Demo-PCA

February 7, 2020

[1]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

[2]:
```
from Python_code import examples as eg
import numpy as np
from numpy import *
import dionysus
```

The circular coordinates pipeline for examining different smoothness cost-functions:

```
Step 1. Getting the point cloud
Step 2. Computing the Vietoris-Rips filtration and its cohomology
Step 3. Selecting the Cocycle
Step 4. First smoothing using Least Squares (Optional)
Step 5. Second smoothing using a new cost function
```

0.1 Perform PCA (with different components) and compute the bottleneck distance between the PCA+CC persistent diagrams and the CC persistent diagrams above.

[3]:
```
import numpy as np
from Python_code import PCAtool as PCAtool
X=np.random.rand(300,2)
print(X)
X_new=PCAtool.pca(X,K=2)
import matplotlib.pyplot as plt
X_cen=X-np.mean(X,axis=0)
plt.scatter(X_cen[:,0], X_cen[:,1])
plt.scatter(X_new[:,0], X_new[:,1])
plt.show()
```

```
[[8.48141350e-01 5.91122008e-01]
 [5.31436310e-01 2.43213398e-01]
 [5.27922545e-02 1.97619252e-01]
 [3.75610841e-01 1.81404888e-01]
 [2.29323441e-01 9.08750886e-01]
 [6.81797283e-01 3.22961919e-01]
```

```
[4.70108448e-01 2.71129849e-01]
[2.54643050e-01 6.64800274e-01]
[6.11247141e-01 2.75887820e-02]
[2.73996954e-01 2.53968500e-01]
[9.33215848e-01 5.92169672e-01]
[4.23109570e-02 7.51093804e-01]
[1.49921863e-01 3.23487535e-01]
[6.83144592e-01 7.56521992e-03]
[9.34487000e-01 8.96733653e-01]
[4.84946225e-01 7.55841809e-01]
[6.96879292e-01 1.13879895e-02]
[3.82419922e-01 1.49896113e-01]
[1.27803105e-01 7.59223579e-01]
[8.45810700e-01 1.47600482e-01]
[5.19067340e-01 6.40215610e-01]
[3.90989066e-01 8.04568989e-01]
[2.81833363e-01 8.85366082e-01]
[6.69233921e-01 2.98863759e-01]
[7.84221348e-01 5.39541792e-01]
[7.78285966e-02 4.65502250e-01]
[8.72253070e-01 4.91712814e-01]
[5.24459489e-01 7.60544475e-01]
[9.82031814e-01 6.07326326e-01]
[1.90689468e-02 1.91466288e-01]
[5.90248842e-01 9.95469508e-01]
[6.27470424e-01 1.82545973e-01]
[7.55542251e-01 2.38947978e-01]
[2.71022640e-01 3.75245757e-01]
[1.21086283e-01 6.53042146e-01]
[5.59112984e-01 7.64261890e-01]
[6.40917365e-01 3.98631721e-01]
[7.91917263e-01 6.66032411e-01]
[4.52395761e-01 8.42085329e-01]
[2.58968926e-01 1.11069240e-01]
[1.70011631e-01 1.56629797e-01]
[8.26003323e-01 2.96180547e-01]
[4.56856036e-01 5.94146809e-01]
[5.04276478e-01 5.42067269e-01]
[4.39412472e-02 2.52716923e-01]
[8.93963616e-01 1.74218868e-01]
[4.08320810e-01 4.56586514e-02]
[8.89085682e-01 7.16474702e-01]
[2.76559474e-01 7.14472152e-01]
[9.92754096e-01 8.15606015e-01]
[9.95146190e-01 8.10103234e-01]
[5.18339989e-01 4.68537496e-01]
[5.21854008e-01 8.14103271e-01]
[3.65136432e-01 2.14956010e-02]
```

```
[7.37586602e-01 4.72429085e-01]
[6.99166136e-02 1.41242283e-01]
[7.91881591e-01 4.52400150e-01]
[5.26042993e-01 8.65426768e-01]
[8.23066693e-01 8.89292298e-02]
[6.28884527e-01 5.98527119e-01]
[4.67029198e-01 6.08253753e-01]
[5.82043160e-01 3.37974600e-01]
[5.53283520e-01 1.88655177e-01]
[2.28076339e-01 7.64111125e-01]
[8.25113589e-01 4.79834621e-01]
[4.33109168e-01 3.22714510e-01]
[7.07843205e-01 3.84581765e-01]
[9.93602805e-01 9.59014686e-01]
[9.76159005e-01 4.34268621e-01]
[2.78273827e-01 8.78856055e-01]
[8.40238994e-01 1.81407936e-01]
[3.97367873e-01 7.01643409e-01]
[8.16446512e-01 3.32916995e-01]
[2.86355249e-01 2.82342987e-02]
[1.26954942e-01 1.62711141e-02]
[2.60516458e-01 5.03050677e-01]
[2.04036518e-01 6.02389220e-01]
[1.98891607e-02 8.75009986e-01]
[3.41881960e-01 7.85567818e-01]
[4.75227528e-02 2.48564580e-01]
[5.04193427e-01 6.30951897e-01]
[4.83838658e-02 5.48686460e-01]
[3.82844890e-02 5.56426865e-01]
[2.90286290e-01 7.62715710e-01]
[6.95051774e-01 1.75307204e-01]
[2.52606312e-01 8.27522873e-01]
[7.90255112e-01 4.52357923e-01]
[3.31087661e-02 2.05313321e-01]
[9.96116180e-01 3.81534945e-01]
[3.85813149e-01 3.53683288e-01]
[8.56598587e-01 6.44516484e-01]
[2.30325327e-01 9.43867168e-01]
[9.15547975e-01 6.72128611e-01]
[2.92926762e-01 3.84598856e-01]
[5.73929847e-01 6.88971770e-01]
[3.59754994e-01 5.36101569e-01]
[1.14450396e-03 4.98212706e-01]
[2.31315381e-01 7.05663108e-01]
[8.76083730e-01 1.03132871e-02]
[7.94933562e-02 3.01344604e-01]
[8.58384064e-01 9.22368808e-01]
[5.91658320e-01 8.78758614e-01]
```

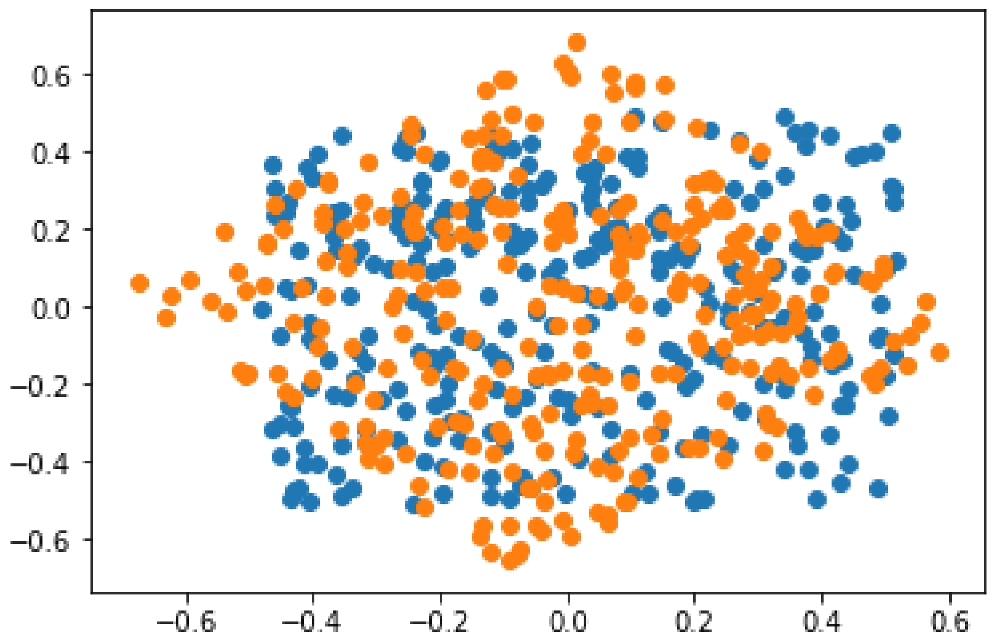
```
[9.99214810e-01 6.25604439e-01]
[4.73822886e-01 5.60268549e-01]
[3.93432751e-01 1.37940683e-02]
[3.65916071e-01 7.68892022e-01]
[4.18930430e-01 2.76698612e-01]
[1.30266487e-01 9.46243966e-01]
[1.15032049e-01 7.31430263e-01]
[7.89366513e-01 8.12727698e-01]
[9.19850074e-01 7.67171683e-01]
[8.44685447e-01 6.62634942e-01]
[9.87678729e-01 2.23802928e-01]
[5.11425753e-01 8.56305460e-01]
[4.15072244e-01 6.71254926e-02]
[5.39049465e-01 6.65480112e-01]
[7.83929073e-01 8.89241172e-01]
[4.21290307e-01 8.74935168e-01]
[5.47581401e-01 7.09302136e-01]
[8.43382648e-01 9.56197236e-01]
[2.15848354e-01 7.18249037e-01]
[8.07355594e-01 4.56091360e-01]
[3.69897546e-01 2.92135726e-01]
[8.71986259e-01 6.55312019e-01]
[6.51621116e-01 4.78602035e-02]
[7.89458916e-01 3.11251712e-01]
[4.09396326e-01 6.64761720e-01]
[5.22601229e-01 8.48062012e-01]
[4.45944957e-02 7.78012650e-01]
[1.77509865e-01 6.18291214e-01]
[6.65172936e-01 1.40938061e-01]
[3.32499099e-01 7.52326029e-01]
[5.95698792e-02 6.49698614e-01]
[3.01496945e-02 4.35586074e-01]
[2.40746539e-01 8.54327989e-04]
[5.24894382e-01 5.32082990e-01]
[7.35891031e-01 5.01900325e-01]
[2.91264418e-01 6.09360946e-01]
[8.84727933e-01 7.76681429e-01]
[5.15241494e-02 3.05698148e-02]
[3.05265675e-02 1.25376205e-01]
[2.31194972e-02 8.11585873e-01]
[9.18771299e-01 3.70417373e-01]
[9.24610577e-01 9.96587082e-02]
[6.19771064e-01 7.30298099e-01]
[4.79139208e-01 2.62093619e-02]
[4.60088557e-01 6.46545831e-02]
[5.50888025e-01 1.24817386e-01]
[8.67937210e-01 3.72009942e-01]
[9.14804131e-01 2.50698061e-01]
```

```
[9.27264651e-01 2.95891311e-01]
[7.29435523e-01 3.70606113e-01]
[3.75586147e-01 9.08017979e-01]
[7.36673779e-01 1.52616242e-01]
[6.31106252e-01 9.85202359e-01]
[5.92816000e-01 8.95874951e-01]
[7.76380225e-02 4.26827126e-01]
[5.03960296e-02 1.21421854e-02]
[6.26574933e-01 6.31082546e-01]
[1.52920270e-01 3.92122577e-01]
[9.30281873e-01 7.30234847e-01]
[4.26677642e-01 3.95571826e-02]
[3.54249783e-01 8.14498169e-01]
[4.61436714e-01 4.63320509e-01]
[1.56397372e-01 6.63291987e-01]
[1.45613293e-01 3.95710293e-02]
[4.48984584e-01 3.91877588e-01]
[8.61452325e-01 8.97944649e-02]
[2.47685626e-01 6.27104642e-01]
[5.92089305e-02 4.18212208e-02]
[7.96611803e-02 1.46433787e-03]
[3.29358128e-01 6.57776688e-01]
[6.97490460e-01 5.53165341e-01]
[5.17583987e-01 7.91219566e-01]
[6.45935253e-01 6.31302508e-01]
[6.08301873e-01 8.34384863e-02]
[2.75627283e-01 3.56036553e-01]
[4.23624722e-01 6.90115022e-01]
[8.60292582e-01 9.65767785e-01]
[8.66193633e-01 4.02936508e-01]
[2.30275905e-01 2.38103377e-01]
[8.10244290e-01 6.08443112e-01]
[5.20573977e-01 3.41964380e-01]
[3.77978561e-01 3.70847275e-01]
[7.71244921e-01 4.94422404e-01]
[7.68394836e-01 7.78079995e-01]
[5.87175495e-01 2.65355087e-02]
[1.90919494e-01 2.65659911e-01]
[9.73218471e-01 3.73354215e-02]
[9.16868612e-02 9.02350277e-01]
[1.47272607e-01 6.37977739e-01]
[1.15341845e-01 7.46788887e-01]
[2.94831745e-01 3.06799262e-01]
[2.87573080e-01 9.22025863e-02]
[2.71657524e-01 1.69876176e-01]
[5.63531087e-01 7.17340766e-01]
[1.38700085e-01 2.71283050e-01]
[1.65042790e-01 1.87947534e-01]
```

```
[2.76871233e-01 4.57746546e-01]
[6.29013647e-01 6.99291395e-01]
[1.29737553e-01 1.50634906e-01]
[4.85493958e-01 2.68530845e-01]
[2.17119121e-01 1.60928998e-01]
[2.99468170e-01 2.12102228e-01]
[5.95586433e-01 8.64984364e-01]
[7.05676134e-01 9.60350040e-01]
[7.48339582e-01 5.88848428e-01]
[8.96862239e-01 4.36863333e-01]
[8.99026697e-01 3.55974369e-01]
[9.44178253e-01 9.02891906e-01]
[5.57836569e-01 6.81734685e-01]
[9.66550336e-01 9.09722743e-01]
[6.63543880e-01 3.61814908e-01]
[6.69430551e-01 6.30355696e-01]
[1.70330354e-01 4.29592557e-01]
[2.17372308e-01 2.94023118e-01]
[8.61903436e-02 4.11693492e-01]
[3.69776618e-01 3.72261848e-01]
[7.07640852e-01 5.12719335e-01]
[7.28196195e-02 5.64941821e-01]
[2.55755089e-01 6.73199209e-01]
[3.13741057e-01 4.32443480e-01]
[1.17434795e-01 6.87973718e-01]
[3.63390999e-01 6.35222809e-02]
[9.05810875e-01 6.93460841e-01]
[9.07215403e-02 1.02009085e-01]
[4.17206805e-01 5.95054433e-01]
[2.88327663e-01 7.52521434e-01]
[6.04674972e-01 2.67246054e-01]
[6.98300673e-02 9.80327510e-02]
[2.17688020e-01 7.49843711e-01]
[9.21246794e-01 2.56004402e-01]
[4.34565382e-01 4.92568432e-01]
[4.23188333e-01 7.79450993e-01]
[8.54149449e-01 4.37077604e-01]
[2.55278300e-01 7.82061480e-01]
[7.39672298e-01 6.03125766e-01]
[1.67583249e-01 1.56655645e-01]
[7.61034115e-01 5.88453473e-01]
[3.66124404e-01 9.20097713e-01]
[5.75755889e-01 8.92120003e-01]
[3.89981833e-01 4.51344649e-01]
[4.44683631e-01 8.15742258e-01]
[2.49887324e-01 3.88595729e-01]
[3.28868152e-01 7.65517101e-01]
[7.94129373e-01 2.17863236e-01]
```

```
[7.46643303e-01 8.09750639e-01]
[1.06713492e-01 3.69515177e-01]
[5.54063504e-01 8.28516432e-01]
[9.11736248e-01 5.15324964e-02]
[8.62858697e-01 6.59516077e-01]
[5.46543099e-01 5.22157045e-01]
[3.16945383e-01 2.20007528e-01]
[9.85009438e-01 3.29223005e-01]
[1.39917024e-01 5.37160783e-01]
[9.71134518e-01 4.28506717e-01]
[8.95878386e-01 9.50267878e-01]
[6.76807199e-01 6.61312197e-01]
[3.65755368e-01 9.42367409e-01]
[2.81409285e-01 7.11648953e-01]
[7.43773033e-02 8.65335563e-01]
[6.32922800e-01 5.05014167e-01]
[8.38592719e-02 8.39315309e-01]
[2.91419691e-01 2.64186088e-01]
[1.19726868e-01 7.04648722e-02]
[2.29292083e-01 6.86277125e-01]
[2.20098437e-01 9.13233204e-01]
[9.73805138e-01 5.12658271e-01]
[6.30676710e-01 7.52307415e-01]
[2.92092674e-01 6.84105981e-01]
[2.42247507e-01 5.22937211e-01]
[7.51981593e-01 9.34291941e-01]
[2.71218021e-01 5.96405684e-01]
[2.14936380e-02 7.44018201e-01]
[9.95960161e-01 7.75779670e-01]
[4.21630587e-01 8.08547130e-01]
[2.47211728e-01 9.56295290e-01]
[8.22800475e-01 8.45134436e-01]
[2.54427449e-01 8.31798809e-01]
[3.96115006e-01 2.44631798e-01]
[4.25354082e-01 9.30018828e-01]
[6.84595395e-01 1.48206265e-02]
[8.08820949e-01 5.95850750e-01]
[6.47650335e-01 3.90330794e-01]
[8.23011335e-01 9.96171952e-01]
[6.56656104e-01 6.51995209e-01]
[3.65136183e-01 9.49321640e-01]
[3.98320599e-01 6.62810815e-01]
[1.67288121e-01 3.60850135e-01]
[3.39620793e-01 7.24875994e-01]
[1.15859762e-01 2.83024519e-01]
[3.96308798e-01 9.16326112e-01]
[3.14676487e-01 1.59979171e-01]
[3.63514710e-01 1.86395151e-01]
```

```
[3.37559077e-01 4.18252361e-01]
[4.13893987e-01 6.81748989e-01]
[4.91053306e-01 2.23290266e-01]
[5.00371032e-01 1.14932142e-01]
[5.29170955e-01 6.83737599e-01]
[3.63946317e-01 3.93252954e-01]]
```



[4]:
```
#Let us compute the bottle-neck distance between full dataset and k-PCA dataset.
↪ k denotes the number of principal components.
prime=23
D=20
dat1=np.random.rand(50,D)
vr = dionysus.fill_rips(dat1, 2, 2.) #Vietoris-Rips complex
cp = dionysus.cohomology_persistence(vr, prime, True) #Create the persistent␣
↪cohomology
dgms = dionysus.init_diagrams(cp, vr) #Calculate the persistent diagram using␣
↪the designated coefficient field.
#record for plottings..
A=np.array([0,1,1])


for k in range(D):
    dat_k = PCAtool.pca(dat1,K=k+1)
    vr_k = dionysus.fill_rips(dat_k, 2, 4.) #Vietoris-Rips complex
```

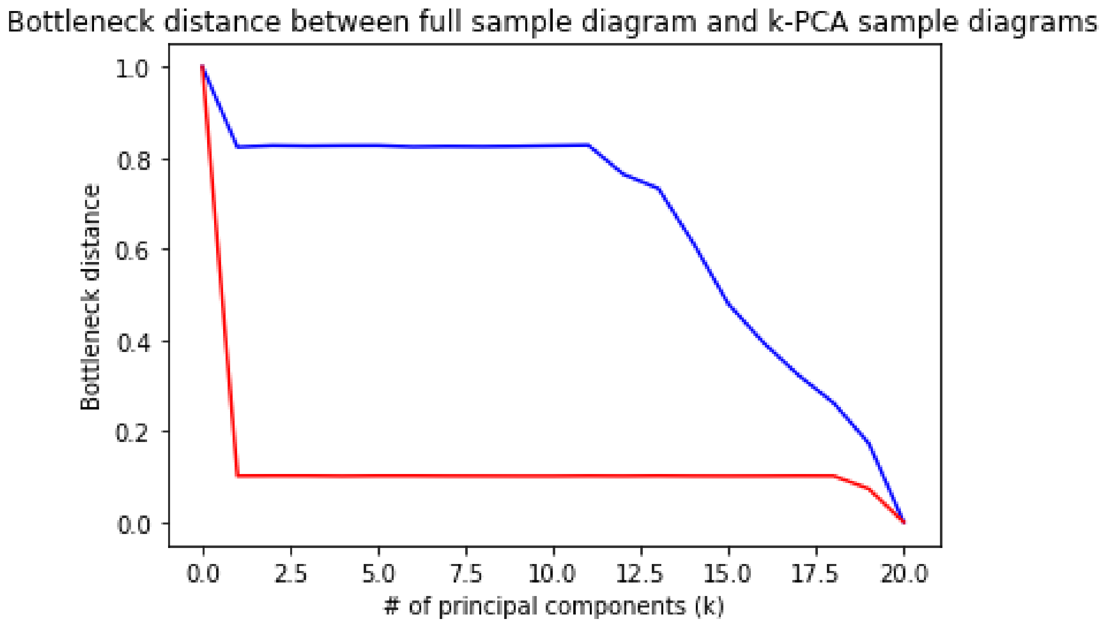
```
    cp_k = dionysus.cohomology_persistence(vr_k, prime, True) #Create the␣
↪persistent cohomology
    dgms_k = dionysus.init_diagrams(cp_k, vr_k) #Calculate the persistent␣
↪diagram using the designated coefficient field.
    bdist_0 = dionysus.bottleneck_distance(dgms[0], dgms_k[0])
    bdist_1 = dionysus.bottleneck_distance(dgms[1], dgms_k[1])
    #dionysus.plot.plot_diagram(dgms_k[1], show=True)
    newrow=[k+1,bdist_0,bdist_1]
    #print(newrow)
    A = numpy.vstack([A, newrow])

print(A)
print(A[1,2])

fig, ax = plt.subplots()
ax.plot(A[:,0],A[:,1],c='b')#dimension 0 diagram distance
ax.plot(A[:,0],A[:,2],c='r')#dimension 1 diagram distance
ax.set(xlabel='# of principal components (k)', ylabel='Bottleneck distance',
       title='Bottleneck distance between full sample diagram and k-PCA sample␣
↪diagrams')
```

```
[[0.00000000e+00 1.00000000e+00 1.00000000e+00]
 [1.00000000e+00 8.23484540e-01 1.01692155e-01]
 [2.00000000e+00 8.27278376e-01 1.02037489e-01]
 [3.00000000e+00 8.26246023e-01 1.02003641e-01]
 [4.00000000e+00 8.27122927e-01 1.01320669e-01]
 [5.00000000e+00 8.27346683e-01 1.01903439e-01]
 [6.00000000e+00 8.24257314e-01 1.01928234e-01]
 [7.00000000e+00 8.25257838e-01 1.01540744e-01]
 [8.00000000e+00 8.24735820e-01 1.01427302e-01]
 [9.00000000e+00 8.25644016e-01 1.01265386e-01]
 [1.00000000e+01 8.27018797e-01 1.01263583e-01]
 [1.10000000e+01 8.28346431e-01 1.01770513e-01]
 [1.20000000e+01 7.64238894e-01 1.01451471e-01]
 [1.30000000e+01 7.32936800e-01 1.01933949e-01]
 [1.40000000e+01 6.12984419e-01 1.01462640e-01]
 [1.50000000e+01 4.79252160e-01 1.01335749e-01]
 [1.60000000e+01 3.94367605e-01 1.01475343e-01]
 [1.70000000e+01 3.22798133e-01 1.01808742e-01]
 [1.80000000e+01 2.61978060e-01 1.01690233e-01]
 [1.90000000e+01 1.73889399e-01 7.36008584e-02]
 [2.00000000e+01 1.20002767e-07 2.39416863e-07]]
0.10169215500354767
```

