ИСПОЛЬЗОВАНИЕ МАШИННОГО ОБУЧЕНИЯ В УПРАВЛЕНИИ БАЗАМИ ДАННЫХ

Максим СТРОПША

Департамент Программной Инженерии и Автоматики, группа TI-192 F/R, Факультет Вычислительной Техники, Информатики и Микроэлектроники, Технический Университет Молдовы, Кишинев, Республика Молдова

Автор корреспондент: Максим СТРОПША, e-mail: stropsa.maxim@iis.utm.md

Научный руководитель: Дориан САРАНЧУК, DISA, FCIM, UTM

Резюме: *Доклад посвящен применению машинного обучения для управления базами данных. Традиционные методы управления данными (например, разделение данных, настройка ручек, переписывание запросов, оценка стоимости) носят эмпирический характер и в значительной степени зависят от вмешательства человека. Эти методы не способны удовлетворить требования высокой производительности для различных приложений и масштабируемости. Объем корпоративных данных увеличивается, и очень важно найти эффективные способы их обработки. Машинное обучение может автоматизировать и оптимизировать процессы управления базами данных.*

Ключевые слова: машинное обучение, управление базами данных, большие данные, оптимизация производительности.

Введение

Традиционные методы управления данными (например, разделение данных, настройка ручек, переписывание запросов, оценка стоимости) носят эмпирический характер и в значительной степени зависят от вмешательства человека для настройки и обслуживания баз данных. Однако существующие эмпирические методы не способны удовлетворить требования высокой производительности для различных приложений, крупномасштабных экземпляров баз данных и различных пользователей [1].

Ожидается, что общий объем корпоративных данных, генерируемых каждый год, увеличится с 18,2 ЗБ в 2015 г. до 175,8 ЗБ к 2025 г. Согласно опросу Seagate, только 32% этих данных доступны в состоянии готовности к анализу. Поскольку объемы данных растут быстрее, чем предприятия могут за ними угнаться, очень важно найти эффективные способы хранения, управления и анализа данных. Один из подходов заключается в использовании машинного обучения (ML) для улучшения структуры базы данных и оптимизации запросов [2].

Цель статьи - понять, как машинное обучение может влиять на стратегию проектирования хороших баз данных и оптимизировать операции с базами данных. Понимание современного состояния этих областей может помочь бизнесу управлять и извлекать полезную информацию из собранных данных.

Проект адресует три вопроса, связанные с базами данных: создание эффективного индекса, неэффективные запросы JOIN и возможность выполнять моделирование и анализ внутри базы данных, а не платить за передачу данных в внешний инструмент.

Связанная работа

В статье 2000 года Xindong Wu [3] идентифицировал будущие вызовы в комбинации машинного обучения и технологии баз данных для создания систем Интеллектуального обучающего баз данных (ILDB). Wu [3] идентифицировал две основные области, которые фокусируются на эффективном представлении данных и методах машинного обучения, способных обрабатывать системы баз данных предприятий. Он также отметил, что реальные

базы данных обычно шумны и содержат отсутствующие или неверные данные, что представляет собой вызов для систем обучения. Хотя статья Wu [3] не современная, она предоставила основное представление возможных проблем для будущего.

В обзоре литературы 2015 года Najafabadi [4] идентифицировали возможные проблемы при адаптации текущих методов машинного обучения к современным сценариям с базами данных. Авторы определили следующие области: потоковые данные(streaming data), многомерные данные(high-dimensional data), масштабируемость моделей и распределенных вычислений. Эти области связаны с проблемами алгоритмов машинного обучения, а не с проблемами использования машинного обучения для создания более совершенных систем баз данных.

Индексация

Разработчики баз данных создают структуру данных, называемую индексом, для упорядочения данных, чтобы к ним можно было быстро получить доступ для эффективного выполнения запросов. Тип используемого индекса определяется как структурой данных, так и типами запросов, которые будут выполняться чаще всего. Наиболее распространенными индексами являются BTree-Index для поиска диапазонов, Hash-Index для одиночного поиска и Index для определения наличия ключа.

Все эти индексы создаются с фиксированными структурами данных, которые не меняются в ответ на основные шаблоны данных. Бейтель и др. утверждали, что эти индексы не используют шаблоны данных для дальнейшей оптимизации работы базы данных.

