Санкт-Петербургский государственный университет
Прикладная математика и информатика
Статистическое моделирование

Кухтина Дарина Александровна

Фильтр Блума

Конспект

Санкт-Петербург
2016

Оглавление

Введение

В последнее время объемы информации становятся очень большими, и для того, чтобы снизить затраты на ее обработку нужно использовать вероятностные методы. Ключевая их идея в том, что жертвую некоторой информацией можно значительно сократить время обработки запроса. Обычно они используется для уменьшения числа запросов к несуществующим данным в структуре данных с более дорогостоящим доступом (например, расположенной сетевой базе данных), то есть для "фильтрации"запросов к ней.

Глава 1

Основные понятия

Фильтр Блума — это реализация вероятностного множества, придуманная Бёртоном Блумом в 1970 году, позволяющая компактно хранить элементы и проверять принадлежность заданного элемента к множеству.

Поясним что значит вероятностное множество.

Определение 1. *Вероятностное множество — структура данных, способная добавлять элемент в множество, а также выполнять запросы проверки принадлежности элемента множеству. При этом существует возможность получить или положительный, но неопределенный ответ (элемента в множестве нет, но структура данных сообщает, что он есть), или отрицательный определенный ответ (элемент точно не содержится в данном множестве).*

По сути, вероятностное множество это структура, позволяющая определить:

- элемент точно не принадлежит множеству
- элемент принадлежит множеству с некоторой вероятностью.

Для определения Фильтра Блума также нужно дать определение хеш-функции.

Определение 2. *Хеширование – это преобразование массива (входных данных) определенной длины в выходную битовую строку (выходные данные) фиксированной длины. Функция, выполняющая данное преобразование называется хеш-функцией, значение хеш-функции называется хеш-код.*

Перейдем к определению Фильтра Блума.

Пусть у нас есть n – количество элементов в множестве. Фильтр Блума представляет собой битовый массив из m бит и k различных хеш-функций $h_1 \dots h_k$, равновероятно отображающих элементы исходного множества во множество $\{0, 1, \dots m-1\}$, соответствующее номерам битов в массиве. Изначально, когда структура данных хранит пустое множество, все m бит обнулены.

Формализуем данное определение. У нас есть kn независимых случайных величин ξ_i, равномерно распределенных на $(1, \dots, m)$ – это наши k хеш-функций на n объектах. Так

как значения хеш-функций должны быть вычислены по объекту, то мы считаем, что имеем реализацию kn- мерной случайной величины. При добавлении нового объекта мы добавляем еще k случайных величин.

Глава 2

Алгоритмы добавления элемента, проверки элемента множеству

2.1. Добавление элемента

Опишем процедуру добавления элемента в Фильтр. Входные параметры:

- m – размер битового массива,
- h_i – хеш-функции,
- k – количество хеш-функций,
- n – количество элементов во множестве.

Как было сказано выше – изначально все m бит обнулены.
Для добавление элемента e необходимо записать 1 на каждую из позиций $h_1(e), \ldots, h_k(e)$ битового массива.
Формально алгоритм на псевдокоде выглядит следующим образом: Входные параметры: x – объект для добавления
Обозначения: B – битовый массив
for j: $1 \ldots k$ do
$i \leftarrow h_j(x)$
if $B_i == 0$ then
$B_i \leftarrow 1$
end
end
.

2.2. Пример

Пусть m=7, размер нашего битового массива, $k = 3$ – количество наших хеш-функций. Мы хотим добавить в пустой фильтр $F = \{0, 0, 0, 0, 0, 0, 0\}$ элемент A, для которого $h_1(A) = 2$, $h_2(A) = 3$ и $h_3(A) = 6$. Тогда после добавления элемента A в изначально

пустой фильтр получим, что на позиции 2, 3 и 6 в массиве F появятся единицы, то есть $F = \{0, 1, 1, 0, 0, 1, 0\}$.

2.3. Проверка принадлежности элемента множеству

Для того, чтобы проверить принадлежность элемента c множеству, нужно проверить, что равны 1 все биты $h_1(c), \ldots, h_k(c)$ битового массива. Если хотя бы один из этих элементов равен нулю, то элемент не принадлежит массиву.

Формально алгоритм выглядит так:

Входные параметры: x – объект для проверки

Обозначения: B – битовый массив

$l \leftarrow 1$;

$j \leftarrow 1$;

while $m == 1$ and $j \leq k$ do

$i \leftarrow h_j(x)$;

if $B_i == 0$ then

$l \leftarrow 0$;

end

$j \leftarrow j + 1$;

end

return l;

Видно, что алгоритм может допустить ложноположительное срабатывание – считаем, что элемент принадлежит множеству, хотя он не принадлежит. В этом случае 1 на местах $h_1(c), \ldots, h_k(c)$ появились в результате добавления других элементов: Также стоит

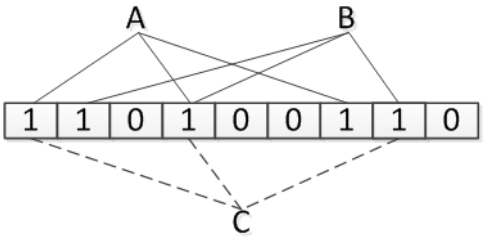

Рис. 2.1. C не принадлежит множеству!

