КЛАСТЕРИЗАЦИЯ ДАННЫХ МЕТОДОМ УСТОЙЧИВЫХ ГОМОЛОГИЙ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Бубнов Я. В.

Иванов Н. Н. − канд. физ-мат. наук, доцент

Задача кластеризации в анализе данных может быть представлена как разбиение большого конечного множества векторов на подмножества на основании выбора подходящей метрики векторного пространства. В общей постановке топологический анализ данных предоставляет методику анализа данных со слабой чувствительностью к конкретным метрикам, размерности пространства и высокой устойчивостью к зашумленности обрабатываемых данных.

Теория гомологии – это раздел алгебраической топологии, изучающий геометрическую структуру данных. В приложениях теории особенное внимание уделяется вычислительной сложности алгоритмов построения и исследования структуры данных, а именно, алгоритмам с линейной и полиномиальной сложностью.

Классический подход в применении гомологии для кластеризации данных состоит в представлении образов (совокупностей векторов информационных признаков) в виде множества симплексов $\sigma^n \subseteq K\ P$ многомерного полиэдра $P \subset R^N$ (замкнутого *N*-мерного многогранника) [1].

Основанием для разбиения множества образов на кластеры является *n*-мерная гомологическая группа $H_n(P)$ полиэдра P, которая представляет собой факторгруппу $Z_n(P) / B_n(P)$ *n*-мерных циклов по *n*-мерным границам и определяет совокупность топологических инвариантов (такие как дыры и пустоты), которые и дают основание для разбиения пространства образов.

Пространство образов получается путем объединения симплексов в комплексы на основании их близости, т.е. используется некоторая метрика *d.* При этом при построении группы, операция взятия границы симплекса вычисляется путем понижения размерности симплекса, то есть поочередным «отбрасыванием» вершин симплекса v_j [2]:

$$\partial B = \sum_{i=0}^{p} (-1)^i [v_0, v_1, \ldots, \hat{v}_i, \ldots, v_p].$$

Например, отбрасывание вершины треугольника (двумерной фигуры) оставляет в результате две вершины, то есть отрезок (одномерная фигура). Однако, рассмотрение статического топологического пространства, построенного по метрике *d*) не позволяет судить в целом о структуре пространства [3]. Для этого используется фильтрации комплекса *K* [4]:

$$K_0 \subset K_1 \subset K_2 \subset \ldots \subset K,$$

$$K_k = \{\sigma \in K : f(\sigma) \le \varepsilon_k\},$$

где $K_0, K_1, \ldots, K_k$ – симплициальные комплексы, причем объединение (склеивание) двух *n*–мерных симплексов σ_i^n, σ_j^n некоторой гранью e_{ij}^n осуществляется только в том случае, если расстояние между ними не превышает заданной величины ε – это основа кластеризации образов.

Далее вводится последовательность отображений гомологических групп фильтрации путем применения гомологического функтора к каждому элементу фильтрации:

$$0 = H_n^0 \to H_n^1 \to H_n^2 \to \ldots \to H_n^m = H_n(K),$$

причем отображение $f_{i,j} : H_n^i \to H_n^j$ является инъективным на основании вложенности множеств $K_i \subset K_j$. Тогда совокупность интервалов всех размерностей будут образовывать так называемые диаграммы устойчивости [4].

Алгоритм вычисления интервалов устойчивости гомологий базируется на концепции порождающих и поглощающих симплексов фильтрации $K^i = \{\sigma^i \mid 0 \le j \le i\}$. Порождающим называется *n*–симплекс σ_i^n, полученный в результате *i*-ого этапа фильтрации, который будет принадлежать *n*-циклу c_i^n, и поглощающим в противном случае. Порождающие симплексы определяют создание гомологичного класса, а поглощающие симплексы определяют слияние класса в граничную группу и формируют границы интервала устойчивости гомологии.

Для вычисления интервалов гомологической устойчивости используется граничная матрица, где на пересечении *j*-го столбца и *i*-ой строки расположена 1, если (*n-1*)-мерный σ_i^{n-1} симплекс является границей *n*-мерного симплекса σ_j^n, сформированных на *n*-ом этапе фильтрации топологического комплекса и 0 иначе.

