

Spherical Voronoi Diagrams

A PyData London story unfolded

Nikolai Nowaczyk

08/05/2016

1 Planar Voronoi Diagrams

- Example: Traveling in London
- Formal description
- Realization in Python

2 Spherical Voronoi Diagrams

- Example: Traveling around the world
- Problem description
- PyData story
- Realization in Python
- Applications

3 Appendix

Outline

1 Planar Voronoi Diagrams

- Example: Traveling in London
- Formal description
- Realization in Python

2 Spherical Voronoi Diagrams

- Example: Traveling around the world
- Problem description
- PyData story
- Realization in Python
- Applications

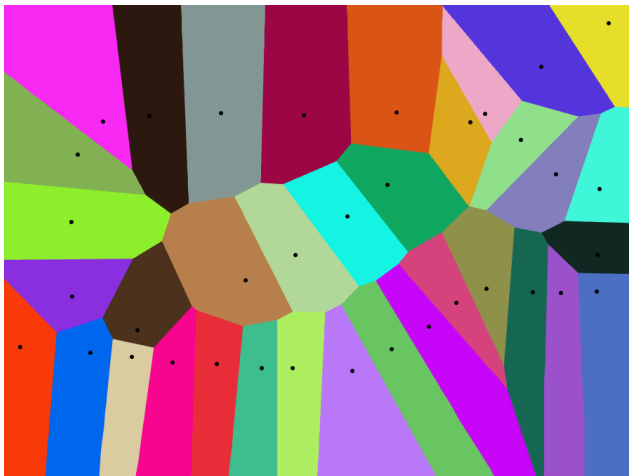
3 Appendix

Voronoi diagrams in the plane



Input: Generator Points

Voronoi diagrams in the plane



<http://alexbeutel.com/webgl/voronoi.html>

Output: Voronoi Regions (coloured)

Mathematical definition

Definition (Voronoi diagram)

Let (X, d) be a metric space (for instance \mathbf{R}^2) and $p = (p_1, \dots, p_K)$ be a finite collection of K distinct points called *generators*.

- 1 For each $1 \leq k \leq K$, the set

$$R_k := \{x \in X \mid \forall j \neq k : d(x, p_k) \leq d(x, p_j)\}$$

is the *k-th Voronoi region*.

- 2 The collection $(R_k)_{1 \leq k \leq K}$ is the *Voronoi diagram* defined by p .

Mathematical definition

Definition (Voronoi diagram)

Let (X, d) be a metric space (for instance \mathbf{R}^2) and $p = (p_1, \dots, p_K)$ be a finite collection of K distinct points called *generators*.

- 1 For each $1 \leq k \leq K$, the set

$$R_k := \{x \in X \mid \forall j \neq k : d(x, p_k) \leq d(x, p_j)\}$$

is the *k-th Voronoi region*.

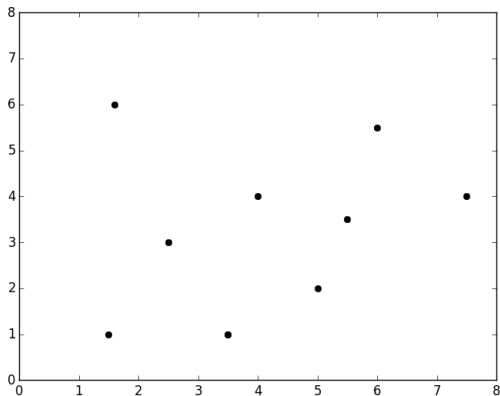
- 2 The collection $(R_k)_{1 \leq k \leq K}$ is the *Voronoi diagram* defined by p .

Theorem

If (X, d) is \mathbf{R}^2 with the Euclidean distance function, then every Voronoi region of every Voronoi diagram is a convex polytope.

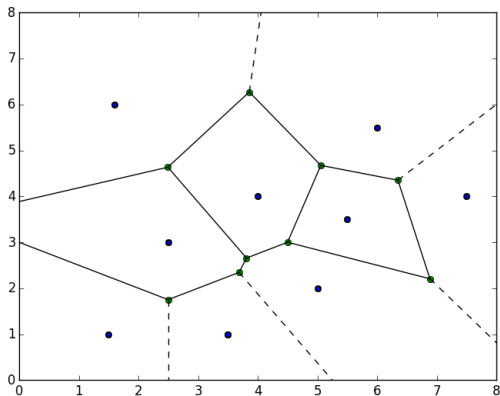
Python: Plotting the generator points

```
points = numpy.array([[1.5, 1.], ...])  
import matplotlib.pyplot as plt  
plt.plot(points[:, 0], points[:, 1], 'o', color='black')  
plt.show()
```



