Реализация вероятностного алгоритма вычислительной топологии

Автор: Леонид Владимирович Сенюков, 17.Б.10-мм
Научный руководитель: д.ф.-м.н., проф. А.Н.Терехов
Консультант: научный сотрудник института Макса Планка Биологической Кибернетики Л.А.Федоров

Санкт-Петербургский государственный университет
Направление "Системное программирование"

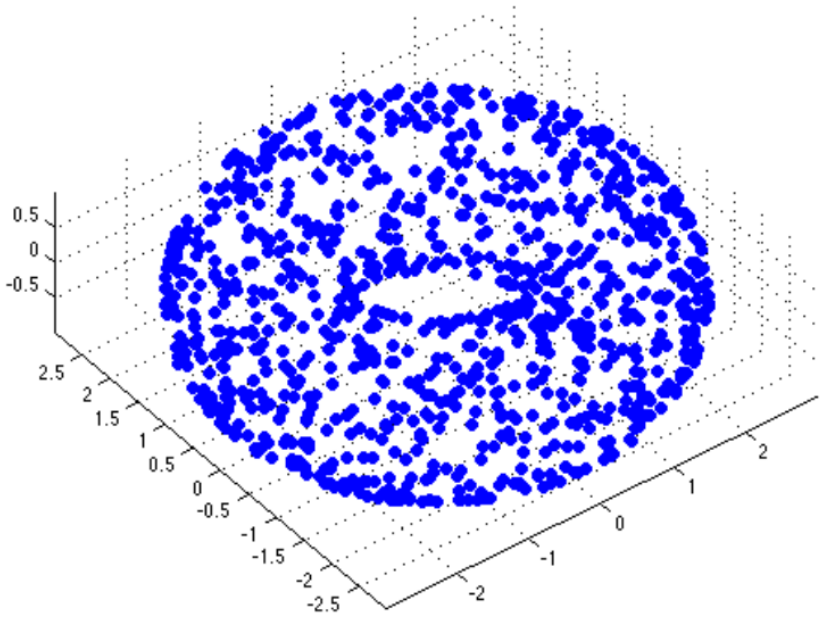
11 мая 2020г.

Введение

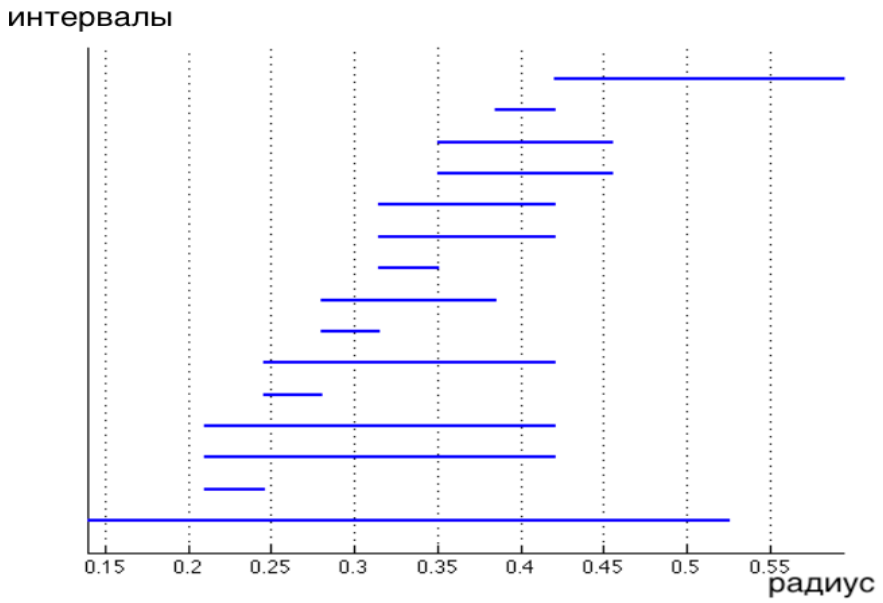
Мотивация:

- Анализ функциональных сетей мозга

Введение



Аппроксимация поверхности тора



Полученная персистентная диаграмма в размерности 2

1

[1] P.Bubenik, 2012, *Towards statistical topology: homology, persistent homology and persistence landscapes*

Обзор предметной области

	1	2	3	4	5	6	7
1				1	1		
2				1		1	
3					1	1	
4							1
5							1
6							1
7							

Матрица граничного оператора над полем $\mathbb{Z}_2$

Стандартный алгоритм построения персистентного ландшафта

Этапы: [2]

- Построение фильтрации
- Составление матрицы граничного оператора
- Редуцирование матрицы
- Считывание персистентной диаграммы
- Преобразование диаграммы в персистентный ландшафт

Проблема:

- Экспоненциальная сложность

Решение :

- Вероятностный метод [3]

---

[2] N. Otter, 2017, *A Roadmap for the Computation of Persistent Homology*

[3] F. Chazal, 2015, *Subsampling methods for persistent homology*

Постановка задачи

Целью работы является реализация алгоритма, применимого для решения задачи построения персистентного ландшафта вероятностным методом

