

Final Project using Ripser

Manu Samala

May 3, 2021

0.1 Objective

The objective is to calculate the barcode for PH_1 of the torus in \mathbb{R}^4 , or more commonly referred to as the Clifford Torus.

0.2 Methodology

To calculate the persistence homology of the \mathbb{R}^4 , we are to utilize software called Ripser.py that calculates the Vietoris-Rips barcodes. We shall utilize python with numpy to generate 30 unit vectors with varying angle values θ and ϕ with the condition shown in eq. 1. Then we shall utilize Ripser.py to calculate the persistent homology of the dataset.

$$\mathbb{S}^1 \times \mathbb{S}^1 = (\cos\theta, \sin\theta, \cos\phi, \sin\phi \mid 0 \leq \theta \leq 2\pi, 0 \leq \phi \leq 2\pi) \quad (1)$$

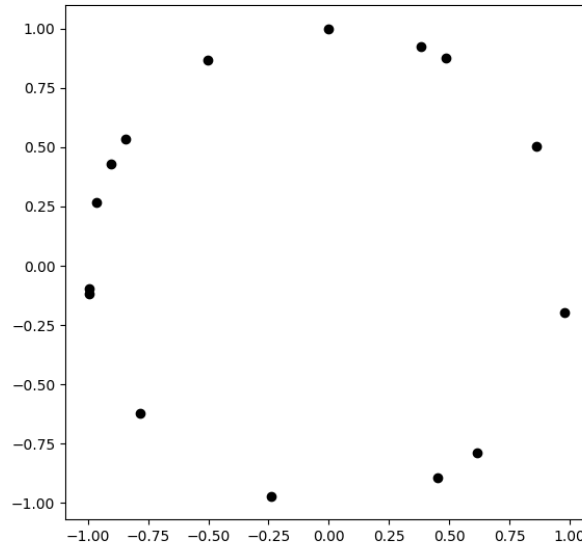


Figure 1: 15 randomized datapoints on \mathbb{R}^2 as a function of θ .

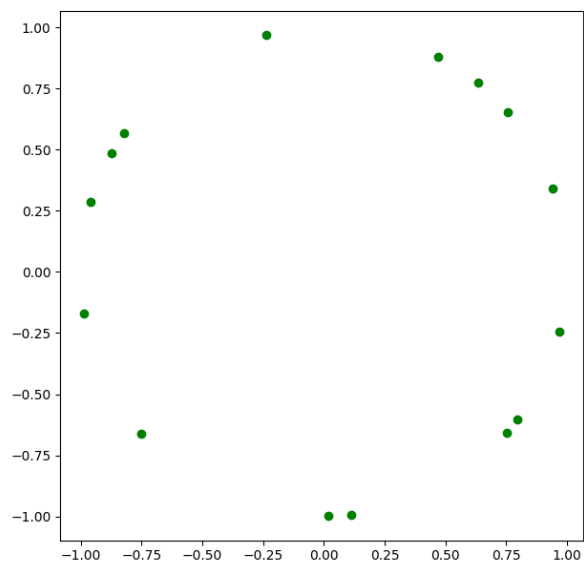



Figure 2: 15 randomized datapoints on \mathbb{R}^2 as a function of ϕ .



```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 ## Generate circles with varying angles
5 ## output is csv file
6
7 N = 15 # number of samples
8 r = 1 # radius
9 u = 2 * np.pi # upper bound
10 T = np.zeros((N, 4)) #Torus numpy array
11 ## angles t(theta), p(phi)
12 t = np.random.uniform(0, u, N)
13 s = np.random.uniform(0, u, N)
14 T[:, 0] = r * np.cos(t)
15 T[:, 1] = r * np.sin(t)
16 T[:, 2] = r * np.cos(p)
17 T[:, 3] = r * np.sin(p)
18
19 np.savetxt("torus.csv", T, delimiter=",")
20
21 plt.scatter(T[:, 0], T[:, 1], color='purple')
22 plt.scatter(T[:, 2], T[:, 3], color='r')
23 plt.show()

```

Figure 3: gen.py: Generate 30 datapoints in \mathbb{R}^4 with varying angles where 15 samples are generated for each \mathbb{R}^2 space. T is a numpy array where rows 0 and 1 are one \mathbb{R}^2 space and rows 2 and 3 are another \mathbb{R}^2 space. The output is a chart and datafile "torus.csv".



```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from mpl_toolkits.mplot3d import Axes3D
4 from ripser import ripser
5 from persim import plot_diagrams
6
7 ## Objective: Import data from csv and calculate
8 ## > persistent homology
9
10 T = np.genfromtxt('torus.csv', delimiter=',') ## Import csv
11 d = 1.5
12
13 ## vary threshold, it is the diameter
14 diagram = ripser(T, maxdim=1, thresh=d)['dgms']
15 plot_diagrams(diagram, show=True)
```

Figure 4: persist.py: Take datafile from gen.py and calculate the 0^{th} and 1^{st} persistent homology of an \mathbb{R}^4 torus. The output is a persistence diagram as seen in Fig. 5 through Fig. 11.

