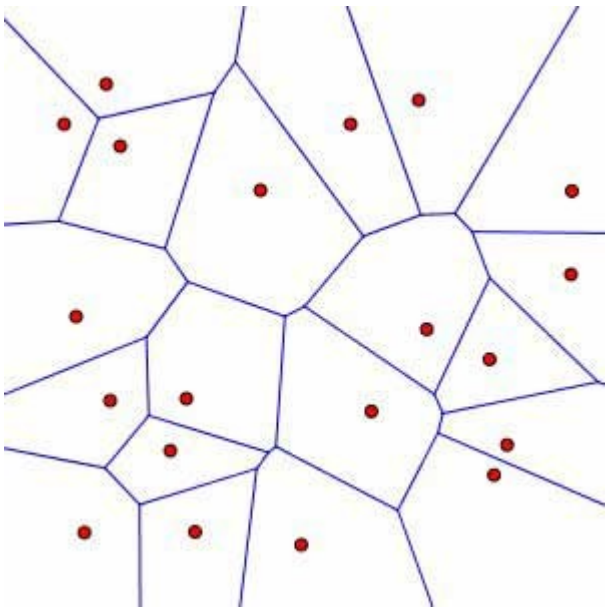
Πολυδιάστατες Δομές Δεδομένων και Υπολογιστική Γεωμετρία – Αναφορά Εργασίας

Θέμα εργασίας: Υλοποίηση αλγορίθμων συσταδοποίησης voronoi σε 2 και 3 διαστάσεις & Οπτικοποίηση.

Σοφικίτης Ευάγγελος, 1047195
Φραδέλος Γεώργιος, 1047143
Ζούλφος Γεώργιος, 1047141

Στόχος της παρούσας εργασίας είναι η υλοποίηση αλγορίθμων Voronoi σε πραγματικά δεδομένα, σε γλώσσα Python. Από μαθηματικής άποψης, τα Voronoi είναι διαγράμματα τα οποία δεδομένων κάποιων σταθερών σημείων/"κέντρων" (sites), χωρίζουν το επίπεδο (ή το χωρο 3ων διαστάσεων) σε περιοχές (cells) ως εξής:

- Καθε "κεντρο" ανήκει σε 1 μόνο cell.
- Οποιοδήποτε σημείο μιας περιοχής (cell) είναι πιο κοντά στο "κέντρο" αυτού του cell από οποιδήποτε άλλο δεδομένο "κέντρο".



Τα Voronoi διαγράμματα βρίσκουν πληθώρα εφαρμογών σε δίαφορους τομείς της επιστήμης όπως για παράδειγμα στις Φυσικές Επιστήμες, στις Επιστήμες Υγείας, στη Μηχανική, στην Πληροφορική κα.

Βασική Ιδέα

Η εργασία αυτή είναι μια δική μας υλοποίηση και η βασική ιδέα είναι η εξής:

1. Αρχικά δέχεται ένα σταθερό πραγματικό dataset και από αυτό αποθηκεύει συγκεκριμμένες τιμες που αντιστοιχούν στα κέντρα του διαγράμματος που θα βρόυμε.

2. Έπειτα διατρέχει τα κέντρα και για κάθε ένα από αυτά βρίσκει όλους τους δυνατούς συνδυασμούς τριγώνων με τα βαρύκεντρά τους/εν δυνάμει σημεία Voronoi.

3. Εξεταζει για κάθε βρύνκεντρο αν είναι όντως σημείο Voronoi, και αν είναι το αποθηκεύει σε μία δομή.

4. Στο επόμενο βήμα γίνονται κάποιοι υπολογισμοί για την εύρεση των ακμών του διαγράμματος.

5. Έπειτα υπολογίζουμε τις ημιευθείες που χωρίζουν τα άκρα του διαγράμματος.

6. Τέλος προβάλλουμε τα αποτελέσματα μαζί με τα κέντρα σε ένα διάγραμμα όπως το παραπάνω.

