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 2)

Выполнил:
студент группы БПМИ185 _____ Александра Игоревна Лежанкина
Подпись И.О. Фамилия

_____
Дата

Принял:
руководитель проекта Андрей Андреевич Паринов
Имя, Отчество, Фамилия

_____
Должность

_____
Место работы

Дата _____2020 _____ _____
Оценка (по 10-тибалльной шкале) Подпись

Москва 2020

Содержание

Задачи первого этапа:

Изучение и сравнение алгоритмов кластеризации

Ward hierarchical clustering

Теория

Данный вид кластеризациии является восходящим. Изначально каждый студент – это отдельный кластер, затем объекты сливаются во всё более крупные кластеры: на каждом шаге выбирается пара кластеров с минимальным расстоянием друг от друга и объединяется. При этом данное расстояние хранится для последующего подсчёта оптимального числа кластеров. Объединение происходит до тех пор, пока все студенты не сольются в один кластер.

Структура данных

Для хранения и обработки данных использую структуру Непересекающихся множеств, которая позволяет легко объединять множества (кластеры), хранит информацию в виде списка родителей, где каждый i-й индекс соответствует i-ому студенты. Реализую функцию как класс с методами

- «find» – поиск родителя;
- «union_sets» – объединение множеств;
- «get_item» – возвращени элемента, соответствующего определённому индексу;
- «count_sets» – поиск числа кластеров;
- «count_elems» – поиск числа элементов в кластере;
- «result» – кластеры.

Формула расстояния

Формула для подсчёта расстояния между кластерам U и V. ρ – расстояние в Евклидовой метрике

- **Метод Уорда** (англ. *Ward's method*)

$$\mathrm{R}_{\mathrm{ward}}(U,V) = \frac{|U|\cdot|V|}{|U|+|V|}\rho^2\left(\sum_{u\in U}\frac{u}{|U|},\sum_{v\in V}\frac{v}{|V|}\right)$$

Оптимальное число кластеров

Изначально запускается функция кластеризации с единичным параметром на месте числа желаемых кластеров, при этом минимальное расстояние, найденное на каждом шаге алгоритма, записывается в список.

Далее мы проходим по образованному списку в поисках максимальной разницы между соседними расстояниями. Шаг, на котором будет найдено это расстояние, будет являться переменной I в формуле num = N – i + 1, которая отвечает за число кластеров, где N – число студентов.

Затем функция кластеризации запускается повторно, но уже с параметром num. Итоговое разбиение является ответом.

Входные данные

Хранятся вводимые данные в виде списка списков, где каждому студенту соответствует определённый индекс.
Среди характеристик полезны:

- Оценки за дисциплины;
- Количество часов, затраченных на освоение предмета;
- Идентификаторы посещения различных курсов или другие полезные скиллы (1 при наличии, 0 иначе).

Выходные данные

В итоге мы имеем набор кластеров оптимального размера. Данные выводятся в виде ключа – номера кластера и набора студентов, которые вошли в кластер. Пока в реализации это просто индекс, но в зависимости от ввода это могут быть имена студентов или набор характеристик.

Стандартизация

Так как данные могут иметь различный масштаб: например, оценки колеблются от 1 до 10, а число часов может доходить до сотни и более, будет полезно стандартизировать данные так, что их среднее значение будет равно 0, а отклонение 1. Для этого буде использовать StandartScaler и функцию fit_transform.

Equivalence Classification Algorithm (Eclat)

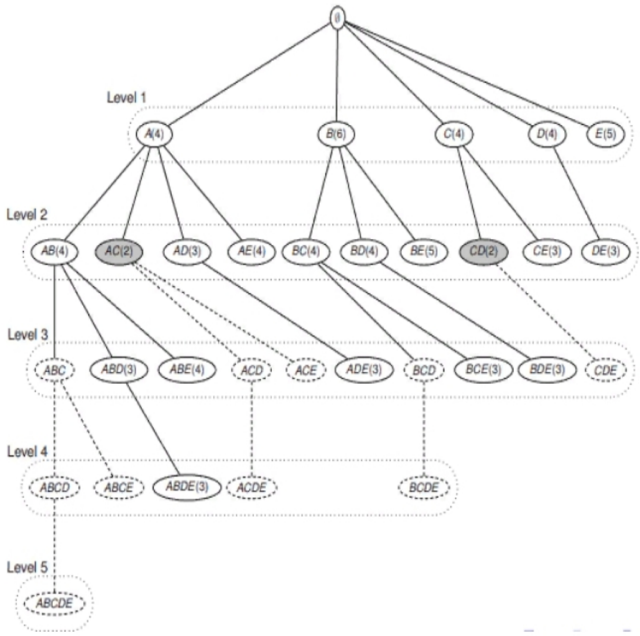
Теория

В отличие от других алгоритмов поиска ассоциативных правил данный использует поиск в глубину. Также в данном алгоритме база данных сканируется только единожды.

