Constructing Geometric Simplicial Complexes

Weituo DAI
Changqing LIU

Contexte

L'analyse topologique des données (TDA) est un domaine récent qui connait un succès croissant depuis quelques années. Il vise à comprendre, analyser et exploiter la structure topologique et géométrique de données souvent représentées par des nuages de points dans des espaces euclidiens ou des espaces métriques plus généraux.

Le complexe simplicial filtré dérivé d'un nuage de points et son homologie permenant sont deux outils fondamentales. L'objectif de ce project est de construire certains complexes simplicials et de calculer leur valeurs de filtration. Dans les parties suivantes, on se limite en $\mathbb{R}^3$ pour la simplicité.

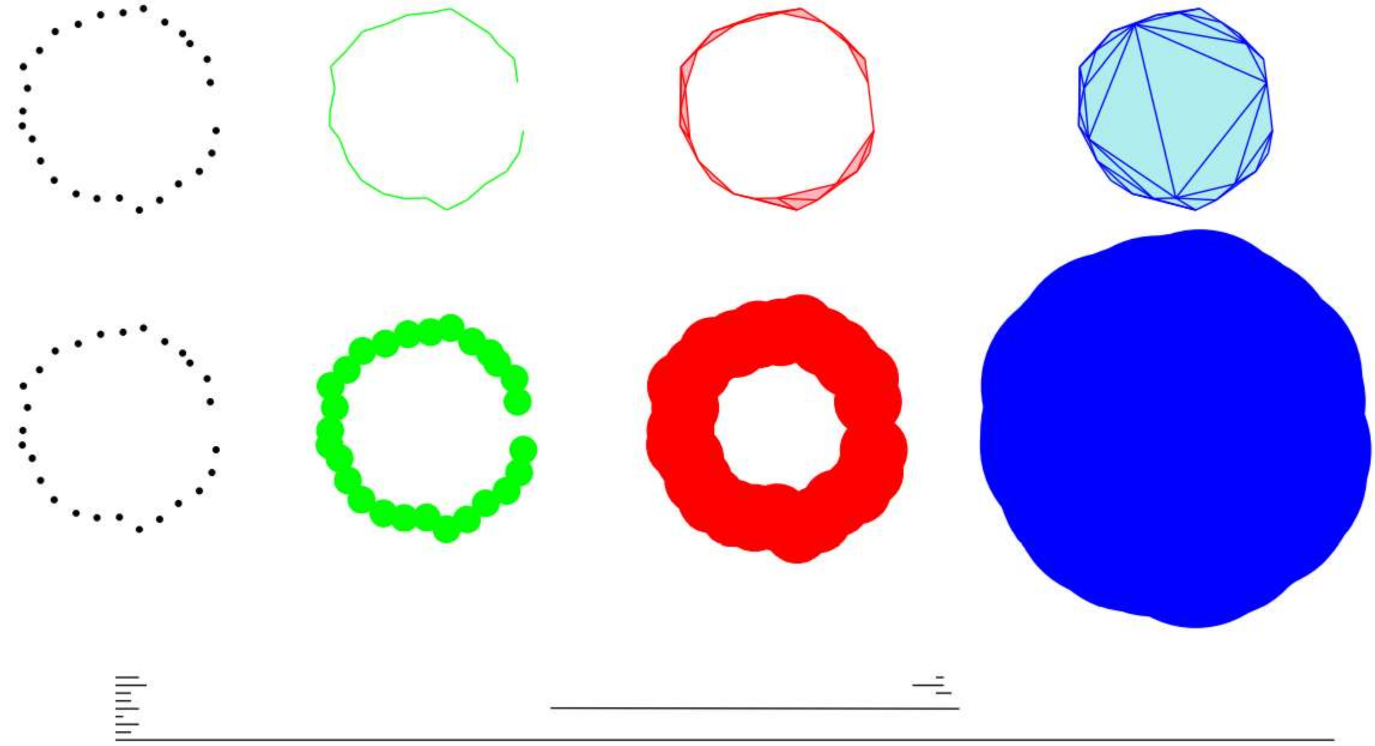
Preliminaries

Complexe Simplicial

Le complexe simplicial est un génération de graphe en dimension supérieur. En général, un k-simplexe est défini comme l'enveloppe convexe d'un ensemble de $k+1$ points affinement indépendants. Les faces d'un k-simplexe sont les simplexes définis par les sous-ensembles de ces $k+1$ points. Comme dans le cas de graphe, si un complexe comprend un simlexe, alors il comprend tous les faces de ce simplexe.