[4]:

```
[Text(0,0.5,'Bottleneck distance'),
 Text(0.5,0,'# of principal components (k)'),
 Text(0.5,1,'Bottleneck distance between full sample diagram and k-PCA sample
```

```
diagrams')]
```



[5]:
```
type(dgms)
type(dgms[0])
#From diagrams to arrays
#np.array(dgms[0])
#From arrays to diagrams
#print( type([(0,1),(0.5,0.7)]) )
dionysus._dionysus.Diagram([(0,1),(0.5,0.7)])
```

[5]:
```
Diagram with 2 points
```

[6]:
```
def denoise_dgm(dgms,threshold=0.1):
    threshold = .1
    bars_0 = [np.array(bar).tolist() for bar in dgms[0] if bar.death-bar.birth␣
 ↪> threshold] #choosing cocycle that persist at least threshold=1.
    bars_1 = [np.array(bar).tolist() for bar in dgms[1] if bar.death-bar.birth␣
 ↪> threshold] #choosing cocycle that persist at least threshold=1.
    #cocycles = [cp.cocycle(bar.data) for bar in bars]
    #plt is the matplotlib incarnation.
    #print( type(bars_1) )
    #bars_1=np.array(bars_1)
    #print(type(bars_1[1]))
    #print(bars_1[1])
    #print(type(bars_1))
```

```
    dgm_denoise_0=bars_0
    dgm_denoise_1=bars_1
    #print(type(dgm_denoise_0))
    #dionysus.plot.plot_diagram(dgm_denoise_0, show=False)
    #dionysus.plot.plot_diagram(dgm_denoise_1, show=False)
    return([dgm_denoise_0,dgm_denoise_1])

'''
#Red highlight cocyles that persist more than threshold value on barcode, when␣
↪more than one cocyles have persisted over threshold values, this plots the␣
↪first one.
dionysus.plot.plot_bars(dgms[1], show=False)
plt.plot([[bar.birth,bar.death] for bar in dgms[1] if bar.death-bar.birth >␣
↪threshold][0],[[x,x] for x,bar in enumerate(dgms[1]) if bar.death-bar.birth␣
↪> threshold][0],'r')
plt.title('Showing the selected cycles on bar codes (red bars)')
plt.show()

#Red highlight ***ALL*** cocyles that persist more than threshold value on␣
↪diagram.
dionysus.plot.plot_diagram(dgms[1], show=False)
Lt1 = [[point.birth,point.death] for point in dgms[1] if point.death-point.
↪birth > threshold]
for Lt3 in Lt1:
    #print(Lt3)
    plt.plot(Lt3[0],Lt3[1],'ro')
plt.title('Showing the selected cycles on diagram (red points)')
plt.show()
'''
print(dgms_k)
test1=denoise_dgm(dgms_k)
test1[0]
```

```
[Diagram with 50 points, Diagram with 49 points, Diagram with 18424 points]
```

[6]:
```
[(0,inf),
 (0,1.40403),
 (0,1.3332),
 (0,1.4289),
 (0,1.40229),
 (0,1.403),
 (0,1.34595),
 (0,1.32887),
 (0,1.37156),
 (0,1.29584),
 (0,1.39248),
 (0,1.64584),
```

```
(0,1.23297),
(0,1.47286),
(0,1.39124),
(0,1.19594),
(0,1.12187),
(0,1.28163),
(0,1.4383),
(0,1.4698),
(0,1.20281),
(0,1.25048),
(0,1.31941),
(0,1.33678),
(0,1.17107),
(0,1.34541),
(0,1.42859),
(0,1.32976),
(0,1.54038),
(0,1.23253),
(0,1.32246),
(0,1.42239),
(0,1.2203),
(0,1.34523),
(0,1.22457),
(0,1.22807),
(0,1.33558),
(0,1.32693),
(0,1.08975),
(0,1.24706),
(0,1.20635),
(0,1.31939),
(0,1.53156),
(0,1.13285),
(0,1.01112),
(0,1.13553),
(0,1.30935),
(0,1.36507),
(0,1.27896),
(0,1.54502)]
```

[9]:

```
'''
#This is legacy code used to construct a dionysus.plot.plot_diagram object from␣
↪raw, it is necessary to compute bottleneck distance.
dgms_tmp=denoise_dgm(dgms_k,0.1)
dgms_base_0=dionysus._dionysus.Diagram()
dgms_base_1=dionysus._dionysus.Diagram()
#pybind issue, need to be correct instead of succint.
for itr in range(len(dgms_tmp[0])):
```

```
    dgms_base_0.append(dgms_tmp[0][itr])
for itr in range(len(dgms_tmp[1])):
    dgms_base_1.append(dgms_tmp[1][itr])
print(type(dgms_base[0]))
dionysus.bottleneck_distance(dgms_base_0, dgms_base_1)
'''
```

[9]:

```
'\ndgms_tmp=denoise_dgm(dgms_k,0.1)\ndgms_base_0=dionysus._dionysus.Diagram()\nd
gms_base_1=dionysus._dionysus.Diagram()\n#pybind issue, need to be correct
instead of succint.\nfor itr in range(len(dgms_tmp[0])):\n
dgms_base_0.append(dgms_tmp[0][itr])\nfor itr in range(len(dgms_tmp[1])):\n    d
gms_base_1.append(dgms_tmp[1][itr])\nprint(type(dgms_base[0]))\ndionysus.bottlen
eck_distance(dgms_base_0, dgms_base_1)\n'
```