Для граничной матрицы определяется операция редукции как построковое суммирование столбцов с одинаковыми значениями индекса крайнего ненулевого элемента, удовлетворяющих следующему условию:

$$j_{i+i_0} = j_i + j_{i0}, \; i < i_0,$$

где i_0 является индексом рассматриваемого столбца. Редуцированная описанным образом результирующая матрица будет содержать в каждом столбце информацию об образовании и слиянии новых гомологических классов.

В качестве примера использования метода устойчивых гомологий, рассмотрим решение задачи кластеризации для 16 кластеров, сформированных 1024 образами с тремя информативными признаками. Исходные данные для эксперимента взяты из проведенного автором эксперимента по выявлению угроз безопасности локальной сети. Фильтрация симплексов выполнена с помощью альфа-комплексов, базирующихся на триангуляции Делоне.

На рисунке 2(а) представлены устойчивости гомологий топологической структуры, образованной нульмерными симплексами рассматриваемого множества образов, и характеризуют компоненты связности образованного пространства.

Из результатов видно, что некоторые топологические пустоты остаются стабильными на достаточно продолжительном интервале фильтрации, что стоит рассматривать как кластеризацию пространства образов. На рисунке 2(б) изображена диаграмма устойчивости для одномерных симплексов.

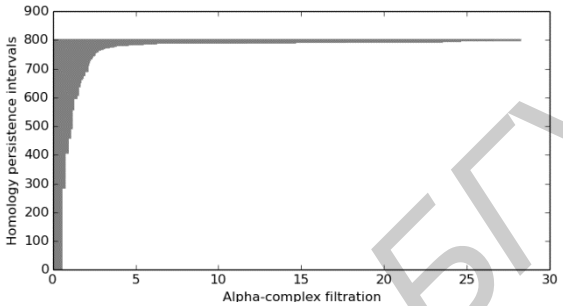

Рис. 1. Диаграммы устойчивости гомологий 0-мерных Alpha-комплексов.

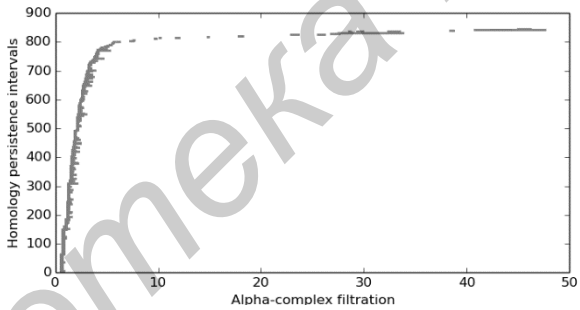

Рис. 2. Диаграммы устойчивости гомологий 1-мерных Alpha-комплексов.

При фильтрации комплексов большей размерности наблюдается низкая продолжительность существования гомологий (рис. 3) в процессе фильтрации, что говорит, в целом о неразделимости образов в пространстве рассматриваемой размерности.

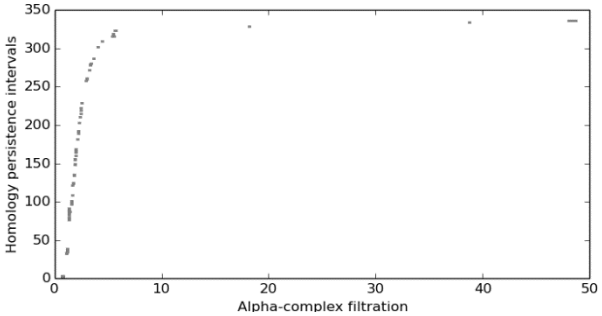

Рис. 3. Диаграмма устойчивости гомологий 2-мерных Alpha-комплексов.

Таким образом, одной из немаловажных проблем использования статистических методов кластеризации является необходимость априорного задания количества кластеров, на которое будет осуществляться разбиение исходной совокупности образов. Благодаря кластеризации методом устойчивых гомологий проблема выбора количества кластеров переносится в другую плоскость, так как результатом данного алгоритма является совокупность всех возможных разбиений, и вопрос уже ставится о необходимой степени различия между ними. Известные алгоритмы вычисления интервалов устойчивости гомологий имеют кубическую сложность [5].

Список использованных источников:

1. Johnson, J. Topological graph clustering with thin position / J. Johnson // Cornell University Library – Topological graph clustering with thin position. – Stillwater, 2012. – 2 с.
2. Хатчер, А. Алгебраическая топология / Хатчер А. // Алгебраическая Топология. Пер. с англ. В. В. Прасолова под ред. Т.Е. Панова – Москва, 2011. – 138 с.
3. Carlson, G. Topology and data / G. Carlson // American mathematical society. Topology and data. – California, 2009. – 256 с.
4. Edelsbrunner, H. Topological Persistence and Simplification / H. Edelsbrunner, D. Letscher, A. Zomorodian // Topological Persistence and Simplification. – Illinois, 2002. – 3 с.
5. Edelsbrunner, H. Computational topology: an introduction / H. Edelsbrunner, J. Harer // American mathematical society. Computational topology: an introduction. – Duke, 2010 – 182 с.

ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ: ПРИНЦИПЫ И ПРИЛОЖЕНИЯ К РЕШЕНИЮ АКТУАЛЬНЫХ ЗАДАЧ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Хвалько Д. В.

Искра Н. А. – старший преподаватель

Логическое программирование радикально отличается от других парадигм программирования. Сегодня в связи с техническим развитием в области параллельных вычислений и BigData стало возможным эффективное практическое применение логического программирования.

Отличительной чертой логической парадигмы программирования является наличие кванторов существования - утверждений о том, что существуют некоторые объекты с заданными свойствами. При работе с этой парадигмой используется конструктивный метод доказательства целевых утверждений: если требуемое было доказано, то в процессе доказательства находятся элементы, соответствующие ранее не определённым объектам, которые, однако, входят в целевое утверждение. Это соответствие и создаёт результат вычислений. Буквально это можно воспринимать как два выражения: программа – множество аксиом, вычисление – конструктивный вывод целевого утверждения из программы (множества аксиом).

Парадигма логического программирования имеет следующие особенности:

1. В логических языках отсутствуют операторы присваивания и побочные эффекты.
2. Применяется естественная математическая модель вычислений.
3. Есть возможность возвратов и перебора.
4. В язык встроены возможности по представлению списков, деревьев и др. коллекций.
5. Высоко развиты возможности мета-программирования.

Аппарат логического программирования, как и первые языки, основанные на этой парадигме, были разработаны в шестидесятые годы двадцатого века. С тех пор, за неимением необходимых технологий и достаточных ресурсов, про эту парадигму почти забыли, но в наши дни, в след за появлением параллельных вычислений, облачных вычислений и больших данных, благодаря увеличению вычислительных мощностей компьютеров, парадигма логического программирования вышла на новый виток развития. Логическое программирование в наши дни позволяет частично решать проблемы неоднозначности человеческого языка и проблему обобщения знаний в задачах искусственного интеллекта и построения экспертных систем [1].

Языки логического программирования представляют собой формализованные языки, где программа есть описание требуемого решения в терминах математической логики, а решение задачи строится в процессе логического вывода по заданному описанию. Существует много подходов к разработке программ на логических языках программирования: например, с использованием индуктивной логики или с использованием ограничений.

Исторически первым языком логического программирования был язык Planner. Этот язык был основан на автоматическом выводе результатов из данных и заданных правил перебора вариантов (которые и назывались планом, отсюда и название языка). Planner использовался, например, для снижения требований к вычислительным ресурсам с помощью поиска с возвратом.

Немного позднее был разработан язык Prolog, который не требовал плана перебора вариантов и был, в некотором роде, упрощением языка Planner.

От языка Planner в свою очередь произошли такие языки, как Conniver, Popler, QA-4, и QLISP. Языки программирования Datalog, Mercury, Visual Prolog были выделены как подмножества языка Prolog. Существуют также альтернативные языки логического программирования, которые не используют поиск с возвратом, например, Ether.

В последние годы, Datalog, который является синтаксическим подмножеством языка Prolog для представления и обработки данных, вновь возник в центре внимания создателей широкого спектра новых приложений, в том числе в таких сферах как:

- интеграция данных;