Метрические методы классификации
Approximate nearest neighbour search

Абдумуталов Рустам
Баранов Юрий
25 сентября 2017

Задача классификации

В задачах классификации часто классы образуют компактно локализованные подмножества. Это предположение принято называть гипотезой компактности.

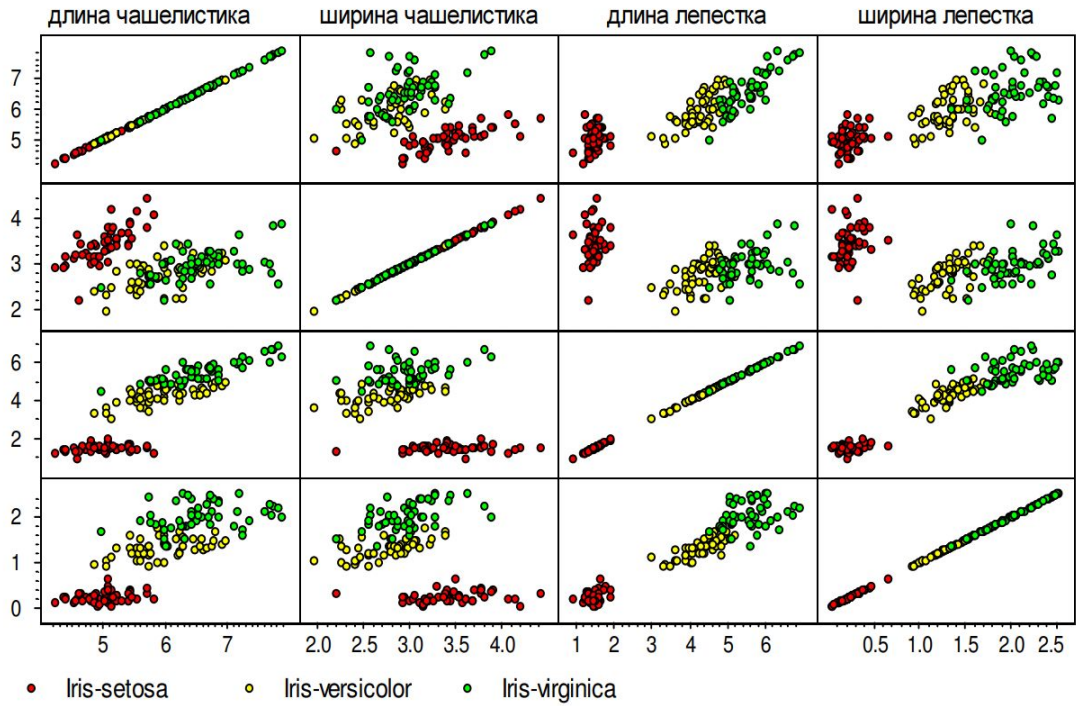
X - объекты, Y - ответы $X^l = (x_i, y_i)_{i=1}^l$ - обучающая выборка

Для формализации понятия «сходства» вводится функция расстояния в пространстве объектов X:

$$\rho : X \times X \to [0, \infty)$$

может не удовлетворять аксиомам метрики

$n = 4$ признака, $|Y| = 3$ класса, длина выборки $\ell = 150$.



Классификация

Для произвольного объекта u строится вариационный ряд

$$\rho(u, x_u^{(1)}) \leqslant \rho(u, x_u^{(2)}) \leqslant \cdots \leqslant \rho(u, x_u^{(\ell)})$$

ответ метрического алгоритма для объекта u по обучающей выборке X вычисляется по формуле

$$a(u; X^\ell) = \arg\max_{y \in Y} \Gamma_y(u, X^\ell); \qquad \Gamma_y(u, X^\ell) = \sum_{i=1}^{\ell} [y_u^{(i)} = y]\, w(i, u)$$

где весовая функция w(i, u) оценивает степень важности i-го соседа для классификации объекта u функция Г(u, X) называется оценкой близости объекта u к классу y

Разновидности метрических классификаторов

Метрический классификатор определён с точностью до весовой функции w. Обычно она выбирается неотрицательной, не возрастающей по i. Это соответствует гипотезе компактности, согласно которой чем ближе объекты u и x (i) u , тем выше шансы, что они принадлежат одному классу.

