Бандиты в Query Selection

Шестаков Владимир

Аннотация

В работе исследуются способы решения задачи Query Optimization. Рассматриваются конкретные решения, использующие метод многоруких бандитов. Среди таких решений выделяется одно конкретное с названием Bao [1], использующее сэмплирование Томпсона. Данное решение проверяется на практике и исследуется на возможность доработки.

Keywords: Query Optimization, Multiarmed Bandits, Thompson sampling

1 Введение

В современном мире требуются новые инструменты для эффективной обработки данных. В основе подобных систем лежат базы данных, но с увеличением сложности возникает задача оптимизации запросов. Методы, использующие статические правила и эвристики, оказываются не способными выдержать динамические изменения. Современные подходы, использующие машинное обучение, сталкиваются с рядом ограничений: длительное время обучения; неспособность адаптации к изменениям в данных; проблема "хвостовых"случаев, связанная с тем, что запросы исполняются быстро в среднем, но даже редкие случаи длительного исполнения неприемлемы в реальном мире. Таким образом, многие методы становятся непрактичными для их использования.

Среди различных методов выделяется Bao (Bandit Optimizer) [1] — один из способов оптимизации запросов, который успешно преодолевает указанные ограничения. В отличие от других подходов, Bao не заменяет традиционный оптимизатор, а дополняет его, работая как надстройка над уже существующим. Основная идея, заложенная в Bao, это выделение нескольких различных подсказок — изменений в текущем запросе, — которые оптимизатор может применить. Некоторые конечные подмножества таких подсказок используются как "руки"в методе многорукого бандита. Таким образом, сформулировав задачу как контекстного многорукого бандита, Bao использует сэмплирование Томпсона для нахождения лучшего набора подсказок среди предложенных.

В статье будет воспроизведена работа Bao, проверена его эффективность на тестовых данных, предложены и проверены возможные улучшения. Цель работы —

показать, каким образом различные изменения в данном алгоритме могут отразиться на результатах в оптимизации исполнения запросов.

2 Постановка задачи

Для начала введём несколько определений, относящихся к самой задаче Query Selection — выбор оптимальной стратегии для определённого запроса, а также к сведению Query Optimization к ней.

Обозначим соответствующий набор подсказок как $HSet_i$, будем использовать далее это обозначение как соответствующую ему функцию, сопоставляющую запросу $q \in Q$, где Q — множество всевозможных запросов, дерево плана запроса $t \in T$, где T — все деревья плана запроса, какие только могут быть. Все наборы подсказок должны содержаться в одном определённом конечном семействе F, зафиксированном изначально. Обозначим метрику, по которой будет происходить оптимизация, как функцию $P : T \to R$, т.е. действующую из множества деревьев плана запроса в множество действительных чисел. Данная метрика в базовом понимании является временем, затраченным на исполнение плана запроса. Также обозначим функцию выбора набора подсказок как $B : Q \to F$. Нам нужно минимизировать значение функции, так что определим потерю R_q как $(P(B(q)(q)) - \min\limits_i P(HSet_i(q)))^2$. Задача состоит в минимизации как отдельных потерь, так и в среднем, учитывая также время обучения и время выбора руки. К сожалению, время обучения трудно описать математической моделью, поскольку время обучения зависит от объёма данных и архитектуры модели, а также метода обучения. Время выбора руки же всегда зависит напрямую от архитектуры модели, при этом для каждой модели оно фиксированное. В постановке задачи пренебрежём данным временем, поскольку оно несущественно.

Теперь определим то, как выбирается стратегия. Пусть дано множество моделей M, предсказывающих по запросу, какую из рук выбрать, т.е. на запрос q выдают определённый набор подсказок (условно будем считать его числом от 1 до $|F|$ и сопоставлять числу i набор подсказок $HSet_i$). Проверяя выбранный план запроса, значения параметров модели обновляются. Среди таких моделей рассмотрим несколько фиксированных, соответствующих конкретным стратегиям многорукого контекстного бандита. Заодно получаем ответ, каким образом получается задача о контекстном бандите: контекстом является сам запрос (преобразованный в вектор фиксированной длины), множество наборов подсказок служит руками такого бандита.

