kNN и приближенный поиск ближайших соседей

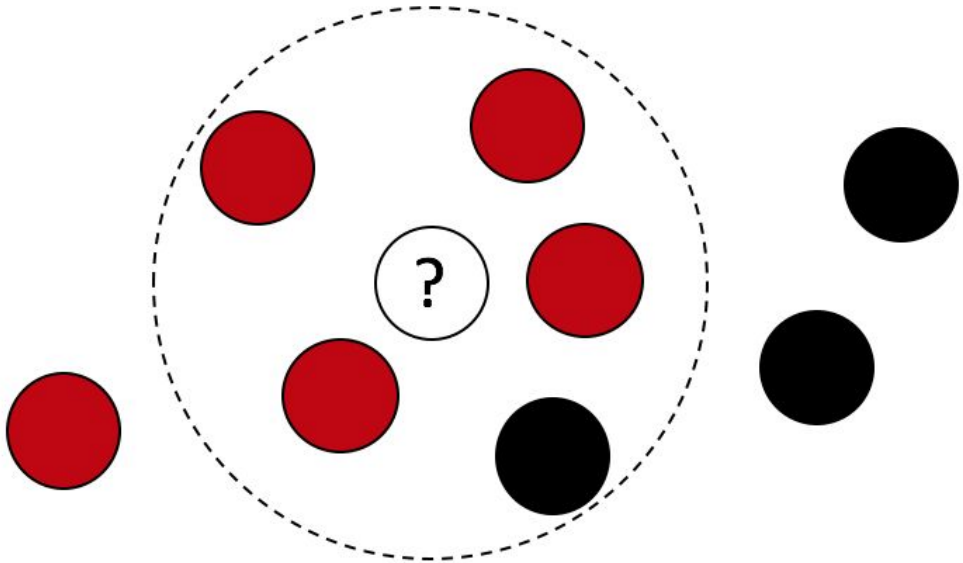
Дарья Виноградова
Сергей Исаев

22.09.2020

Задача kNN

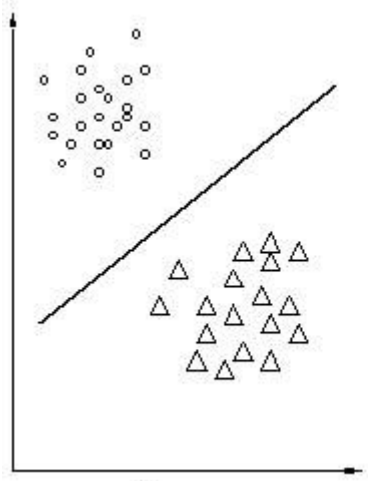
- задача классификации
- есть обучающая выборка
- хотим определять класс для потока новых объектов
- посмотрим на ближайшие объекты из обучающей выборки
- определим лидирующий класс
- отнесем к нему очередной объект

Задача kNN



Гипотеза компактности

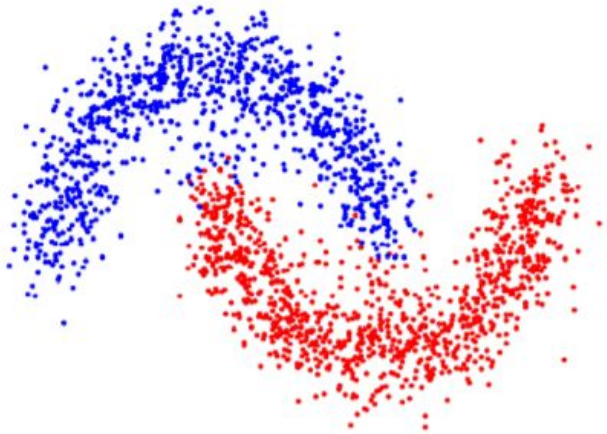
Если мера сходства объектов введена достаточно удачно, то схожие объекты гораздо чаще лежат в одном классе, чем в разных, а классы при этом образуют компактные области.



Формализация похожести объектов

- евклидово расстояние
- Манхэттенское расстояние
- расстояние Левенштейна (для строк)
- ...

k=1



Преимущества:

- простота реализации
- вывод на основе прецедентов

Недостатки:

- приходится хранить всю выборку целиком
- отсутствие настраиваемых параметров
- неустойчивость к выбросам
- низкое качество классификации

Увеличиваем k

Что исправили:

- появился параметр (отбор с помощью LOO)
- меньше чувствительность к шуму

Новая проблема:

- неоднозначность классификации

Что делать с неоднозначностью?

- ввести строго убывающую последовательность вещественных весов, задающих вклад i-го соседа в классификации

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

- метод парзеновского окна (фиксированной и переменной ширины)

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

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

Применение

- поиск
- рекомендательные системы (похожие профили или треки)
- распознавание лиц
- выявление дубликатов
- медицина
- юриспруденция
- ...

