Финальный отчёт о проделанной работе

Павел Захаров

3 мая 2022 г.

Содержание

1 Данные

Прежде чем начнём, давайте взглянем на данные. Во избежание наведённых эффектов перейдём от абсолютных цен к приращениям. Все тренды останутся на месте, а у нас отпадёт необходимость нормировать данные. Следующий вопрос: что делать с пропусками? Если в данных идут два NaN подряд, то можно заменить их оба на некоторое константное значение, например 1. Если мы находимся на стыке NaN и числа, то можно так же выставить 1. Проанализируем на примере разных типов данных:

- Мало NaN-ов. В таком случае у нас мало и стыков, и непрерывных участков с NaN-ами так что значения на них не играют особой роли.

- Мало не NaN-ов. В таком случае у нас всё ещё мало стыков, поэтому давайте посмотрим на идущие подряд NaN-ы.

  Мы не хотим, чтобы на пустом (буквально) месте возникали какие-либо тренды, так что заполнение их одним и тем же значением – самый корректный вариант.

  Насчёт того, какое брать значение. Если ставим один, то подобные акции приравниваются к тем, цены которых близки к постоянным. Если любым другим, то возникают тренды (т.к. на данных промежутках данные акции либо постоянно растут в цене, либо постоянно падают). Так что либо данные акции пойдут в «мусорный» кластер, либо в один из существущих. Так как мусорные кластеры я буду раскидывать по другим, то особой разницы нет.

2 Описание алгоритма

Потоковые алгоритмы кластеризации слишком заточены под конкретную форму данных, так что будем конструировать свой. Будем считать, что данные ведут себя разумно, то есть общие тренды у двух близких акций сохраняются на протяжении всего временного периода (если нет, то мы перестаём считать их близкими). Тогда давайте задаим

какое-нибудь априорное разбиение, после чего будем его понемного модифицировть по ходу дела.

3 Фаза 1

Априорное разбиение будем делать с помощью k-means. Одна из его особенностей в том, что он в сути своей использует евклидову метрику. Я долго пытался встроить в него произвольную метрику, но данный подход неэффективен по веремени, так как он требует решить произвольную задачу оптимизации, что на практике работает сколь угодно долго (с евклидовой метрикой задача оптимизации превращается в least squares).

По итогу я наткнулся на статью, где описывается использование Dynamic Time Wrapping вместе с k-means. Это было то, что надо. Давайте поговорим про особенности и метрики.

Качество разбиения считать смысла нет (на данном этапе), так как особенность k-means в том, что он оптимизирует внутре/междукластерные расстояния (а качество разбиения имеет смысл мерять именно по выбранной метрике).

Так что давайте поговорим про такой аспект, как независимость от стартовой точки. Для измерения близости двух кластеризаций буду использовать rand index. Для двух разных стартовых инициализаций k-means он равен примерно 0.9 (что, кстати, весьма неплохо, так как k-means славится своей нестабильностью. Выходит, наши данные достаточно хорошо разделимы).

При сдвиге временного окна на какой-то значимый промежуток получаем rand index между исходным и новым разбиением равен примерно 0.855 (при нескольких замерах), так что можно смело говорить, что априорная кластеризация практически не зависит от выбора начального разбиения.

Также было решено на каждом шаге выбрасывать вырожденные кластеры (с < 10) элементов, дабы не засорять наше разбиение. Выбрасывание происходит следующим образом: для каждого из элементов из выбрасываемых кластеров я определяю наиближайший центроид среди прочих кластеров, после чего к этому кластеру его и присоединяю.

Выбор числа кластеров. Я решил, что буду ориентироваться на GICS в контексте разбиения, то есть я хочу получить схожее число кластеров. После начального разбиения оно будет более-менее стабильное, так что давайте подберём число k так, чтобы после выбрасывания осталось около 12 кластеров. Взял 18 (запускал и смотрел на количество выброшенных).

4 Фаза 2

Теперь мы хотим кластеризовать данные, передаваемые в виде потока. Я выбрал следующий вариант:

- Разбиваю данные на промежутки (скажем, по одному месяцу длиной);
- Беру самый ранний из нерассмотренных промежутков. На нём будем пересчитывать наше текущее разбиение.
- Что мы пересчитываем:
  - Удаляем из рассмотрения компании, которые ушли с рынка. Здесь никаких особых действий не требуется.
  - Распределяем компании, которые зашли на рынок. Здесь тоже всё несложно. Так как разбиение на кластеры у нас уже имеется, то достаточно каждую из новых компаний закинуть в наиближайший кластер (вычисляем все центроиды и берём ближайший).

– Модифицируем имеющиеся кластеры. Тут будет два вида действий: разбивание некоторых кластеров на два меньших и раскидывание некоторых кластеров по другим.

Длина промежутка (30 дней) взята из соображений устойчивости разбиения. Т.е. если мы хотим каждые 30 дней изменять кластеризацию – берём такую длину промежутка.

Давайте обсудим последний пункт: модификацию имеющихся кластеров (ибо именно там возникают разного рода параметры).

Для оценки качества подобных преобразований я использовал Silhouette Coefficient. При «ликвидации» маленьких кластеров данный показатель меняется незначительно.

При дроблении наибольшего кластера у нас меняется только внутрекластерное расстояние на данных точках. В подавляющем большинстве случаев разбиение ведёт к увеличению данной метрики, так что качества данное телодвижение не ухудшит.

Теперь давайте зафиксируем саму процедуру. При каких обстоятельствах мы хотим дробить кластер? Наверное, если он впринципе хорошо делится. Давайте это проверим: на каждом кластере запустим k-means и будем оценивать качества разбиения данного кластера как отношение размера меньшей доли к большей. Чем оно ближе к единице – тем данный кластер лучше разделим.

Давайте отталкиваться от этого. Будем делить каждый кластер на два именно с такой вероятностью (на каждом шаге). Закодив такую процедуру, мы видим, что устойчивость пострадала несильно.

Но с другой стороны если в кластере, положим, 12 элементов, то с немаленькой вероятностью мы разделим его на два кластера по 6 элементов. Вопрос: устраивает ли это нас? Нет, так как устойчивость сильно подрывается, а само разбиение на кластеры принципиально лучше не станет.

Теперь про «ликвидацию» кластеров. Уничтожать большие кластеры мы определённо не хотим, ибо они представляют из себя какие-то реальные группы компаний, а не выбросы. От маленьких кластеров мы хотим избавляться в любом случае, так что давайте при маленьком размере кластера (скажем, < 10) будем его «ликвидировать».

Можно «ликвидировать» и другие маленькие кластеры (выбранные по какому-то принципу), но на Silhouette Coefficient это влияет слабо, а по устойчивости бьёт.

На этом описание процедуры закончено.