отметить что алгоритм не допускает ложноотрицательного срабатывания – считаем,

что элемент не принадлежит множеству, хотя он принадлежит. Это полезное свойство следует напрямую из алгоритма добавления элемента.

Глава 3

Минимизация ложноположительного срабатывания

Ложноположительным срабатыванием мы называем ситуацию, когда фильтр говорит, что объект принадлежит множеству, но на самом деле это не так – все проверяемые биты заполнены при добавлении других элементов. Ясно, что мы хотим уменьшить вероятность такого события.

Рассмотрим событие $B_j = \{\xi_i = j$ хотя бы для одного $i = 1, \ldots, nk\}$, $j = 1, \ldots, m$. $P(B_j) = 1-(1-\frac{1}{m})^{nk}$ Формально задача сводится к вычислению вероятности $P(B_1, \ldots, B_k) \approx P(B_1) \times \ldots \times P(B_k) = (1 - (1 - \frac{1}{m})^{nk})^k$.

Мы определили чему равно вероятность ложноположительного срабатывания,и хотим ее минимизировать. То есть $(1 - (1 - \frac{1}{m})^{nk})^k \to min$. Для достаточно большого m в силу второго замечательного предела имеем:$(1 - (1 - \frac{1}{m})^{kn})^k \approx (1 - \exp^{-\frac{kn}{m}})^k$. Минимизирую, получаем $k_{opt} = \frac{m}{n} ln(2)$.

Глава 4

Выбор ключевых параметров

Выбор параметров играет большую роль для минимизации ложноположительного срабатывания. Рассмотрим что произойдет при увеличении значений параметров? Зафиксируем оптимальное количество хеш-фенкций $k = k_{opt}$. Вспомним, что вероятность ложноположительного срабатывания при фиксированном k_{opt} равна: $P = (1 - \exp^{-\frac{k_{opt}n}{m}})^k_{opt}$. Следовательно:

- m – размер битового массива – чем больше, тем меньше вероятность ложноположительного срабатывания
- n – число элементов в множестве – чем больше, тем больше вероятность ложноположительного срабатывания.

Хеш-функция является ключевым элементом вероятностного фильтра. Заметим , что для фильтра Блума вероятность ложноположительного срабатывания зависит только от отношения $\frac{m}{n}$, и не зависит от выбора хеш-функции. Однако для уменьшения времени и увеличенения мощности выбор хеш-функции играет важную роль. Учитывая объемы данных это стоит принять во внимание.

Глава 5

Операции над Фильтрами Блума

Как было сказано ранее, стандартный фильтр Блума не поддерживает операции удаления элементов. Возможно поддержание операции удаления элемента из фильтра, если использовать второй фильтр Блума который будет хранить удаленные элементы. Проблема этого подхода состоит в том, что для второго фильтра также возможно ложноположительное срабатываение, что в свою очередь приведет к тому, что для перового фильтра будет ложноотрицательное срабатывание – то есть мы получим, что элемента нет в фильтре, хотя он не был удален. Естественно такой подход не является хорошим. Однако, Фильтр Блума поддерживает операции объединининения и пересечения. Это может быть выполнено при условии, что они одинакового размера и с одинаковым набором хеш-функций. Эти операции могут быть реализованы путем побитовых операций объединения и пересечения. Как при этом меняется вероятность ложноположительного срабатывания? Есть теоретические результаты:

Теорема 1. *Вероятность ложноположительного срабатывания для* $BF(B \cup A) \geq$ *вероятности ложноположительного срабатывания для* $BF(A)$ *и* $BF(B)$.

Теорема 2. *Если* $BF(A \cap B)$,$BF(A)$ *и* $BF(B)$ *имеют одинаковые* m *и* h_i, *то* $BF(A \cap B) = BF(A) \cap BF(B)$ *с вероятностью*

$$(1 - \frac{1}{m})^{k^2|A-A\cap B||B-A\cap B|}$$

Глава 6

Модификация Фильтра Блума: удаление элемента

Для стандартного фильтра Блума невозможна операция удаления элемента. Однако было бы довольно удобно, если бы фильтр поддерживал данную операцию. Это возможно, если используется модификация фильтра Блума со счетчиками. При добавлении элемента мы увеличиваем счетчик, при удалении элемента – уменьшаем. Формально алгоритм добавления для модификации Фильтра Блума со счетчиками выглядит так:

Входные параметры: x – объект для добавления

Обозначения: B – битовый массив, C – массив счетчиков

for j: $1 \dots k$ do

$i \leftarrow h_j(x)$

$C_i \leftarrow C_i + 1$

if $B_i == 0$ then

$B_i \leftarrow 1$

end

end

Рассмотрим также удаление элемента из Фильтра: Алгоритм удаления для модификации Фильтра Блума со счетчиками:

Входные параметры: x – объект для удаления

Обозначения: B – битовый массив, C – массив счетчиков

for j: $1 \dots k$ do

$i \leftarrow h_j(x)$

$C_i \leftarrow C_i - 1$

if $C_i \leq 0$ then

$B_i \leftarrow 0$

end

end

Поиск элемента происходит таким же образом, что и для стандартного фильтра Блума.