Для работы нам понадобится ввести несколько определений:

- Support (Поддержка) – идентификатор того, насколько часто встречается объект среди транзакций. Вычисляется, как число транзакций, содержащих те объекты, чью support мы ищем, делённое на количество транзакций. Измеряется в процентах (*100). Набор характеристик будет представлять интерес, если его Support выше, введённого ранее значения. Этот параметр (Минимальная поддержка) будет зависеть от того, с какой базой данных мы работаем, и сколько в ней студентов. На практике будет понятнее, какой размер минимальной поддержки делает работу эффективнее. Условно можно считать его равным 20%. Представленный тип хранения данных для алгоритма Eclat позволяет быстро рассчитывать поддержку. Суммарное число транзакций нам дано изначально, а число транзакций, в которых содержится наш объект item хранится как множество TID set. Чтобы посчитать число транзакций, в которых встречается несколько объектов, мы просто объединяем их и считаем мощность. Это можно сделать очень просто, если хранить данные в контейнере set.
- Confidence (Достоверность/Вероятность) – идентификатор корректности работы ассоциативного правила. Confidence(i | j) = Support(ij) / Support(i). Также необходимо задать какие-то ограничения максимальной достоверности, иначе правила окажутся слишком очевидными, и алгоритм будет не таким полезным. Мне кажется, можно зафиксировать этот параметр равным 60.

В ходе алгоритма мы поочередно просматриваем все возможные объединения объектов с некоторым ограничением. Сначала берем каждый объект сам по себе и считаем его поддержку, если она меньше заданной минимальной, отбрасываем его. Из оставшихся образовываем пары, считаем поддержку, отбрасываем. Затем тройки, четверки и так до тех пор, пока число рассматриваемых объектов в объединении не превзойдёт максимальное число характеристик для некоторого студента. Будет образовано подобное дерево при помощи поиска в глубину.



Входные данные

{item : TID set}, item – название объекта, TID set – множество номеров транзакций, содержащих данный объект item. В нашем случае объектами являются характеристики, относящиеся к студентам, а транзакциями, соответственно, наборы этих характеристик для каждого студента. Номер транзакции – номер студента в базе данных. Например, пусть исходная база данных содержит несколько студентов и набор характеристик для каждого. Характеристикой может быть оценка за дисциплину, затраченное на её прохождение число часов и т.д. Тогда для успешной работы алгоритма Eclat необходимо преобразовать каждую характеристику в объект item (Например, 10 по матанализу и 8 по программированию это два разных объекта-item). Номером транзакций будет являться номер студента, который имеет данную оценку среди своих характеристик.

Данные удобно будет хранить в виде дерева префиксов, а также отдельно словарь со значениями Support и Confidence для каждого ключа-узла из образованного дерева. Это обеспечит наглядный показатель того, какие отношения будут образовывать правила.

Выходные данные

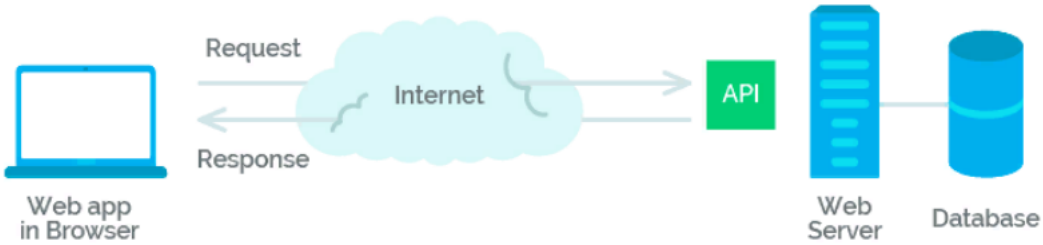
Если выводить Confidence от введённых пользователем собственных характеристик, то это будет вероятность получения желаемой оценки. Иначе можно вывести вероятный результат работы нашего ассоциативного правила для заданных характеристик, тогда студент получит наиболее вероятный результат своей деятельности.

Задачи второго этапа:

Проектирование своего RESTful API.

Проектирование API

По завершении реализации алгоритмов необходимо было разработать собственный веб-API. Это полезно для спецификации структуры запросов и ответов между клиентом и сервером. API позволяет взаимодействовать пользователю с лежащей внутри базой данных благодаря паре уровней между ними – уровень абстракции данных и уровень менеджера ресурсов.



Теория

Для решения задачи проектирования API (application programming interface) использовался довольно стандартный подход REST (representational state transfer) – архитектура при дизайне веб-сервисов и веб-API. Важно, что в ней внутренняя реализация скрыта от клиента. Его принципы:

- клиент-серверная архитектура – существует разделение между клиентом, использующим сервер, и самим сервером. Клиенты взаимодействуют с сервером, обмениваясь представлениями ресурсов;
- любые данные – ресурс (унификация) – программный интерфейс сервера стандартизирован;
- любой ресурс имеет ID – универсальный код ресурса URI, который однозначно идентифицирует этот ресурс;
- ресурсы связаны между собой;
- используются стандартные методы (запросы) протокола HTTP – они должны быть независимы друг от друга и могут задаваться в любом порядке. Для каждого запроса существует свой статус, необходимый для грамотного оформления ответа и предоставления его клиенту;
- кэшируемость;
- сервер не хранит все сессии в памяти – никакие данные клиента не сохраняются на сервере.

  Этот подход позволяет предоставить данные для пользователя в понятном виде.

Основная идея заключается в том, что каждый запрос на сервер преобразует приложение клиента. RESTful API – практическое воплощение архитектурного стиля REST. При разработке RESTful веб-сервиса у нас имеется некоторый список url-адресов, с помощью которых можно с ним взаимодействовать.

Поподробнее о методах HTTP. Зачастую каждый из методов служит для выполнения какого-то действия из концепции программирования CRUD (create, read, update, delete). Всего их существует 7: OPTIONS, GET, HEAD, PUT, POST, DELETE, PATCH. Основными являются только 4:

- GET (read) – получает информацию о ресурсе в удобном для пользователя формате. Возможно применение как для полного списка ресурсов (в нашем случае для всех экспериментом), так и для какого-то конкретного. Не меняет состояние ресурса;
- POST (create) – создаёт новый ресурс;
- PUT (update) – обновляет ресурсов. Изменяет состояние ресурса, но допускается неоднократное повторение;
- DELETE (delete) – удаляет ресурс. Также изменяет состояние.

Для взаимодействия с пользователем в стандарте HTTP существует около 70 статус кодов. Это информация о статусе завершения операции. Нужно заметить, что ошибки из серии – 4xx client error – должны отлавливаться, обрабатываться, но их нельзя возвращать в ответ пользователю. В моей реализации применялось всего 5:

- 200 – OK – успешный запрос;
- 201 – OK – в результате запроса был создан новый ресурс (используется в методе POST);
- 204 – OK – ресурс удалён (в методе DELETE);
- 400 – Bad Request – невалидный запрос;
- 404 – Not found – ресурс не найден (в методах GET, PUT и DELETE).

Практика

При практической реализации использовались: Python3, Flask, Flask-RESTful – расширение для Flask. Помимо Flask также предлагались такие фреймворки как Eve или Django, но выбранный оказался наиболее удобным и обладающим обширной и подробной документацией.

В нашем случае ресурсом является объект JSON, суммарно представляющий из себя базу данных. Это довольно лёгкий и понятный для работы формат. Процесс заключался в обработке JSON файла: на сервер передаётся информация о студентах: идентификатор, год поступления, пол, город; информация об оценках: идентификатор студента, идентификатор оценки, название дисциплины, год получения оценки, дата получения, тип, оценка Важно заметить то, что идентификатор студента и оценки не изменяется даже при изменении самого объекта, он сохраняется постоянным. На выходе пользователь получает идентификаторы студента и эксперимента и кластер, к которому отнесён студент.

Flask – это библиотека Python, представляющая из себя фреймворк для создания веб-приложений. Она идеально подходит для быстрого и качественного проектирования благодаря простому, но легко расширяемому функционалу.

Из расширения flask_restful нам потребуются классы Api и Resourse, а также интерфейс парсинга запросов reqparse.

В моей реализации использовалось 3 HTTP-метода. Все они принимают на вход идентификатор эксперимента и работают с алгоритмом Ward Hierarchical clustering.

Подробнее о методах. Они применялись как для целого списка экспериментов, так и для конкретных с определённым идентификатором. Для списка:

- GET – возвращает список экспериментов;
- POST – добавляет новый эксперимент;
- DELETE – удаляет эксперимент.
  Для отдельного эксперимента:
- GET – возвращает результат эксперимента;
- POST – запускает алгоритм с полученными параметрами.

Использованные материалы

Ward hierarchical clustering

- https://www.researchgate.net/publication/51962445_Ward's_Hierarchical_Clustering_Metho d_Clustering_Criterion_andAgglomerative_Algorithm (Ward's Hierarchical Clustering Method: Clustering Criterion and Agglomerative Algorithm)
- https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
- https://scikit-learn.org/stable/modules/clustering#hierarchical-clustering
- https://stackabuse.com/hierarchical-clustering-with-python-and-scikit-learn/
- https://neerc.ifmo.ru/wiki/index.php

Equivalence Classification Algorithm (Eclat)

- http://oaji.net/articles/2015/1948-1434984559.pdf (International Journal of Computer Techniques -– Volume 2 Issue 3, May – June 2015 FREQUENT ITEMSET MINING USING ECLAT WITH RELATIVE PROFIT AND PRICE)
- https://www.geeksforgeeks.org/ml-eclat-algorithm/
- https://habr.com/ru/company/ods/blog/353502/

Api

- «Пишем API на Python» https://temofeev.ru/info/articles/pishem-api-na-python-s-flask-i-rapidapi/
- Проектирование RESTful API с помощью Python и Flask https://habr.com/ru/post/246699/
- How to build JSON API with Python https://www.freecodecamp.org/news/build-a-simple-json-api-in-python/
- Руководство по проектированию API https://docs.microsoft.com/ru-ru/azure/architecture/best-practices/api-design
- Подходы к проектированию RESTful API https://habr.com/ru/company/dataart/blog/277419/
- Лучшие практики проектирования REST API https://jazzteam.org/ru/technical-articles/restful-services-manual/