0.3 Results

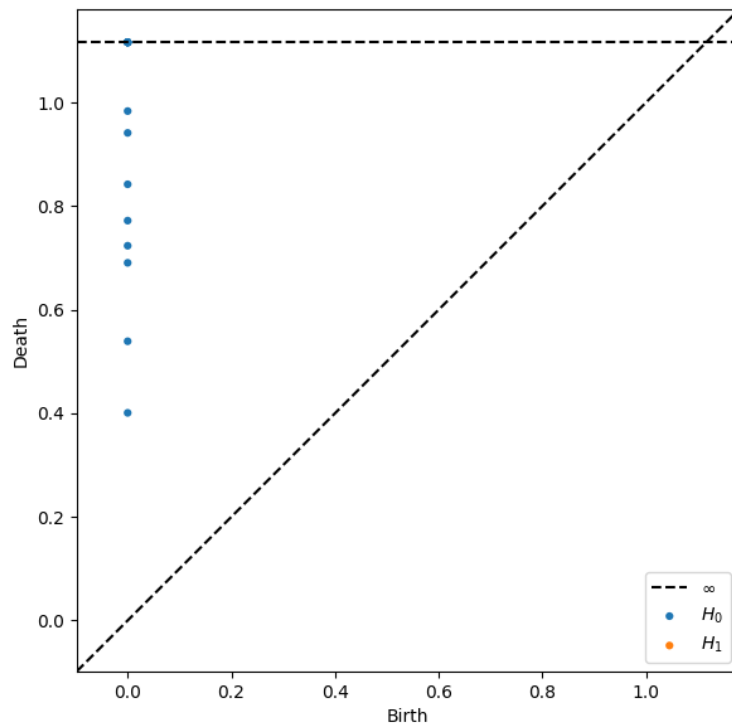


Figure 5: 0^{th} and 1^{st} persistent homology with radius=0.5.

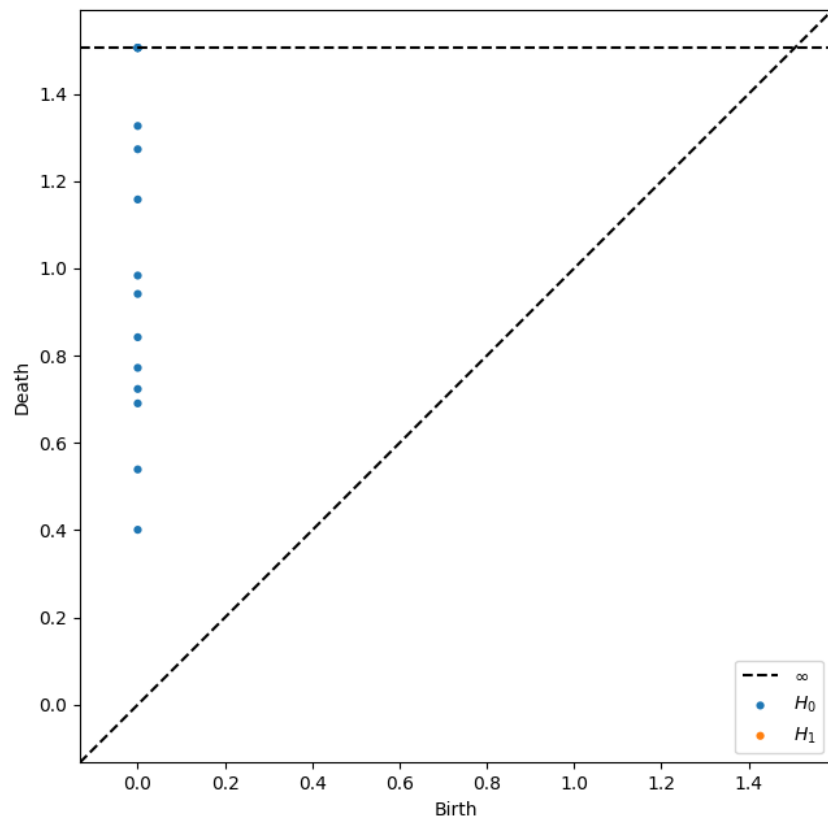


Figure 6: 0^{th} and 1^{st} persistent homology with radius=0.75.