Homologie Permenant

Homologie permenant nous permet de tracer l'évolution d'un complexe filtré. Pour un ensemble de points $p_1, \ldots, p_n$, on considère une famille de boules de rayon r : $B(p_1, r), \ldots, B(p_n, r)$. Les components fusionnent progressivement à partir de plusieurs points isolés, voir figure.



Théoriquement, cette union de boules est homotopiquement équivalente à son **nerve**, un complexe simpliciale avec un sommet pour chaque boule, un k-simplexe si les $k+1$ boules correspondantes ont de point en commun (ou de manière équivalente, il y a une boule de rayon r qui contient les points $p_{i_1}, \ldots, p_{i_{k+1}}$ dont le centre est le point en commun). Ce complexe simplicial s'appelle complexe de Čech de parametre r. Pour chaque simplexe dans le complexe, on lui associe une valeur de filtration, qui est le plus petit r où le simplexe apparaît.

Complexe de Čech

Dans cet partie on construira le complexe de Čech.

Remarque

Tout d'abord, on cherche à déterminer la valeur de filtration pour chaque simplexe. La condition équivalente audessus nous implique que cette valeur corresponde au rayon de la boule minimum qui couvre le simplexe (**MEB**). Donc on écrira un algorithme qui calcul le MEB.

On pose C^k le sous-complexe de Čech avec les simplexes de dimension au plus k , et C_l^k où on ne garde que ceux dont le valeur est inférieur à l .

Problème de la Boule Minimum

P est un ensemble de points dans $\mathbb{R}^3$, on cherchera la boule de rayon minimum qui couvre tous les points de P.

Pour cela, on utilise l'algorithme de welzl :

```
Algorithm MEB(P, R):
Input:  un ensemble de points P, un ensemble R avec |R| ≤ 4
Output:  la boule minimum avec R sur sa borne qui couvre P

if |P| = 0 or |R| = 4 then
    return MEB_trivial(R)
    Quand l'ensemble est petit, on calcule la MEB manuellement
p := a random point in P
D := MEB(P\{p},  R)
if p is in D then
    return D
return MEB(P\{p}, R∪{p})
```

et

```
Algorithm MEB(P)
Input:  un ensemble de points P
Output:  la boule minimum qui couvre P

return MEB(P, ∅)
```

D'après la théorie, *https://en.wikipedia.org/wiki/Smallest-circle_problem* *(https://en.wikipedia.org/wiki/Smallest-circle_problem)*
l'espérance de la complexité est a priori $O(n)$

Construction

On veut maintenant construire le complexe de Čech de manière naïve à l'aide de l'algorithme MEB.

```
Algorithm cechNaif(P, k):
Input:  un ensemble de points P, un entier k qui marque la dimension
Output:  le complexe C^k avec la valeur de filtration de chaque simplexe

Complex = ∅
for s in subsets of P with at most k+1 elements:
    add {[the simplex induced by s, and its filtration value]} into Complex
return Complex
```

Pour construire le complexe C_l^k de manière plus efficace, on considère son skeleton : un graphe dont les sommets sont les 0-simplexe et les arêtes sont les 1-simplexe. Les arêtes d'un simplexe dans C_l^k sont forcément contenus dans le skeleton. Donc on peut juste parcourir la list d'adjacence du skeleton. Voici le pseudo code :

```
Algorithm cechEfficient(P, k, l):
Input: un ensemble de points P, un entier k qui marque la dimension, un
       nombre limite de la filtration
Output: le complexe C^k_l avec la valeur de filtration de chaque simplexe

Complex := ∅
G := skeleton of C^k_l
un graphe des sommets dans P. Si le MEB de deux sommets < l, on les relie avec
un arête.
for d := 0 to k:
    for v in P:
        for k-simplex in the adjacence list of v:
            r = MEB(k-simplex)
            if r < l:
                add [k-simplex, r] into Complex
return Complex
```

Le temps pour construire le skeleton est de $O(\binom{n}{2}) = O(n^2)$. Pour récupérer les d-simplexes ($d \leq k$), on va d'ailleurs calculer le MEB chaque fois dont la complexité est de $O(d+1) = O(1)$, donc la complexité de ce processus est de $O(n\binom{N}{d+1})$, où N est le nombre moyenne de voisins d'un point. Quand l est petit, N est aussi petit. En conséquant, la complexité totale est de $O(n^2 + \sum_{d=1}^{k+1} n\binom{N}{d}) = O(n^2 + N^{k+1}n)$

α-complex

Etant donné P un ensemble de points, A un sous ensemble, on cherche à calculer une boule B avec rayon minimale tel que tous les points de A est dans la surface de B, et qu'aucun point de P est dans B.

