**Исследовательская работа**

По теме «Кластеризация данных»

Выполнила

Студентка КММ0-01-23

Плахотина Ю. С.

Москва, 2023–2024 г.

Оглавление

[Введение 3](#_Toc155462076)

[Генерация распределений 4](#_Toc155462077)

[Кластеризация изображений 6](#_Toc155462078)

[Генерация изображений 10](#_Toc155462079)

[Заключение 13](#_Toc155462080)

# Введение

В программном обеспечении ClustSystem были реализованы такие методы кластеризации, как BIRCH, CURE и ROCK с помощью библиотек pyclustering и scikit-learn.

Целью работы является оценка работоспособности и качества ПО в генерации и кластеризации данных.

Задача – провести кластеризацию данных различной размерности, поработать с неоднородными данными и рассмотреть влияние задаваемых параметров на конечный результат.

# Генерация распределений

В рамках исследования работы программы были рассмотрены различные наборы данных.

|  |  |  |  |
| --- | --- | --- | --- |
| *Рисунок 1 – входные данные генерации распределения* | | Сгенерируем 1000 точек в пространстве размерности 3. Возьмем стандартное нормальное распределение с и . Сравним результаты работы алгоритмов BIRCH-P и BIRCH-S. Можно заметить, что при одинаковом количестве кластеров количество элементов в них существенно отличается. Похожая проблема наблюдается в CURE и ROCK. Дело в том, что данные алгоритмы реализованы с помощью устаревшего pyclustering, а BIRCH-S используется из scikit-learn. | |
|  |  | |

*Рисунок 2 – результат кластеризации BIRCH-S алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 0.015625 |
| Показатель DunnIndex | 0.01788602205943323 |
| Показатель DunnIndexMean | 0.421281049512545 |
| Показатель DBi | 1.4776828069561347 |

|  |  |
| --- | --- |
|  |  |

*Рисунок 3 – результат кластеризации BIRCH-P алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 0.453125 |
| Показатель DunnIndex | 0.1474016514473182 |
| Показатель DunnIndexMean | 0.5068028560462193 |
| Показатель DBi | 0.43871971301986923 |

|  |  |
| --- | --- |
|  |  |

*Рисунок 4 – результат кластеризации ROCK алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 120.578125 |
| Показатель DunnIndex | 0.06046279402787573 |
| Показатель DunnIndexMean | 0.47066552694140146 |
| Показатель DBi | 0.8245495601067467 |

|  |  |
| --- | --- |
|  |  |

*Рисунок 5 – результат кластеризации CURE алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 4.359375 |
| Показатель DunnIndex | 0.10668970743209803 |
| Показатель DunnIndexMean | 0.5269471447786228 |
| Показатель DBi | 0.7834504266624879 |

Данные алгоритмы могут работать и с большим количеством точек, однако в рамках данного исследования мы столкнулись с нехваткой вычислительных мощностей, поэтому рассматриваем более малоразмерные примеры.

На данном примере заметим, что BIRCH-S лучше всего показывает себя по времени выполнения. Проблему с несбалансированным количеством элементов в кластерах можно решить, задавая для каждого алгоритма новые параметры.

Рассмотрим это на примере кластеризации изображения.

# Кластеризация изображений

|  |  |
| --- | --- |
|  |  |

*Рисунок 6 – результат кластеризации изображения CURE алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 14.671875 |

Поскольку подсчет расстояния между кластерами при обработке изображений рассматриваемыми алгоритмами требует большого количества времени, показатели DunnIndex, DunnIndexMean и DBi в данных примерах опустим.

Также не будет рассмотрен алгоритм ROCK, поскольку в данных условиях он выдаёт слишком медленный результат.

|  |  |
| --- | --- |
|  |  |

*Рисунок 7 – результат кластеризации изображения BIRCH-P алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 30.828125 |

|  |  |
| --- | --- |
|  |  |

*Рисунок 8 – результат кластеризации изображения BIRCH-S алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 5.8125 |

|  |  |  |
| --- | --- | --- |
| *Рисунок 9 – смена метрики* | Попробуем улучшить результат работы алгоритма *BIRCH-P*. Для этого, вместо Евклидовой, выберем другую метрику в задаваемых параметрах кластеризации. | |
|  | |  | |

*Рисунок 10 – результат кластеризации изображения BIRCH-P алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 184.828125 |

Время выполнения после смены метрики увеличилось в 6 раз.

Попробуем изменить тип конвертации изображения на YUV.

|  |  |
| --- | --- |
|  |  |

*Рисунок 11 – результат кластеризации изображения BIRCH-P алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 184.828125 |

|  |  |
| --- | --- |
|  |  |

*Рисунок 12 – результат кластеризации изображения CURE алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 13.171875 |

|  |  |
| --- | --- |
|  |  |

*Рисунок 13 – результат кластеризации изображения BIRCH-S алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 2.671875 |

BIRCH, реализованный на pyclustering, все так же плохо справляется с поставленной задачей. А вот CURE показывает уже более приемлемый результат.