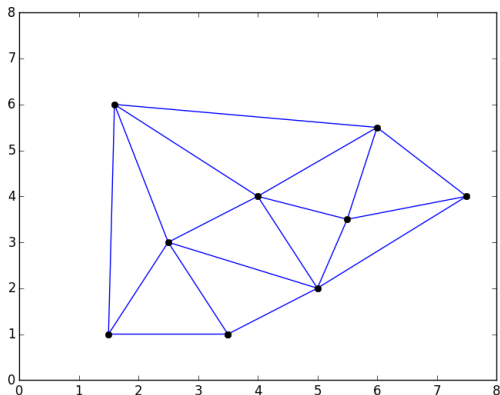
Python: Calculating Voronoi diagram

```
from scipy.spatial import Voronoi, voronoi_plot_2d  
vor = Voronoi(points)  
voronoi_plot_2d(vor, ax)  
plt.show()
```



Python: Delaunay triangulation

```
from scipy.spatial import Delaunay
tri = Delaunay(points)
ax.triplot(points[:,0], points[:,1], tri.simplices.copy(), color='blue')
plt.show()
```



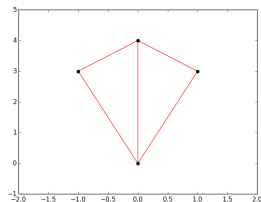
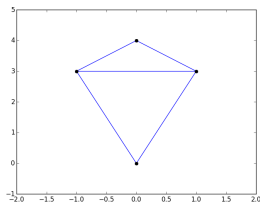
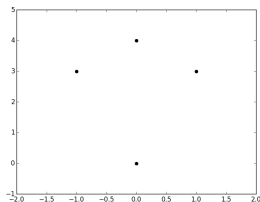
Delaunay triangulation

Definition (Delaunay triangulation)

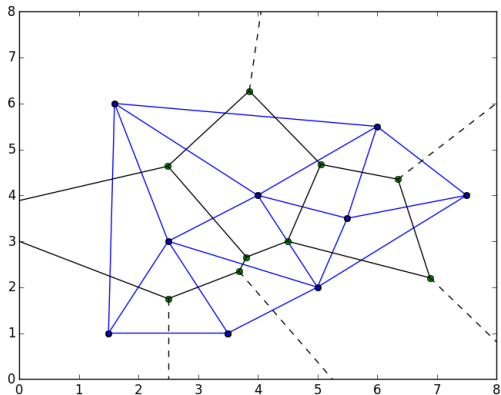
Let P be a set of points in the d -dimensional Euclidean space \mathbf{R}^d . A triangulation $T(P)$ of P is a *Delaunay triangulation* if the circum-hyperspheres of any simplex in $T(P)$ contain no point of P .

Theorem

For any discrete set P in \mathbf{R}^d in *general position*, there exists a unique Delaunay triangulation.



Python: Delaunay triangulation and Voronoi diagram



Outline

1 Planar Voronoi Diagrams

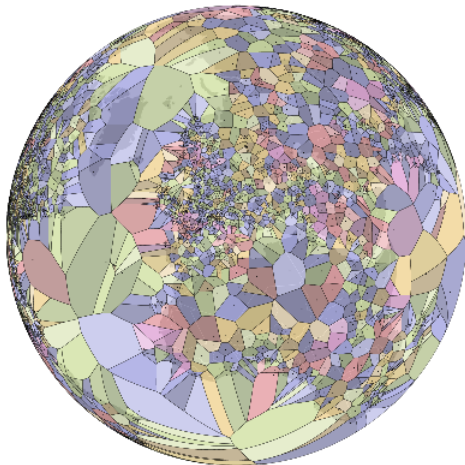
- Example: Traveling in London
- Formal description
- Realization in Python

2 Spherical Voronoi Diagrams

- Example: Traveling around the world
- Problem description
- PyData story
- Realization in Python
- Applications

3 Appendix

World Aiport Map



<https://www.jasondavies.com/maps/voronoi/airports/>

A long time ago (PyData London 2015)...



<https://www.youtube.com/watch?v=gxNa9BD5CnQ>

Spherical Voronoi Diagrams

- Exactly the same problem as planar Voronoi diagram, just on a sphere instead of a plane.
- Many theoretical papers exist.
- Found no ready-to-go implementation in Python.