Бейтель и др. продемонстрировали повышение скорости выполнения запросов на 44% и сокращение памяти на 75%, необходимой для хранения индекса с использованием индексов нейронной сети, по сравнению с BTree-Index в базе данных только для чтения. Использование индексов на основе ML имеет некоторые недостатки, такие как необходимость планировать ложноотрицательные результаты (т. е. индекс ML сообщает, что ключ не существует, когда он существует), которые не встречаются с традиционными индексами, и индексы ML возвращают приблизительные местоположения данных, а не точные местоположения записей.

Для решения этих проблем Бейтель и др. объединять изученные индексы с традиционными индексами, позволяя более быстрому индексу ML выполнять большую часть поиска, в то время как традиционный индекс гарантирует правильность поиска. Хотя в этом исследовании рассматривались только базы данных только для чтения, авторы утверждают, что индекс онлайн-обучения ML может обрабатывать доступные для записи базы данных.

Рабочие нагрузки на системы управления базами данных (СУБД) обычно меняются со временем. Например, приложение для отслеживания автобусов может иметь высокую рабочую нагрузку в течение дня, когда автобусы ходят, и мало запросов ночью, когда автобусы не ходят.

Согласно Ma[4] и соавт., СУБД, которая может прогнозировать свою рабочую нагрузку, может оптимизировать свои индексы, чтобы работать лучше по мере изменения рабочей нагрузки и работать автономно. Администраторы-люди несут ответственность за ручную настройку оптимизации в текущей СУБД. Ma [4] и др. представить QueryBot 5000, структуру прогнозирования рабочей нагрузки, которая предсказывает будущие модели рабочей нагрузки с использованием ансамблевой модели линейной регрессии, рекуррентной нейронной сети и кластеризации.

Кластеризация запросов имеет решающее значение для снижения сложности моделирования рабочей нагрузки и обеспечения эффективного прогнозирования. Ma и др. обнаружил, что пять крупнейших кластеров могут моделировать 95% рабочей нагрузки экспериментальных систем. Эта структура прогнозирует рабочую нагрузку как в краткосрочной, так и в долгосрочной перспективе, позволяя автоматизированной СУБД выбирать наилучшие оптимизации.

В отличие от оптимизирующей платформы Бейтель и др., платформа QueryBot 5000 использует онлайн-модели машинного обучения, которые адаптируются к изменениям

поступающих рабочих нагрузок в режиме реального времени. Одним из ограничений этой структуры является то, что по-прежнему требуется некоторая ручная настройка порога, при котором рабочая нагрузка изменилась достаточно, чтобы вызвать необходимость повторной кластеризации.

Индексация в базах данных - исследования включают использование машинного обучения для создания эффективных индексов. Цель состоит в удалении зависимости DBMS от экспертов в домене для выбора индексов, и вместо этого дизайн индексов основан на схеме базы данных и нагрузке, которую она получает. Архитектура, названная SmartIX, абстрагирует задачу анализа и принятия решения о том, будут ли кандидатские столбцы для индекса улучшать производительность, задачу, которую обычно выполняют люди. Вместо этого SmartIX содержит агента обучения по награде, который исследует возможные кандидатские столбцы для индекса. Он также содержит среду, где агент получает награды за свои решения и в которой его следующие действия передаются в третью компоненту - интерфейс DBMS, который записывает команды для изменения базы данных и также читает статистику о текущей производительности индексов. Обучение агента было проведено с использованием двухслойной нейронной сети перцептрона за 64 000 шагов времени, где статистика обучения собиралась каждые 128 шагов и использовалась для предоставления графического представления результатов. При сравнении производительности конфигурации индекса, созданной SmartIX, с конфигурациями, созданными генетическими алгоритмами и др

Оптимизация запросов на соединение

Объединение нескольких таблиц вместе для построения результата запроса — одна из самых затратных операций с базой данных. Операции соединения являются дорогостоящими, поскольку база данных должна определить наиболее эффективный порядок таблиц, участвующих в объединении. Это решение обычно принимается с использованием эвристики, чтобы сузить область поиска.

) был разработан Марклом и его коллегами в 2003 году как первый пример оптимизатора запросов, который использует машинное обучение для повышения производительности за счет изучения исторических данных.

Чтобы исправить ошибки моделирования в предыдущих планах выполнения запросов, постепенно корректирует оценки кардинальности и избирательности для предикатов. LEO может обнаруживать ошибки оценки во время выполнения запроса и автоматически повторно оптимизировать запрос. LEO работает, регистрируя фактические мощности и учась уменьшать разницу между фактическими и прогнозируемыми мощностями. Проблема сходимости, в которой скорость обучения модели сильно зависит от данных и предположений о данных, была одной из проблем, с которыми столкнулись Маркл и его коллеги, это все еще распространено в современном машинном обучении.

Авторы решили эту проблему, позволив LEO работать в двух режимах: исследовательском, который выбирает рискованные, но недорогие планы на основе предположений, и конкретном, который отдает предпочтение планам, основанным на опыте и реальных фактах. LEO также решает проблему коррелированных соединений, обнаруживая разницу между предсказанным и фактическим числом элементов. Еще одна проблема, поднятая авторами, заключается в том, чтобы определить, когда стоит повторно оптимизировать запрос в середине выполнения из-за нелинейности в модели затрат, проблема, позже решенная Кришнаном и др.

Лю и др. использовали трехслойную нейронную сеть для оптимизации запросов на соединение. Чтобы сгенерировать начальные обучающие запросы для своей нейронной сети, они разработали собственный алгоритм.

Нейронная сеть может считывать из запросов только нестрогие операторы диапазона, такие как "=" или ">=", но они разработали методы для перезаписи всех других реляционных операторов, таких как "=" и "" или ">". Они сравнили нейронную сеть с оптимизатором

запросов, встроенным в IBM DB2, систему управления базами данных. Результаты показали, что нейронные сети превзошли методы оценки, основанные на статистике, при оценке кардинальности запросов.

Однако Лю и соавт. заявили, что хотели бы проделать больше работы, например, иметь возможность сравнивать столбцы с операторами отношения, потому что на данный момент можно сравнивать значения столбцов только с константами (c1 >= 3, но не c1 >= c2). Во-вторых, они хотели поэкспериментировать с различными типами нейронных сетей, такими как RNN и глубокие нейронные сети. В более поздних разработках в этой области использовались более сложные модели.

Было показано, что оптимизатор соединения RL создает лучшие планы оптимизации, а также работает более чем в десять раз быстрее, чем традиционные методы оптимизации, такие как левосторонняя эвристика. Одним из ограничений этого исследования является то, что авторы рассматривают только один тип операции соединения, который составляет большинство протестированных тестов.

Моделирование в базе данных

Другая интересная область исследования - разработка новых языков программирования для создания моделей машинного обучения в базе данных. Эти модели могут использовать близость данных и избежать необходимости длительных конвейеров для перемещения данных в другое хранилище для анализа.

MADlib — это аналитическая библиотека для Greenplum и SQL разработал Хеллерштейн. MADlib поддерживает реализация традиционных алгоритмов машинного обучения такие как SVM и K-means. Он также поддерживает множество контролируемых и неконтролируемых алгоритмов машинного обучения, является открытым исходным кодом. Его кодовая база получает вклад как от промышленности, так и от научных кругов. Цель авторов заключалась в том, чтобы «ускорит инновации и передачу технологий в Сообщество Data Science через общую библиотеку масштабируемой аналитики базы данных». Машинное обучение и другие продвинутые методы анализа с MADlib реализованы с использованием простые SQL-скрипты. Чтобы запустить последовательность этих операторов SQL или выполнить итеративный анализ, можно также использовать файл драйвера на основе Python для запуска MADlib. Пользователи также имеют доступ к шаблонным запросам, которые помогают, когда схема для входные таблицы модуля не обязательно фиксированы. MADlib нашла применение в сотрудничестве с исследователями из Университета Висконсина, а также из Университета Флориды, и в Беркли. MADlib еще не нашел коммерческого применения. Другой похожий язык — MLog, разработанный Ли, декларативный язык программирования, который взаимодействует непосредственно с базой данных. MLog улучшает MADlib на позволяя пользователям создавать более сложные модели. MLog позволяет пользователям создавать сложные модели глубокого обучения в базы данных таким же образом, как пользователи могут открывать сеанс SQL и выполнять простые запросы к данным на месте. Глубокий модели обучения обычно требуют большого объема обучения данные должны быть точными. MLog позволяет создать модель где находятся данные, экономя время и упрощая процесс построения модели. Программное обеспечение работает, просматривая данные как тензоры, и включает в себя функциональные возможности для автоматической оптимизации процесса построения модели. Модели машинного обучения, созданные с помощью Декларативный язык запросов MLog может быть скомпилирован в нативные программы TensorFlow с сопоставимой производительностью до оптимизированных вручную моделей. Авторы обнаружили, что в некоторых случаях программы TensorFlow, созданные с помощью MLog, все еще могут быть до двух раз медленнее, чем код, настроенный вручную. Другое ограничение заключается в том, что язык MLog не был полностью интегрирован с SciDB.

