# ПРИМЕНЕНИЕ МАШИННОГО ОБУЧЕНИЯ ДЛЯ УВЕЛИЧЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ СУБД POSTGRESQL

Белошедов Е. С., Гуринович А. Б.
Кафедра информационных технологий автоматизированных систем,
Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: 2595118@mail.ru, gurinovich@bsuir.by

*Машинное обучение занимается поиском скрытых закономерностей в данных, что позволяет получить исключительные результаты. С этим связан растущий рост интереса к этой теме в ИТ-сообществе. Распознавание речи и отсканированных документов, поисковые машины всё это функционирует на базе машинного обучения. Исследуются способы применения методов машинного обучения для увеличения производительности СУБД.*

## Введение

SQL представляет собой декларативный язык. То есть пользователь указывает только на те операции, которые должны быть проделаны с данными. За выбор способа выполнения этих операций отвечает СУБД. Можно разделить процесс поиска оптимального плана на две части. Во-первых, нужно уметь оценивать стоимость любого плана — количество ресурсов, необходимых для его выполнения. В случае, когда на сервере не выполняются другие задачи и запросы, оцениваемое время выполнения запроса прямо пропорционально количеству потраченных на него ресурсов. Поэтому можно считать, что стоимость плана — это его время выполнения в некоторых условных единицах. Во-вторых, требуется выбрать план с минимальной оценкой стоимости. Легко показать, что число планов растет экспоненциально с увеличением сложности запроса, поэтому нельзя просто перебрать все планы, оценить стоимость каждого и выбрать самый дешевый. Для поиска оптимального плана используются более сложные алгоритмы дискретной оптимизации: динамическое программирование по подмножествам для простых запросов и генетический алгоритм для сложных [1].

## I. Поиск существующего оптимального плана выполнения запроса

В PostgreSQL для плана предсказываются две стоимости: стоимость запуска (start-up cost) и общая стоимость (total cost). Стоимость запуска показывает, сколько ресурсов план потратит до того, как выдаст первую запись, а общая стоимость — сколько всего ресурсов потребуется плану для выполнения. Эта задача разделяется на две подзадачи. Сначала для каждой вершины плана предсказывается, сколько кортежей будет отобрано в ней. Затем на основе этой информации оценивается стоимость выполнения каждой вершины, и, соответственно, всего плана. В случаях, когда предположение о независимости условий не выполняется, модель PostgreSQL работает не совсем корректно. Следующие способы позволяют преодолеть эту проблему. Первый способ заключается в построении многомерных гистограмм. Проблема этого способа заключается в том, что с увеличением размерности, многомерная гистограмма требует экспоненциально растущее количество ресурсов для сохранения той же точности. Поэтому приходится ограничиваться гистограммами маленькой размерности (2-8 измерений). Отсюда следует вторая проблема этого метода: нужно каким-то образом понять, для каких пар столбцов имеет смысл строить многомерные гистограммы, а для каких необязательно. Чтобы решить эту проблему, требуется либо хороший администратор, который будет изучать планы ресурсоемких запросов, определять корреляции между столбцами и вручную указывать, какие гистограммы нужно достроить, либо программное средство, которое с помощью статистических тестов попытается найти зависимые друг от друга столбцы. В настоящий момент существуют патчи, позволяющие использовать в PostgreSQL многомерные гистограммы, но в них администратору требуется вручную задавать, для каких столбцов эти многомерные гистограммы должны быть построены [2].

## II. Использование машинного обучения для оценки выборочности

Альтернативный подход — это применение машинного обучения для нахождения совместной выборочности нескольких условий. Как уже говорилось выше, машинное обучение занимается поиском закономерностей в данных. Данные — это набор объектов. В данном случае объектом является совокупность условий в одной вершине плана. По этим условиям и их маргинальным выборочностям требуется предсказать совместную выборочность.Наблюдаемыми признаками вершины плана будут являться маргинальные выборочности всех её условий. Необходимо считать эквивалентными между собой все условия, отличающиеся только в константах. Мож-

но рассматривать данное допущение как типичный прием машинного обучения — hashing trick — примененный для уменьшения размерности пространства. Предполагается, что вся необходимая для предсказания информация о константах условия содержится в его маргинальной выборочности. Можно показать это строго для простых условий вида a < const: здесь по выборочности условия возможно восстановить значение константы, то есть потери информации не происходит (см. рис. 1).

| Selectivity | users.age > const | users.city = const | messages.sender_id = users.id |
|---|---|---|---|
| 0.25 | 0.25 | - | - |
| 0.23 | 0.25 | 0.6 | - |
| 0.3 | 0.5 | 0.6 | - |
| 0.0005 | - | 0.5 | 0.001 |
| ... | ... | ... | ... |

Рис. 1 – Задача машинного обучения

Можно заметить, что, если теперь использовать линейную регрессию, то в качестве частного случая получается текущую модель PostgreSQL.

## III. Методы тестирования

Для тестирования различных подходов был использован бенчмарк TPC-H. В качестве простых регрессоров были использованы следующие методы:

- Гребневая линейная регрессия и стохастический градиентный спуск. Этот метод хорош тем, что позволяет использовать динамическое обучение (online learning), поэтому не требует хранить никаких наблюдаемых объектов.
- Множество гребневых линейных регрессий и стохастический градиентный спуск.
- Множество гребневых линейных регрессий и аналитическое решение методом Гаусса.

Ответы регрессора являются входными значениями для оптимизатора, который ищет оптимальный план. Наблюдаемые объекты (исполняемые планы), являются выходными значениями оптимизатора. Поэтому наблюдаемые объекты зависят от ответов регрессора. Такие системы с обратной связью намного сложнее для изучения, чем системы, в которых регрессор не влияет на окружающую среду. Именно в этих терминах аналитическое решение методом Гаусса является нестабильным — оно быстро обучается, но предлагает более рискованные решения, поэтому в целом система работает хуже [3].

## IV. Использование машинного обучения

После детального изучения линейной модели было обнаружено, что она недостаточно полно описывает данные. Поэтому наилучшие результаты из опробованных нами методов показал kNN. Существенный минус этого метода заключается в необходимости сохранения в памяти всех объектов с последующей организацией быстрого поиска по ним. Существенно улучшить эту ситуацию можно, используя алгоритм отбора объектов. Идея наивного алгоритма отбора объектов: если предсказание на объекте достаточно хорошее, то запоминать этот объект не нужно. Следующим этапом является тестирование системы для алгоритма kNN (см. рис. 2).

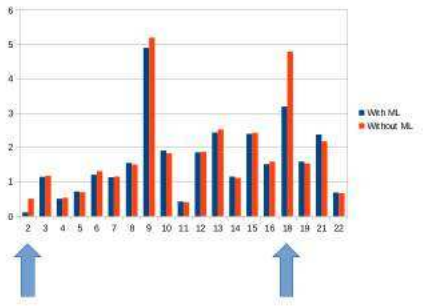


Рис. 2 – Результаты машинного обучения

Можно заметить, что предложенный подход в самом деле ускоряет время работы СУБД.

## V. Заключение

Текущий алгоритм гарантирует, что в тех планах, к которым сходится алгоритм, предсказания выборочности будут правильными. Однако это не гарантирует глобальной оптимальности выбранных планов. Поиск глобально оптимальных планов или хотя бы лучшего локального оптимума — это отдельная задача. С внедрением машинного обучения возможно прервать выполнение плана, в котором были допущены серьезные ошибки в предсказании выборочности, учесть полученную информацию и выбрать новый лучший план для выполнения. В большинстве случаев новый план будет существенно отличаться от предыдущего. В процессе работы СУБД меняются данные и типичные запросы. Следовательно, данные, полученные в прошлом, могут быть уже неактуальными.

## VI. Список литературы

1. Mangrum, J. M. The evaluation and management of bradycardia / J. M. Mangrum, J. P. DiMarco // N. Engl. J. Med. – 2000. – Vol. 342, № 10. – P. 703–709.
2. Neil Conway. Доклады про внутреннее устройство PostgreSQL.
3. Bruce Momjian. Explaining the Postgres Query Optimizer.