ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

Отчет о программном проекте

на тему разработка системы предсказания успешного завершения учебной дисциплины
(промежуточный, этап 1)

Выполнил:
студент группы БПМИ186 _____ Шелковников Вячеслав Михайлович
Подпись И.О. Фамилия
7.02.2020
Дата

Принял:
руководитель проекта Андрей Андреевич Тарасов
Имя, Отчество, Фамилия
младший научный сотрудник
Должность
МНУЛ ИССА ФКН НИУ ВШЭ
Место работы

Дата 07.02.2020 10 _____
Оценка (по 10-тибалльной шкале) Подпись

Москва 2020

Содержание

Задачи этапа

Изучение и сравнение алгоритмов кластеризации

Алгоритм Clara

Описание

Алгоритм Clara - алгоритм кластеризации, основой которого служит другой алгоритм кластеризации PAM. Большим недостатком PAM является его неэффективность на большом объеме данных, Clara решает эту проблему.

Используемые термины

- N - Объем данных
- K - кол-во кластеров
- Медоид - объект, принадлежащий кластеру различие которого с другими объектами в наборе данных или кластере минимально.
- Data set - данные
- Subdata set - подмножество данных
- S - размер Subdata-set
- PAM - алгоритм кластеризации
- Штраф - евклидовое расстояние между объектами кластера и медоидом

Шаги алгоритма

- Случайным образом сгенерировать subdata set размера S (при этом если это не первая итерация в subdata set надо обязательно положить лучшие K-медиоды на данный момент)
- Использовать алгоритм PAM на полученном subdata set и найти с его помощью K-медоидов
- Соотнести точки всего data set с полученными медоидами
- Посчитать штраф
- Обновить лучшие K-медоиды, если текущие показали лучшие результаты
- Повторять нужное кол-во раз

Алгоритмическая сложность и применение

Алгоритм PAM имеет Асимптотику $O(k\ (n-k)^2)$, то есть квадратичную от N Clara, в свою очередь, снижает ее до линейной от N - $O(k(s-k)^2+nk)$, что при больших N и удачном выборе S, намного обгоняет PAM при этом не сильно снижая качество кластеризации. Clara используется при большом объеме данных, когда k-medoid и k-means алгоритмы не справляются по времени(миллион наблюдений и более). Так же стоит понимать, что как и k-medoid алгоритмы Clara подходит для небольших значений K.

FP Growth

Описание

FP Growth - один из самых эффективных алгоритмов поиска ассоциативный правил, в отличие от Apriori не проседает по времени на больших данных, так как не требует большое количество раз проходить по базе данных. В основе алгоритма лежит построение дерева транзакций по некоторым правилам.

Используемые термины

- Набор транзакций - параметры объекта базы данных, где каждый параметр - транзакция
- FP - дерево - дерево построенное алгоритмом FPG
- Префикс набора - путь по дереву до выбранной конечной вершины
- Суффикс набора - выбранная конечная вершина
- Корень дерева -старший узел дерева
- Узел дерева - вершина дерева
- Индекс узла - частота появления узла среди транзакций
- Минимальная поддержка - минимальный удовлетворяющий нас индекс узла
- Популярный набор - набор, в котором все транзакции удовлетворяют минимальной поддержки

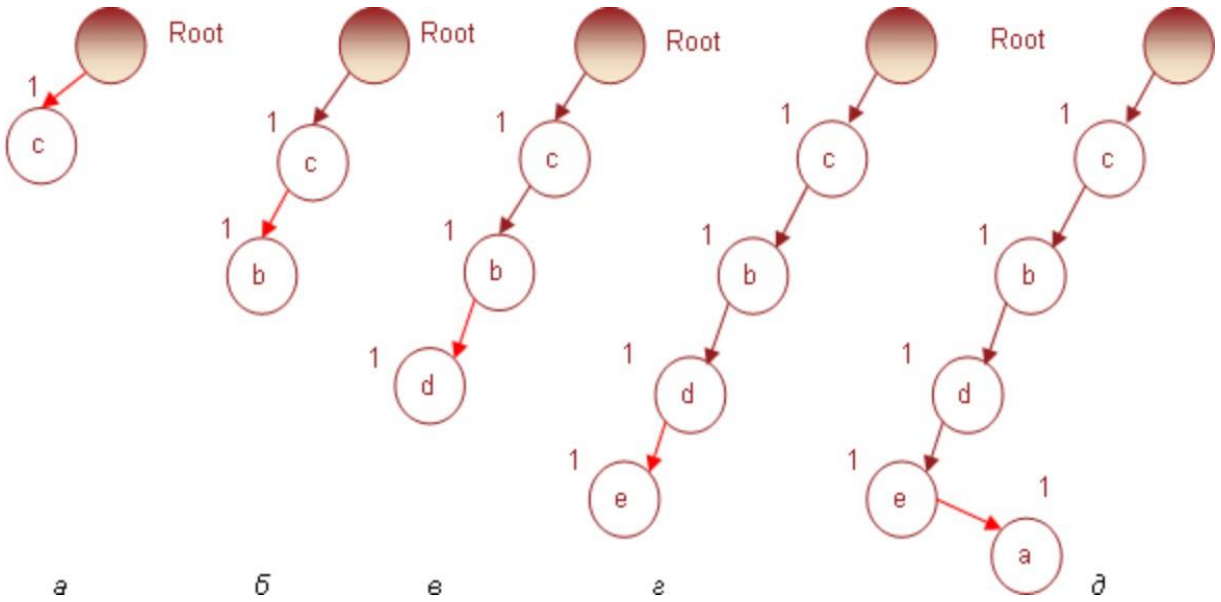
Шаги алгоритма

- Построение FP дерева
- Построение условного FP дерева по выбранной транзакции
- Построение популярных наборов

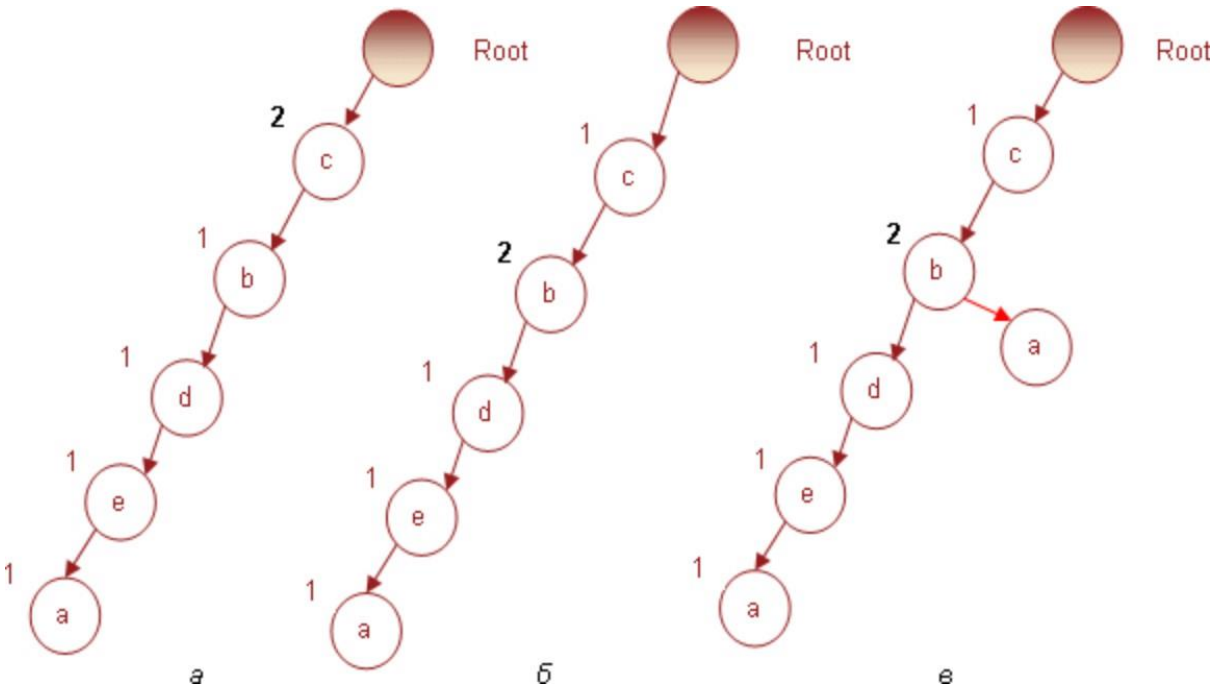
Построение FP дерева

- Посчитать частоты каждой транзакции среди всей БД
- Отсортировать транзакции в каждом наборе по убыванию частот
- Далее для каждого набора делаем следующий алгоритм действий
- Начинаем присоединять транзакции по следующему правилу : если узел транзакция уже существует, увеличиваем индекс узла на 1, иначе создаем новый узел с транзакцией, переходим к следующей транзакции, переходя на 1 уровень ниже

Пример построение дерева по набору c b d e a



Пример добавления набора c b a



Построение условного FP дерева и формирование популярных наборов

Чтобы получить популярные наборы для какой-то транзакции, делаем следующий алгоритм

- Формируем новые наборы транзакций, в каждый набор входит путь от корня дерева до узла с выбранной нашей транзакцией. В итоге получаем множество всех путей до искомого узла
- По данному множеству построим FP дерево по выше приведенному алгоритму
- Пересчитываем индексы всех узлов учитывая их повторы по всему дереву (суммируем все индексы каждого узла)
- Добавляем к каждому листу узел с выбранной нами заранее транзакцией
- Начиная с корня идем до выбранной нашей вершины, формируя наборы транзакций, при этом добавляя только те вершины, которые проходят минимальную поддержку (ее мы предварительно выбрали)

Реализация клиент-серверной архитектуры

В проекте была использовалась клиент-серверная архитектура Json API, позволяющая посылать запросы типа “post” и “get” на сервер, чтобы создать или получить данные об эксперименте. Более того, система удовлетворяет правилам REST.

Rest

REST (от англ. Representational State Transfer — «передача состояния представления») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы.

Набор ограничений

- **Модель клиент-сервер**. Отделение потребности интерфейса клиента от потребностей сервера, хранящего данные, повышает переносимость кода клиентского интерфейса на другие платформы, а упрощение серверной части улучшает масштабируемость.
- **Отсутствие состояния**. Протокол взаимодействия между клиентом и сервером требует соблюдения следующего условия: в период между запросами клиента никакая информация о состоянии клиента на сервере не хранится
- **Кэширование**
- **Единообразие интерфейс**
    1. Идентификация ресурсов
    2. Манипуляция ресурсами через представление
    3. Каждое сообщение содержит достаточно информации, чтобы понять, каким образом его обрабатывать.
    4. Гипермедиа как средство изменения состояния приложения

Использование Flask для реализации

Flask является микрофреймворком для создания вебсайтов на языке Python.

Было реализовано два класса *Experiment* и *ExperimentList.* В каждом классе реализованы два метода *get* и *post*. *Get* возвращает результаты эксперимента и список экспериментов соответственно. *Post* запускают новый эксперимент с заданными параметрами и создает новый эксперимент соответственно

Clara implementation

https://pastebin.com/xKP0aAkJ

Api

https://pastebin.com/uy6k9yrV

Источники информации

Clara

- https://www.datanovia.com/en/lessons/clara-in-r-clustering-large-applications/#clara-concept
- https://ranalytics.github.io/data-mining/101-Partitioning-Algos.html
- https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/CLARA

FP Growth

- https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Frequent_Pattern_Mining/The_FP-Growth_Algorithm
- https://www.geeksforgeeks.org/ml-frequent-pattern-growth-algorithm/
- https://basegroup.ru/community/articles/fpg

Rest

- https://medium.com/@andr.ivas12/rest-%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%8B%D0%BC-%D1%8F%D0%B7%D1%8B%D0%BA%D0%BE%D0%BC-90a0bca0bc78
- https://ru.wikipedia.org/wiki/REST