Математические методы распознавания образов, ВМК МГУ
Семинар №2

За основу взяты материалы Соколова Евгения.
Модифицировано и дополнено Афанасьевым Глебом
и Алексеевым Ильей

1 Метод k ближайших соседей

§1.1 Описание алгоритма

Пусть дана обучающая выборка $X = \{(x_i, y_i)\}_{i=1}^{\ell}$ и функция расстояния $\rho : \mathbb{X} \times \times \mathbb{X} \to [0, \infty)$, и требуется классифицировать новый объект $u \in \mathbb{X}$. Расположим объекты обучающей выборки X в порядке возрастания расстояний до u:

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

где через $x_u^{(i)}$ обозначается i-й сосед объекта u. Алгоритм k *ближайших соседей* относит объект u к тому классу, представителей которого окажется больше всего среди k его ближайших соседей:

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

Параметр k обычно настраивается с помощью кросс-валидации.

В классическом методе k ближайших соседей все объекты имеют единичные веса: $w_i = 1$. Такой подход, однако, не является самым разумным. Допустим, что $k = 3$, $\rho(u, x_u^{(1)}) = 1$, $\rho(u, x_u^{(2)}) = 1.5$, $\rho(u, x_u^{(3)}) = 100$. Ясно, что третий сосед находится слишком далеко и не должен оказывать сильное влияние на ответ. Эта идея реализуется с помощью весов, обратно пропорциональных расстоянию:

$$w_i = K\left(\rho(u, x_u^{(i)})\right),$$

где $K(x)$ — любая монотонно убывающая функция.

С помощью метода k ближайших соседей можно решать и задачи регрессии. Для этого нужно усреднить значения целевой функции на соседях с весами:

$$a(u; X, k) = \frac{\sum_{i=1}^{k} w_i y_u^{(i)}}{\sum_{i=1}^{k} w_i},$$

где $y_u^{(i)}$ — значение целевой переменной на объекте $x_u^{(i)}$.

§1.2 Примеры функций расстояния

1.2.1 Метрика Минковского

Метрика Минковского определяется как:

$$\rho_p(x,z) = \left(\sum_{j=1}^{d} |x_j - z_j|^p\right)^{1/p}$$

для $p \geqslant 1$. При $p \in (0,1)$ данная функция метрикой не является, но все равно может использоваться как мера расстояния.

Частными случаями данной метрики являются:

- Евклидова метрика ($p = 2$). Задает расстояние как длину отрезка прямой, соединяющей заданные точки.
- Манхэттенское (Краснодарское) расстояние ($p = 1$). Минимальная длина пути из x в z при условии, что можно двигаться только параллельно осям координат.
- Метрика Чебышева ($p = \infty$). Максимальная разница координат:

  $$\rho_\infty(x,z) = \max_{j=1,\ldots,d} |x_j - z_j|.$$

- «Считающее» расстояние ($p = 0$), равное числу координат, по которым векторы x и z различаются:

  $$\rho_0(x,z) = \sum_{j=1}^{d} [x_j \neq z_j].$$

Главной интуицией при выборе p для конкретной задачи является то, что чем больше p, тем значительнее для метрики большие изменения координат, а не маленькие. Иллюстрацией может стать пример в коде:

```
>>> x = np.arange(128)/128
>>> y = x + 1/128   # close to x
>>> z = x * 20      # distant to x
>>> dist = lambda x,y,p: np.sum(np.abs(x-y)**p)**(1/p)
>>> dist(x,y,1)
1.0
>>> dist(x,z,1)
1206.5
>>> dist(x,y,100)
0.008200911590805212
>>> dist(x,z,100)
18.9650404848902
>>> 18.9650404848902/0.008200911590805212
2312.552729644547
```

Видим, что $\rho_{100}(x,z)/\rho_{100}(x,y) \geqslant \rho_1(x,z)/\rho_1(x,y)$ (разница почти в два раза). С ростом размерности векторов и показателя p разница увеличивается. То есть при малом p, как малые так и большие изменения замечаются этой метрикой примерно в равной степени. Но при большом p малые изменения для такой метрики почти не заметны, а весь фокус смещается на большие разности координат. Особенно явно это видно на примере $p = 0$ и $p = \infty$.

1.2.2 Метрика Минковского с весами

Если признаки имеют разный масштаб, то метрика может перестать быть информативной, какой p бы ни был выбран. Например, если x_1 и x_2 — это площадь квартиры и число комнат соответственно, то прямое использование метрики вряд ли передаст информацию о числе комнат, потому что основной вклад сделает дельта по площади.

Первый способ решения такой проблемы, который может прийти в голову, — это отмасштабировать признаки так, чтобы они принимали значения из одного диапазона (к примеру, из отрезка $[0,1]$). Это можно записать следующим образом:

$$\rho_w(x,z) = \left(\sum_{j=1}^{d} w_j |x_j - z_j|^2\right)^{1/2}, \quad w_j \geqslant 0.$$

Предполагается, что веса подобраны так, что в новом признаковом пространстве $\tilde{x}_j = x_j\sqrt{w_j}$ все признаки вносят одинаковый вклад в метрику и никто их них не теряется.

Разный масштаб признаков геометрически означает, что в новом признаковом пространстве линии уровня метрики Минковского являются окружностями, тогда как в старом они были эллипсами.

$$\rho^2(\tilde{x},0) = \sum_{j=1}^{d} \tilde{x}_j^2 = \sum_{j=1}^{d} w_j x_j^2 = \rho_w^2(x,0).$$

У приведённого решения есть один серьёзный недостаток: если один из признаков лишний, то после масштабирования он будет вносить такой же вклад, как и полезные признаки. Поэтому веса стоит подбирать не только с целью отмасштабировать признаки, но и с целью уменьшить вклад ненужных. Например, после масштабирования можно использовать эмпирический коэффициент корреляции (Пирсона, Спирмена, Кендалла) между признаком и целевым вектором, или любую другую меру важности признака (взаимную информацию, статистику хи-квадрат и проч.):

$$\rho_{w,h}(x,z) = \left(\sum_{j=1}^{d} |w_j h_j| \cdot |x_j - z_j|^2\right)^{1/2}.$$

1.2.3 Расстояние Махалонобиса

Как мы выяснили, метрика Минковского с весами эквивалентна обычной метрике над признаками, полученными с помощью покоординатного преобразования.

Цель этого преобразования — подстроить функцию расстояния под задачу. Оно выглядит следующим образом:

$$\widetilde{x} = Wx,\ W = \mathrm{diag}(\sqrt{w_1}, \ldots, \sqrt{w_d}).$$

Одно лишь масштабирование координатных осей не всегда является достаточно мощным преобразованием. Что если добавить поворот осей?

Определим расстояние Махалонобиса следующим образом:

$$\rho_S(x, z) = \sqrt{(x - z)^T S^{-1} (x - z)},$$

где S — симметричная положительно определенная матрица. Для неё справедливо разложение по собственным значениям:

$$S = Q\Lambda Q^T,\ \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_d),\ Q^T Q = I.$$

Легко убедиться, что метрика Махалонобиса, как и метрика Минковского с весами, эквивалентна евклидовому расстоянию над линейно преобразованными признаками:

$$\rho_S^2(x, 0) = x^T S^{-1} x = x^T (Q\Lambda Q^T)^{-1} x = x^T Q \Lambda^{-1} Q^T x = \sum_{i=1}^{d} \tilde{x}_i^2 = \rho^2(\tilde{x}, 0),$$

где $\tilde{x} = \Lambda^{-1/2} Q^T x$. На этот раз исходные эллипсы не только отмасштабированы (соответственно собственным значениям матрицы S), но и повернуты (соответственно собственным векторам матрицы S).

Подбор матрицы S — это отдельная задача оптимизации, которая будет рассмотрена в теме «Обучение метрик».

1.2.4 Косинусное расстояние.

Как мы выяснили выше, используемая метрика всегда должна учитывать признаковое пространство, которому принадлежат объекты выборки. Рассмотренные выше метрики хорошо работают с произвольным векторным пространством. Сейчас мы рассмотрим отдельный часто встречающийся тип признакового пространства: векторы имеют примерно одинаковую длину и/или они распределены по поверхности сферы. В этом случае подсчет евклидового расстояния между двумя векторами является несколько избыточной операцией, поскольку достаточно лишь знать, насколько два вектора сонаправлены.

Косинусным расстоянием называют угол между векторами:

$$\rho_{\cos}(x, y) = \arccos\left(\frac{\langle x, y\rangle}{\|x\|\|y\|}\right).$$

Причем часто используют просто значение косинуса. Косинус обратно пропорционален углу, поэтому в этом случае говорят не о расстоянии, а о косинусной *близости*.

Косинусную метрику можно эффективно использовать для измерения схожести между текстами. Пусть имеется вокабулярий V, который представляет собой список слов $v_i,\ i = \overline{1, |V|}$. Каждому тексту можно поставить в соответствие вектор x, каждая компонента x_i которого равна числу вхождений слова v_i в данный текст. Такой способ векторизации текста называется bag of words (мешок слов).

Пример.

```
[Пес, сел, на, пень]                                  # первый текст
[Кот, сел, на, ель]                                   # второй текст
{Пес,   Кот,    ель,    на,     сел,    пень}        # словарь
[1,     0,      0,      1,      1,      1]           # первый мешок слов
[0,     1,      1,      1,      1,      0]           # второй мешок слов
```

Косинус между двумя векторами, мешками слов, будет тем больше, чем больше у двух текстов общих слов.

Один из плюсов косинусной меры состоит в том, что в ней производится нормировка на длины векторов. Благодаря этому она не зависит, например, от размеров сравниваемых текстов.

1.2.5 Редакторское расстояние

Для измерения посимвольного сходства между двумя строками можно использовать различные виды *редакторского расстояния*. Например, расстояние Левенштейна определяется как минимальное число элементарных операций, с помощью которого можно преобразовать одну строку в другую. Среди операций доступны замена, вставка, удаление. Пример: $\rho(\text{ЛАБРАДОР}, \text{ГИБРАЛТАР}) = 5$. Эта метрика является симметричной. Применяется для сравнения не только текстов, но и ДНК-последовательностей. Эффективное вычисление расстояния редактирования для двух заданных строк является популярной задачей на знание алгоритмов и структур данных [9].

1.2.6 KL-дивергенция.

Достаточно экзотичен, но все-таки реален случай, когда необходимо измерить расстояние между двумя вероятностными распределениями. Дивергенция Кульбака–Лейблера для дискретных распределений p, q определяется следующим образом:

$$D_{\mathrm{KL}}(p \parallel q) = \sum_{i=1}^{d} p_i \log\left(\frac{p_i}{q_i}\right).$$

KL-дивергенция по определению означает матожидание логарифмической разности между двумя распределениями. Эта метрика не является симметричной. Под собой она имеет глубокие основания из математической статистики [10].

1.2.7 Функции расстояния на категориальных признаках

Категориальные признаки не имеют никакой явной структуры, и поэтому достаточно сложно ввести на них разумное расстояние. Как правило, ограничиваются сравнением их значений: если у двух объектов одинаковые значения категориального признака, то расстояние равно нулю, если разные — единице. Тем не менее, существуют определенные соображения по поводу того, как измерять сходство для таких признаков.

Будем считать, что метрика записывается как взвешенная сумма расстояний по отдельным признакам с некоторыми весами:

$$\rho(x, z) = \sum_{j=1}^{d} w_j \rho_j(x_j, z_j).$$

Способы измерения расстояния для вещественных признаков обсуждались выше. Обсудим некоторые варианты для категориальных признаков. Введем следующие обозначения:

1. $f_j(m) = \sum_{i=1}^{\ell} [x_{ij} = m]$ — количество раз, которое j-й признак принимает значение m на обучающей выборке;
2. $p_j(m) = \frac{f_j(m)}{\ell}$ — частота категории m на обучающей выборке;
3. $p_j^{(2)}(m) = \frac{f_j(m)(f_j(m)-1)}{\ell(\ell-1)}$ — оценка вероятности того, что у двух случайно выбранных различных объектов из обучающей выборки значения признака будут равны m.

Тогда расстояние на категориальных признаках можно задавать, например, одним из следующих способов:

1. Индикатор совпадения:

   $$\rho_j(x_j, z_j) = [x_j \neq z_j]$$

2. Сглаженный индикатор совпадения. Чем выше частота у значения признака, тем больше расстояние (если оба человека живут в Москве, то эта информация не очень важна, поскольку вероятность такого совпадения высока; если оба человека живут в Снежинске, то это важная информация, так событие является достаточно редким):

   $$\rho_j(x_j, z_j) = [x_j \neq z_j] + [x_j = z_j] \sum_{q : p_j(q) \leqslant p_j(x_j)} p_j^{(2)}(q)$$

3. Чем более частые значения оказались при несовпадении, тем больше расстояние (если оба человека из разных, но очень маленьких городов, то можно считать их похожими; если один человек из Москвы, а второй — из Питера, то они сильно отличаются):

   $$\rho_j(x_j, z_j) = [x_j \neq z_j] \log f_j(x_j) \log f_j(z_j)$$

   (обратите внимание, что для борьбы с численными проблемами имеет смысл добавлять единицу под логарифмом: $\log(f_j(x_j) + 1)$, $\log(f_j(z_j) + 1)$)

Более подробный обзор функций расстояния на категориальных признаках можно найти в работе [2].

1.2.8 Мера Джаккарда.

Это мера сходства между двумя множествами:

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

Пример. Бинарные изображения (Ч/Б) часто используют как маски для областей интереса на каком-то другом (цветном) изображении. Популярный способ сравнить похожесть двух таких областей — это взять меру Джаккарда между ними.

1.2.9 «Проклятие размерности»

«Проклятием размерности» в контексте метрических методов обычно называют ряд явлений:

1. Трудоемкость вычислений
2. Необходимость хранения огромного количества данных
3. Метрика становится неинформативной

Подробнее остановимся на последнем пункте. Во-первых, метрика становится неинформативной в силу закона больших чисел: эмпирическое среднее d случайных величин стремится к фиксированному числу (теоретическому мат. ожиданию). Метрика Минковского является суммой случайных величин:

$$\frac{1}{d}\sum_{j=1}^{d} x_j^2 \xrightarrow{d\to\infty} \mathbb{E}[x_j^2]$$
$$\Rightarrow \rho^2(x,0) \approx d \cdot \mathbb{E}[x_j^2],\ d \gg 1.$$

Значит, с увеличением размерности пространства все объекты примерно равноудалены друг от друга и не может идти речи о «соседях».

Во-вторых, метрика становится неинформативной из-за геометрии многомерных пространств. Пусть объекты выборки — это точки, равномерно распределенные в d-мерном шаре радиуса R. Рассмотрим вложенный шар радиуса $R - \varepsilon$. Формула объема шара:

$$V(R,d) = \frac{\pi^{d/2} R^d}{\Gamma(d/2+1)}.$$

Долю объектов, попавших во вложенный шар, можно посчитать разделив объем вложенного шара на объем внешнего:

$$p(d,R) = \frac{V(R-\varepsilon,d)}{V(R,d)} = \left(\frac{R-\varepsilon}{R}\right)^d \xrightarrow{d\to\infty} 0$$

Таким образом, объекты сосредоточены вдоль границ шара. Многомерное метрическое пространство сильно разрежено.

2 Методы поиска ближайших соседей

Данный раздел посвящен алгоритмам поиска ближайших соседей, эффективных с точки зрения вычислительной сложности. Эта проблема актуальна не только для решения задач классификации и регрессии методом KNN, но и для современных ретривных и поисковых систем, которым необходимо хранить миллиарды объектов и быстро находить релевантные. Постановка задачи следующая: из выборки $X \in \mathbb{R}^{n \times d}$ извлечь k объектов, наиболее похожих на объект-запрос $q \in \mathbb{R}^d$ с точки зрения функции расстояния $\rho(q, \cdot)$. В промышленных масштабах такая задача должна решаться за $O(\log n)$ сравнений.

§2.1 Точные методы.

Наивный алгоритм. Отсортировать всю выборку по значению $\rho(q, \cdot)$ и взять срез из первых k элементов. Сложность по времени $O(n \log n)$.

Поиск k-ой порядковой статистики. Существует алгоритм поиска k-ой порядковой статистики, сложность которого $O(n)$ сравнений. Этот алгоритм изменяет порядок в массиве таким образом, что первые k элементов меньше, чем все последующие.

kd-tree. Данный метод заключается в построении сбалансированного дерева, листья которого соответствуют объектам выборки X. Рассмотрим процедуру построения такого дерева. Пусть $x_i \in \mathbb{R}^n$ — это j-ый столбец матрицы признаков.

1. Найти медиану x_1 и с ее помощью разделить X на две половины.
2. Для каждой из половин отдельно найти медианы x_2 и разделить эти половины еще раз пополам.
3. Повторять эту процедуру вплоть до x_d и затем продолжить с x_1.
4. Если какое-то разбиение содержит лишь один объект, то получен лист.
5. И так далее до тех пор, пока все объекты не будут распределены в свои листья.

Если размерность пространства небольшая (10-20), то сложность построения дерева $O(n \log n)$. Заметим, что функция расстояния не используется при построении дерева.

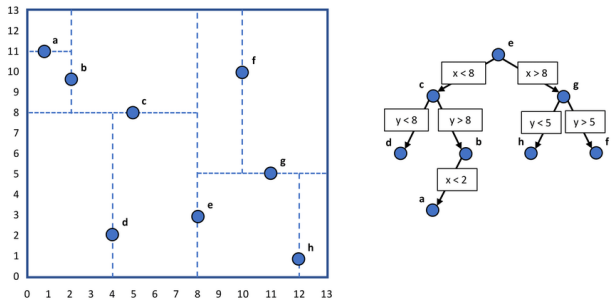

Рис. 1. Пример разбиения kd-tree.

Поиск осуществляется путем передвижения от корня к листьям и постоянными проверками на факт того, нет ли в соседнем сплите более близких соседей, чем в данном. Экспериментально было установлено, что в пространствах большой размерности сложность поиска ближайшего соседа в kd-дереве сильно ухудшается и приобретает линейный порядок сложности [3].

Ценность kd-дерева в том, что его можно построить один раз и затем многократно использовать для поиска. Процедуру построения структуры данных с такой целью называют индексированием, а саму структуру – индексом. Все методы, о которых пойдет речь дальше имеют своей целью построение эффективного индекса.

ball-tree. То же самое, но вместо полупространств шары. Вместо медианы на каждом шаге нужно искать геометрический центр и радиус шара. При малых размерностях сложность построения $O(n \log^2 n)$, а сложность поиска $O(\log n)$.

§2.2 Приближенные методы

Искать k ближайших соседей приближенно, то есть разрешать результату поиска быть чуть дальше от нового объекта, чем k его истинных соседей. В данном разделе мы разберем несколько примеров этого подхода. Цель всех методов, представленных ниже, состоит в построении такого индекса, который минимизирует число объектов выборки X, с которыми необходимо сравнить запрос q для отыскания ближайших к нему соседей.

2.2.1 kd-tree и ball-tree

В эти алгоритмы можно внести такую модификацию, что поиск в них станет быстрее, но результат будет отличаться от истинного. Дело в том, что необязательно хранить в каждом листе лишь один объект. Вместо этого построение дерева можно останавливать в тот момент, когда разбиение станет достаточно малым. Тогда на этапе поиска можно сначала спуститься по дереву до данного листа, а затем искать ближайших соседей только в нем (или еще в соседних листах). Таким образом возникает эффект квантования признакового пространства.

2.2.2 LSH

Хэш-функции тоже осуществляют квантование произвольного множества объектов на отдельные корзины. Когда два объекта оказываются в одной корзине, говорят, что произошла коллизия. Обычные хэши придумывают такими, чтобы они минимизировали число коллизий в среднем для всех объектов. Однако идея locality-sensitive hashing, наоборот, в том, чтобы максимизировать число коллизий для близких объектов и минимизировать для далёких.

Рассмотрим следующую хэш-функцию:

$$h_w(x) = [\langle w, x \rangle > 0].$$

Она соответствует некоторой гиперплоскости, проходящей через начало координат, и возвращает для каждого x либо 0, либо 1 в зависимости от того, по какую сторону от этой гиперплоскости он находится. Причем чем ближе два вектора друг другу, тем

вероятнее, что они окажутся по одну сторону от случайной гиперплоскости. Значит, мы сократили пространство поиска в двое. Если использовать сразу m гиперплоскостей, то это будет эквивалентно разбиению пространства признаков на $2m$ областей.

В случае, когда этих областей меньше, чем объектов в выборке ($2^m < n$), получается эффект квантования. Если же $2m \gg n$, то получается ерунда, поскольку каждая область содержит только один объект (или ни одного). Или не ерунда? Оказывается, что это даже лучше. Заметим, что область, в которую попал отдельный вектор x, кодируется битовой цепочкой, а именно, значениями хэшей от данного вектора. Причем соседние области соответствуют соседним битовым цепочкам. В случае $2m \gg n$ эту битовую цепочку можно воспринимать как новые координаты x и мерять от них расстояние Хэмминга. Это дает два плюса: 1) расстояние Хэмминга можно эффективно вычислять побитовыми операциями, 2) битовые цепочки занимают меньше памяти.

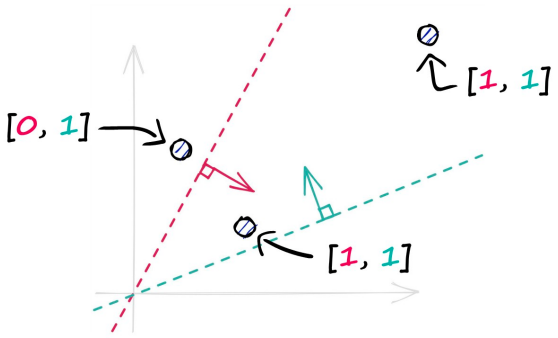

Рис. 2. Пример разбиения LSH. Подписаны битовые коды каждого объекта.

Это лишь один пример хэш-функции, обладающей свойством LSH. Для других метрик и доменов данных существуют другие. Например, для текстов часто применяют MinHash [11], для евклидового расстояния можно использовать p-stable LSH [12].

2.2.3 Annoy

Данный метод расшифровывается как Approximate Nearest Neighbors Oh Yeah. Он реализован в одноимённой библиотеке и применяется в Spotify для рекомендательных систем, чтобы быстро находить похожих пользователей, похожие треки.

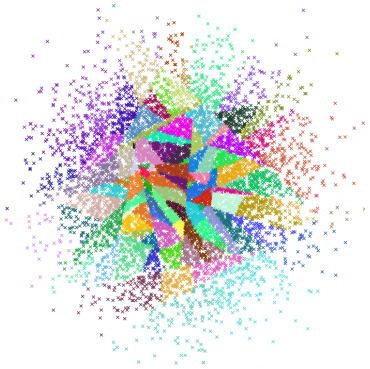

Рис. 3. Пример разбиения Annoy.

1. Выбираем два произвольных объекта из X и проводим гиперплоскость, равноудалённую от них. Делим всю выборку на две части с помощью этой гиперплоскости.

2. В каждой из полученных половин проделываем то же самое, пока все разбиения не станут достаточно малыми. Получаем дерево, листья которого содержат объекты из X.

3. Строим еще несколько таких деревьев.

Теперь чтобы произвести поиск, необходимо спуститься в самый низ каждого дерева, объединить эти листья и среди них выбрать ближайших соседей. Причем число деревьев, используемых для поиска, является гиперпараметром, отвечающим за компромисс между точностью и скоростбю поиска.

2.2.4 HNSW

Данный метод — один из самых быстрых на сегодня. Он является скрещением двух других методов поиска: probability skip list и navigable small world.

Probability skip list (PSL). Двоичный поиск по массиву можно представить как спуск по многоуровневой структуре, где с каждым уровнем уточняется регион поиска. Автор PSL решил заложить подобную логику явно в структуру данных. Список состоит из нескольких уровней. Верхние уровни содержат мало элементов массива, и чем ниже, тем их больше. Если один элемент появился на уровне t, то он есть и на всех уровнях ниже. Таким образом, начав поиск с верхнего уровня, получается определить регион, а спускаясь все ниже, в этом регионе появляются новые и новые элементы, уточняя зону поиска до тех пор, пока нужный элемент не найдется.

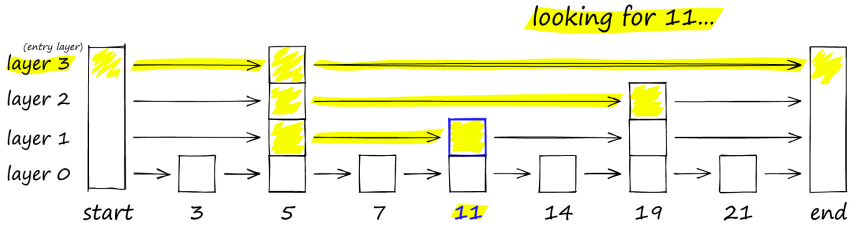

Рис. 4. Пример PSL.

Navigable Small World (NSW). Существует парадокс, обнаруженный исследователями социальных сетей: почти все люди знакомы друг с другом через небольшое число промежуточных знакомств. То есть люди образуют граф, в котором можно добраться из одной точки в любую другую за малое число шагов. Много исследователей использовали эту идею для создания индексов, обладающих подобными свойствами, поскольку это очень удобно, когда из любой точки можно за небольшое число шагов найти ближайшего соседа.

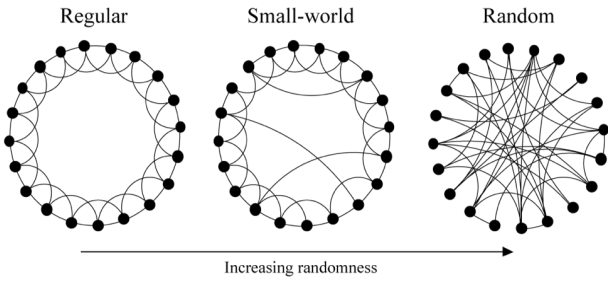

Рис. 5. Разница между графом с регулярной структурой, случайной и графа NSW.

Hierarchical Navigable Small World (HNSW). Из слияния PSL и NSW родился метод HNSW. Это граф, который состоит из нескольких уровней. Уровни несут ту же роль, что и в PSL — они уточняют регион. А чтобы делать это максимально быстро, каждый уровень строится как NSW.

Данный метод реализован в библиотеке FAISS, которая является одним из лучших инструментов для создания огромных индексов для ретривных систем.

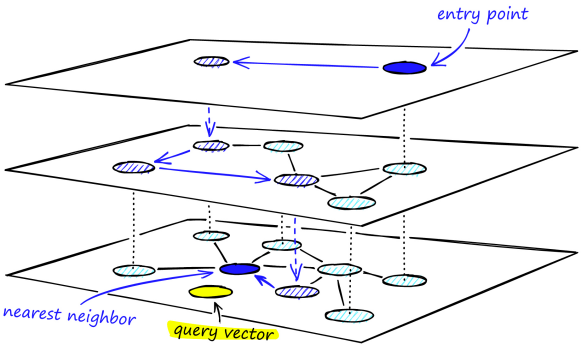

Рис. 6. Пример HNSW.

Список литературы

[1] *Weber, R., Schek, H. J., Blott, S.* (1998). A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. // Proceedings of the 24th VLDB Conference, New York C, 194–205.

[2] *Boriah, S., Chandola, V., Kumar, V.* (2008). Similarity measures for categorical data: A comparative evaluation. // In Proceedings of the 2008 SIAM International Conference on Data Mining (pp. 243–254).

[3] *Weber, R., Schek, H. J., Blott, S.* (1998). A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. // Proceedings of the 24th VLDB Conference, New York C, 194–205.

[4] *Andoni, A., Indyk, P.* (2008). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. // Communications of the ACM, 51(1), 117.

[5] *Datar, M., Immorlica, N., Indyk, P., Mirrokni, V. S.* (2004). Locality-sensitive hashing scheme based on p-stable distributions. // Proceedings of the twentieth annual symposium on Computational geometry - SCG '04, 253.

[6] *Bawa, Mayank and Condie, Tyson and Ganesan, Prasanna* (2005). LSH Forest: Self-tuning Indexes for Similarity Search. // Proceedings of the 14th International Conference on World Wide Web.

[7] *Wang, J., Liu, W., Kumar, S., Chang, S.-F.* (2015). Learning to Hash for Indexing Big Data - A Survey. http://arxiv.org/abs/1509.05472

[8] *Nearest neighbors and vector models – part 2 – algorithms and data structures*, Erik Bernhardsson blog.

[9] *LeetCode — Edit Distance,* https://leetcode.com/problems/edit-distance/

[10] *Kullback-Leibler Divergence — Wikipedia*, https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence

[11] *MinHash — Wikipedia*, https://en.wikipedia.org/wiki/MinHash

[12] *Euclidean LSH — ML Wiki*, http://mlwiki.org/index.php/Euclidean_LSH