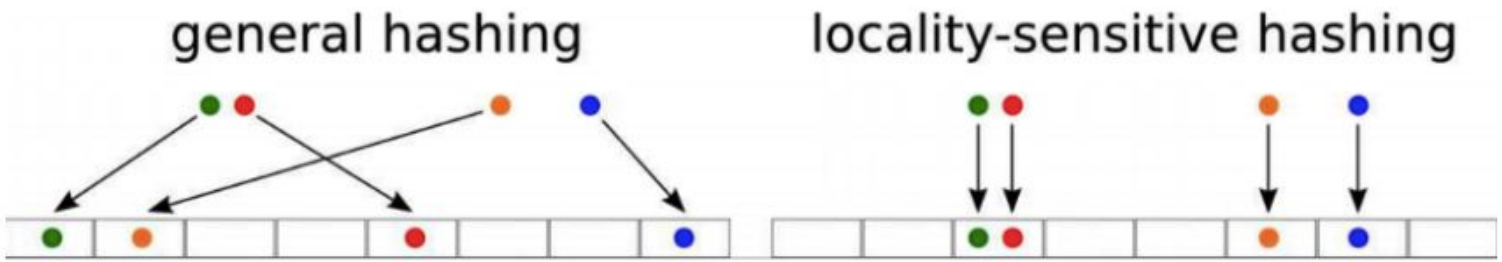
Какие в итоге недостатки?

- всю выборку надо хранить целиком
- выборка должна быть репрезентативной
- долгий поиск соседей

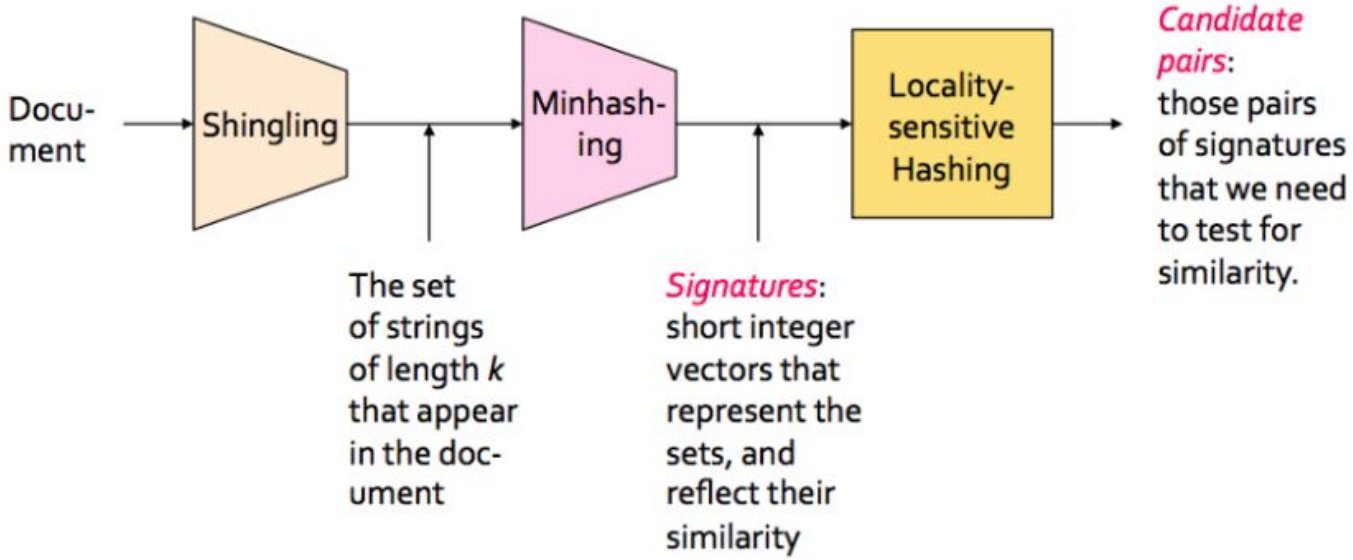
Сконцентрируемся на решении последней проблемы.

MinHash+LSH

Идея: хэшировать объекты так, чтобы для похожих объектов результаты хэш-функции также были похожими



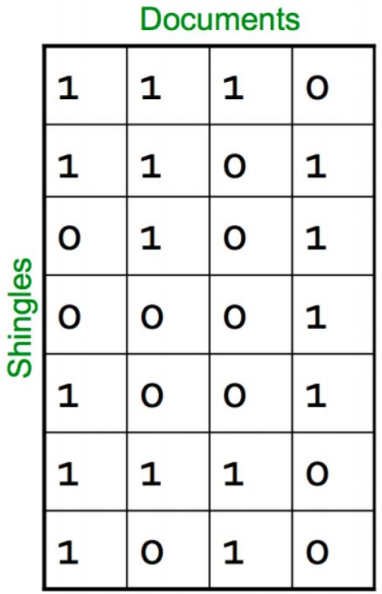
MinHash+LSH


Docu-
ment
Shingling
Minhash-
ing
Locality-
sensitive
Hashing
Candidate
pairs:
those pairs
of signatures
that we need
to test for
similarity.
The set
of strings
of length k
that appear
in the doc-
ument
Signatures:
short integer
vectors that
represent the
sets, and
reflect their
similarity

MinHash+LSH. Shingling (разбиение на k-грамы)

Nadal ⟶ { ‘Na’, ‘ad’, ‘da’, ‘al’ } - 2-грамы
Nadal ⟶ { ‘Nad’, ‘ada’, ‘dal’ } - 3-грамы

- у похожих документов много одинаковых k-грам
- перестановка абзацев мало меняет похожесть

