Поиск похожих объектов с помощью хэширования

Сивухин Никита

1 Введение

Задача поиска наиболее похожего объекта из заданного множества находит применение различных областях: алгоритмы машинного обучения, компьютерного зрения, статистика и т.д. Мы сконцентрируем свое внимание на двух конкретных задачах поиска:

- Задача поиска ближайшей точки в пространстве $\mathbb{R}^d$ с евклидовой метрикой
- Задача поиска множества с максимальным коэффициентом Жаккара J относительно заданного множества A, где $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$

Для решения задач в евклидовом пространстве существуют эффективные детерминированные алгоритмы для небольших размерностей ($d \leq 3$), однако данные алгоритмы не дают существенного приемущества относительно наивного линейного алгоритма для пространств с большим количеством измерений.

Больше успехов удалось достичь в решении задачи приближенного поиска соседа. Допустим мы у нас есть метрическое пространство S с метрикой $d(x, y)$. Сформулируем три, тесно связанные друг с другом, задачи поиска ближайшего соседа в этом пространстве:

Задача 1 (поиск ближайшего соседа)**.** *Для фиксированного множества элементов $P \subset S$ и запроса $q \in S$ мы хотим найти такой элемент $NN(q) \in P$, что $NN(q) = \arg\min_{p \in P} d(p, q)$*

Задача 2 (c-приближенный поиск ближайшего соседа)**.** *Для фиксированного множества элементов $P \subset S$ и запроса $q \in S$ мы хотим найти такой элемент $ANN_c(q) \in P$, что $d(ANN_c(q), q) \leq c \cdot d(NN(q), q)$.*

Задача 3 (c-приближенный вероятностный поиск ближайшего соседа)**.** *Для фиксированного множества элементов $P \subset S$ и запроса $q \in S$ мы хотим найти $ANN_c(q)$ с вероятностью p.*

Алгоритмы, основанные на хэшировании, позволяют быстро решать последнюю задачу в теории. На практике результаты данных подходов довольно хороши, но не все практические результатые также хороши, как теоретические.

2 Locality sensitive hashing

Определим понятие LSH-семейства хэш-функций, для которых существует фреймворк, позволяющий решать задачу приближенного вероятностного поиска соседа:

Определение 1 (LSH-семейство хэш-функций)**.** *Семейство хэш-функций $H : S \to U$ называется (r_1, r_2, p_1, p_2)-чувствительным, если для любых двух элементов $a, b \in T$ и случайной функции $h \in H$ выполняется два свойства:*

1. *Если $d(a, b) \leq r_1$, то $Pr[h(a) = h(b)] \geq p_1$*
2. *Если $d(a, b) \geq r_2$, то $Pr[h(a) = h(b)] \leq p_2$*

Имеет смысл рассматривать только LSH-семейства, где $r_1 < r_2$ и $p_1 > p_2$, что будет дальше неявно подразумеваться во всех рассуждениях.

Определим еще одну вспомогательную задачу, непосредственно для решения которой разработан фреймворк на основе LSH хэш-функций:

Задача 4 ((R, cR)-приближенный поиск ближайшего соседа)**.** *Для фиксированного множества элементов $P \subset S$ из n элементов и запроса $q \in S$, в случае, если $d(NN(q), q) \leq R$, мы хотим найти такой элемент $ANN_c^R(q) = p$, что $d(p, q) \leq cR$. Иначе, алгоритм может как найти, так и не найти точку с такими свойствами (см. рисунок 1).*

Задачу 2 можно свести к только что сформулированной задаче с помощью бинарного поиска, замедлив общее решение в $O(\log \log R)$ раз, где $R = \max_{p_1, p_2 \in P} \frac{d(q,p_1)}{d(q,p_2)}$. Однако, чтобы иметь возможность оценить замедление непосредственно через входные параметры задачи, необходимо использовать более продвинутые алгоритмы (Ring-Cover trees), которые позволяют добиться замедления в $\log(\frac{|P|}{\epsilon})$, где $c = 1 + \epsilon$.

Теперь опишем вероятностный алгоритм решения задачи 4:

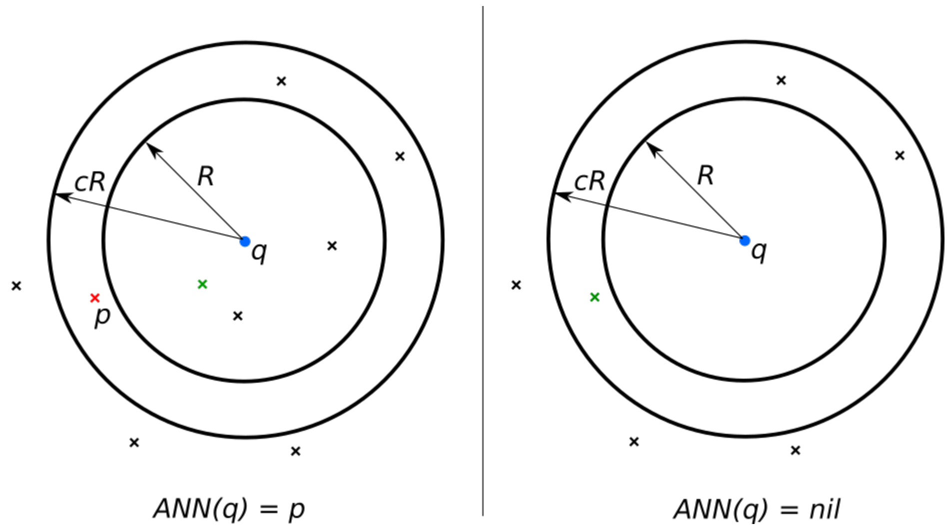

Рис. 1: Не левой картинке решением задачи может быть как точка p, так и две пять остальных точек в круге cR. На правой картинке алгоритм может и не найти ни одну из точек внешнего круга

Теорема 1. *Если в метрическом пространстве S существует (R, cR, p_1, p_2)-чувствительное семейство $H : S \to U$, то задачу 4 можно решить с некоторой константой вероятностью в следующих условиях:*

- *Время на запрос: $O(n^\rho \log_{1/p_2} n \times T_H + n^\rho \times D_S)$, где $\rho = \frac{\log p_1}{\log p_2}$, T_H — время на вычисление значения хэш-функции из семейства H, а D_S — время на вычисления расстояния между точками в пространстве S*

- *Время на построение: $O(n^\rho \log_{1/p_2} n \times T_H)$*

- *Используемую память: $O(n^{1+\rho} + M_S)$, где M_S — пространство, необходимое для сохранения элементов P пространства S*

Для доказательства теоремы, нам нужно будет построить несколько составных хэш-функций. Для начала определим $g(x) = (h_1(x), h_2(x), \cdots, h_K(x))$ — составная функция из K случайных функций семейства H. Также, для работы алгоритма нам понадобится семейство из L случайных функций g_i. Для уменьшения размера пространства значений функций g_i построим хэш-таблицу, отображающую кортежи $(h_1(x), \cdots, h_K(x))$ в префикс натуральных чисел $[t]$.

Алгоритм, доказывающий теорему, будет работать следующим образом:

- Каждая точка исходного множества P будет распределена в L хэш-корзинок согласно функциям g_i, причем в корзинках, где находится более одной точки, мы оставим прозвольную

- Для каждой точки запроса q будут рассмотрены не более L кандидатов из исходного множества, находящиеся в корзинках, соответствующих значениям $g_1(q), g_2(q), \cdots, g_L(q)$. Для всех кандидатов вычислим расстояние и вернем того, кто удовлетворяет условиям задачи

Для доказательства теоремы определим два свойства:

1. назовем точку $p \in P$ «плохой», если для запроса q при условии, что $d(p,q) \geq cR$ выполняется равенство $g_i(p) = g_i(q)$

2. назовем функцию g_i «хорошей», если для запроса q и некоторой точки $p \in P : d(p,q) \leq R$ и $g_i(p) = g_i(q)$.

Т.к. $Pr[p - \texttt{«плохая»}] \leq p_2^K$, то положим $K = -\log_{p_2} 2n$, тогда $Pr[p - \texttt{«плохая»}] \leq \frac{1}{2n}$. Оценим матожидание числа «плохих» точек при таком выборе K: $E \leq L \times n \times \frac{1}{2n} \leq \frac{L}{2}$. Воспользуемся неравенством Маркова, чтобы оценить вероятность события, когда количество «плохих» точек достаточно мало: $Pr[\texttt{количество «плохих» точек} < L] \geq 1 - \frac{L/2}{L} \geq \frac{1}{2}$.

Теперь оценим вероятность появления «хорошей» функции $Pr[g_i - \texttt{«хорошая»}] \geq p_1^K = p_1^{-\log_{p_2} 2n} = p_1^{\log_{p_1} 2n / -\log_{p_1} p_2} = (2n)^{-1/\log_{p_1} p_2} \geq n^{-1/\log_{p_1} p_2} = n^{-\log p_1 / \log p_2} = n^{-\rho}$.

Таким образом, выбрав $L = t \times n^{\rho}$ мы получим структру, где хорошая функция найдется с вероятностью $1 - \frac{1}{e^k}$ и при этом с вероятностью хотя бы $\frac{1}{2}$ количество плохих точек не будет превосходить $2L$. Значит мы решили исходную задачу с константой вероятностью и необходимыми ограничениями по времени и памяти. $\square$

3 Пример с метрикой Хэмминга

Предложение 1. *Пусть $S = \{0,1\}^d$ и $d(a,b)$ — расстояние Хэмминга между двумя битовыми строками. Тогда для любых c и R, семейство хэш-функций $H = \{h_i : h_i((b_1, b_2, \cdots, b_d)) = b_i\}$ является $(R, cR, 1 - \frac{R}{d}, 1 - \frac{cR}{d})$-чувствительным.*

Следствие 1. *Для любого $c > 1$ для задачи (R, cR)-приближенного поиска соседа в метрическом пространстве $\{0,1\}^d$ с расстоянием Хэмминга существует решение со следующими характеристиками:*

- *Время на запрос:* $O(n^{1/c}(\log_{1/p_2} n + d))$
- *Время на построение:* $O(n^{1/c} \log_{1/p_2} n)$
- *Используемая память:* $O(n^{1+1/c} + nd/\log n)$ *машинных слов*

Несложно показать, что $\rho = \frac{\log p_1}{\log p_2} \leq \frac{1}{c}$ (достаточно расписать вероятности и применить неравенство Бернулли). □

В случае, если нас только интересуют задачи, где $\frac{cR}{d} = O(1)$, то множитель асимптотики $\log_{1/p_2} n$ можно заменить на $\log n$.

4 Применение LSH для поиска похожих множеств

Поставим перед собой следующую задачу:

Задача 5. *Для заданной коллекции множеств $\mathcal{C} = \{c_i \subset [u]\}$ и конкретного запроса Q мы хотим найти множество $c_i : J(c_i, Q) \geq j_2$, при условии, что существует $c^* \in \mathcal{C} : J(c^*, Q) \geq j_1$. Имеет смысл рассматривать только случаи, когда $j_1 \geq j_2$.*

Задача 5 является конкретизацией задачи 4 для случая пространства подмножеств с метрикой Жаккара. Заметим, что J — это мера похожести, поэтому в отличие от задачи 4, в текущей формулировке все неравенства заменены на противоположные.

Мы будем пользоваться некоторыми результатами без какого-либо доказательства:

Теорема 2 (fast similarty-sketch construction)**.** *Для множества $A \subset [u]$ существует алгоритм, который за время $O(t \log t + |A|)$ вычисляет скетч S_t размера t множества A, удовлетворяющий следующему свойству:*

Для пары множества A и B определим случайную переменную

$$X_i = \begin{cases} 1 & \textit{если } S_t(A)[i] = S_t(B)[i] \\ 0 & \textit{иначе} \end{cases}$$

Тогда если $X = \frac{1}{t}\sum_{i \in [t]} X_i$, то $E[X] = J(A, B)$ и отклонение значений переменной X можно оценить с помощью границ Чернова.

Скетчи $S_t(A)$ обладают еще одним замечательным свойством:

Теорема 3. *Для пары множества A и B с мерой Жаккара J построим скетчи размером t. Тогда для произвольного подмножества индексов $I \subset [t]$ размера k верно следующее:*

$$E[\prod_{i \in I} X_i] \leq J^k$$

И если $tJ \geq k - 1$, то

$$E[\prod_{i \in I} X_i] \geq \frac{(tJ)^{\underline{k}}}{t^{\underline{k}}}$$

Последнее свойство говорит о том, что компоненты скетча достаточно независимы друг от друга и могут быть использованы как хэш-функции по отдельности.

Для решения задачи 5 мы должны построить L составных хэш-функций g_i. Каждая такая функция будет состоять из K компонент, случайным образом выбранных из скетча $S_t(A)$. Данный подход можно реализовать с помощью таблицы T размером $L \times K$, где элемент $T[i, j]$ будет равноверноятно случайно выбран из диапазона $[j \cdot t/K \dots (j + 1)t/K)$. Тогда

$$g_i(A) = (S_t(A)[T[i, 0]], S_t(A)[T[i, 1]], \cdots S_t(A)[T[i, K - 1]])$$

.

Множество значений функций g_i для конкретного множества A может быть вычислено за время $O(LK + t \log t + |A|)$. Для дальнейших рассуждений, а также для того, чтобы значения таблицы $T[i, j]$ были корректно определены, положим $t = K \cdot [1 + K(\frac{1}{j_1} - 1)]$.

Ограничим число «плохих» множеств $A : J(A, Q) \leq j_2$, которые могут попасть в корзинки, соответствующие значениям $g_1(Q), g_2(Q), \cdots, g_L(Q)$:

Лемма 1. *Если взять $K = \log_{1/j_2} n$, то для множества $A \in \mathcal{C}$: $J(A, Q) \leq j_2$ вероятность того, что $g_i(A) = g_i(Q)$ не превосходит $\frac{1}{n}$*

Чтобы значения хэш функций совпали, необходимо, чтобы для множества $I = \{T[i, 0], T[i, 1], \cdots T[i, K-1]\}$ было верно равенство $\prod_{i \in I} X_i = 1$, которое выполняется с вероятностью $J(A, B)^K \leq \frac{1}{n}$, согласно теореме 3. $\square$

Из леммы 1 следует, что матожидание числа «непохожих» множеств A, которых мы найдем в корзинках, соответствующих значениям $g_i(Q)$, не превосходит L.

Лемма 2. *Если взять $K = \log_{1/j_2} n$ и $L = j_1^{-K}$, то для множества $A : J(A, Q) \geq j_1$ верно, что $Pr[\exists i : g_i(A) = g_i(Q)] = O(1)$.*

Доказательство оставляется докладчику в качестве упражнения. □

Используя две предыдущие леммы, мы получаем алгоритм, отвечающий на запрос за время $O(L|Q|)$. Чтобы добиться аддитивной асимптотики $O(L + |Q|)$ разберем два случая:

Обозначим за $M = \{(i, A) : g_i(A) = g_i(Q)\}$

- $|M| \geq CL$ для некоторой большой константы C. В этом случае воспользуемся неравенством Маркова, чтобы вычислить вероятность небольшого количества «плохих» элементов: $Pr[\texttt{количество «плохих»} \leq \frac{CL}{2}] \geq 1 - \frac{L}{CL/2} = 1 - \frac{2}{C}$. Таким образом, если мы случайным образом вытащим элемент из $M' \subset M : |M'| = CL$, то с большой вероятностью — это будет «хороший» элемент. Чтобы произвести все эти вычисления, нам понадобится $O(L + |Q|)$ операций.

- $|M| \leq CL$. Для каждого множества из $\mathcal{C}$ и запроса Q построим еще один скетч размером $\Theta(\log n)$. Данное действие не повлияет на итоговую асимптотику, т.к. $t = \Omega(\log n)$. После этого для каждого $(i, A) \in M$ мы воспользуемся алгоритмом, позволяющим с высокой вероятностью определить, что $J(A, B) \leq j_2$, используя скетч размером $\Theta(\log n)$. Матожидания числа шагов для такого алгоритма в случае малой похожести множеств можно оценить как $E[\texttt{количество шагов}] \leq r$, для некоторой большой константы r, при это вероятность негативного срабатывания не превосходит $\frac{1}{n}$.

Таким образом мы получили алгоритм решения задачи 5, занимаюий $O(n^{1+\rho} + \sum_{A \in \mathcal{C}} |A|)$ машинных слов и отвечающий на запрос за $O(n^\rho \log n + |Q|)$ с некоторой константой вероятностью (вычисление базового скетча за $O(t \log t)$ можно опустить в асимптотике, т.к. это слагаемое подавляется слагаемым порядка $O(n^\rho \log n)$).