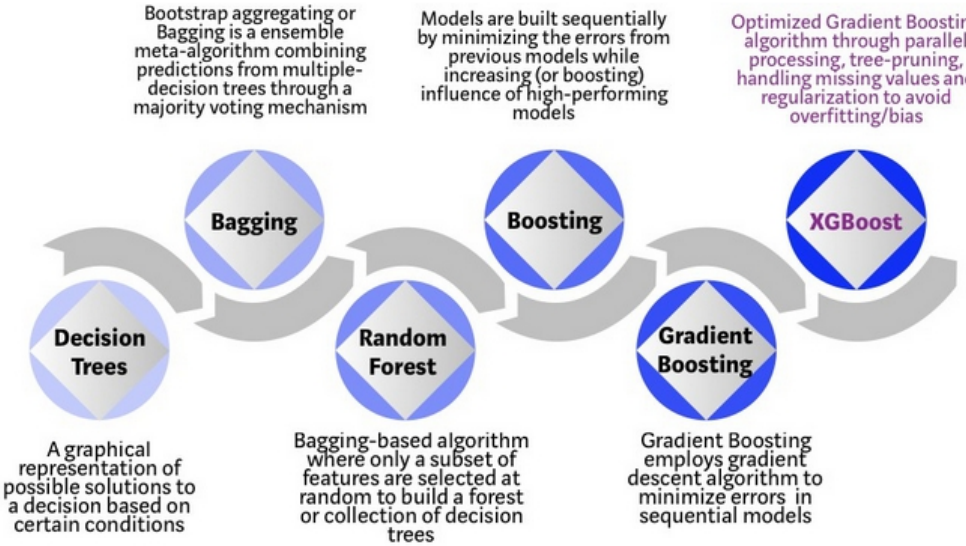
Что такое XGBoost?

XGBoost — это ансамблевый алгоритм машинного обучения на основе дерева решений, в котором используется инфраструктура повышения градиента. В задачах прогнозирования, связанных с неструктурированными данными (изображениями, текстом и т. д.), искусственные нейронные сети, как правило, превосходят все другие алгоритмы или платформы. Однако, когда речь идет о структурированных/табличных данных малого и среднего размера, алгоритмы на основе дерева решений в настоящее время считаются лучшими в своем классе. На приведенной ниже диаграмме показана эволюция древовидных алгоритмов с течением времени.



Evolution of XGBoost Algorithm from Decision Trees

Как создать интуицию для XGBoost?

Деревья решений в их простейшей форме представляют собой легко визуализируемые и довольно интерпретируемые алгоритмы, но построение интуитивного представления для следующего поколения древовидных алгоритмов может быть немного сложным. См. ниже простую аналогию, чтобы лучше понять эволюцию древовидных алгоритмов.

Представьте, что вы менеджер по найму и проводите собеседование с несколькими кандидатами с отличной квалификацией. Каждый шаг эволюции древовидных алгоритмов можно рассматривать как вариант процесса интервью.

1. Decision tree: у каждого менеджера по найму есть набор критериев, таких как уровень образования, количество лет опыта, результаты собеседования. Дерево решений аналогично тому, как менеджер по найму проводит собеседование с кандидатами на основе своих собственных критериев.
2. Bagging: Теперь представьте, что вместо одного интервьюера теперь есть панель интервью, где каждый интервьюер имеет право голоса. Бэггинг или бутстрап-агрегирование включает в себя объединение входных данных всех интервьюеров для принятия окончательного решения посредством демократического процесса голосования.
3. Random forest: это алгоритм на основе пакетов с ключевым отличием, заключающимся в том, что случайным образом выбирается только подмножество функций. Другими словами, каждый интервьюер будет проверять интервьюируемого только по определенным случайно выбранным квалификациям (например, техническое собеседование для проверки навыков программирования и поведенческое интервью для оценки нетехнических навыков).
4. Boosting: это альтернативный подход, при котором каждый интервьюер изменяет критерии оценки на основе отзывов предыдущего интервьюера. Это «повышает» эффективность процесса собеседования за счет развертывания более динамичного процесса оценки.

5. Gradient Boosting: особый случай повышения, при котором ошибки минимизируются алгоритмом градиентного спуска, например. фирмы, занимающиеся стратегическим консалтингом, используют кейс-интервью для отсеивания менее квалифицированных кандидатов.
6. XGBoost: Думайте о XGBoost как о повышении градиента на «стероидах» (ну, не зря это называется «Экстремальное повышение градиента»!). Это идеальное сочетание методов оптимизации программного и аппаратного обеспечения для получения превосходных результатов с использованием меньшего количества вычислительных ресурсов в кратчайшие сроки.

Почему XGBoost работает так хорошо?

XGBoost и Gradient Boosting Machines (GBM) — это методы ансамблевого дерева, в которых применяется принцип повышения слабых учеников (обычно CART) с использованием архитектуры градиентного спуска. Однако XGBoost улучшает базовую структуру GBM за счет системной оптимизации и усовершенствований алгоритмов.

How XGBoost optimizes standard GBM algorithm

Оптимизация системы:

1. **Распараллеливание**: XGBoost приближается к процессу последовательного построения дерева, используя параллельную реализацию. Это возможно из-за взаимозаменяемости циклов, используемых для построения базовых учеников; внешний цикл, который перечисляет конечные узлы дерева, и второй внутренний цикл, который вычисляет признаки. Такое вложение циклов ограничивает распараллеливание, поскольку без завершения внутреннего цикла (более требовательного к вычислительным ресурсам) нельзя запустить внешний цикл. Поэтому для улучшения времени выполнения порядок циклов меняется местами, используя инициализацию через глобальное сканирование всех экземпляров и сортировку с использованием параллельных потоков. Этот переключатель улучшает алгоритмическую производительность, компенсируя любые накладные расходы на распараллеливание в вычислениях.
2. **Обрезка дерева**: Критерий остановки для разделения дерева в рамках GBM является жадным по своей природе и зависит от отрицательного критерия потерь в точке разделения. XGBoost сначала использует параметр «max_depth», как указано вместо критерия, и начинает обрезку деревьев в обратном порядке. Такой подход «сначала в глубину» значительно повышает производительность вычислений.
3. **Аппаратная оптимизация**: этот алгоритм был разработан для эффективного использования аппаратных ресурсов. Это достигается за счет осведомленности о кеше путем выделения внутренних буферов в каждом потоке для хранения статистики градиента. Дополнительные усовершенствования, такие как «внеъядерные» вычисления, оптимизируют доступное дисковое пространство при обработке больших фреймов данных, которые не помещаются в память.

Алгоритмические улучшения:

1. **Регуляризация**: он наказывает более сложные модели с помощью регуляризации LASSO (L1) и Ridge (L2), чтобы предотвратить переобучение.
2. **Осведомленность о разреженности**: XGBoost естественным образом допускает разреженные функции для входных данных, автоматически «изучая» лучшее пропущенное значение в зависимости от потерь при обучении и более эффективно обрабатывает различные типы шаблонов разреженности в данных.
3. **Взвешенный квантильный эскиз**: XGBoost использует алгоритм распределенного взвешенного квантильного эскиза для эффективного поиска оптимальных точек разделения среди взвешенных наборов данных.
4. **Перекрестная проверка**. Алгоритм поставляется со встроенным методом перекрестной проверки на каждой итерации, что избавляет от необходимости явно программировать этот поиск и указывать точное количество повышающих итераций, необходимых для одного запуска.

Так должны ли мы все время использовать только XGBoost?

Когда дело доходит до машинного обучения (или даже жизни в этом отношении), бесплатных обедов не бывает. Как специалисты по данным, мы должны протестировать все возможные алгоритмы на имеющихся данных, чтобы определить алгоритм-лидер. Кроме того, недостаточно выбрать правильный алгоритм. Мы также должны выбрать правильную конфигурацию алгоритма для набора данных, настроив гиперпараметры. Кроме того, есть несколько других соображений для выбора выигрышного алгоритма, таких как вычислительная сложность, объяснимость и простота реализации. Это как раз тот момент, когда машинное обучение начинает уходить от науки к искусству, но, честно говоря, именно здесь и происходит волшебство!