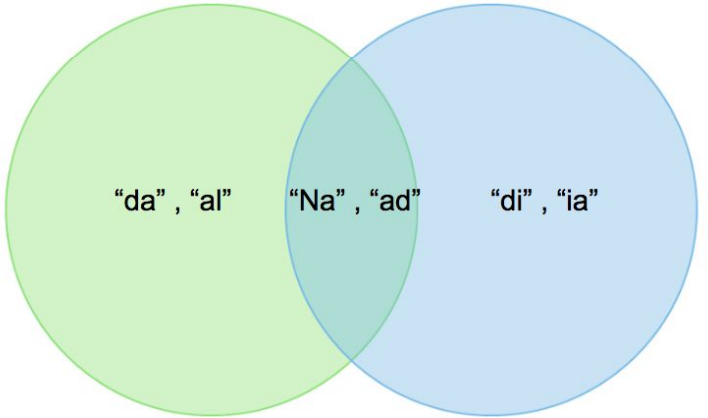


MinHash+LSH. Shingling

Мера похожести: индекс Жаккара

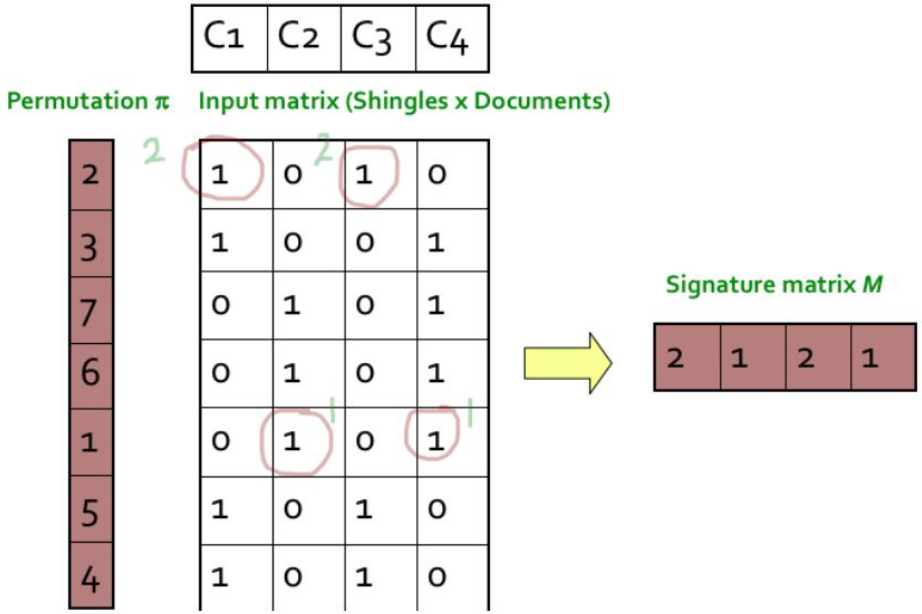
$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Пример. У документов ‘Nadal’ и ‘Nadia’ индекс Жаккара равен 1/3



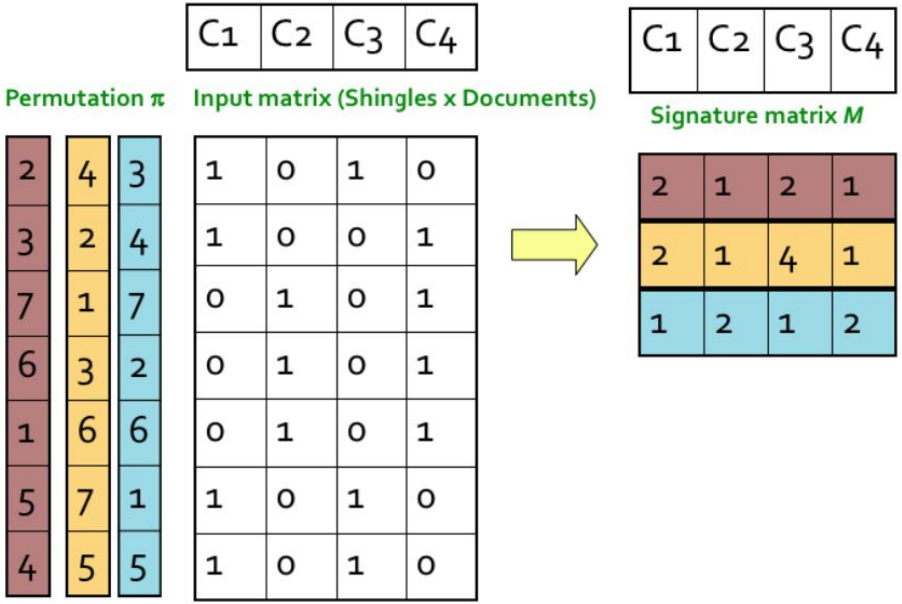
MinHash+LSH. MinHash

Мотивация: хотим понизить размерность матрицы
Хэш-функция: минимальный индекс случайной перестановки, которому соответствует 1 в документе



MinHash+LSH. MinHash

Применим несколько раз:

