Игорь Кураленок ikuralenok@gmail.com

1. Выиграть соревнование по сжатию логов с помощью DictExpansion

Есть библиотека создания словарей сжатия по потоку записей, сгенерированных единым случайным процессом. Она даже работает, к ней нужно добавить нерегулярную часть (когда в записи присутствует нерегулярная, повторяющаяся часть, например id пользователя), доказать теоретическую оптимальность полученного алгоритма в предельном случае, провести сравнение сжатия на данных http://mattmahoney.net/dc/text.html, https://quixdb.github.io/squash-benchmark/

2. **Сделать term sharded поиск по wikipedia на оптимальных n-gramm'ах**

Существующие поисковые системы построены по принципу шардирования по данным. Следуя этому принципу надо разложить порции инвертированного списка на сервера постранично, вычислить ранжирующую функцию, передать результаты для выбора лучших вверх по иерархии (в данном случае к L1.5 или L2). Такой подход прост в реализации ранжирующей функции, но неэффективен по распределению нагрузки: на каждый запрос, не попавший в кеш мы запрашиваем все машинки. Существует альтернативный подход, так называемый term sharded, при котором локальным является индекс по одному терму, а вычисление ранжирующей функции происходит на L2. Было несколько попыток реализации данного подхода (например Microsoft Maguro/Tiger), которые неудались из-за недостаточного учета связей между разными терминами при фильтрации. Есть идея как эту проблему излечить с помощью умного алгоритма выделения n-gramm. Необходимо взять данные википедии, разделить их на 100 нодов, реализовать шардирование по данным, побомбить запросами из логов Яндекс (данные РОМИП). Замерить отставание от прямого поиска. Реализовать term sharded и это отставание уменьшить.

3. Жадные регионы vs деревья

Деревья решений — удобный и общепринятый способ формулирования решающей функции. Однако, структура дерева для некоторых подходов излише сложна, в частности, она предполагает биение всего пространства жесткими границами и для введения мягких границ приходится вставать на уши. Есть способ подбирать не целое дерево а регион, задаваемый предикатом, основанном на свойствах точки, и константы или распределения, ассоциированный с подходящими по условию точками. При этом способ не проигрывает существующим деревьям при объединении в ансамбль. Хочется описать существующий метод, показать, что он работает на реальных данных, научиться использовать не только конъюнкции при объединении условий, но полный КНФ. Показать, что данная конструкция лучше приспособлена к использованию категориальных признаков.

4. Оптимальное построение деревьев решений: log (n + 1) регуляризация, оценка дисперсии сверху, учет Stein paradox при вычислении среднего, линейные функции в листьях дерева, априорная дискретизация признаков

За последние несколько лет в деревьестроении были найдены приемы, позволяющие как существенно превзойти классические деревья по качеству модели, так и пересмотреть само отношение к деревьям решений. Эти наработки хочется как исследовать с теоретической точки зрения, так и показать их эффективность на практических данных открытых коллекций. Берем с десяток коллекций, прогоняем на всех, видим профит, публикуем статьи.

5. Категориальные признаки в построении деревьев: online features, другие свертки

В последнее время все большее внимание исследователей привлекает использование т.н. сырых данных в обучении. Одним из примеров подобных данных являются категориальные признаки, отличающиеся от своих обычных собратьев не только дискретностью, но и отсутствием отношения порядка. Есть несколько приемов, позволяющих использовать подобные данные в классическом обучении. Хочется их исследовать и обосновать.

6. Распределение как решающая функция: гаммы в листьях (Василий Ершов)

Как известно, деревья решений являются кусочно-постоянной функцией на пространстве факторов. Однако, при их построении возникает понимание не только о среднем значении внутри того или иного листа, но и о характере этого распределения. В последнее время деревья редко используют поодиночке и часто в решающей функции принимают участие тысячи и более деревьев (в Я есть модельки по 100к, например). В этих условиях, «неточности» информации о распределении сглаживаются и могут породить новый мощный класс решающих функций. Хочется попробовать моделировать распределение в листе разложением в гамма и посмотреть, что же получится на открытых данных.

7. Факторизация матрицы dL/ds_{ij} для построения легкой решающей функции мультиклассификатора (Руслан Соловьев)

Задача мультиклассификации обычно связана с выбором из трех альтернатив: построение one vs. all классификаторов, построение единого мультиклассификатора по multi-logit и построение системы бинарных классификаторов, связанных модельной матрицей (см. статью multiclass to binary). Мы предлагаем еще одну альтернативу, совмещающую все три подхода. Кажется она работает, более того, даже данные уже посчитаны, осталось только статью написать.

8. Построение оптимального решающего конечного автомата с помощью нейронных сетей (последовательности)

Обучение на сырых данных ныне популярная тема. Одним из наиболее важных типов данных являются последовательности (от биологии до банковской сферы). Предлагается устроить бустинг из слабых решающих функций, сформулированных в терминах DFA/NFA. Алгоритм подбора есть, нужно перевести его на GPU, научиться делать boosting (видимо на AdaBoost) и оценить качество по сравнению с текущими подходами.

9. Продуктовое моделирование задач BigData и динамическое построение оптимального плана исполнения

На сегодняшний момент есть 2.5 подхода к большим данным: каменный век, в котором прибывает научная братия (Hadoop, Spark, etc. из коробки), медный век (ms aether, Я.Нирвана, аналоги от amazon, google) и бронзовый век, который был в Я до последнего времени. Характерная черта каменного века в том, что если данные удалось обработать — это чудо, а чудеса нельзя делать пачками, это всегда ручная работа. В медном веке все чуть лучше, данные уже удается обработать, но способов получить один и тот же результат тыщщи, и в каждом свои ошибки и особенности, в итоге все пишут 100001, свой, метод получить из данных А табличку Б, и энтропия нарастает. В бронзовом веке система целенаправленно борется с энтропией путем стандартизации процессов, выведением отношений между ними и поддержкой набора «кошерных» популярных операций. Хочется сделать попытку прорыва в век железный. Для этого можно попробовать переориентироваться с мышления процессами на мышление в терминах типов данных. После этого поворота можно воспользоваться всеми прелестями теории компиляторов, начиная с языков описания, заканчивая уже существующими алгоритмами вывода. В этом месте есть небольшие наработки, которые сейчас используются внутри Я, но особо не прогрессируют там, которые можно поразвивать в opensource.