Υλοποίηση στις 2 Διαστάσεις

Η κλάση `class VoronoiSite:` sites περιγράφει τα “κέντρα” από τα οποία προκύπτει το διάγραμμα. Τα αντικείμενα της κλάσης δημιουργούνται με τις συντεταγμένες x,y και περιέχει τις ιδιότητες:

```
self.x = x
self.y = y
self.voronoi_points = []
self.perpendiculars = []
self.vectors = []
self.semi_lines = []
self.has_cell = False
```

Βοηθητικές Συνατρήσεις:

Στο πρόγραμμά μας έχουμε φτιάξει κάποιες βοηθητικές συνατρήσεις που υλοποιούν τα παραπάνω βήματα και παρουσιάζονται παρακάτω.

`def distance(site, point):`

Η συνάρτηση αυτή δέχεται ως όρισμα δύο κέντρα ή ένα κέντρο και ένα σημείο και υπολογίζει την απόστασή μεταξύ τους.

`def intersection(line1, line2):`

Η συνάρτηση αυτή δέχεται δύο ευθείες και επιστρέφει το σημείο τομής τους αν υπάρχει, αλλίως επιστρέφει None.

`def perpendicular(point1, point2):`

Η συναρτηση αυτή δέεχεται δύο σημεία και βρίσκει τη μεσοκάθετό τους

`def find_center(point1, point2, point3):`

Η σύνάρτηση δέχεται τρία σημεία και χρησιμοποιόντας την perpendicular και την inersection βρίσκει το βαρύκεντρό του τριγώνου που ορίζουν.

`def points_in_circle(center, radius, sites_table):`

Η συνάρτηση δέχεται ένα βαρύκεντρο με την απόσταση (ακτίνα) από τις κορυφές του τριγώνου του και τη λίστα με τα “κέντρα”. Εξεταζει αν υπάρχουν “κέντρα” εσωτερικά του κύκλου στον οποίο εγγράφεται το τρίγωνο.

def is_vector_perpendicular(this_vector, this_site, sites_table):

Δέχεται ένα ευθύγραμμο τμήμα που έχει δημιουργηθεί από δύο σημεία Voronoi, το “κέντρο” στο οποίο ανήκουν και τη λίστα με τα “κέντρα” . Αυτό που κάνει είναι να ελέγχει αν το ευθύγραμμο τμήμα ισαπέχει από δύο “κέντρα” δηλαδή αποτελεί ακμή του διαγράμματος.

def number_of_intersections(line, line_table):

Δέχεται ένα ευθύγραμμο τμήμα και μια λίστα με όλες τις ακμές Voronoi και χρησιμοποιώντας συναρτήσεις της βοβλιοθήκης Shapely βρίσκει το πλήθος των τομών του με αυτές.

Βασικές συναρτήσεις:

def find_voronoi_points(sites_table):

Δέχεται το σύνολο των “κέντρων” και συνδυάζει τις βοηθητικές συναρτησεις ***distance (site, point)*** , ***points_in_circle (center, radius, sites_table)***, ***perpendicular(point1, point2)*** , ***find_center(point1, point2, point3)*** για να βρεί τους κόμβους του διαγράμματος Voronoi.

Σημείωση: Γίνεται ένας extra έλεγχος για την περίπτωση που υπάρχουν συνευθειακά σημεία οπότε και δεν ορίζεται τρίγωνο ώστε να βρεθεί το βαρύκεντρο. Τότε αποθηκεύονται οι δύο κατάλληλες μεσοκάθετοι.