PyData London 2015 initiated quite a collaboration between...

- Tyler Reddy,
- Nikolai Nowaczyk,
- Ross Hemsley,
- Edd Edmondson,
- Joe Pitt-Francis,
- Mark Sansom,
- Ralf Gommers,
- Pauli Virtanen,
- Juan Luis Cano Rodríguez,
- CJ Carey,
- ...

Result of PyData 2015: scipy.spatial.SphericalVoronoi

<https://github.com/scipy/scipy/pull/5232>

scipy / scipy

Watch 203 Star 2,115 Fork 1,221

Code Issues 783 Pull requests 125 Wiki Pulse Graphs

ENH: adding spherical Voronoi diagram algorithm to scipy.spatial #5232

Merged pv merged 80 commits into scipy:master from tylerjereddy:master 2 days ago

Conversation 51 Commits 80 Files changed 4

+450 -0



tylerjereddy commented on 8 Sep 2015

Background

On July 30th I proposed the idea of adding spherical Voronoi code to scipy (details / motivation described in Issue #5097), which was positively received. As suggested, I posted a similar proposal to the mailing list, which did not receive any responses. While recognizing that an addition this major may be rejected outright, I nonetheless thought I'd move forward with an initial preview of the code, its documentation, and associated unit tests, to give a more concrete idea of how it behaves. The unit tests I wrote all passed after the code migration was complete (all failed initially).

Question / Advice

Before investing more time into refinements I thought it would be useful to receive feedback, if only to confirm that items on my checklist need to be addressed, etc.

My current to-do list for improving the code (if we move forward with this):

- ☒ miscellaneous code improvements suggested by CJ Carey (see below)
- ☒ Python 3 compatibility (assess with 2to3 diff first)

Labels

scipy.spatial

Milestone

0.18.0

Assignee

No one assigned

Notifications

Unsubscribe

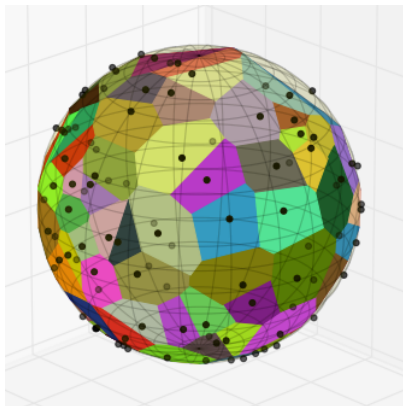
You're receiving notifications because you were mentioned.

8 participants



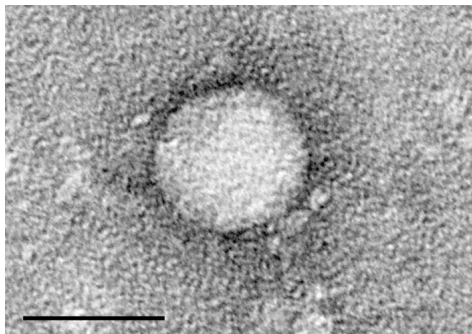
Will be in the 0.18.0 release of scipy.

PyData London 2016: scipy.spatial.SphericalVoronoi



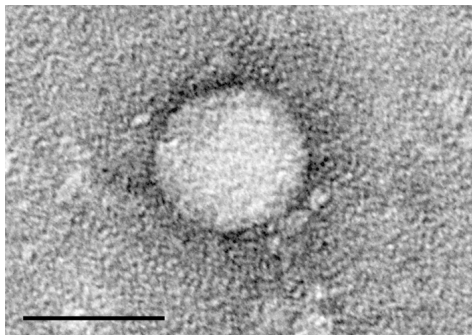
```
>>> points = np.array([...])  
>>> from scipy.spatial import  
      SphericalVoronoi  
>>> sv = SphericalVoronoi(points)
```

Possible Application: Hepatitis C



- 200mio people infected worldwide
- can cause liver disease, cirrhosis, liver cancer
- 350.000 deaths per year
- new medication available (more successful, but expensive)

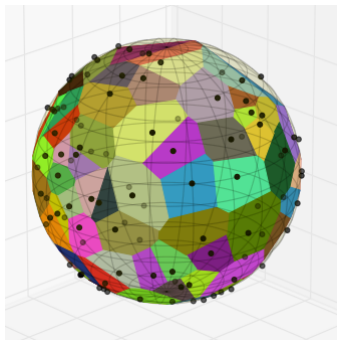
Possible Application: Hepatitis C



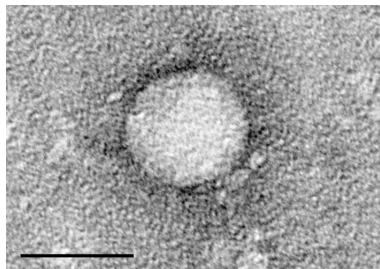
- 200mio people infected worldwide
- can cause liver disease, cirrhosis, liver cancer
- 350.000 deaths per year
- new medication available (more successful, but expensive)

no vaccine

Modelling Hepatitis C



This could be a virus!