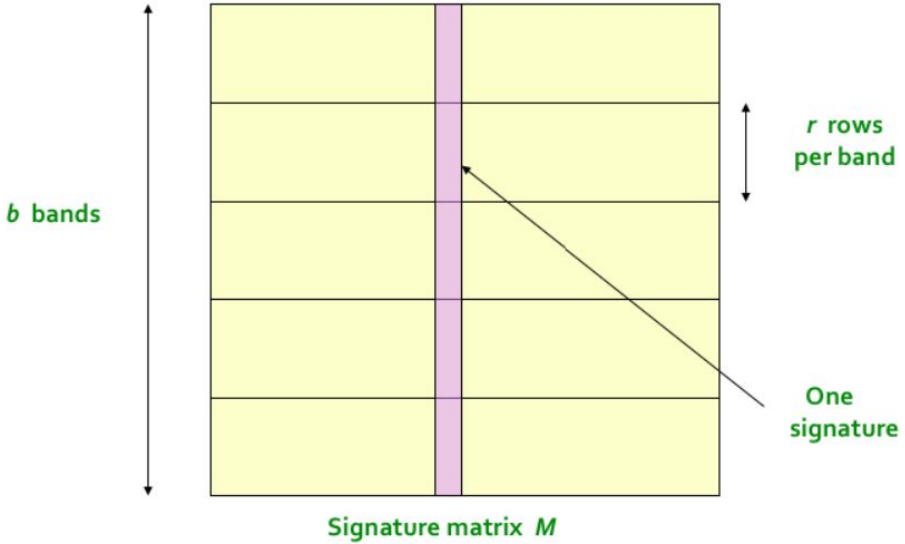


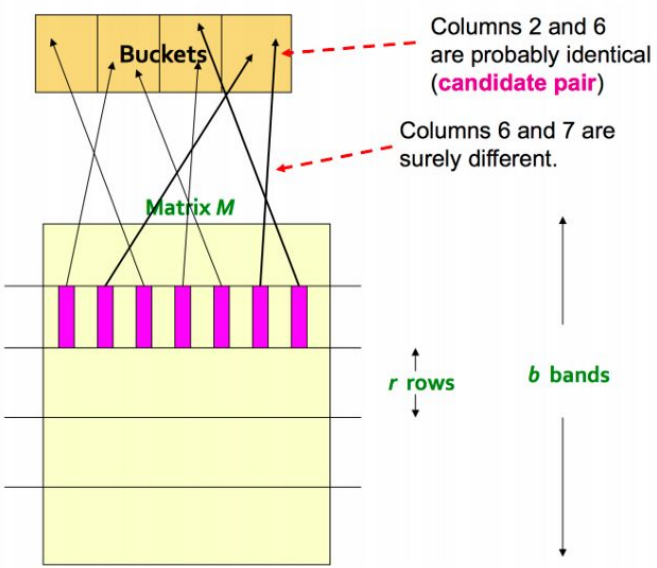
MinHash+LSH. LSH

Мотивация: найти документы, похожесть которых хотя бы t (критерий того, что документы попадают в один класс).

- делим матрицу документов на b блоков по r строк
- к каждому столбцу каждого блока применяем хэш-функцию, выдающую одно из k значений
- Если результаты исходных векторов совпали хотя бы для одного блока - считаем их похожими

MinHash+LSH. LSH





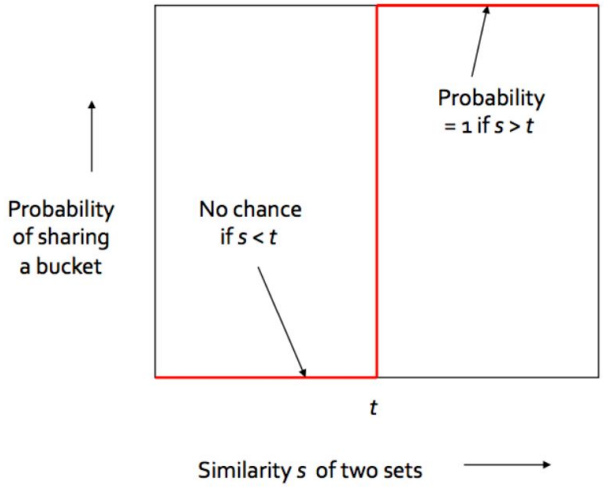
MinHash+LSH. Как варьировать b и r

- 100к документов, signature-матрица 100 x 100000
- t=80%
- b=20, r=5

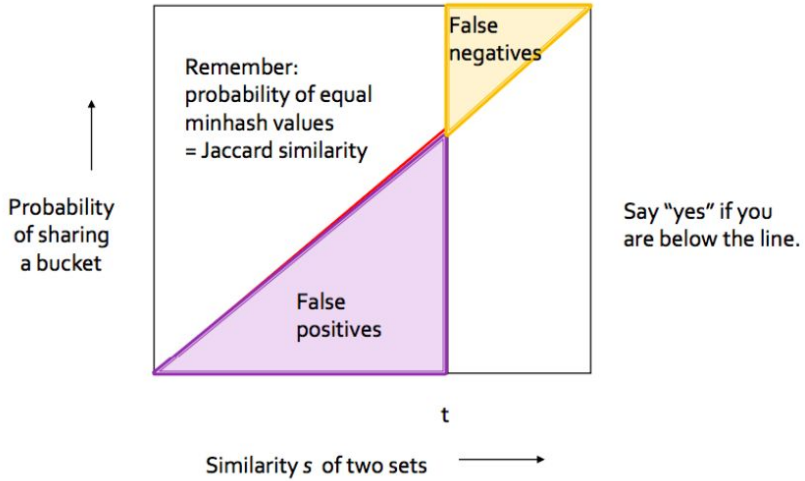
P (пара одинакова в одном блоке) = (0.8)^5 ~ 0.328

P (пара различна во всех блоках) = (1 - 0.328)^20 ~ 0.00035 - это доля False-Negatives