On va prouver que ce problème est de LP-type. Ici, on fixe le sous ensemble A, pour chaque sous ensemble C de P, on pose la fonction $f(C) =$ *le rayon de la boule minimum* B telle que tous les points de A est dans la surface de B, et qu'aucun point de C est dans B, et $f(C) =$ *l'infini* si cette boule n'existe pas.

Preuve:

On voit directement, $f(C_1) \leq f(C_2)$ si $C_1 \subset C_2$. Supposons que $C_1 \subset C_2 \subset P, x \in P$, on a $f(C_1) = f(C_2) = f(C_1 \cup \{x\})$. On montrera $f(C_2 \cup \{x\}) = f(C_2)$. Et c'est évident:

Sans perte de généralité, on suppose que $f(C_1) = f(C_2) < \infty$. Comme $f(C_1) = f(C_2) = f(C_1 \cup \{x\})$, C_1 , C_2 , $C_1 \cup \{x\}$ ont la même boule minimum. En effet cet boule satisfait : la boule minimum telle que tous les points de A est dans la surface de

B, et qu'aucun point de $C_1 \cup C_2 \cup \{x\}$ est dans B. Donc elle est aussi la boule minimum pour $C_2 \cup \{x\}$

En conséquant, on peut appliquer l'algorithme de LP-type à cette question:

```
Algorithm MEB_alpha(P,R)
Input: un ensemble de points P, un sous-ensemble R qui forme un simplexe
Output: la boule minimum avec R sur sa borne qui contient aucun point de A
        à l'intérieur ou l'infini si cette boule n'existe pas

if |R| > 4:
    return [0,0,0], infinity
    # the ball doesn't existe. We use a number sufficiently large to
      represent it
if |P| = 0:
    return circumball(R)
randomly choose a point p from P
D := MEB_alpha(P\{p}, R)
if p not in D or D.radius = infinity:
    return D
return MEB_alpha(P\{p}, R∪{p})
```

et

```
Algorithm verification(P, R)
Input:  un ensemble de points P, un sous-ensemble R qui forme un simplexe
Output:  la valeur de filtration de ce simplex si il est dans le complexe;
         sinon, l'infini

return radius of MEB_alpha(P, R)
```

L'Exactitude:

Comme $|P|$ décroît à chaque pas de la récurrence, l'algorithme se termine toujours.

On va prouver l'exactitude de MEB_alpha par récurrence.

Si $|P| = 0$, c'est trivial.

Supposons que MEB_alpha est correct pour $|P| = n$. Alors pour $|P| = n + 1$, si $|R| > 5$, MEB_alpha renvoie l'infini, qui est tout à fait correct. Si $|R| < 5$, on déterminera la relation entre le point p sélectionné et la boule minimum.

En fait par l'hypothèse, **MEB_alpha(P\{p},R)** donne le bon résultat. Si p n'est pas dans **MEB_alpha(P\{p},R)**, alors la boule minimum **MEB_alpha(P,R)** est tout à fait **MEB_alpha(P\{p},R)** ; si **MEB_alpha(P\{p},R)** est l'infini, par la croissance du problème de LP-type, on sait que **MEB_alpha(P,R)** est aussi l'infini; si p est dans la boule, alors le resultat est soit l'infini, soit le boule minimum avec $R \cup \{p\}$ sur sa borne, ceux qui donne **MEB_alpha(P\{p},R∪{p})**.

Donc par récurrence, l'algorithme **MEB_alpha** nous donne la bonne réponse.

Complexité

Comme l'algorithme MEB au-dessus, l'espérance de la complexité est aussi a priori $O(n)$

Construction

Par la suite, on construira le α-complexe. A l'aide de **MEB_alpha**, on peut le construire naïvement:

```
Algorithm alpha(P, k, l)
Input: un ensemble de points P, un entier k qui marque la dimension, un
       nombre k la limite de la filtration
Output: les simplexes de la dimension au plus k et de la valeur de filtration
        au plus l, avec la valeur de filtration de chaque simplexe

Complex = ∅
k = min{k,3}
il n'y a pas de alpha-simplex de dimension plus que 3
for d := k+1 to 1:
    for R in d-element subsets:
    filtration_value := verification(P,R)
    if filtration_value != l'infini:
        add [R,filtration_value] to Complex
```