An introduction to Topological Data Analysis with Gudhi
TP 1 - MVA 2017-18

Frédéric Chazal

October 31, 2017

Abstract

Documentation for the Python interface of Gudhi:
http://gudhi.gforge.inria.fr/python/latest/

1 Simplicial complexes and simplex trees

In Gudhi, (filtered) simplicial complexes are encoded through a data structure called simplex tree. Here is a very simple example illustrating the use of simplex tree to represent simplicial complexes. See the Gudhi documentation for a complete list of functionalities. Try the following code and a few other functionalities from the documentation to get used with the Simplex Tree data structure.

```
import numpy as np
import gudhi as gd
import random as rd
import matplotlib.pyplot as plt

st = gd.SimplexTree() #Create a simplex tree

#Simplices can be inserted one by one
#Vertices are indexed by integers
if  st.insert([0,1]):
    print("first simplex inserted!")
st.insert([1,2])
st.insert([2,3])
st.insert([3,0])
st.insert([0,2])
st.insert([3,1])

L = st.get_filtration() #Get a list with all the simplices
#Notice that inserting an edge automatically insert its vertices (if they were
#not already in the complex)
for splx in L:
    print(splx)


#insert the 2-skeleton giving some filtration values to the faces
st.insert([0,1,2],filtration=0.1)
st.insert([1,2,3],filtration=0.2)
```

```
st.insert([0,2,3],filtration=0.3)
st.insert([0,1,3],filtration=0.4)

#if you add a new simplex with a given filtration values, all its faces that
#were not in the complex before are added with the same filtration value
st.insert([2,3,4],filtration=0.7)
L = st.get_filtration()
for splx in L:
    print(splx)

###########################################################################
#Many operations that can be done on simplicial complexes (see also the Gudhi
#documentation and examples):
###########################################################################

st.set_dimension(2) #Warning! For the moment, the dimension of the simplicial
#complex has to be set manually


print("dimension=", st.dimension())

st.initialize_filtration()
print("filtration=", st.get_filtration())
print("filtration[1, 2]=", st.filtration([1, 2]))
print("filtration[4, 2]=", st.filtration([4, 2]))

print("num_simplices=", st.num_simplices())
print("num_vertices=", st.num_vertices())

print("skeleton[2]=", st.get_skeleton(2))
print("skeleton[1]=", st.get_skeleton(1))
print("skeleton[0]=", st.get_skeleton(0))
```

2 Filtrations and persistence diagrams

Filtrations are easy to define and the computation of their persistence diagrams is done in the following way:

```
###########################################################################
#Filtrations and persitence computation
###########################################################################

#Currently, the function to assign a filtration value to a simplex that is
# already in the filtration has not been included in the Python version of
#Gudhi. It will be in the next release (but it is already existing in the C++
#library. A trick to overcome this issue is the following:

st2 = gd.SimplexTree() #The new filtered complex
L = st.get_filtration()
for splx in L:
    #We assign to each simplex its dimension as filtration value
    st2.insert(splx[0],filtration=len(splx[0])-1.0)
```

```
L = st2.get_filtration()
for splx in L:
    print(splx)

#To compute the persistence diagram of the filtered simplex
st2.initialize_filtration()
diag2=st2.persistence()
print(diag2)
#To plot a persistence diagram
gd.plot_persistence_diagram(diag2)
gd.plot_persistence_barcode(diag2)

#To compute bottleneck distance between diagrams
st3 = gd.SimplexTree()
st3.insert([0,1],filtration=0.0)
st3.insert([1,2],filtration=0.1)
st3.insert([2,0],filtration=0.2)
st3.insert([0,1,2],filtration=0.5)
st3.set_dimension(2)
st3.initialize_filtration()
diag3 = st3.persistence()
gd.plot_persistence_diagram(diag3)


diag2_0 = st2.persistence_intervals_in_dimension(0)
diag3_0 = st3.persistence_intervals_in_dimension(0)
dB0 = gd.bottleneck_distance(diag2_0,diag3_0)



diag2_1 = st2.persistence_intervals_in_dimension(1)
diag3_1 = st3.persistence_intervals_in_dimension(1)
dB1 = gd.bottleneck_distance(diag2_1,diag3_1)
```