MinHash+LSH. Как варьировать b и r

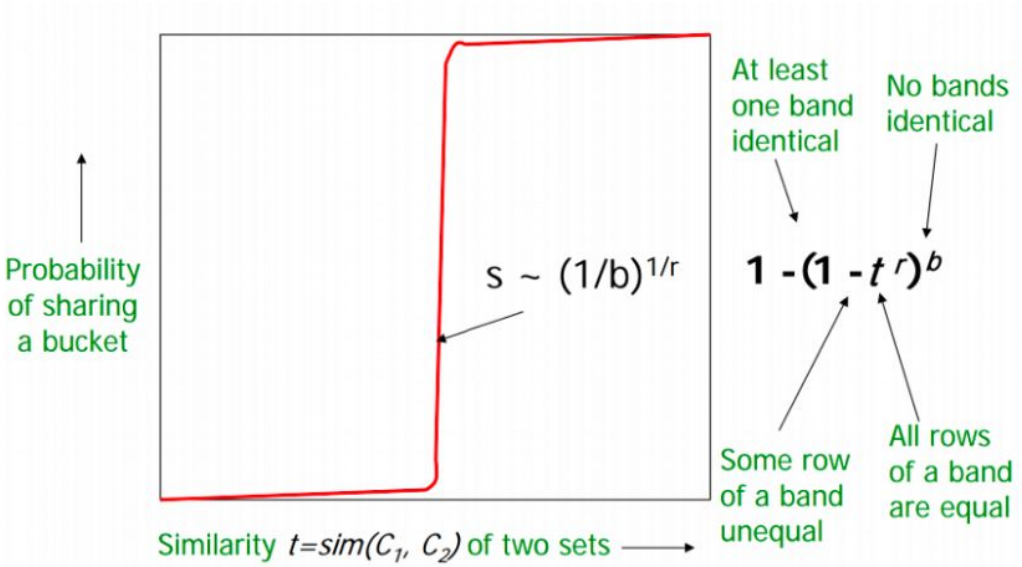


Идеальный мир



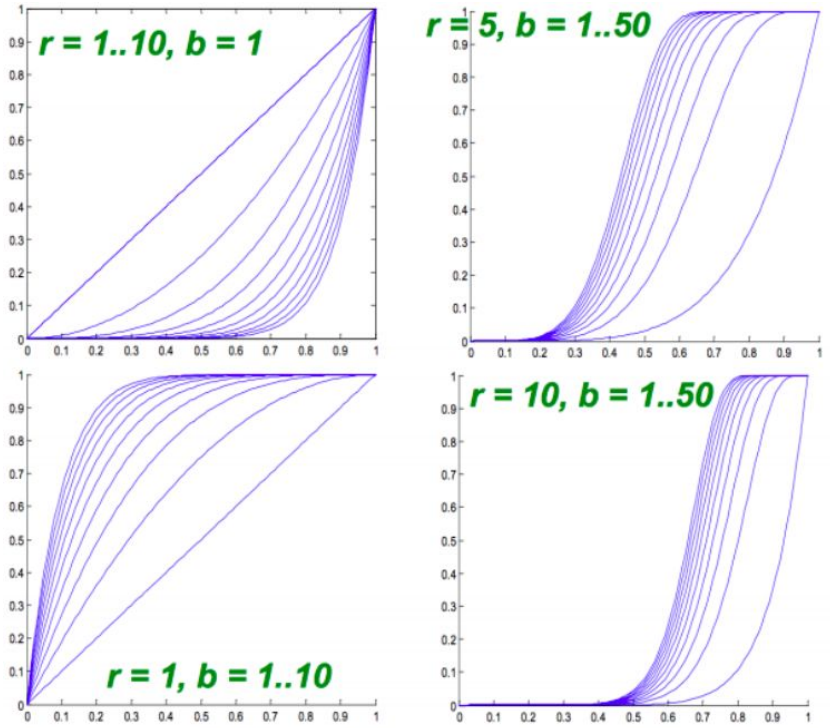
Худший случай

MinHash+LSH. Как варьировать b и r



Общий случай

MinHash+LSH. Как варьировать b и r



Invertex index. Пример

rest_id	famous_for
1	Pure Veg, South Indian
2	Pure Veg, South Indian
3	South Indian, Beverages
4	Pure Veg, South Indian
5	North Indian, Mughlai, Chinese
. . . .	. . . .
144	Deserts,Ice Cream
145	Chinese,Fast food
146	Ice Cream,Deserts
147	North Indian, South Indian,Chinese
148	Fast food, Caf
149	North Indian, Chinese, Manglorean, Malwani,Seafood
150	Fast food, Tex-mex

I. Dataset

word	word_present
Pure Veg	1, 2, 4, 11, 20, 25, 40, 57, 95, 99, 143
South Indian	1, 2, 3, 4, 9, 11, 12, 18, 20, 22, 23, 25, 26, 29,124, 141, 147
North Indian	5, 7, 8, 11, 13, 14, 15, 16, 112, 113, 115, 118, 119, 124, 125, 129, 133, 135, 147, 149
Mughlai	5, 7, 13, 14, 15, 16, 38, 58, 62, 65, 82, 93
. . .	. . .
Goan	112, 113, 117, 132
Konkan	117, 132
Finger food	125
Gujarati	126
Rajasthani	126
Snack Bar	134
Manglorean	149

II. Inverted index for keyword search (from table I)

Invertex index. Формальная постановка

- разбиваем пространство на регионы
- в каждом выбираем центр
- находим центр, ближайший к объекту-запросу
- отбираем кандидатов среди точек, лежащих в регионе с этим центром

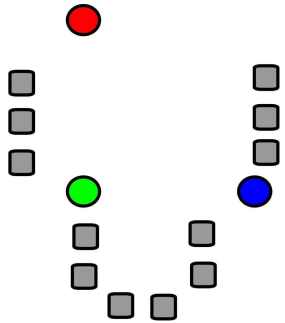
Метод k-средних (k-means)

Задача кластеризации:

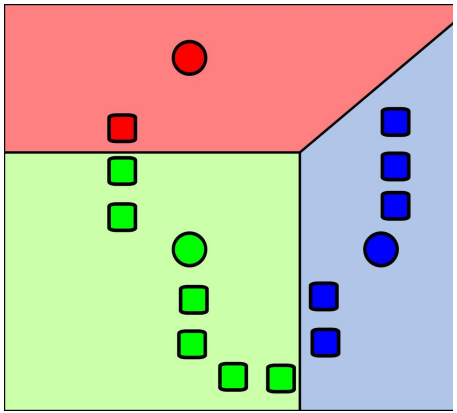
- дано d-мерное пространство в точками на нём.
- необходимо разделить точки на кластеры, так чтобы близкие точки были в одном кластере.
- если формально, то нужно минимизировать суммарное квадратичное отклонение точек кластеров от центров масс этих кластеров.