Одним из кандидатов в такие модели, предложенный в статье, является стратегия, основанная на сэмплировании Томпсона. Для этого внутри стратегии оптимизируются на каждом шаге параметры модели, что должна предсказывать возможное значение метрики, полученной на планах запросов. Оптимизация происходит путём минимизации суммы потерь на значениях метрики от уже полученных результатов. Говоря формально, до этапа обучения было выполнено несколько планов выполнения запросов, пусть это будет множество пар (запрос, значение метрики) T_{prev}, на этапе обучения пусть оптимизируется параметр θ, что для модели m, принимающей условно θ и x, воз-

вращая $y = m(\theta, x)$, тогда выбирается параметр $\theta' = \underset{\theta}{\operatorname{argmin}} \sum_{(x,y)\in T_{prev}} f(m(\theta, x), y)$, где f — функция потерь, условно MSE. Сама модель m в оригинале является Tree Convolution Network — специальной моделью, разработанной чтобы предсказывать по бинарным деревьям некие значения (именно бинарным деревом и является дерево плана запросов). Как и свёрточные нейросети, в ней используются знания об узлах, включая пары родитель-ребёнок среди узлов. Подробнее об этой модели можно прочитать в данной статье[2]. На этапе выбора рычага выбирается тот, что даёт наименьшее предсказанное моделью значение метрики.

Кроме различных моделей далее будут рассмотрены другие способы оптимизации, такие как выбор иного метода обучения модели, что позволит либо сильнее приблизить значение искомого параметра к оптимальному, либо сделать это же быстрее, чем происходит с исходной моделью.

3 Предлагаемые решения

Из предложенных решений выделяются два вида, изменение модели и изменение метода обучения. Далее в этой части статьи будет много предположений, которые затем будут проверены на практике.

Начнём с первого вида. В оригинальной статье[1] приводится несколько примеров того, что будет, если заменить модель Bao на Random Forest или линейную, но остаётся ещё много вариантов модификаций изначальной нейронной сети.

Первый вариант, с помощью которого можно изменить модель, является изменение одного или нескольких слоёв на байесовские. В байесовком слое каждый параметр теперь считается не как некое фиксированное число, а как принадлежащий распределению со своими фиксированными параметрами. Таким образом, при каждом вызове модели на одном и том же значении могут быть получены самые разные числа. Таким образом можно достичь нескольких целей: во-первых, с помощью этого уменьшается риск переобучения, что очень важно в случае небольшого объёма тренировочных данных, во-вторых, поскольку в тестовых данных о значении метрики определённо есть шум, что появляется из-за невозможности ни точно измерить время исполнения дерева плана запроса, ни воспроизвести этот процесс одним и тем же образом каждый раз, поэтому если получится предсказать данный шум с помощью сэмплирования параметров из некоторого распределения, значение функции потерь может также уменьшиться в среднем. Параметры самих распределений находятся с помощью метода максимального правдоподобия. Подробнее о байесовских нейронных сетях можно прочесть здесь[3].

Второй вариант, это использовать иную стратегию контекстного многорукого бандита. Например, одну из тех стратегий, что предсказывает не значение, а матожидание от всех возможных значений, после чего выбирается рычаг с минимальным из них. Можно также предсказывать верхнюю границу интервала достоверности.

Перейдём ко второму виду, затрагивающему обучение модели.

Первое, что можно сделать, это реализовать перебор гиперпараметров. Гиперпараметры модели сильно влияют качество модели. При этом сами оптимальные гиперпараметры могут меняться в зависимости от количества предоставленных данных.

Второе, в предоставленном алгоритме отсутствует разделение на тренировочную и тестовую выборки. Из-за этого возрастает риск переобучения модели. Как вариант, можно добавить данное разделение, плюс использовать валидационную выборку для нахождения лучшей модели.

Далее идут ещё нереализованные способы.

Третье, в текущей реализации обучение и использование модели последовательно сменяются друг другом. Довольно логично использовать модель и обучать новую параллельно, чтобы при этом не было проблем с задержками из-за времени обучения между запросами.

Четвёртое, из-за накопления данных, в том числе одинаковых, время обучения заметно возрастает. Для избежания этого можно поставить ограничение сверху на допустимый объём хранимых данных и время от времени прореживать их.

Пятое, вместо прореживания данных, можно поступить иначе. Если обучать модель не с нуля, а используя как отправную точку состояние предыдущей модели, дообучая на новых данных, время обучения не будет увеличиваться с каждой новой итерацией. Минусом же является то, что точность предсказаний модели может заметно упасть.

Список литературы

[1] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. Bao: Making learned query optimization practical. *Proceedings of the 2021 International Conference on Management of Data*, June 2021.

[2] Lili Mou, Ge Li, Lu Zhang, Tao Wang, and Zhi Jin. Convolutional neural networks over tree structures for programming language processing. 2014.

[3] Tulupyev, Nikolenko, and Sirotkin. *Cycles in Bayesian networks: probabilistic semantics and relations with neighboring nodes*, volume 1. SPIIRAS, March 2014.