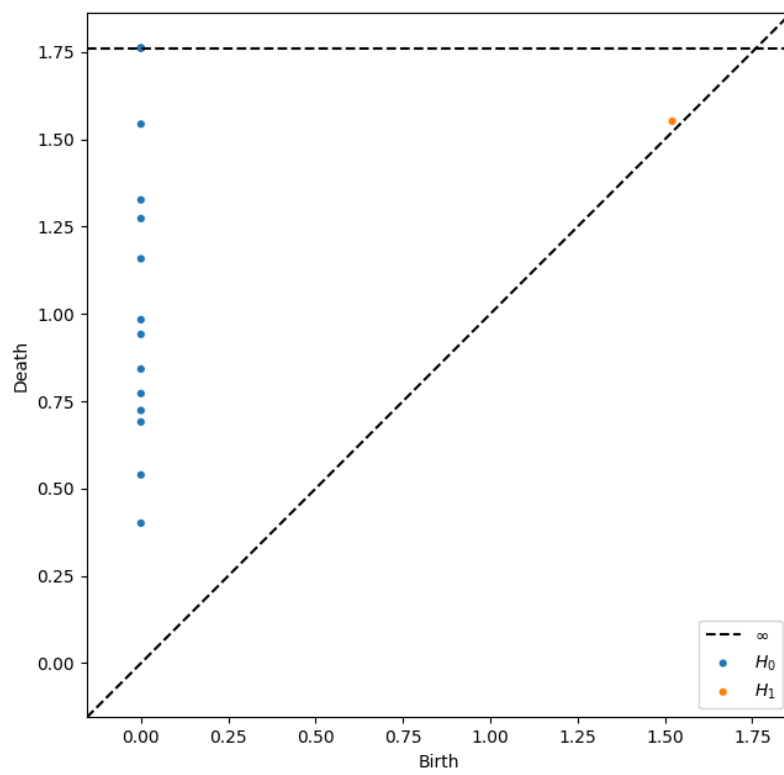


Figure 7: 0^{th} and 1^{st} persistent homology with radius=0.78.

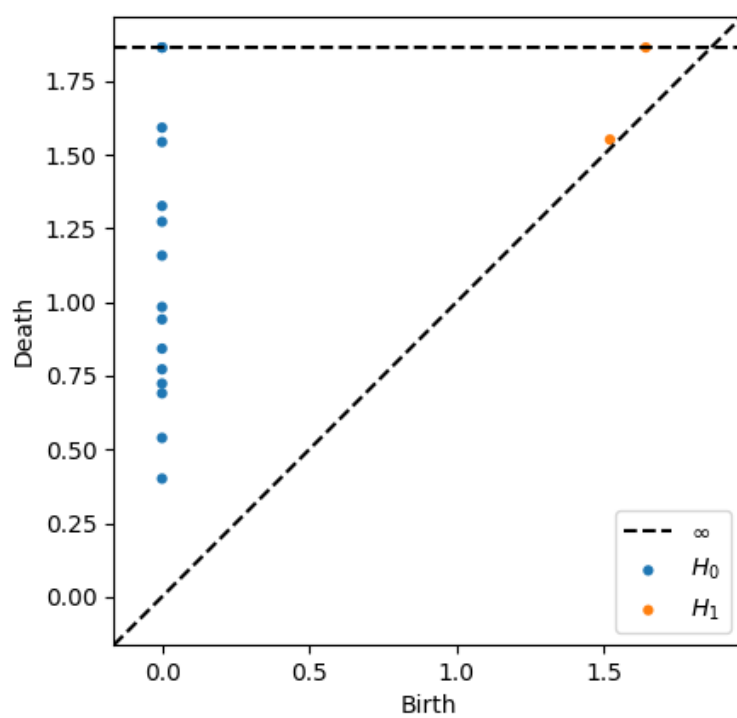


Figure 8: 0^{th} and 1^{st} persistent homology with radius=0.85.

