

# Отчет №1

Амосов Федор

6 ноября 2013 г.

## 1 Постановка задачи

Пусть задано метрическое пространство  $(\mathbb{R}^m, d)$ . Дан набор точек  $P = \{p_i\}_{i=1}^n$  из  $\mathbb{R}^m$ . Требуется построить структуру на  $P$ , которая позволит эффективно отвечать на следующие запросы:

- Найти ближайшую точку из  $P$  к данной точке  $q$  (NN),
- Найти  $k$  ближайших точек из  $P$  к данной точке  $q$  (kNN),

**Вопрос 1.** Я прав, что под эффективностью мы будем подразумевать не алгоритмическую сложность, меньшую, чем  $\Theta(n)$ , а хорошее время работы на реальных датасетах?

## 2 Структуры данных

### 2.1 M-tree

Это абстрактная структура данных. Она представляет собой дерево поиска, листьями которого являются точки из  $P$ , а остальными вершинами — шары в  $\mathbb{R}^m$ . Такие вершины содержат ссылки либо на поддеревья, либо на листья. Все шары в поддереве должны содержаться в шаре корня этого поддерева. Если вершина ссылается на лист, то соответствующая точка должна содержаться в шаре этой вершины.

#### 2.1.1 Построение

Алгоритм построения на точках  $P$  выглядит следующим образом.

- Разбиваем  $P$  на части, соответствующие ближайшим соседям некоторых точек из  $P$ ,
- В каждой части строим  $M$ -дерево,
- Собираем полученные деревья в одно дерево, устанавливая над ними корневую вершину.

**Замечание 1.** Построение надо как-то аккуратно делать, чтобы покрыть шарами «все точки».

**Замечание 2.** Корневой вершине можно дать шар с центром в центроиде  $P$  и радиусом, покрывающим  $P$ .

**Замечание 3.** Базой этого рекурсивного алгоритма можно считать следующее. Если размер  $P$  достаточно мал, то можно закончить текущее построение, сделав листьями точки  $P$ .

**Вопрос 2.** Как тут дела со сбалансированностью? В статье написано пару строк на эту тему, но как-то непонятно. Там говорят, что для балансировки нужно проводить еще операции (split и т.п.). Я не совсем понимаю, к чему приведут эти операции.

**Вопрос 3.** Стоит ли нам задумываться об онлайн-построении структур? Т.е. стоит ли разбирать операции insert и т.п.?

### 2.1.2 Запрос

Разберем запрос NN.

**Вопрос 4.** Как нормально делать запрос в таком дереве? В статье я что-то не вижу. На Википедии этого тоже нет. Мой вариант нахождения соседа точки  $q$ :

- Если потомки текущей вершины являются точками из  $P$ , то находим среди них самую близкую к  $q$  и возвращаем ее.
- Иначе, находим все шары, соответствующие потомкам текущей вершины, в которых есть точка  $q$ . Если таких нет, то прекращаем работу.
- Делаем NN запрос в каждое поддерево. Получаем набор точек-кандидатов. Среди них выбираем лучшую и возвращаем.

**Вопрос 5.** Что делать, если запрос  $q$  находится далеко-далеко от множества  $P$  и не покрывается ни одним шаром?

**Вопрос 6.** Как тут дела обстоят с аналитическими оценками на время работы? Это касается и построения тоже.

## 2.2 R-tree

**Вопрос 7.** Правда, что R-tree — это реализация абстрактной структуры M-tree, при построении использующая метрику  $L_\infty$ ?

## 2.3 VoR-tree

Это R-tree, в котором в каждом листе, помимо самих точек из  $P$ , хранится еще и информация об их ячейках Вороного. Опишем, как это может ускорить процесс поиска. Рассмотрим NN запрос для точки  $q$ .

Поиск мы проводим с помощью BFS в R-tree. В BFS мы используем очередь с приоритетами из вершин дерева. В качестве приоритетов мы используем расстояние от точки  $q$  до прямоугольника, соответствующего вершине дерева. На этапе обработки листьев, мы пытаемся найти ту вершину из листа, в чьей ячейке Вороного находится  $q$ .

Для запроса kNN используется BFS на графе Делоне от NN вершины. Там все просто.

**Вопрос 8.** Зачем хранить информацию в листьях R-дерева о соседях точек из  $P$  в графе Делоне (см. рисунок 2 в соответствующей статье)? Мы используем граф Делоне, но не в самом R-tree (см. kNN)

## 3 Видимые мной задачи

- Создать проект. Написать интерфейс структуры, решающей нашу задачу (NN и kNN). Сделать простейшую(наивную) реализацию.
- Найти интересный датасет. Сделать тесты с помощью наивной реализации.
- Написать R-tree как реализацию интерфейса. Сравнить эффективность наивной реализации и R-tree.
- Написать быстрое построение диаграммы Вороного. Подойдет, к примеру, алгоритм Форчуна.
- Улучшить R-tree до VoR-tree. Сравнить эффективность R-tree и VoR-tree.
- Придумать, как распараллелить построение VoR-tree, используя интерфейс MapReduce.
- Реализовать распараллеливание, используя Hadoop.
- Научиться строить SQL индекс по VoR-tree.
- Написать фреймворк, реализующий весь pipeline. На вход — набор точек, на выход — SQL индекс.
- Добавить в фреймворк другие алгоритмы для анализа сравнительной эффективности.