[10]:

```
#Let us compute the bottle-neck distance between full dataset and k-PCA dataset.
 ↪ k denotes the number of principal components.
prime=23
D=10
#Here is the data insertion part
dat1=np.random.rand(50,D)
vr = dionysus.fill_rips(dat1, 2, 2.) #Vietoris-Rips complex
cp = dionysus.cohomology_persistence(vr, prime, True) #Create the persistent␣
 ↪cohomology
dgms = dionysus.init_diagrams(cp, vr) #Calculate the persistent diagram using␣
 ↪the designated coefficient field.
#record for plottings..
A=np.array([0,0,1,1])

for thres in [0,0.1]:
    ######Thresholding the features on the persistent diagrams describing␣
 ↪persistent cohomology of VR complex(CC)
    dgms_tmp=denoise_dgm(dgms,thres)#thresholding CC diagrams
    dgms_base_0=dionysus._dionysus.Diagram()#take persistent points of dim 0
    dgms_base_1=dionysus._dionysus.Diagram()#take persistent points of dim 1
    #pybind issue, need to be correct instead of succint.
    for itr in range(len(dgms_tmp[0])):
        dgms_base_0.append(dgms_tmp[0][itr])
    for itr in range(len(dgms_tmp[1])):
        dgms_base_1.append(dgms_tmp[1][itr])
    #print(type(dgms_base[0]))
    #dionysus.bottleneck_distance(dgms_base_0, dgms_base_1)
    #######
    for k in range(D):
        dat_k = PCAtool.pca(dat1,K=k+1)
        vr_k = dionysus.fill_rips(dat_k, 2, 4.) #Vietoris-Rips complex
        cp_k = dionysus.cohomology_persistence(vr_k, prime, True) #Create the␣
 ↪persistent cohomology
```

```
        dgms_k = dionysus.init_diagrams(cp_k, vr_k) #Calculate the persistent␣
↪diagram using the designated coefficient field.
        #######Thresholding the features on the persistent diagrams describing␣
↪persistent cohomology of kPCA-VR complex(kPCA)
        #############Here we use the same thresholding for fairness. But this␣
↪configuration is easy to change.
        PCA_thres=thres
        #############
        dgms_k_tmp=denoise_dgm(dgms_k,PCA_thres)#thresholding CC diagrams
        dgms_k_0=dionysus._dionysus.Diagram()#take persistent points of dim 0
        dgms_k_1=dionysus._dionysus.Diagram()#take persistent points of dim 1
        #pybind issue, need to be correct instead of succint.
        for itr in range(len(dgms_k_tmp[0])):
            dgms_k_0.append(dgms_k_tmp[0][itr])
        for itr in range(len(dgms_k_tmp[1])):
            dgms_k_1.append(dgms_k_tmp[1][itr])
        #print(type(dgms_base[0]))
        #dionysus.bottleneck_distance(dgms_base_0, dgms_base_1)
        #######
        bdist_0 = dionysus.bottleneck_distance(dgms_base_0, dgms_k_0)
        bdist_1 = dionysus.bottleneck_distance(dgms_base_1, dgms_k_1)
        #dionysus.plot.plot_diagram(dgms_k[1], show=True)
        newrow=[k+1,thres, bdist_0,bdist_1]
        #print(newrow)
        A = numpy.vstack([A, newrow])
print(A)
#Column names of A
#1 number of principal components k;
#2 threshold used for filtering out topological features by persistence;
#3 Bottleneck distance between dimension 0 diagrams of CC and PCA
#3 Bottleneck distance between dimension 1 diagrams of CC and PCA
```

```
[[0.00000000e+00 0.00000000e+00 1.00000000e+00 1.00000000e+00]
 [1.00000000e+00 0.00000000e+00 5.32168150e-01 8.48593563e-02]
 [2.00000000e+00 0.00000000e+00 5.33484817e-01 8.48593563e-02]
 [3.00000000e+00 0.00000000e+00 5.33195853e-01 8.48593563e-02]
 [4.00000000e+00 0.00000000e+00 5.33411741e-01 8.46603960e-02]
 [5.00000000e+00 0.00000000e+00 5.30841947e-01 8.50518048e-02]
 [6.00000000e+00 0.00000000e+00 4.94617045e-01 8.49809349e-02]
 [7.00000000e+00 0.00000000e+00 3.85514319e-01 8.47640336e-02]
 [8.00000000e+00 0.00000000e+00 3.00773650e-01 8.47401023e-02]
 [9.00000000e+00 0.00000000e+00 1.68912739e-01 8.53575021e-02]
 [1.00000000e+01 0.00000000e+00 1.19829949e-07 5.98311871e-08]
 [1.00000000e+00 1.00000000e-01 5.32168150e-01 8.48593563e-02]
 [2.00000000e+00 1.00000000e-01 5.33484817e-01 8.48593563e-02]
 [3.00000000e+00 1.00000000e-01 5.33195853e-01 8.48593563e-02]
 [4.00000000e+00 1.00000000e-01 5.33411741e-01 8.46603960e-02]
```

```
[5.00000000e+00 1.00000000e-01 5.30841947e-01 8.50518048e-02]
[6.00000000e+00 1.00000000e-01 4.94617045e-01 8.49809349e-02]
[7.00000000e+00 1.00000000e-01 3.85514319e-01 8.47640336e-02]
[8.00000000e+00 1.00000000e-01 3.00773650e-01 8.47401023e-02]
[9.00000000e+00 1.00000000e-01 1.68912739e-01 8.53575021e-02]
[1.00000000e+01 1.00000000e-01 1.19829949e-07 5.98311871e-08]]
```

[ ]: