla

Roll No: MDS202027

TOPOLOGICAL DATA ANALYSIS

ASSIGNMENT 2
DUE DATE: 15TH FEBRUARY

Write an interactive Python code that does the following job.

(1) Let n be a positive integer bigger equal 3.

(2) Let X_n be the following set of n-tuples of points in $\mathbb{R}^2$

$$X_n = \{(x_1, \ldots, x_n) \in (\mathbb{R}^2)^n \mid ||x_i|| = 1, \forall i, x_n = (0,-1)^T \text{ and } \sum x_i = (0,0)^T\}.$$

Write a piece of code that will genarate, say 200, random points of X_n.

(3) The point cloud generated above is your data set.

(4) Compute the persistent homology of the generated point cloud (in dimensions $0, 1, 2$) and express it in terms of a persistence diagram.

(5) Your submission should be the code (along with intermediate explainatory notes at appropriate places) in a notebook. Also prepare a table of values $n, \beta_0(X_n), \beta_1(X_n)$ and $\beta_2(X_n)$ for n at least 6 or 7.

Import libraries

In [1]:
```
#!pip install gudhi
```

In [2]:
```
import numpy as np
import gudhi as gd
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
```

(1) Let n be a positive integer bigger equal 3.
(2) Let X_n be the following set of n-tuples of points in $\mathbb{R}^2$

$$X_n = \{(x_1, \ldots, x_n) \in (\mathbb{R}^2)^n \mid ||x_i|| = 1, \forall i, x_n = (0, -1)^T \text{ and } \sum x_i = (0, 0)^T\}.$$

Write a piece of code that will genarate, say 200, random points of X_n.

In [3]:
```
n = 4   # n>3
N = 200 # N = 200
```

In [4]:
```
def generate_function(n):
    points = np.random.uniform(-1, 1, (n-2, 2))
    points = np.vstack([points, np.array([[0, -1]])])
    points /= np.linalg.norm(points, ord=2, axis=-1, keepdims=True)
    points = np.vstack([points, -points.sum(axis=0)])
    return points
point_cloud = lambda N, n: np.array([generate_function(n) for _ in range(N)])
```

In [5]:
```
point_cloud_new = point_cloud(N, n)
point_cloud = point_cloud_new.reshape(N, -1)
```

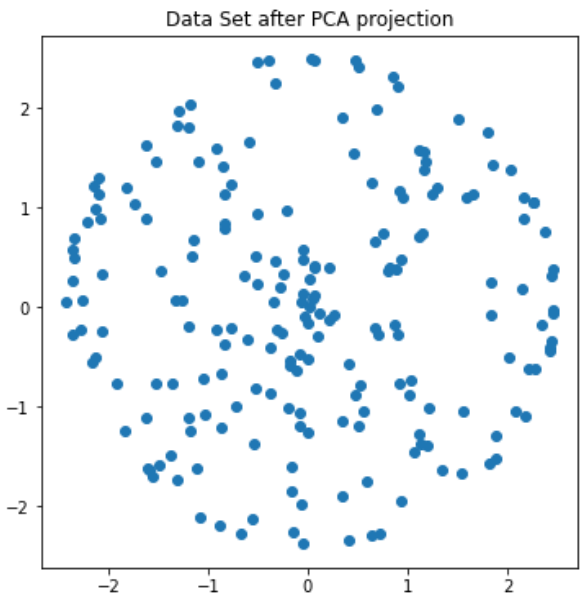
(3) The point cloud generated above is your data set.

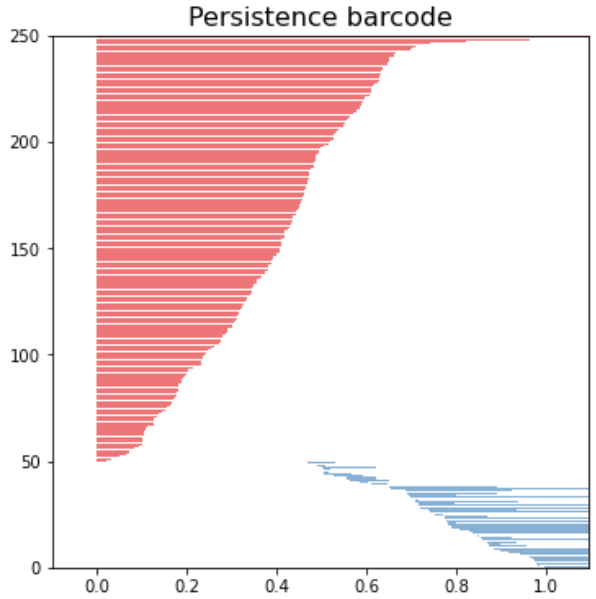
(4) Compute the persistent homology of the generated point cloud (in dimensions $0, 1, 2$) and express it in terms of a persistence diagram.

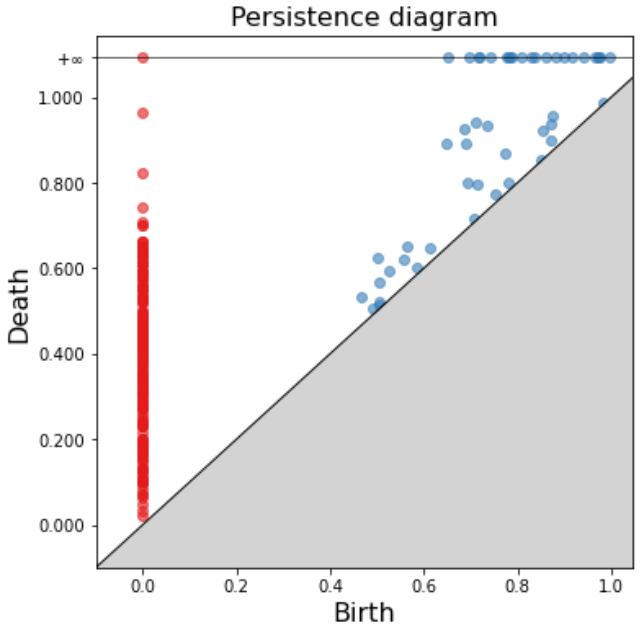
In [6]:
```
def persistence_diag(length, max_dimension=2):
    rc = gd.RipsComplex(points=point_cloud, max_edge_length=length)
    tree = rc.create_simplex_tree(max_dimension=max_dimension)
    diag = tree.persistence()
    data = PCA(n_components=3).fit_transform(point_cloud)
    fig, ax = plt.subplots(1, 3, figsize=(20, 6))
    # Plot 1
    ax[0].scatter(data[:, 0], data[:, 1])
    ax[0].set_title("Data Set after PCA projection")
    # Plot 2
    gd.plot_persistence_barcode(diag, axes=ax[1])
    # Plot 3
    gd.plot_persistence_diagram(diag, axes=ax[2])
    plt.show()
```

In [7]:
```
persistence_diag(length=1.0)
```

```
usetex mode requires TeX.
```







(5) Your submission should be the code (along with intermediate explainatory notes at appropriate places) in a notebook. Also prepare a table of values n, $\beta_0(X_n)$, $\beta_1(X_n)$ and $\beta_2(X_n)$ for n at least 6 or 7.

In [8]:

```
class Betti_Number:
    def __init__(self, N, n):
        self.point_cloud = point_cloud

    def tree(self, max_edge_length, max_dimension=2):
        rc = gd.RipsComplex(points=self.point_cloud, max_edge_length=max_edge_length)
        self.tr = rc.create_simplex_tree(max_dimension=max_dimension)
        self.diag = self.tr.persistence()

    def betty_numbers(self, homology_coeff_field=2, max_betty_num=3):
        self.tr.compute_persistence(homology_coeff_field=homology_coeff_field)
        b_nums = self.tr.betti_numbers()
        if len(b_nums) >= max_betty_num:
            return  b_nums[:max_betty_num]
        return b_nums + [0]*(max_betty_num - len(b_nums))
```

In [9]:
```
obj = Betti_Number(N = 200, n = 4)
obj.tree(max_edge_length = 1.0)

for i, y in enumerate(obj.betty_numbers()):
    print(f"B_{i} = {y}")
```

```
B_0 = 1
B_1 = 22
B_2 = 0
```

In [10]:
```
list1 = []
for n in range(3, 8):   # to ensure n>=3 and n goes atleast till 6 or 7.
    obj = Betti_Number(N = 200, n = n)
    obj.tree(max_edge_length = 3)
    row = obj.betty_numbers()
    row.insert(0, n)
    list1.append(row)
betti_no_dataframe = pd.DataFrame(list1, columns=['n', 'B0', 'B1', 'B2'])
```

In [11]:
```
betti_no_dataframe
```

Out[11]:

	n	B0	B1	B2
0	3	1	0	0
1	4	1	0	0
2	5	1	0	0
3	6	1	0	0
4	7	1	0	0

Above are the Betti numbers for different values of n.