Вызовы

Ван и др. обсуждает, как обе системы глубокого обучения может быть улучшена с точки зрения базы данных и как можно улучшить приложения баз данных с помощью глубокого обучения методы. Они обнаружили, что глубокое обучение может использовать преимущества методов оптимизации, используемых в области баз данных. в обучении своих моделей. Такие методы, как операция планирование, управление памятью, параллелизм и параллелизм/непротиворечивость. Точно так же поле базы данных может использовать обработка естественного языка, отличительная черта глубокого обучения, интерпретировать неформальные запросы к базе данных и переводить их в формальные запросы. Кроме того, планирование запросов, традиционно сделать с использованием сложных моделей затрат в базе данных, вместо этого может быть сделать более эффективно с помощью модели RNN. Наконец, глубокое обучение может помочь в анализе пространственно-временных данные, хранящиеся в базах данных, путем сопоставления проблемы с образную структуру, а затем с помощью сверточной нейронной сеть для использования пространственной локализации данных. В общем, обмен методами и представлениями между сообщества баз данных и глубокого обучения принесут большую пользу обеим областям. Guo и Daudjee описывают ограничения недавнего использования глубокое обучение для оптимизации запроса на соединение. Они обнаружили, что в худшем случае эти модели создают планы выполнения, которые на порядки дороже, чем самый дешевый план и может даже привести к сбою СУБД, требуя слишком много памяти.

Они утверждают, что дополнительное время, необходимое для выполнения худшего планирует, что результаты модели могут перевесить сэкономленное время с помощью глубокого обучения. Это указывает на необходимость уравновешивания высокая производительность с последовательностью. Авторы считают, что необходимы дополнительные исследования для повышения надежности модели для это приложение. Авторы также утверждают, что лучшая функция кодирование необходимо для расширения запроса на соединение с глубоким обучением оптимизаторы к корпоративным базам данных со многими таблицами, которые могут часто меняются, что требует переобучения модели. Они утверждают эта функция кодирования, устойчивая к изменениям таблицы, должна быть требованием для будущих исследований.

Заключение

Хотя машинное обучение может значительно повысить эффективность и автономию базы данных, все еще существуют проблемы с коммерчески жизнеспособностью этих гибридных систем. Применение машинного обучения для улучшения проектирования и эксплуатации баз данных — это новая и быстро развивающаяся область исследований.

Библиография

1. Machine Learning for Data Management: A System View – [online]. [20.12.2022]– Доступен: http://dbgroup.cs.tsinghua.edu.cn/ligl/papers/icde22-tutorial-paper.pdf, http://www.ijecs.in/index.php/ijecs/article/view/4520
2. 2020. Rethink Data: Put More of Your Business Data to Work—From Edge to Cloud. Technical Report. Seagate. 56 pages. [online]. [20.12.2022]– Доступен: https://www.seagate.com/files/www-content/our-story/ rethink-data/files/Rethink_Data_Report_2020.pdf
3. Xindong Wu. 2000. Building Intelligent Learning Database Systems. AI Magazine 21, 3 (Sept. 2000), 61–68. https://doi.org/10.1609/aimag. v21i3.1524 Number: 3.
4. Maryam M. Najafabadi, Flavio Villanustre, Taghi M. Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. 2015. Deep learning applications and challenges in big data analytics. Journal of Big Data 2, 1 (Feb. 2015), 1. https://doi.org/10.1186/s40537-014-0007-7