Отчет №5

Амосов Федор

12 января 2014 г.

Алгоритм

Итак, вспомним наш алгоритм построения графа Делоне D на наборе точек P через склеивание меньших подграфов.

1. Пусть S_i — случайный поднабор точек из P размера m;
2. Пусть P_i — разбиение точек P по клеткам Вороного C_i набора точек S_i (каждое P_i — это множество точек);
3. Построим граф Делоне D_i на каждом множестве P_i рекурсивным вызовом этого алгоритма. Пусть каждый наш граф Делоне хранит помимо построенного графа еще и L_i — список смежных каждой грани симплексов (треугольников). Далее нам будет полезен тот факт, что размер всех L_i равен $O(n)$.
4. Проделаем с каждым D_i следующую операцию. Возьмем все граничные треугольники (находятся из L_i). Запустим dfs, описанный в предыдущем отчете, на графе соседних треугольников. С помощью него мы найдем все «плохие» треугольники. Выбросим все плохие треугольники из D_i (вместе с соответствующими ребрами и вершинами);
5. Сконструируем множество точек V. Добавим в него все точки границ $ConvP_i$ (находятся из L_i). Так же добавим в него все вершины найденных плохих треугольников;
6. Построим G — граф Делоне на V. Сделаем мы это рекурсивным вызовом этого алгоритма, или каким–нибудь **другим** построителем графов Делоне, если точек в V достаточно мало*;
7. Найдем в G те ребра, которые либо,
   - Связывают вершины разных D_i;
   - Были удалены в ходе уничтожения плохих треугольников.
8. Получим итоговый граф Делоне D вставкой этих ребер в объединение D_i.

На сей момент, корректность этого алгоритма была «проверена» только многочисленными экспериментами. Но сейчас нам будет интересен другой вопрос. Сколько этот алгоритм работает (в количестве операций)?

Сложность на малых размерностях

Итак, пусть $T(n)$ — время работы этого алгоритма на наборе из n d–мерных точек, где d — небольшая константа. Составим рекуррентное соотношение на $T(n)$. Предположения, в которых мы будем это делать,

- Все $ConvP_i$ имеют высокую выпуклость (не выстраиваются в линии)
- Все P_i имеют похожие размеры

Добиться этого можно взяв m (число множеств P_i) достаточно большим. Будем считать m константой.

Итак, из чего складывается $T(n)$,

1. Выбор m случайных точек — $O(m) = O(1)$

2. Разбиение всех точек по m клеткам. С учетом того, что m — константа, мы можем это сделать за $O(nf(m)) = O(n)$, где $f(m)$ — некоторая малая функция типа $\log m$ и т.п.

3. Построение всех D_i — $mT(\frac{n}{m})$

4. Нахождение всех граничных треугольников (из L_i) + запуск всех dfs — $O(n) + O(mg(\frac{n}{m})\log n)$, где $g(n)$ — ориентировочное число точек границы выпуклой оболочки случайного множества из n точек. Для d–мерного случая $g(n) = O(dn^{\frac{d-1}{d}}) = O(n^{\frac{d-1}{d}})$. $\log n$ вылезает из–за поиска точки в круге при проверке треугольника на «хорошесть», если у нас уже построен поисковый индекс на P (или его частях). Тем самым, сложность получается такой, $O(n) + O(dn^{\frac{d-1}{d}}\log n) = O(n)$.

5. Конструирование V — $O(n)$.

6. Построение G — $O(T(|V|)) = O(T(mg\left(\frac{n}{m}\right))) = O(T(n^{\frac{d-1}{d}})) = o(T(n))$, в случае рекурсивного вызова. Опыт показывает, что если n достаточно велико, то вызов внешнего алгоритма будет происходить только на задачах малых, по сравнению с n, размеров, поэтому его временем работы можно пренебречь.

7. Нахождение нужных ребер — $O(|G|)$ = [d не большое] = $O(|V|) = O(n^{\frac{d-1}{d}})$

8. Вставка ребер (удаление уже было произведено на этапе нахождения плохих треугольников) — $O(|G|) = O(n^{\frac{d-1}{d}})$

Итого,

$$T(n) = O(1) + O(n) + mT\left(\frac{n}{m}\right) + O(n) + O(n) + o(T(n)) + O(n^{\frac{d-1}{d}}) + O(n^{\frac{d-1}{d}})$$

$$T(n) = mT\left(\frac{n}{m}\right) + O(n)$$

Вспомним основную теорему о рекуррентном соотношении.

Теорема

Если

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^c)$$

То

$$T(n) = \begin{cases} O(n^c), & c > \log_b a \\ O(n^c \log n), & c = \log_b a \\ O(n^{\log_b a}), & c < \log_b a \end{cases}$$

Воспользуемся ей для нашего случая. Итого получается,

$$T(n) = O(n \log n)$$

И это в том предположении, что m и d достаточно малые. Тем самым, мы разработали алгоритм, который на малых размерностях и на достаточно большом числе точек работает за $O(n \log n)$. Анализ сложности при больших d будет позже.

Открытые задачи

Обозначения те же, что и в алгоритме. Будем рассматривать те D_i, которые были до пункта 4.

1. Рассмотрим следующий неориентированный граф T на $\cup_{i=1}^{m} D_i$. Его вершинами будут треугольники и еще одна выделенная вершина A. Между двумя треугольниками будет ребро, если они имеют общую грань. Между треугольником и A будет ребро, если треугольник будет находиться на границе некоторого D_i. Рассмотрим подграф T', индуцированный на плохие треугольники и на вершину A. Доказать, что T' связен.

2. Рассмотрим граф G' как D без ребер всех D_i. Доказать, что множество ребер G' совпадает с множеством ребер G, которые соединяют разные D_i.

3. Пусть O — центр описанной сферы около некоторого плохого треугольника. Пусть O лежит в клетке Вороного C_i. Доказать, или опровергнуть то, что ближайшая к O точка из P принадлежит либо C_i, либо клетке, соседней к C_i.

4. Найти асимптотику числа граней выпуклой оболочки случайного набора точек в многомерном пространстве.