Hepatitis C virus

- Computational virology tries to understand a virus through computer simulations.
- Calculating spherical Voronoi diagrams is an important tool.

Maybe more collaboration coming up...



- Help with plotting:
<https://github.com/matplotlib/matplotlib/issues/5294>
- Help with surface area:
<https://github.com/scipy/scipy/issues/6069>

Questions, Comments? Get in touch!

- Nikolai Nowaczyk
 - ▶ mail@nikno.de
 - ▶ <https://github.com/niknow>
- Tyler Reddy
 - ▶ tyler.reddy@bioch.ox.ac.uk
 - ▶ <https://github.com/tylerjereddy>
- How to further improve `scipy.spatial.SphericalVoronoi` ?
- Other applications of spherical Voronoi diagrams?
- Get the code examples:
<https://github.com/niknow/pydata-london-2016-voronoi>

Questions, Comments? Get in touch!

- Nikolai Nowaczyk
 - ▶ mail@nikno.de
 - ▶ <https://github.com/niknow>
- Tyler Reddy
 - ▶ tyler.reddy@bioch.ox.ac.uk
 - ▶ <https://github.com/tylerjereddy>
- How to further improve `scipy.spatial.SphericalVoronoi` ?
- Other applications of spherical Voronoi diagrams?
- Get the code examples:
<https://github.com/niknow/pydata-london-2016-voronoi>

THANK YOU!

Outline

1 Planar Voronoi Diagrams

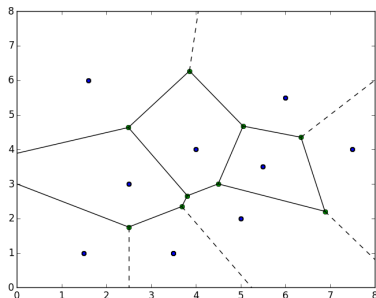
- Example: Traveling in London
- Formal description
- Realization in Python

2 Spherical Voronoi Diagrams

- Example: Traveling around the world
- Problem description
- PyData story
- Realization in Python
- Applications

3 Appendix

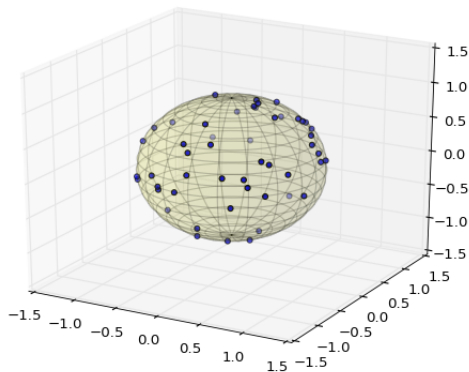
Python: A closer look at the data structures



```
>>> points
[[1.5, 1.0], [3.5, 1.0], [5.0,
 2.0], [2.5, 3.0], [3.5, 1.0],
  [4.0, 4.0], [5.5, 3.5],
  [6.0, 5.5], [7.5, 4], [1.6,
  6.0]]
>>> points.shape
(10L, 2L)
>>> vor = Voronoi(points)
>>> vor.vertices
[[-1.11, 3.55], [2.49, 4.63],
 [3.81, 2.65625], [3.86,
 6.27], [6.87, 2.20], [4.5,
 3.], [ 6.35, 4.35], [5.06,
 4.67], [2.5, 1.75], [3.69,
 2.34]]
>>> vor.vertices.shape
(10L, 2L)
>>> vor.regions
[[-1, 0, 1, 3], [7, 5, 4, 6], [7,
 3, 1, 2, 5], [7, 3, -1, 6],
 [9, -1, 8], [], [-1, 4, 6],
 [9, 2, 5, 4, -1], [8, 0, -1],
 [9, 2, 1, 0, 8]]
```

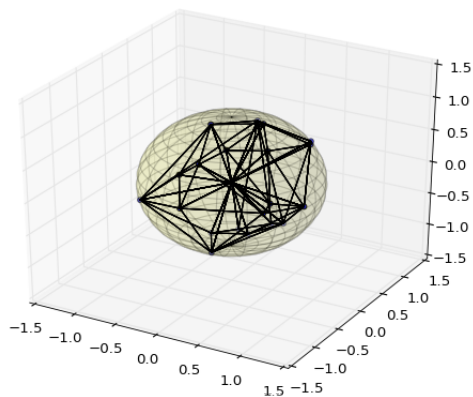
list of tuples of integers. The tuple at the k-th position contains the indices of the Voronoi vertices of the Voronoi region surrounding the k-th generator.

Algorithm:



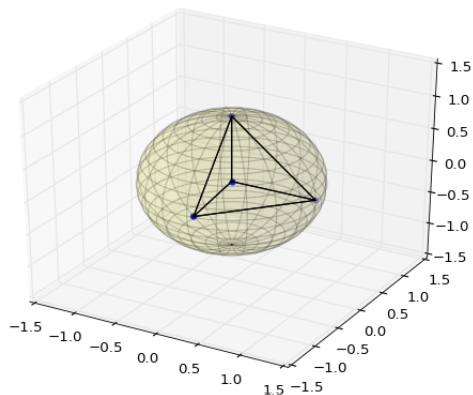
Generator points

Algorithm:



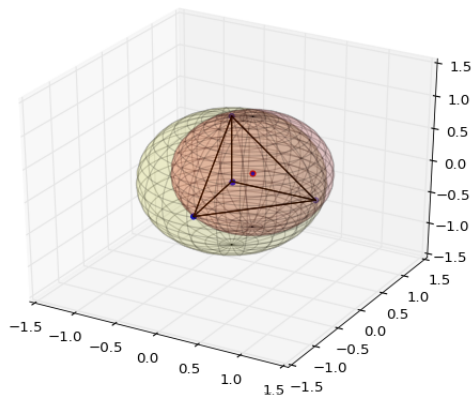
Calculate triangulation

Algorithm:



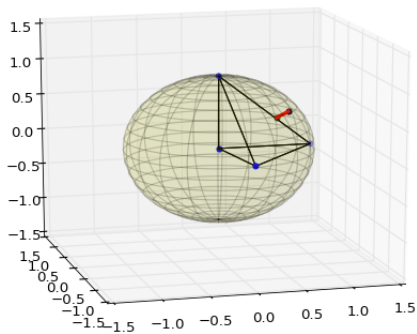
For each tetrahedron in the triangulation...

Algorithm:



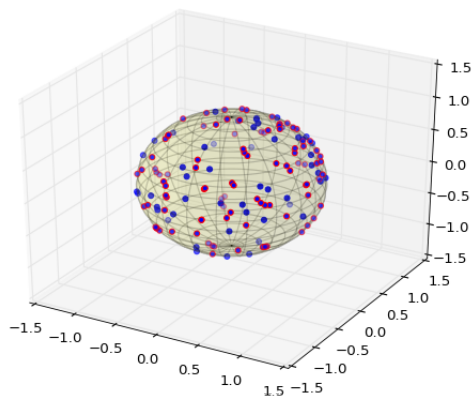
...compute circumcenter of circumsphere.

Algorithm:



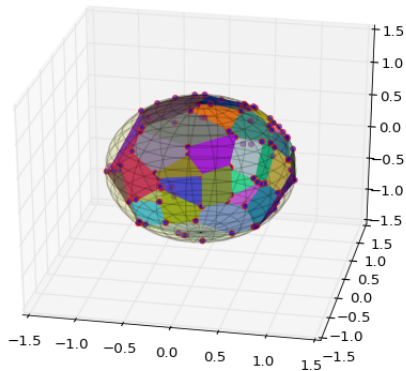
...and project onto sphere.

Algorithm:



Get lots of points on the sphere.

Algorithm:



These points are the Voronoi vertices!

Spherical Voronoi Algorithm: Summary

- 1 Calculate 3D Delaunay triangulation of the generator points.
- 2 Add the center of the sphere to the triangulation and get a tetrahedralization.
- 3 Calculate the circumcenter of the circumsphere of each tetrahedron.
- 4 Project circumcenters to the sphere to get the Voronoi vertices.
- 5 Use the neighbourhood information from the triangulation to infer, which vertices belong to which region.