Метрические алгоритмы относятся к методам ленивого обучения (lazy learning)

Метрические алгоритмы классификации относятся также к методам рассуждения по прецедентам (case-based reasoning, CBR)

Метод ближайших соседей (nearest neighbor, NN)

$$w(i,u) = [i = 1]; \qquad a(u; X^\ell) = y_u^{(1)}$$

- Неустойчивость к погрешностям
- Отсутствие параметров, которые можно было бы настраивать по выборке. Алгоритм полностью зависит от того, насколько удачно выбрана метрика ρ
- Низкое качество классификации

Алгоритм k ближайших соседей (kNN)

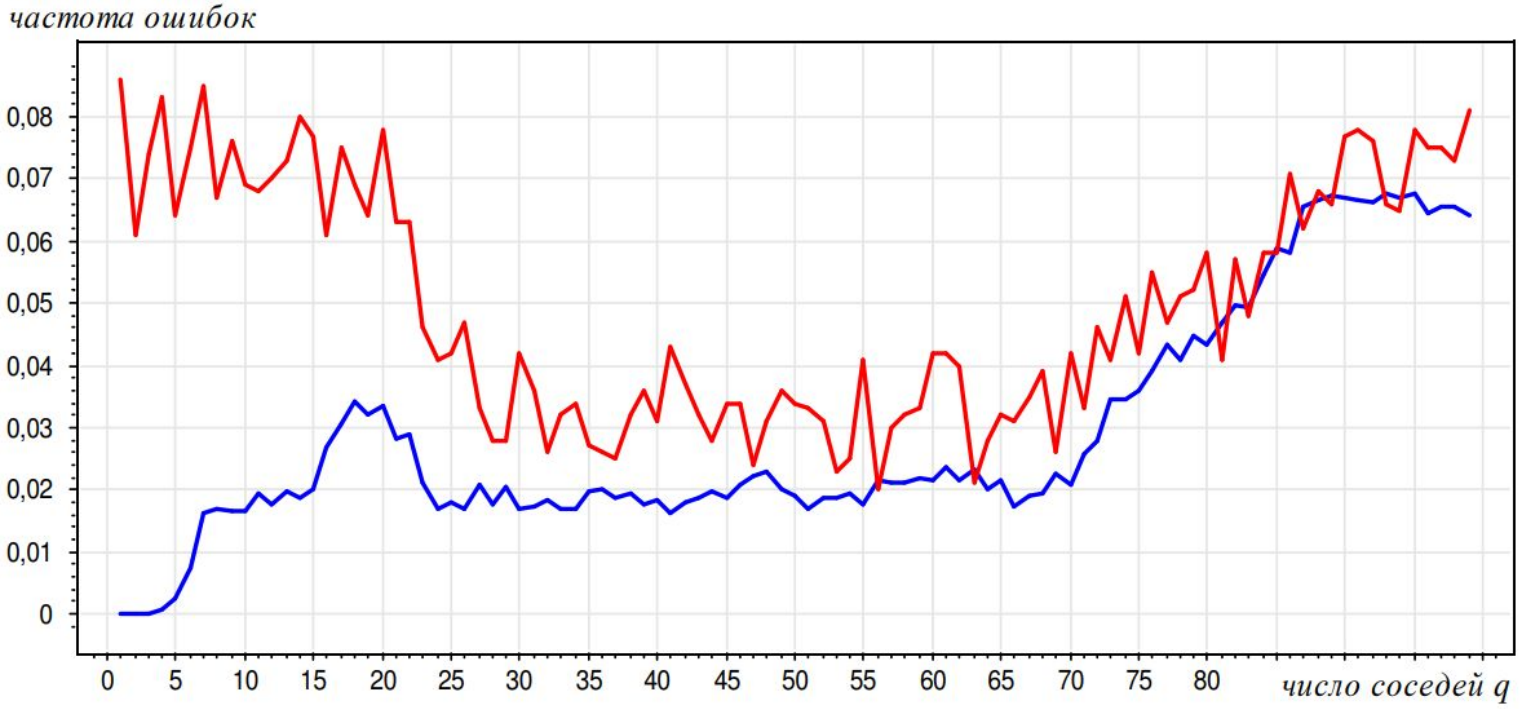
$$w(i,u) = [i \leqslant k]; \qquad a(u; X^\ell, k) = \arg\max_{y \in Y} \sum_{i=1}^{k} [y_u^{(i)} = y]$$