3 Stability of persistence for functions

Exercise 1.
a) Recall the torus is the surface which is homeomorphic to the surface obtained by identifying the opposite sides of a square as illustrated on Figure 1. Using Gudhi, construct a triangulation (2-dimensional simplicial complex) of the Torus. Define a filtration on it, compute its persistence and use it to deduce the Betti numbers of the torus (check that you get the correct result using the function betti_numbers()).
b) Use Gudhi to compute the Betti numbers of a sphere of dimension 2 and of a sphere of dimension 3 (hint: the k-dimensional sphere is homeomorphic to the boundary of a $k+1$-dimensional simplex.
Exercise 2. The goal of this exercise is to illustrate the persistence stability theorem for functions on a very simple example.
The code below allows to define a simplicial complex (the so-called α-complex) triangulating a set of random points in the unit square in the plane.

```
n_pts = 1000
```

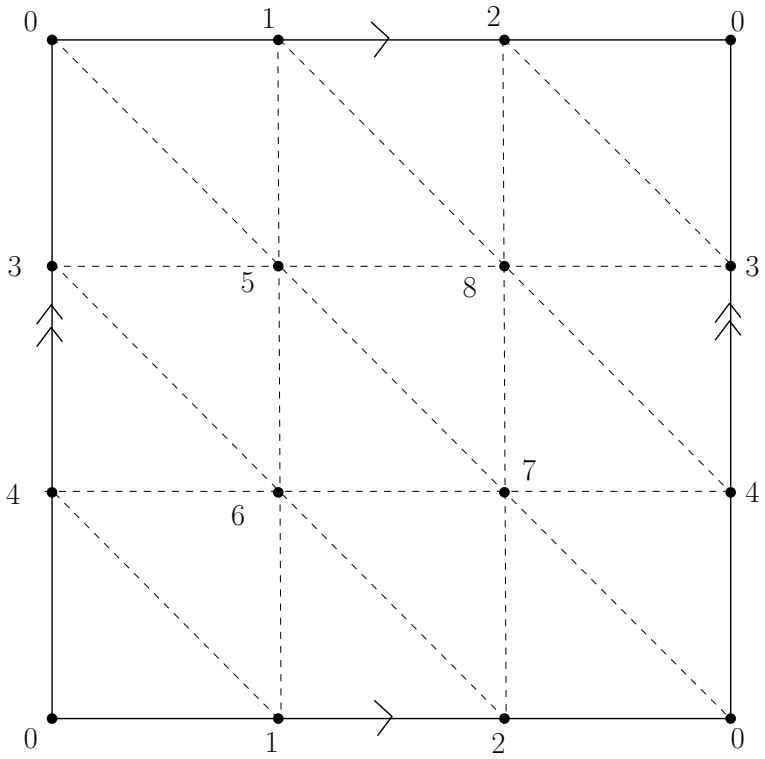

Figure 1: The torus as the quotient of a square

```
#Build a random set of points in the unit square
X = np.random.rand(n_pts,2)
#Compute the alpha-complex filtration
alpha_complex = gd.AlphaComplex(points=X)
st_alpha = alpha_complex.create_simplex_tree(max_alpha_square=1000.0)
```

Let $p_0 = (0.25, 0.25)$ and $p_1 = (0.75, 0.75)$ be two points in the plane $\mathbb{R}^2$ and let $\sigma = 0.05$.
1. Build on such a simplicial complex the sublevel set filtration of the function

$$f(p) = \exp(-\frac{\|p - p_0\|^2}{\sigma}) + 3\exp(-\frac{\|p - p_1\|^2}{\sigma})$$

and compute its persistence diagrams in dimension 0 and 1.
2. Compute the persistence diagrams of random perturbations of f and compute the Bottleneck distance between these persistence diagrams and the perturbated ones. Verify that the persistence stability theorem for functions is satisfied.