Метод k-средних не решает эту задачу.

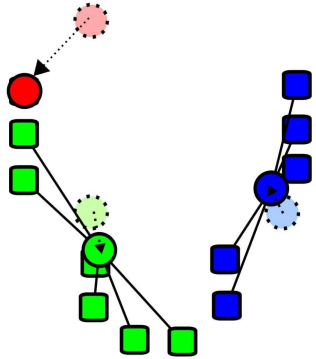
Метод k-средних — алгоритм



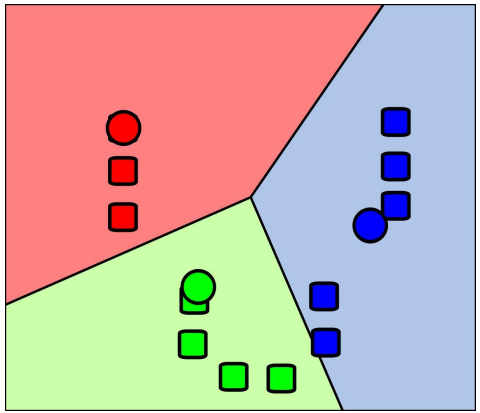
Исходные точки и случайно выбранные начальные точки.



Точки, отнесённые к начальным центрам.



Вычисление новых центров кластеров (Ищется центр масс).



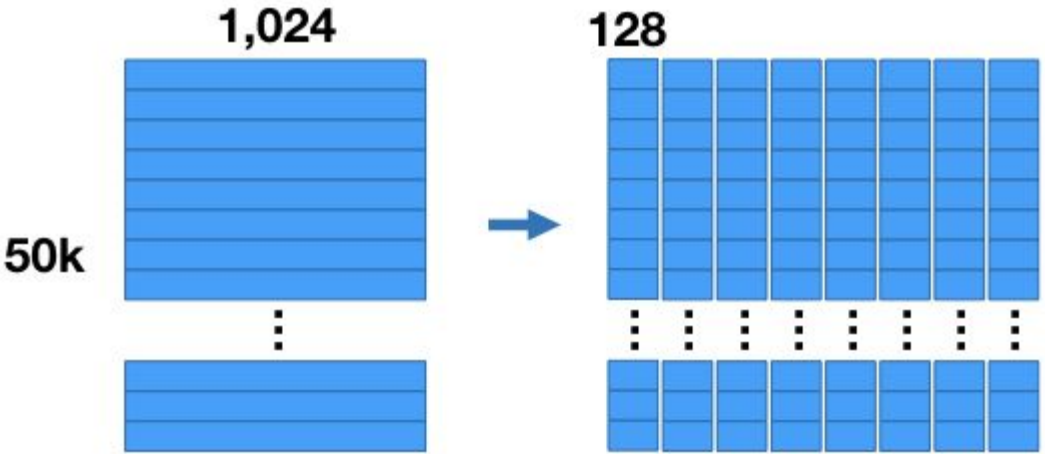
Предыдущие шаги, за исключением первого, повторяются, пока алгоритм не сойдётся.

Product quantization

Алгоритм для быстрого и приближенно подсчёта расстояния между многомерными векторами.

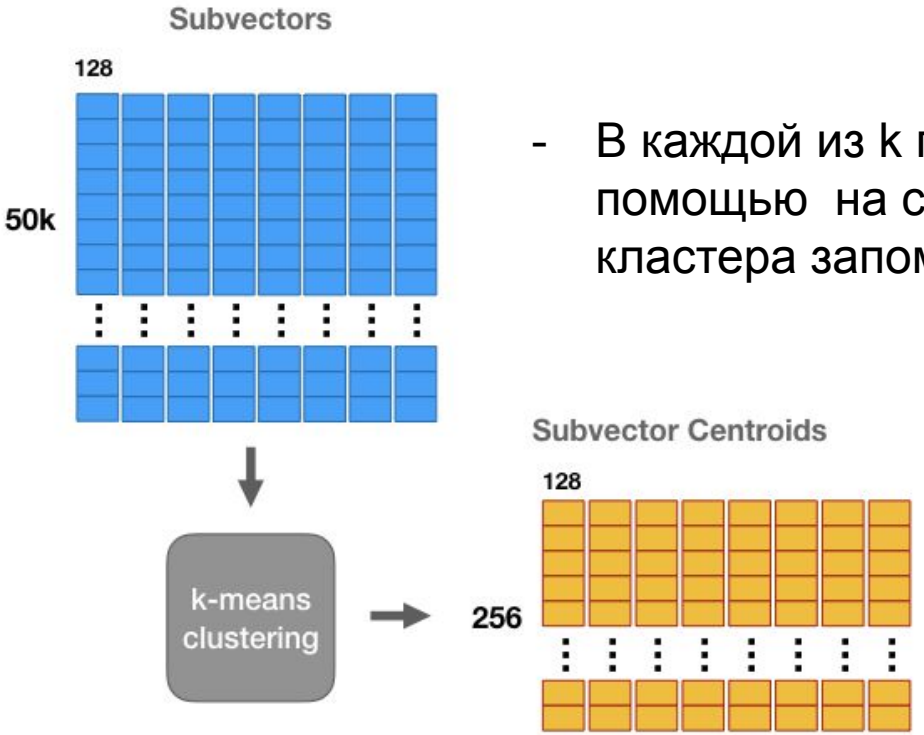
- позволяет ускорить наивное решение задачи kNN (подсчёт расстояний от зафиксированного вектора до всех остальных и их сортировка).
- позволяет более компактно хранить набор векторов признаков.