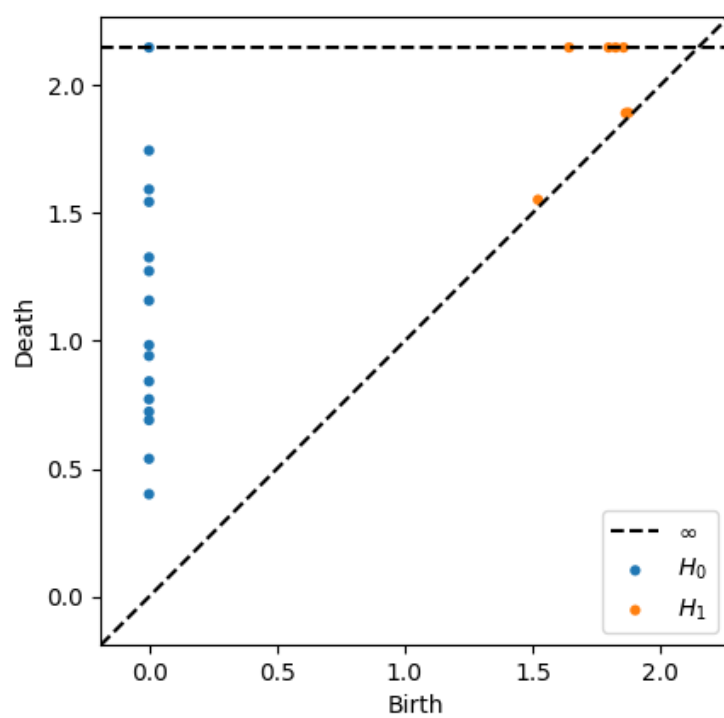


Figure 9: 0^{th} and 1^{st} persistent homology with radius=0.95.

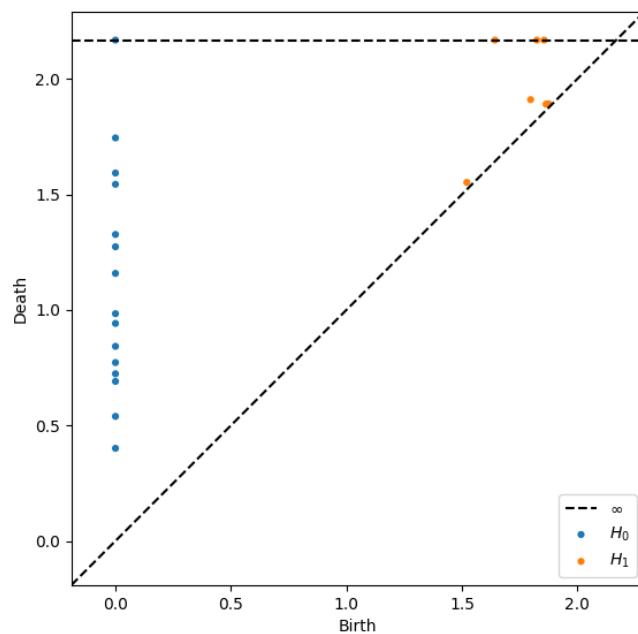


Figure 10: 0^{th} and 1^{st} persistent homology with radius=1.0.

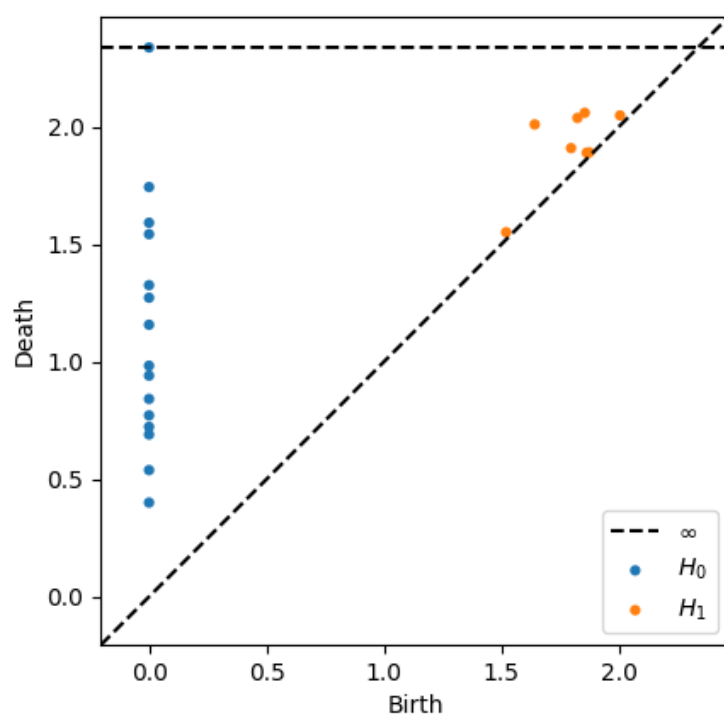


Figure 11: 0^{th} and 1^{st} persistent homology with radius=1.1.

0.4 Analysis

We have estimated the dimension of the 1st persistent homology to be equal to 2. As seen in Fig. 5 and Fig. 6 with a radius less than 0.78, none of the points are connected, and we can not make any conclusions about the topology of the dataset besides the fact that they are disconnected. In Fig. 7 through Fig. 8, we detect a little bit of noise in this dataset. Most likely, this is from the lack of points in this dataset since one of the best ways to reduce noise is to greatly increase the amount of points tenfold. In Fig. 9 and Fig. 10 with a radius greater than or equal to 0.95, we find the persistent homology of the dataset where we have estimated the dimension of the 1st persistent homology to be equal to 2. Finally at Fig. 11, the distance between points is too high with a radius of 1.1 and any features detected from this point is trivial.