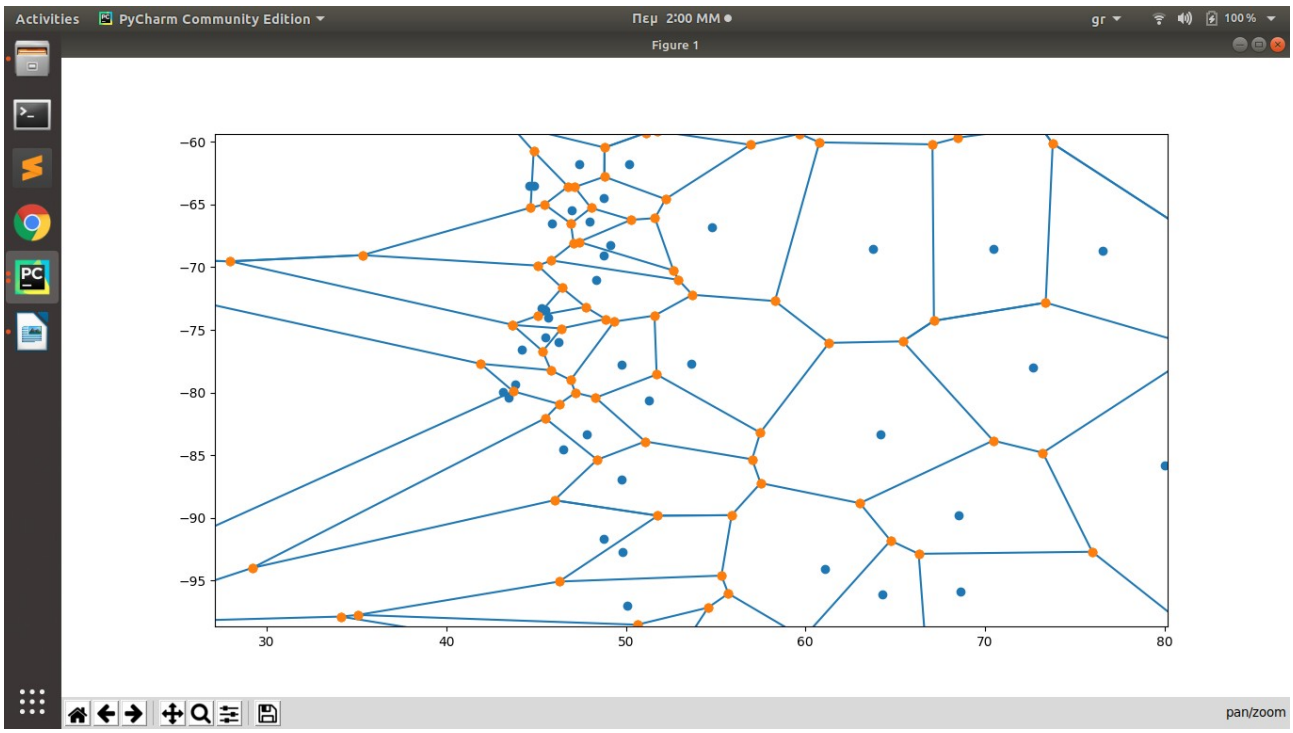
def find_voronoi_cells(sites_table):

Δέχεται το σύνολο των “κέντρων” και για καθένα από αυτά βρίσκει τις ακμές Voronoi που το περικλείουν. Για τη διαδικασία αυτή για ένα κέντρο γνωρίζοντας τα σημεία Voronoi που το περικλείουν βρίσκουμε όλα τα πιθανά ευθύγραμμα τμήματα μεταξύ τους, και χρησιμοποιόντας την ***is_vector_perpendicular(this_vector, this_site, sites_table)*** κρατάμε τα σωστά. Εάν οι ακμές του κέντρου σχηματίζουν κλειστό πολύγωνο τότε η ιδιότητα has cell της κλάσης γίνεται true.

def ***find_voronoi_semi_lines(sites_table):***

Δέχεται το σύνολο των “κέντρων” και για κάθε ένα που δεν έχει cell (άρα βρίσκεται στα άκρα του διαγράμματος) υπολογίζει και αποθηκεύει τις ημιευθείες Voronoi που του αντιστοιχούν.