Product quantization — алгоритм



- Пусть изначально было n (=50000) векторов размерности d (=1024).
- Объединим признаки в k групп, так чтобы в каждой группе было d/k = d’ (=128) признаков.

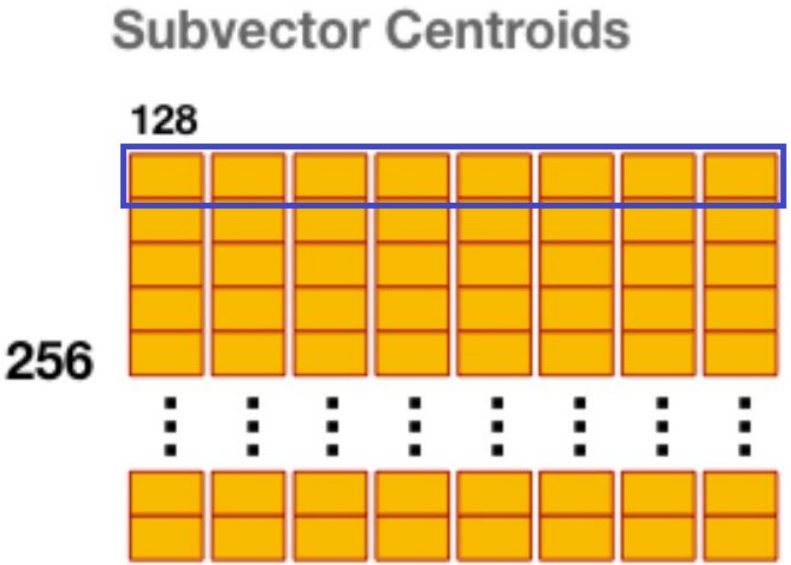
Product quantization — алгоритм



- В каждой из k групп проводим кластеризацию с помощью на с (=256) кластеров, для каждого кластера запоминаем центр масс.

- Заменим в таблице субвекторы из d’ признаков на идентификатор кластера. (Размер набора данных очень сильно уменьшается)

Product quantization — применение в задаче kNN



Предподсчёт:
Находим квадрат расстояния от центра масс кластера группы признаков зафиксированного вектора до каждого центра масс оставшихся 255 кластеров. (и так для всех 128 групп признаков)

Подсчёт:
Теперь, чтобы найти квадрат расстояния от зафиксированного вектора до любого другого, остаётся лишь просуммировать 128 предподсчитанных квадратов расстояний.

k-d tree

Структура данных (двоичное дерево).

Позволяет решать задачу 1NN за O(logn) обычно и O(n) в худшем случае.

Позволяет решать задачу kNN за O(klogn) обычно и O(nlogn) в худшем случае.

Решение задачи kNN можно ускорить (без изменения асимптотики), но с получением приближенного ответа.

k-d tree — построение

- Всё пространство и все точки обозначим за корневую вершину.

- Далее выберем условие, что часть вершин пойдёт в левого сына, а часть в правого. Для этого нужно выбрать ось, по которой происходит деление (например, случайную или самую длинную) и способ разбивки прямоугольника (обычно медианная точка, но бывает и середина).

- Рекурсивно строим вершины, пока в них точек больше, чем мы хотим.

- Насчитываем bounding box для точек в каждой вершине.

k-d tree — решение задачи 1NN

1. Спускаемся по дереву до листовой вершины с зафиксированной точкой.
2. Проверяем всех её соседей в листовой вершине.
3. Поднимаемся по дереву вверх и проверяем bounding box’ы соседних вершины. И если находим что-то более привлекательное спускаемся и в эту ветку и пытаемся обновить результат.

k-d tree — решение задачи kNN

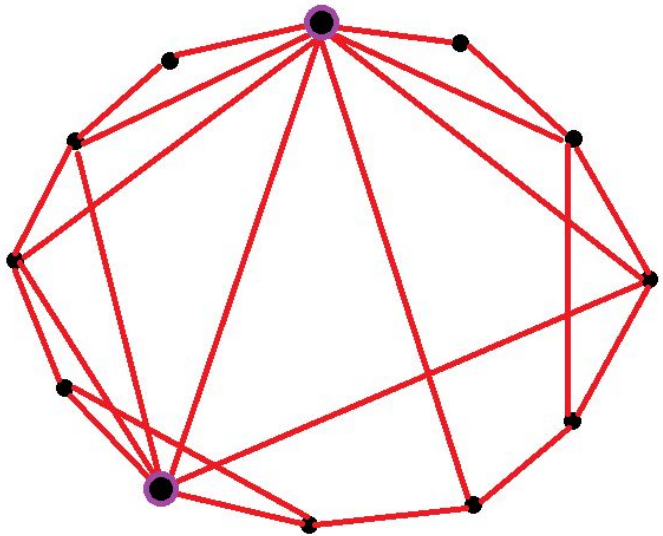
В отличии от предыдущего решения храним не одну самую самую близкую точку, а очередь из k самых близких точек. Расстояние до bounding box’а сравниваем с k-ой точкой.

