Approximate nearest neighbor search

Постановка задачи

Задача поиска ближайшего соседа заключается в нахождении среди множества элементов, расположенных в метрическом пространстве, элементов близких к заданному, согласно некоторой заданной функции близости, определяющей это метрическое пространство.

Формализуем ее для Евклидова пространства:

Пусть у нас есть конечное множество $Y \subset R^D$, состоящее из n векторов.

Тогда наша задача найти NN(x), $x \subset R^D$: $NN(x) = \underset{y \in Y}{\operatorname{argmax}}\, d(x, y)$

Рассматриваемые методы решения

- K-d tree
- Product-quantization

K-d tree

- Определение: несбалансированное бинарное дерево поиска для хранения точек k-мерного пространства.

- Идея: каждый уровень дерева соответсвует одному из измерений, на 2 части элементы множества разбивается в зависимости от значения координаты этого измерения.

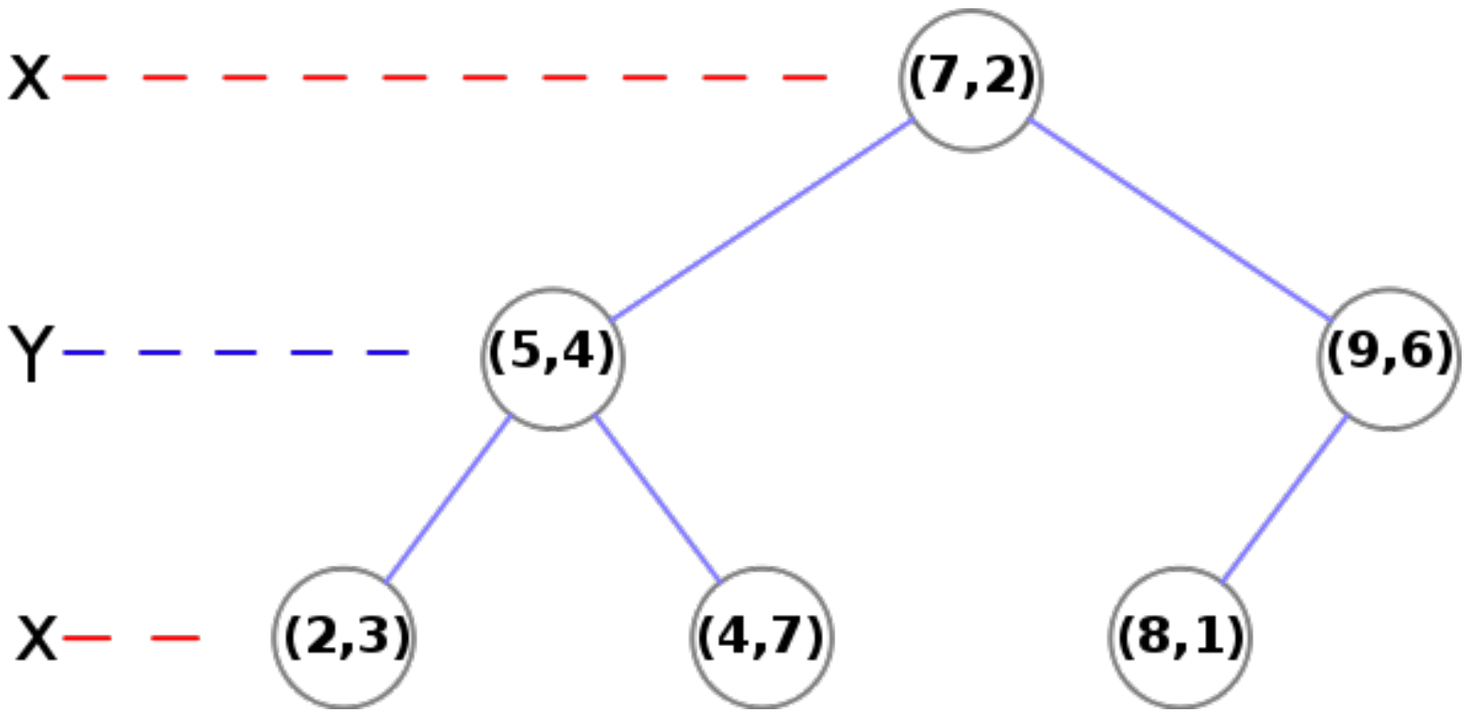
- В каждом листе хранится непосредственно сама точка, а так же bounding box (т.е. прямые, которые ее ограничивают)

Построение k-tree

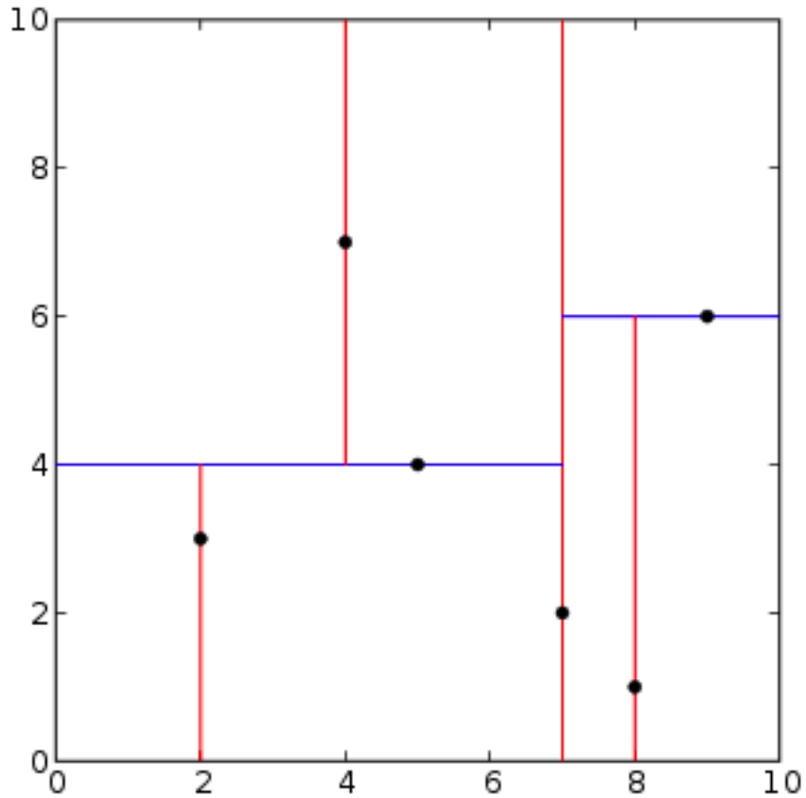
- Разобьём все точки прямой по 1 координате в соотвествии с медианой. Получим подмножества для левого и правого ребёнка.
- Далее проведем аналогичную операцию на полученных подмножествах, уже действую с следующим измерением.
- Будем циклично повторять операцию, пока не закончатся элементы

Пример k-tree

Resulting k-tree



k-d tree decomposition for the point set (2,3), (5,4), (9,6), (4,7), (8,1), (7,2)



Алгоритм поиска ближайшего соседа.

- Найдем ближайшую точку C спускаясь по дереву аналогично построению.
- При рекурсивном подъеме каждый раз проверяем, может ли текущая вершина улучшить ответ, и если может, то C = cur_node.
- Проверяем, нет ли возможности улучшить ответ, опустившись в соседнее поддерево. Пусть мы ищем ответ для Q и рассматриваем поддерево с корнем T. Тогда если dist(C, Q) < dist(Q, T.bounding_box), то мы не можем улучшить ответ и продолжаем подъем. В противном случае в соседнем поддереве могут быть ближайшие точки и мы начинаем их поиск с начала алгоритма.

Время работы и память

- Построение: O(nlogn)
- Поиск элемента: O(logn) в среднем, O(n) - в худшем случае.
- Высота: O(logn)
- Память: O(n)

Product quantization

Formally, a quantizer is a function q from $x \subset R^d$ **to vector** $q(x) \subset C = \{c_i; i \subset I\}$
where the index set I is from now on assumed to be finite: I = 0 . . . k – 1

The reproduction values ci are called centroids

The set of reproduction values C is the codebook of size k

Vector x quantized to its nearest codebook centroid,:

$$q(x) = \underset{c_i \in C}{\operatorname{argmax}}\, d(x, y)$$

Алгоритм поиска

Индексация вектора y

1) quantize y to $q_c(y)$

2) compute the residual $r(y) = y - q_c(y)$

3) quantize $r(y)$ **to** $q_p(r(y))$

4) add a new entry to the inverted list to, corresponding $q_c(y)$ **.**

It contains the vector identifier and the binary code

(the product quantizer's indexes)

Алгоритм поиска

Непосредственно поиск ближайшего соседа

1) quantize x to its w nearest neighbors in the codebook $q_c(y)$

2) compute the squared distance $d(u_j(r(x), c_{j,i})^2$

for each subquantize j and each of its centroid $c_{j,i}$

3) compute the squared distance between r(x) and all the indexed vectors of the inverted list

Using the subvector-to-centroid distances computed in the previous step,

this consists in summing up m looked-up values

4) select the K nearest neighbors of x based on the estimated distances .

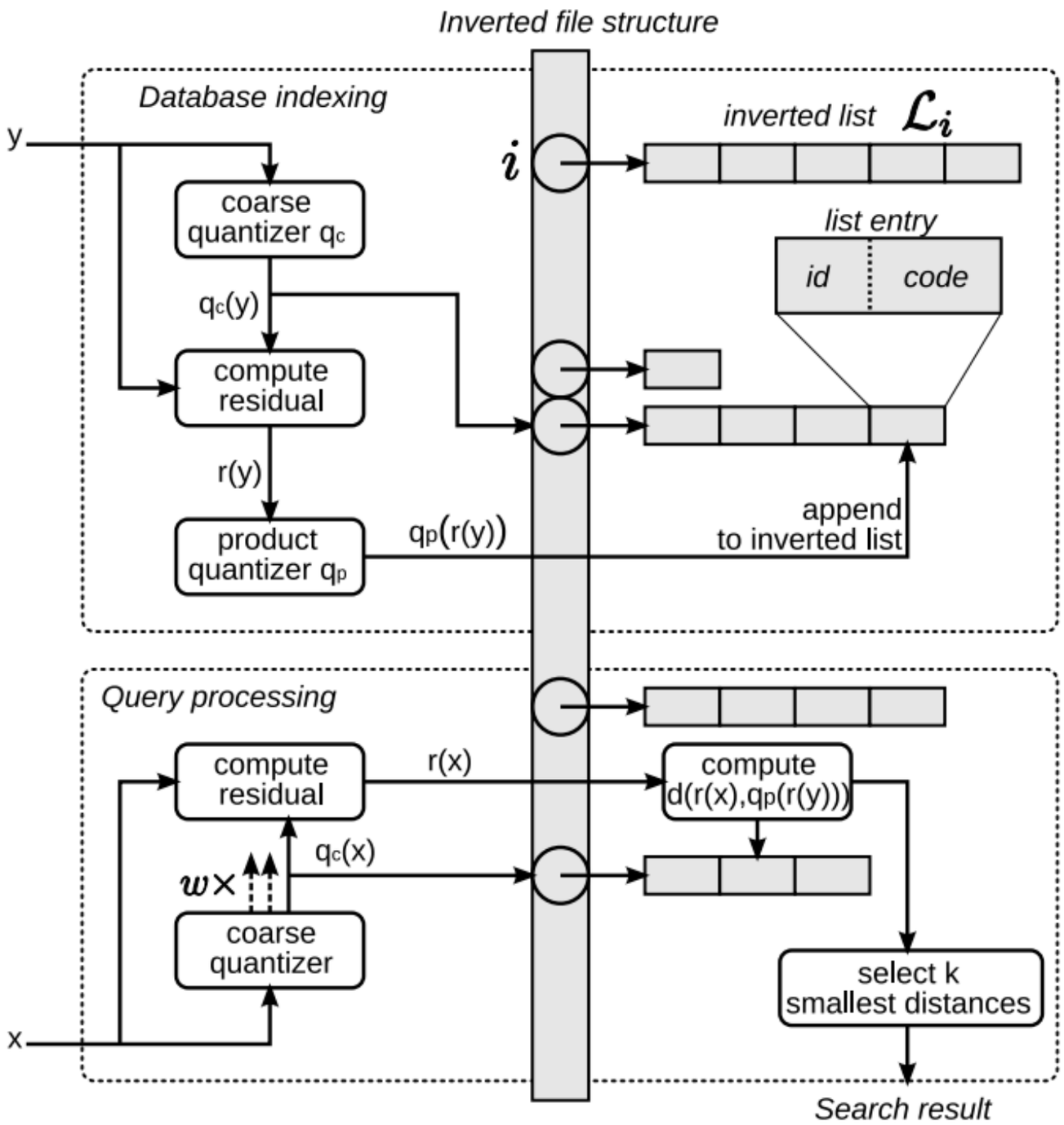
This is implemented efficiently by maintaining a Maxheap structure of fixed capacity,

that stores the K smallest values seen so far

After each distance calculation, the point identifier is added to the structure only

if its distance is below the largest distance in the Maxheap

Схематичное изображение алгоритма



Источники

- https://lear.inrialpes.fr/pubs/2011/JDS11/jegou_searching_with_quantization.pdf

- https://en.wikipedia.org/wiki/K-d_tree#cite_note-Friedman:1977:AFB:355744.355745-9

- https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/kdtrees.pdf

- https://pdfs.semanticscholar.org/c17e/d7402d4cc841df0b098c162e80ef85caa10a.pdf