Τα αποτελέσματα που πήραμε για τα πρώτα 100 στοιχεία του dataset είναι κάπως έτσι:



Υλοποίηση στις 3 Διαστάσεις

Η κλάση `class VoronoiSite:` sites περιγράφει τα “κέντρα” από τα οποία προκύπτει το διάγραμμα. Τα αντικείμενα της κλάσης δημιουργούνται με τις συντεταγμένες x, y, z και περιέχει τις ιδιότητες:

```
self.x = x
self.y = y
self.y = z
self.voronoi_points = []
self.vectors = []
```

`def intersection(plane1, plane2, plane3):`

Η συνάρτηση αυτή δέχεται ως είσοδο 3 επίπεδα και επιστρέφει το σημείο τομής τους. Αρχικά παίρνει τα επίπεδα ανά δύο και ελέγχει εάν παράλληλα οπότε και δεν υπάρχει σημείο τομής. Τότε παίρνει δύο επίπεδα και βρίσκει την εξίσωση της γραμμής στην οποία τέμνονται. Τέλος υπολογίζει το σημείο τομής αυτής της ευθείας με το τρίτο επίπεδο.

`def calcAngle(plane1, plane2):`

Η συνάρτηση αυτή δέχεται ως όρισμα δύο επίπεδα και επιστρέφει την γωνία που σχηματίζουν. Χρησιμοποιείται για να καθοριστεί εάν τα δυο επίπεδα είναι παράλληλα.

`def distance(this_site, point):`

Η συνάρτηση αυτή δέχεται ως όρισμα δύο κέντρα ή ένα κέντρο και ένα σημείο και υπολογίζει την απόστασή μεταξύ τους.

`def perpendicular(point1, point2):`

Η συνάρτηση αυτή δέχεται ως είσοδο δύο σημεία και υπολογίζει το επίπεδο που περνάει από το μέσο της απόστασής τους και είναι κάθετο σε αυτά.
Def findCenter Η συνάρτηση δέχεται τρία σημεία και χρησιμοποιώντας την perpendicular και την inersection βρίσκει το βαρύκεντρο του τριγώνου που ορίζουν.

`def find_center(point1, point2, point3):`

Η σύνάρτηση δέχεται τρία σημεία και χρησιμοποιόντας την perpendicular και την inersection βρίσκει το βαρύκεντρό του τριγώνου που ορίζουν.

```
def points_in_circle(center, radius, sites_table):
```

Η συνάρτηση δέχεται ένα βαρύκεντρο με την απόσταση (ακτίνα) από τις κορυφές του τριγώνου του και τη λίστα με τα “κέντρα”. Εξεταζει αν υπάρχουν “κέντρα” εσωτερικά της σφαίρας στον οποίο εγγράφεται το τρίγωνο.

```
def is_vector_perpendicular(this_vector, this_site, sites_table):
```

Δέχεται ένα ευθύγραμμο τμήμα που έχει δημιουργηθεί από δύο σημεία Voronoi, το “κέντρο” στο οποίο ανήκουν και τη λίστα με τα “κέντρα” . Αυτό που κάνει είναι να ελέγχει αν το ευθύγραμμο τμήμα ισαπέχει από δύο “κέντρα” δηλαδή αποτελεί ακμή του διαγράμματος.

Κύριες συναρτήσεις

```
def find_voronoi_points(sites_table):
```

Δέχεται το σύνολο των “κέντρων” και συνδυάζει τις βοηθητικές συναρτησεις distance , Point in circle , Find Ceneter για να βρεί τους κόμβους του διαγράμματος Voronoi.

```
def find_voronoi_cells(sites_table):
```

Δέχεται το σύνολο των “κέντρων” και για καθένα από αυτά βρίσκει τις ακμές Voronoi που το περικλείουν. Για τη διαδικασία αυτή για ένα κέντρο γνωρίζοντας τα σημεία Voronoi που το περικλείουν βρίσκουμε όλα τα πιθανά ευθύγραμμα τμήματα μεταξύ τους, και χρησιμοποιόντας την `is_vector_perpendicular(this_vector, this_site, sites_table)` κρατάμε τα σωστά. Εάν οι ακμές του κέντρου σχηματίζουν κλειστό πολύγωνο τότε η ιδιότητα has cell της κλάσης γίνεται true.

Τα αποτελέσματα που πήραμε για δοκιμαστικά στοιχεία που δώσμε είναι κάπως έτσι: