Теоретические Основы Аналитики Больших Данных и Алгоритмов Вычислений Реального Времени

Цели курса

Основной целью данного курса является предоставить студентам уникальную возможность приобрести концептуальную основу и математические инструменты, применимые к аналитике больших данных и вычислений в режиме реального времени. Курс дает краткий обзор основных фаз работы с большими данными, таких как извлечение, унификации, обновления и объединение информации и специфических особенностей обработки данных, которая должны быть в высшей степени параллельной и распределенной. Имея в виду эту специфические особенности, мы затем более подробно изучим ряд математических инструментов для анализа больших данных, таких как регрессионный анализ, линейное оценивание, проблемы калибровки, обработку в реальном масштабе времени входящего (потенциально бесконечного) потока данных. Мы увидим, как эти подходы могут быть преобразованы, чтобы соответствовать требованиям больших данных. Мы также обсудим, почему большинство широко используемых алгоритмических языков не вполне подходит для решения таких проблем и наметим альтернативные подходы.

Идеи курса

В рамках традиционных подходов к обработке информации мы должны собрать все данные в одном массиве и применить алгоритм обработки к нему. Если исходные данные распределены на множестве сайтов и их общий объем велик, это сразу приводит к определенным техническим проблемам:

- Накопление всех исходных данные в одном месте потребует чрезмерных ресурсов для их хранения.
- Применение традиционных алгоритмов к огромным массивам данных потребует чрезмерно много оперативной памяти, вычислительной мощности и времени.
- Идея обработки всех данных сразу не раскрывает (и на самом деле затрудняет) возможности для параллельных и распределенных вычислений.

В курсе показано, что вместо того, чтобы собирать вместе все необработанные данные и обрабатывать все сразу, мы можем естественно разделить весь процесс на простые очень независимые части. В частности:

- Извлечение определенной «достаточной» информации из каждого экземпляра исходных данных и представление ее в удобной «канонической» форме.
- Объединение частей канонической информации.

- Обновление накопленной канонической информации, по мере поступления новых данных.
- Вычисление окончательного результата из накопленной информации в канонической форме.

Оказывается, что информация в канонической форме часто имеет фиксированный размер, который не зависит от количества исходных данных, используемых для ее производства. В результате все отдельные стадии извлечения канонической информации из исходных данных, ее объединение и получение конечного результата не требуют чрезмерного объема памяти или вычислительной мощности. После того, как две части канонической информации объединяются в одну, исходные части могут быть уничтожены. Извлечение и объединение частей канонической информации могут выполняться на разных компьютерах без необходимости синхронизации. Это предоставляет широкий спектр естественных вариантов для массивных параллельных распределенных вычислений.

Некоторые темы курса

Обучение в режиме реального времени.

Пример: Линейное оценивание на основе калибровки.

Алгоритм оценивания, который используется для обработки потока данных в режиме реального времени постоянно сам по себе обновляется на основании другого потока «калибровочных» данных.

Проблемы:

- Объем калибровочных данных может быть огромным и быстро растущим. Как следствие, сбор и хранение всех данных калибровки может потребовать чрезмерных ресурсов.
- Кроме того, вычисление обновленной версии алгоритма оценивания, на основе этих данных потребует большого (и даже постоянно растущего) количество времени.

Решения:

- При поступлении калибровочного измерения – не сохранять его, а использовать для обновления некоторой специальной информации (фиксированного размера), которая является достаточной для построения алгоритма оценивания. Такая «упакованная» информация является аналогом достаточной статистики.
- Не вычислить алгоритм регрессии каждый раз «с нуля», а обновлять его, используя только накопленную каноническую информацию.

В курсе: Рассматривается линейная задача калибровки, обсуждается ее вычислительная сложность при используя «традиционного» подхода и подхода, использующего «каноническую информацию».

Обработка сигналов в реальном времени.

Стандартные подходы потребуется записать полный сигнал (например, временной ряд или изображение), а затем применить к нему алгоритм обработки. Хотя такой подход может обеспечить наиболее точную обработку, он не может быть сделано в режиме реального времени, поскольку потребуется записи всего сигнала (или достаточно больших его частей).

Однако, если время имеет решающее значение, обработка может выполняться по мере поступления данных, с помощью «скользящего окна». Такой подход не потребует сохранения исходного сигнала и позволит выбрать баланс между качеством обработки, ее сложностью и временем задержки.

В курсе: Рассматривается проблема обработки для (потенциально) бесконечного массива данных и стоится оптимальный алгоритм с учетом вычислительных ограничений и задержек.

Разложение сложной задачи на простые.

Нередко большая проблема может быть разложена в набор аналогичных, но более мелких проблем, которые могут решаться параллельно. Для разработки соответствующих алгоритмов, который смог бы использовать подобные возможности разложения, необходимо либо явно определить, как вычислить соответствующие части параллельно (что обычно весьма трудоемко и сильно зависит от используемой архитектуры), либо использовать соответствующий язык программирования достаточно высокого уровня, который изначально ориентирован на параллелизм и распределенные вычисления (см. следующий раздел).

В курсе: Сравниваются «стандартный» и «параллельных» (функциональный) подходы на примерах таких базовых задач, как умножение векторов и матриц.

«Параллельное» мышление.

Современное алгоритмическое мышление сильно связано с используемыми языками программирования. В то же время почти все алгоритмические языки, которые используются для прикладных задач, явно определяют порядок выполнения операций. В результате даже если компьютер может обрабатывать несколько проблем одновременно, классический алгоритмический язык не позволяет использовать параллелизм неявно. Чтобы преодолеть, что программист должен четко определять, какие части кода могут выполняться параллельно.

Радикальный подход к этой проблеме состоит в переходе к алгоритмическим языкам, которые не указывают порядок операций (таких, как, например, функциональные языки).

В курсе: Краткое введение в функциональное программирование. Реализация матричной алгебры и алгоритмов обработки информации в стиле функционального программирования.