При k = 1 этот алгоритм совпадает с предыдущим, следовательно, неустойчив к шуму. При k = ℓ, наоборот, он чрезмерно устойчив и вырождается в константу

Подбор k: leave-one-out, LOO

$$\mathrm{LOO}(k, X^\ell) = \sum_{i=1}^{\ell} \Big[a\big(x_i; X^\ell \backslash \{x_i\}, k\big) \neq y_i \Big] \to \min_k$$

Пример. Задача UCI: Iris.



— смещённое число ошибок, когда объект учитывается как сосед самого себя

— несмещённое число ошибок LOO

Алгоритм k взвешенных ближайших соседей

$$w(i,u) = [i \leqslant k]\, w_i; \qquad a(u; X^\ell, k) = \arg\max_{y \in Y} \sum_{i=1}^{k} [y_u^{(i)} = y] w_i$$

В отличии от kNN, неоднозначность ответа устраняется, если взять нелинейно убывающую последовательность, скажем, геометрическую прогрессию, где знаменатель прогрессии q ∈ (0, 1) является параметром алгоритма. Его можно подбирать по критерию LOO, аналогично числу соседей k

Недостатки простейших метрических алгоритмов типа kNN

- Приходится хранить обучающую выборку целиком.
- Поиск ближайшего соседа предполагает сравнение классифицируемого объекта со всеми объектами выборки за $O(\ell)$ операций. Проблема решается с помощью эффективных алгоритмов поиска ближайших соседей, требующих в среднем $O(\log(l))$ операций.
- В простейших случаях метрические алгоритмы имеют крайне бедный набор параметров, что исключает возможность настройки алгоритма по данным.

Метод парзеновского окна

$$a(u; X^\ell, h) = \arg\max_{y \in Y} \sum_{i=1}^{\ell} [y_u^{(i)} = y] \, K\left(\frac{\rho(u, x_u^{(i)})}{h}\right)$$

K - функция ядра, h - ширина окна

Зависимость LOO(h), как правило, имеет характерный минимум, поскольку слишком узкие окна приводят к неустойчивой классификации; а слишком широкие — к вырождению алгоритма в константу

Фиксация ширины окна h не подходит для тех задач, в которых обучающие объекты существенно неравномерно распределены по пространству X.

Окно переменной ширины

Финитное ядро — невозрастающая функция K(z), положительная на отрезке [0, 1] и равная нулю вне его.

$$a(u; X^\ell, k) = \arg\max_{y \in Y} \sum_{i=1}^{k} [y_u^{(i)} = y] \, K\left(\frac{\rho(u, x_u^{(i)})}{\rho(u, x_u^{(k+1)})} \right)$$

Заметим, что при финитном ядре классификация объекта сводится к поиску его соседей, тогда как при не финитном ядре (например, гауссовском) требуется перебор всей обучающей выборки.

Метод потенциальных функций

В силу симметричности функции расстояния ρ(u, x) возможен и другой, двойственный, взгляд на метрическую классификацию - ядро помещается в каждый объект обучающей выборки, и “притягивает” объект к своему классу

$$a(u; X^\ell) = \arg\max_{y \in Y} \sum_{i=1}^{\ell} [y_i = y]\, \gamma_i\, K\left(\frac{\rho(u, x_i)}{h_i}\right), \qquad \gamma_i \geqslant 0,\ h_i > 0$$

1: Инициализация: $\gamma_i = 0;\ \ i = 1, \ldots, \ell;$
2: **повторять**
3: выбрать объект $x_i \in X^\ell;$
4: **если** $a(x_i) \neq y_i$ **то**
5: $\gamma_i := \gamma_i + 1;$
6: **пока** число ошибок на выборке не окажется достаточно мало

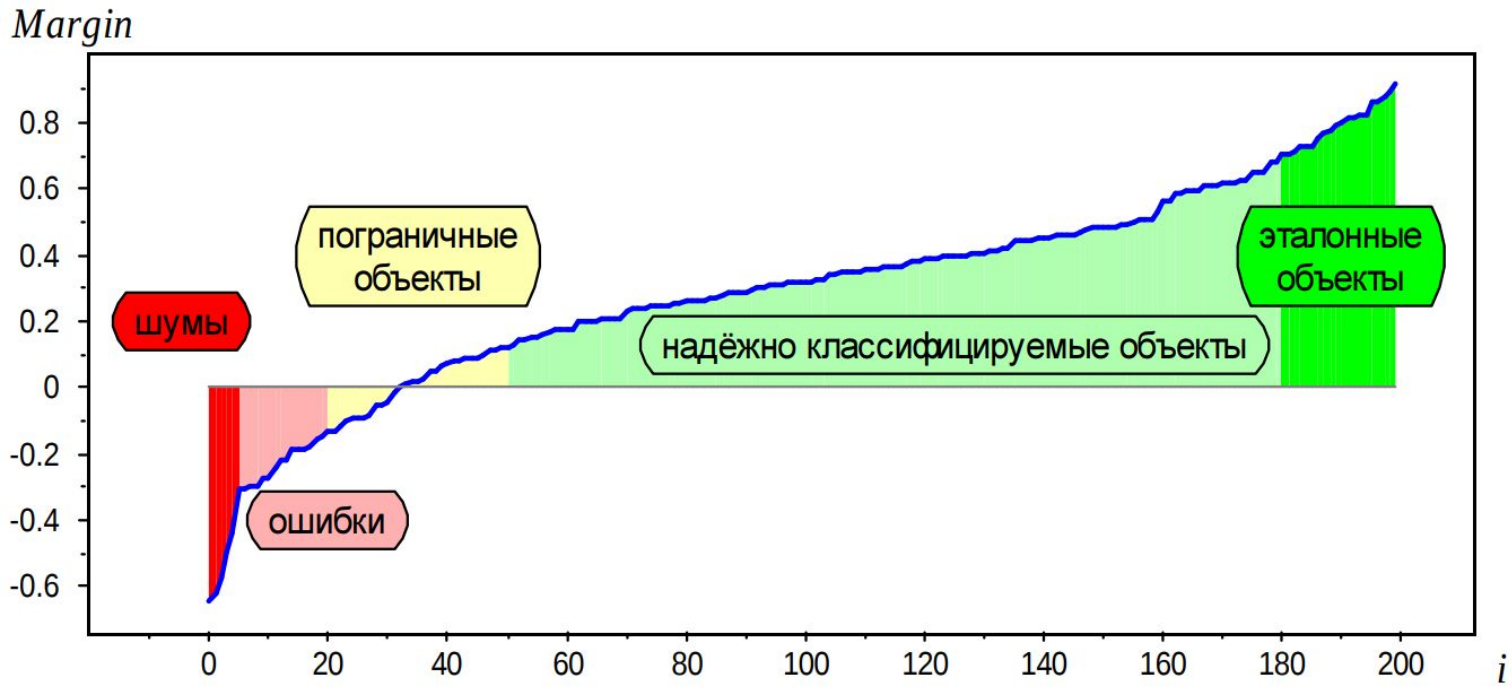
Отбор эталонных объектов

Введем определение отступа объекта (margin): $M(x_i) = \Gamma_{y_i}(x_i) - \max\limits_{y \in Y \setminus y_i} \Gamma_y(x_i)$

Отступ показывает степень типичности объекта. Отступ отрицателен тогда и только тогда, когда алгоритм допускает ошибку на данном объекте.

- Эталонные объекты имеют большой положительный отступ, плотно окружены объектами своего класса и являются наиболее типичными его представителями.

- Неинформативные объекты также имеют положительный отступ. Изъятие из выборки не влияет на качество классификации.


Margin
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
20
40
60
80
100
120
140
160
180
200
i
шумы
пограничные объекты
ошибки
надёжно классифицируемые объекты
эталонные объекты

Алгоритм 3.2. Отбор эталонных объектов STOLP

Вход:

X^ℓ — обучающая выборка;

δ — порог фильтрации выбросов;

ℓ_0 — допустимая доля ошибок;

Выход:

Множество опорных объектов $\Omega \subseteq X^\ell$;

для всех $x_i \in X^\ell$ проверить, является ли x_i выбросом:

  если $M(x_i, X^\ell) < \delta$ **то**

    $X^{\ell-1} := X^\ell \setminus \{x_i\}; \quad \ell := \ell - 1;$

Инициализация: взять по одному эталону от каждого класса:

$\Omega := \left\{ \arg\max_{x_i \in X_y^\ell} M(x_i, X^\ell) \;\middle|\; y \in Y \right\};$

пока $\Omega \neq X^\ell$;

  Выделить множество объектов, на которых алгоритм $a(u; \Omega)$ ошибается:

  $E := \{x_i \in X^\ell \setminus \Omega : M(x_i, \Omega) < 0\};$

  если $|E| < \ell_0$ **то**

    выход;

  Присоединить к Ω объект с наименьшим отступом:

  $x_i := \arg\min_{x \in E} M(x, \Omega); \quad \Omega := \Omega \cup \{x_i\};$

Approximate nearest neighbor search

KD-tree

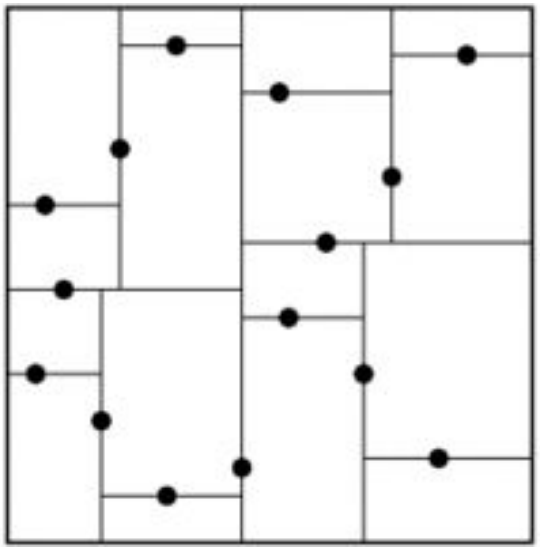
K-мерное дерево - это *несбалансированное* дерево поиска для хранения точек из R^k.

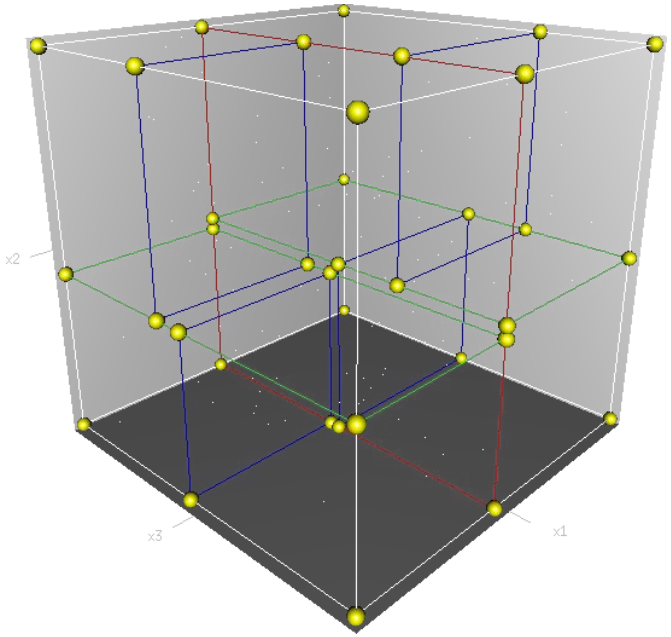
- Каждый *узел* делит множество гиперплоскостью на две части
- Гиперплоскости перпендикулярны осям
- Высота дерева *O(log n),* листья соответствуют 1 элементу выборки

Backtrack - проходим вверх от листа, т.к. в нём не обязательно ближайший

Randomized KD-tree - много деревьев (randomly rotated datapoints)

KD-tree





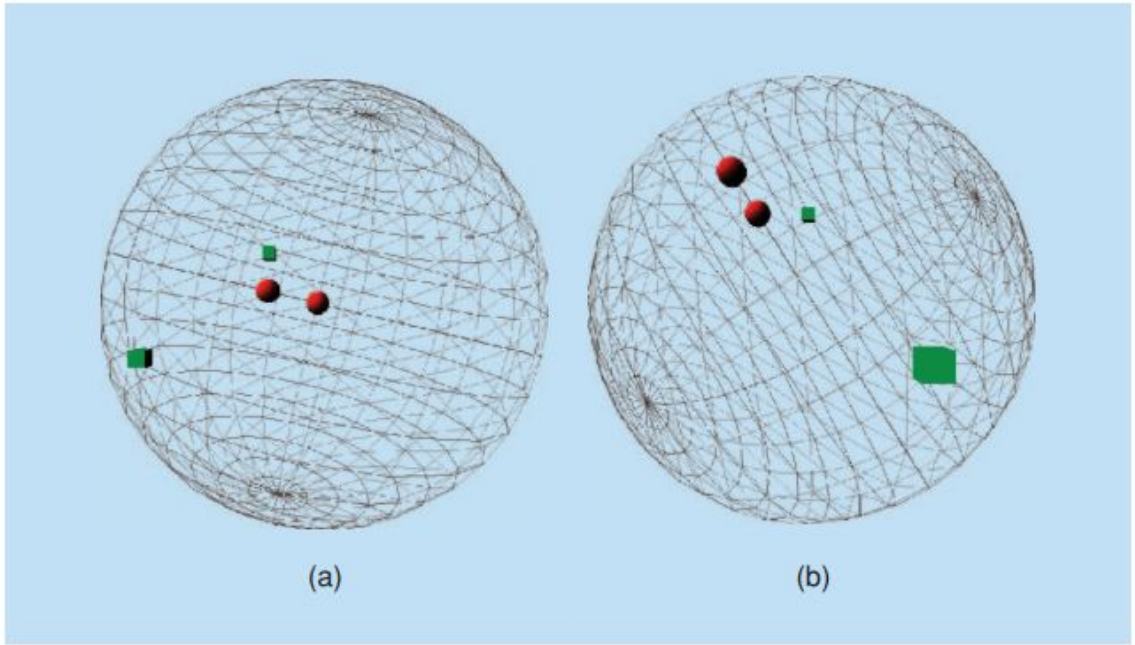
Locality-sensitive hashaing

LSH - вероятностный метод понижения размерности многомерных данных

Основные идеи:

- Подбор хэш функций для некоторых измерений
- Максимизация коллизий для похожих объёктов
- Отображение в множество корзин

Locality-sensitive hashaing



[FIG1] Two examples showing projections of two close (circles) and two distant (squares) points onto the printed page.

The Graph Nearest Neighbor Search Algorithm(GNNS)

Обзор метода:

- Строим **k-NN** graph (*offline*)
- Выполняем **hill-climbing** в **случайном** узле графа (*online*)

GNNS: k-NN Graph

$G = (D, E)$

D - множество узлов (элементы обучающей выборки)

E - множество ребер (связей)

Узел X_i соединён с узлом X_j , если X_j входит в k-NN X_i

Наивное построение - $O(dn^2)$, можно быстрее

GNNS: Approximate *K*-Nearest Neighbor Search

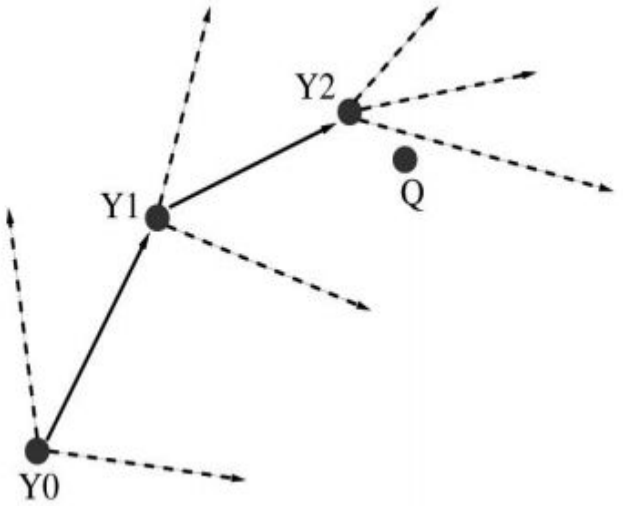
K - количество соседей требуемых вывести для запроса

k - количество соседей для каждого узла в k-NN graph

- Выбираем случайный узел
- *T* раз заменяем узел Y_{t-1} на ближайшего соседа к заданному в запросе
- Возвращаем *K* ближайших из просмотренных узлов

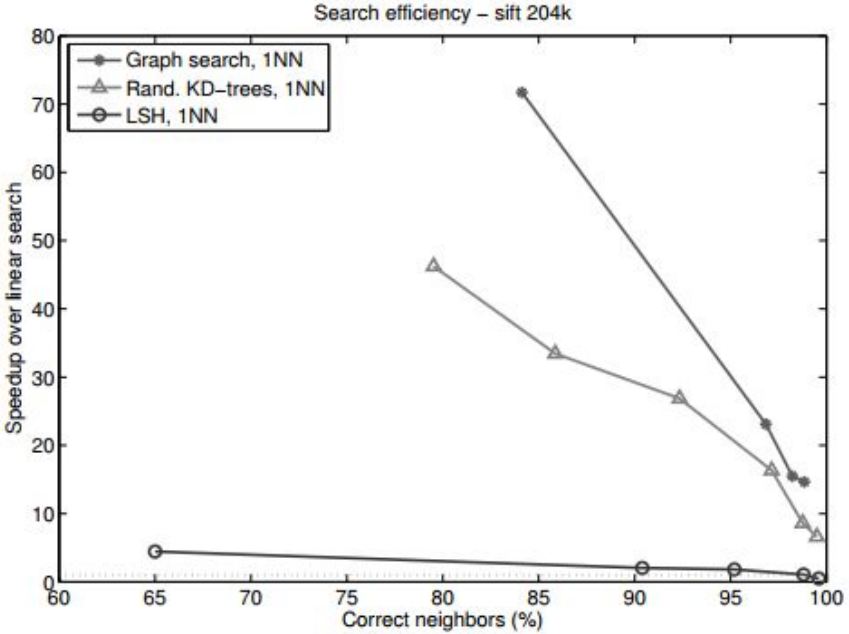
$$Y_t = \underset{Y \in N(Y_{t-1}, E, \mathcal{G})}{\operatorname{argmin}} \rho(Y, Q)$$

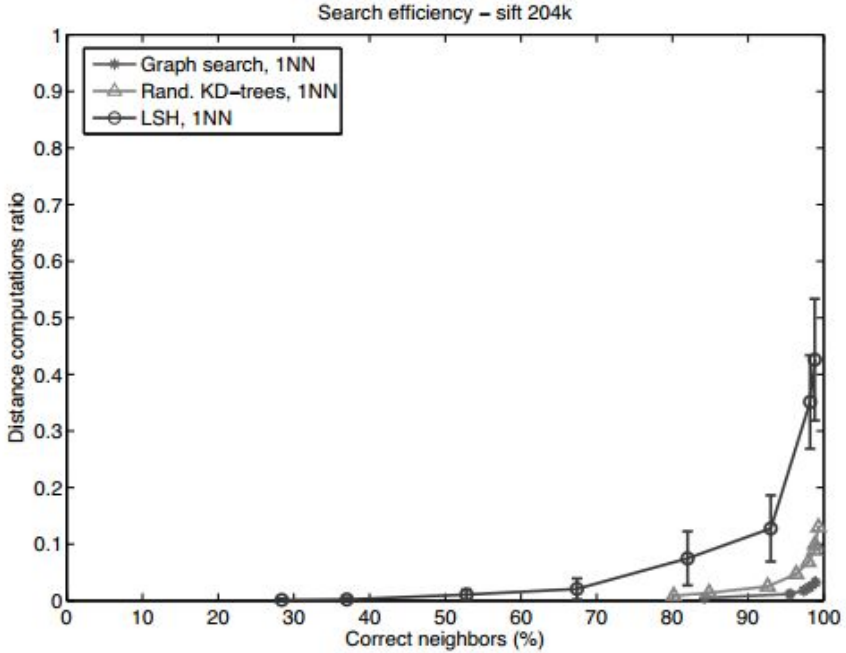
Input: a k-NN graph $\mathcal{G} = (\mathcal{D}, \mathcal{E})$, a query point Q, the number of required nearest neighbors K, the number of random restarts R, the number of greedy steps T, and the number of expansions E.
ρ is a distance function. $N(Y, E, \mathcal{G})$ returns the first E neighbors of node Y in $\mathcal{G}$.
$\mathcal{S} = \{\}$.
$\mathcal{U} = \{\}$.
$Z = X_1$.
for $r = 1, \ldots, R$ **do**
  Y_0: a point drawn randomly from a uniform distribution over $\mathcal{D}$.
  for $t = 1, \ldots, T$ **do**
    $Y_t = \text{argmin}_{Y \in N(Y_{t-1}, E, \mathcal{G})} \rho(Y, Q)$.
    $\mathcal{S} = \mathcal{S} \bigcup N(Y_{t-1}, E, \mathcal{G})$.
    $\mathcal{U} = \mathcal{U} \bigcup \{\rho(Y, Q) : Y \in N(Y_{t-1}, E, \mathcal{G})\}$.
  end for
end for
Sort $\mathcal{U}$, pick the first K elements, and return the corresponding elements in $\mathcal{S}$.



K = 1 and *E = 3*

GNSS vs KD-trees vs LSH





Заключение

- Приближенные методы быстрее
- Однако, менее точны
- На практике приближенные используют чаще

Список литературы и ресурсов

- http://machinelearning.ru/wiki/images/9/9d/Voron-ML-Metric.pdf

- https://ru.wikipedia.org/wiki/K-%D0%BC%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE
- https://ru.wikipedia.org/wiki/Locality-sensitive_hashing
- http://www.ijcai.org/Proceedings/11/Papers/222.pdf
- http://www.slaney.org/malcolm/yahoo/Slaney2008-LSHTutorial.pdf