Оптимизация:

Идём в сыновью вершинину, когда расстояние до bounding box’а менее len/c, а не просто len.

Граф мир тесен (Small World)

- кратчайший путь между двумя случайно выбранными вершинами O(logn).
- средняя степень вершины мала.



Navigable Small World для задачи kNN

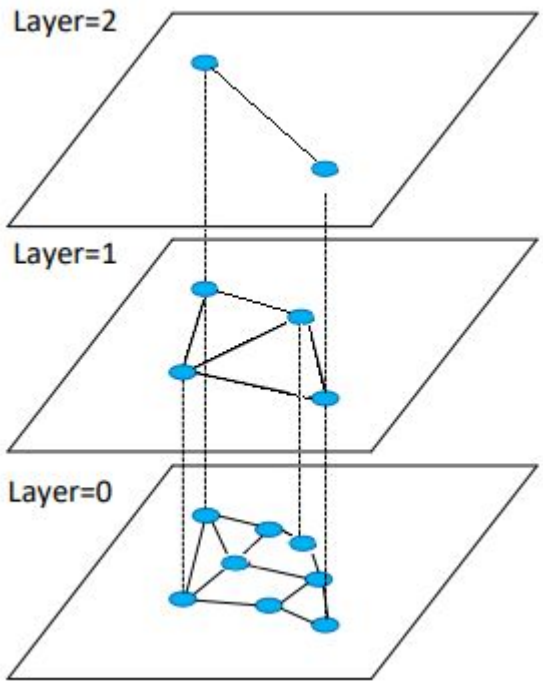
```
knn(G, q, m, k): // Граф, точка запрососа, количество итераций, k - NN.
    C = ∅ // Вершины, которые предстоит проверить.
    V = ∅ // Посещённые вершины.
    for 1...m:
        c = {случайная непосещённая вершина}
        C ∪= {c}, V ∪= {v}
        while True:
            u = {Ближайшая к q вершина из C}
            C \= {u}
            if u дальше чем k-й элемент W:
                break
            for v: (u, v) in G:
                if v ∉ V:
                    C ∪= {v}, V ∪= {v}
    return k ближайших к q вершин из V
```

NSW — проблемы

- Можно “застрять” в локальном минимуме.
- Не очень понятно, как достраивать граф, чтобы он оставался “Маленьким миром”

Иерархический мир тесен (HSW)

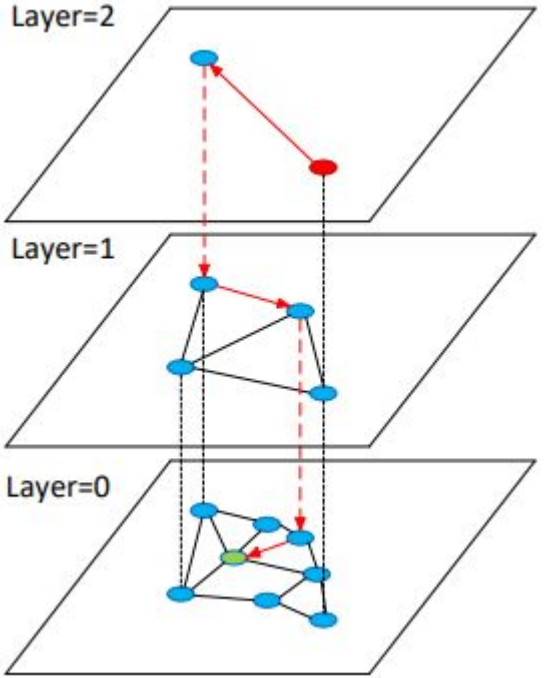
- Состоит из нескольких слоёв графа, где близкие вершины соединены друг с другом.
- Если вершина есть в нижнем уровне, то в верхний она проходит с вероятностью p на следующий слой.



Hierarchical Navigable Small World для kNN

Поиск ближайших соседей во всей структуре:

1. Идём с верхнего уровня до первого:
   - Жадно ищем ближайшую к q вершину на текущем уровне.
   - Спускаемся в соответствующую соседу вершине на уровень ниже.
2. На нулевом уровне жадно ищем k ближайших соседей.



Hierarchical Navigable Small World для kNN

Вставка элемента:

1. Случайным образом выбираем максимальный слой, на котором будет представлена **q**.
2. На каждом уровне, где будет представлена **q**, сверху вниз:
   - Жадно ищем **m** ближайших к **q** вершин.
   - Добавляем связи **q** с ними.
   - Удаляем лишние связи у новообразовавшихся соседей.

Источники

1. k-means:
   - https://ru.wikipedia.org/wiki/Метод_k-средних
2. product quantization:
   - https://mccormickml.com/2017/10/13/product-quantizer-tutorial-part-1/
3. k-d tree:
   - https://www.coursera.org/lecture/ml-clustering-and-retrieval/kd-tree-representation-S0gfp + следующие видео из этого урока
   - https://www.youtube.com/watch?v=Y4ZgLlDfKDg
4. (H)NSW:
   - https://ru.wikipedia.org/wiki/Мир_тесен_(граф)
   - http://neerc.ifmo.ru/wiki/index.php?title=Поиск_ближайших_соседей_с_помощью_иерархического_маленького_мира