Задачи:

- Изучить предметную область
- Реализовать алгоритм, получающий на вход облако точек и строящий по нему усредненный персистентный ландшафт
- Разработать набор тестовых входных данных
- Провести тестирование

Обзор существующих инструментов

Топологический анализ данных:

- GUDHI (2014) [4]
- Ripser (2015)

Работа с матрицами:

- PHAT (2013)

[4] *https://github.com/GUDHI/gudhi-devel*

Первый подход к решению задачи

Алгоритм [5]:

- Генерируем несколько выборок точек из облака
- Вычисляем персистентный ландшафт на каждой выборке
- Считаем средний ландшафт

Подход:

- Вычисления на каждой выборке проводим параллельно друг с другом
- Используем средства языка C++, Gudhi и Ripser
- Используем пул потоков

[5] F. Chazal, 2015, *Subsampling methods for persistent homology*

Тестирование

Тестирование проводилось на Macbook Pro с 2.6GHz 4-ядерным процессором 16 ГБ RAM

- m = 500 точек в трехмерном пространстве
- Координаты точек нормированы
- r = 0.5
- k = 10 выборок размера n = 200

Сравнение времени работы

Алгоритм	Время работы (мс)	CO (мс)
GUDHI	119 283	1 165
Ripser	3 140	151
Алгоритм на основе Gudhi	9 751	502
Алгоритм на основе Ripser	667	47

Второй подход к решению задачи

Подход:

- Сохраним многопоточность
- Хотим переиспользовать результаты редуцирования матриц на предыдущих выборках

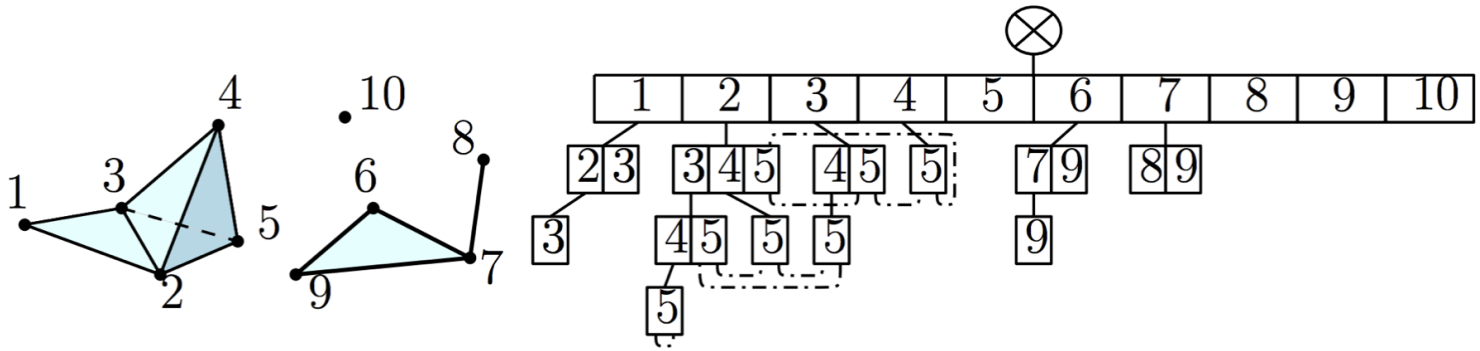
Недостатки:

- Необходима синхронизация

Потребовались библиотеки:

- Потокобезопасные структуры данных – TBB
- Редуцирование матриц – PHAT (поддерживает OpenMP)

Симплициальное дерево



Пример симплициального дерева для облака из 10 точек [6]

- Одно дерево на все потоки
- Каждая вершина – симплекс фильтрации
- Помечаем занулившиеся столбцы матрицы через вершины дерева

[6] Jean-Daniel Boissonnat, Clement Maria, 2012, *The Simplex Tree: An EfficientData Structure for General Simplicial Complexes*

Тестирование

- m = 500 точек в трехмерном пространстве
- Координаты точек нормированы
- r = 0.5
- k = 10 выборок размера n = 200

Полученные результаты:

- Среднее число обрабатываемых в одной выборке симплексов высшей размерности – 2 000 000
- Не стали рассматривать 1 000 000 симплексов
- Уменьшение размеров редуцируемых матриц на 5%

Результаты тестирования

Сравнение времени работы

Алгоритм	Время работы (мс)	CO (мс)
GUDHI	119 283	1 165
Алгоритм на основе GUDHI	9 751	502
Альтернативный алгоритм	63 712	950

Результаты

- Изучены основные подходы, применяемые в области
- Разработана и реализована структура данных "Общее симплициальное дерево"
- Реализовано две версии алгоритма, в том числе на основе разработанной структуры данных
- Разработан набор тестов для оценки эффективности решений
- Проведено тестирование и получены данные об эффективности решений [7]

[7] *https://github.com/leoneed03/SamplingLandscape*