# Генерация изображений

|  |  |  |
| --- | --- | --- |
| *Рисунок 14 – варианты генерации возможных изображений* | Теперь сгенерируем данные в виде спирали ДНК при помощи make\_dna с количеством точек генерации равном 100. | |
|  | |  | |

*Рисунок 15 – результат кластеризации CURE алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 0.0 |
| Показатель DunnIndex | 0.06017677934159759 |
| Показатель DunnIndexMean | 0.4063039186866146 |
| Показатель DBi | 1.6868950740815847 |

|  |  |
| --- | --- |
|  |  |

*Рисунок 16 – результат кластеризации BIRCH-R алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 0.015625 |
| Показатель DunnIndex | 0.03312069932852782 |
| Показатель DunnIndexMean | 0.4548434130839042 |
| Показатель DBi | 1.0541207951739133 |

|  |  |
| --- | --- |
|  |  |

*Рисунок 17 – результат кластеризации BIRCH-S алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 0.0 |
| Показатель DunnIndex | 0.04650108916846331 |
| Показатель DunnIndexMean | 0.7317184756117385 |
| Показатель DBi | 1.5185546493556596 |

|  |  |
| --- | --- |
|  |  |

*Рисунок 18 – результат кластеризации ROCK алгоритмом*

|  |  |
| --- | --- |
| Время работы алгоритма | 0.078125 |
| Показатель DunnIndex | 0.031742586889493135 |
| Показатель DunnIndexMean | 0.05797538276240025 |
| Показатель DBi | 0.6386980255790758 |

Результат алгоритма ROCK можно улучшить, если в качестве параметра радиуса связности взять значение .

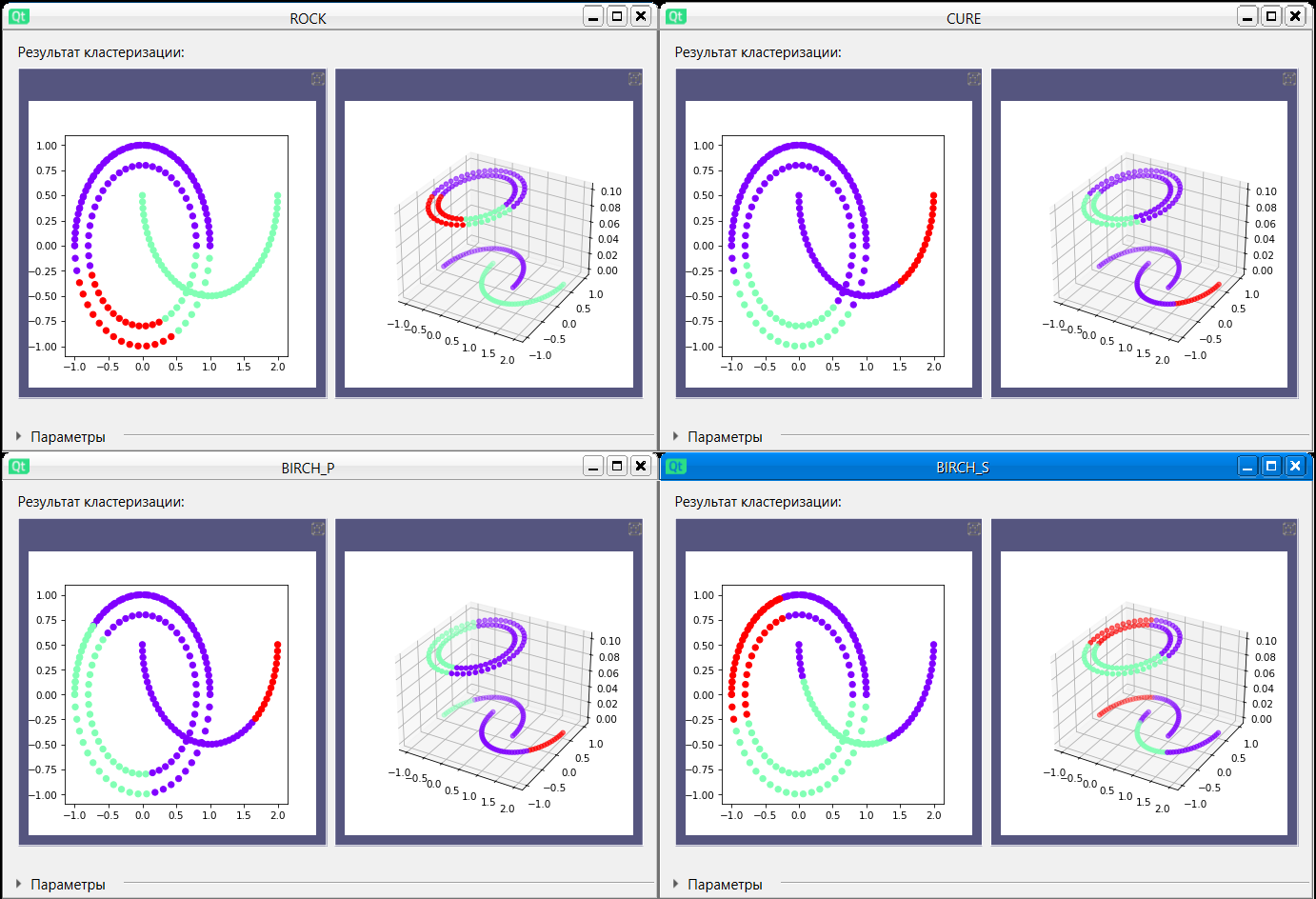
|  |  |
| --- | --- |
|  |  |

*Рисунок 19 – результат кластеризации ROCK алгоритмом при*

|  |  |
| --- | --- |
| Время работы алгоритма | 0.09375 |
| Показатель DunnIndex | 0.04483810491624281 |
| Показатель DunnIndexMean | 0.41606339335052295 |
| Показатель DBi | 1.882802471493716 |

Рассмотрим также случай, когда сгенерированы одновременно круги и «луны». Сместим круги по оси *z* на 0.1.

Все алгоритмы справились с поставленной задачей, однако разбиение на кластеры у каждого алгоритма свое.

  
*Рисунок 20 – результат кластеризации рассматриваемых алгоритмов*

# Заключение

В данном ПО реализован выбор параметров генерации и кластеризации, а также присутствуют данные для анализа результатов, что удовлетворяет заявленным требованиям и может в дальнейшем использоваться для кластеризации.