Классификация в непрерывном пространстве.

Реализация метода KNN на точках плоскости.

Владимирова Элина, Евдокимов Данил, Назаров Максим, Стельмах Татьяна

8 января 2021 г.

1 Резюме

Была решена задача объединения точек на плоскости в кластеры по Евклидовому расстоянию с использованием метода KNN.

Входные данные - требуемое количество кластеров. Посредством генерации определенных наборов пар координат на плоскости и проведения кластеризации точек обучающей выборки методами k-means и hierarchical и последующей классификации всего набора точек методом K-Nearest-Neighbours оказывается получено изображение осуществленного распределения путем рисования элементов каждого кластера уникальным цветом из заданной палитры, визуально сравниваемое затем с результатом разбиения всех точек на группы кластеризатором hierarchical.

2 Постановка задачи

Заданные точки на плоскости ($\geqslant 500$ пар координат) необходимо разбить на известное количество классов следующим образом:

1. Осуществить случайную выборку точек (около $\frac{1}{5}$-$\frac{1}{10}$ общего количества) из заданного массива;
2. Провести кластерный анализ точек тестовой выборки;
3. Используя метод K-Nearest-Neighbours (KNN), осуществить классификацию точек базового набора.

3 Допущения

- Количество точек считается заданным изначально.
- Количество кластеров задается посредством стандартного ввода.
- Кластеры не пересекаются и покрывают все множество точек, т.е. любая 1 точка гарантированно принадлежит ровно 1 кластеру.
- Генерация координат точек осуществляется функциями, выделенными в отдельный файл *pointGenerator.py*. Сделано это было в основном с целью получения нескольких различных простейших типов расположения точек в тестовых данных: обыкновенный случайный разброс, случайный разброс с приближением к прямой, 3 фрагментам различных прямых и вложенным эллипсам.
- Из соображений удобства формирования множества точек и их анализа тестовая выборка генерируется отдельно от общей.

4 Описание решения

Координаты точек основного массива и обучающей выборки генерируются по заданному количеству точек, после чего на тестовом множестве посредством встроенных функций Python производится иерархическая кластеризация (данный метод кластеризации был выбран из изученных как дающий наиболее близкие к истине результаты). После распределения элементов обучающего множества по кластерам производится классификация каждой точки основного множества, осуществляемая посредством применения к ней следующего алгоритма:

1. Ищутся 10 соседей - точек, ближайших к рассматриваемой. Эта выборка включает в себя и тестовые точки, если они окажутся в числе 10-ти ближайших.
2. Ищется ближайшая к рассматриваемой точка из тестовой выборки - ввиду малочисленности и, следовательно, широкого разброса элементов обучающего множества достаточно одной такой точки.
3. Посредством анализа соседей, классы которых уже известны, вычисляется наиболее часто встречающийся среди них класс. При возникновении неопределенности - если все соседи еще ожидают распределения или распределены по некоторым классам равномерно - решающую роль играет принадлежность к классу ближайшей точки из обучающего множества: рассматриваемая точка записывается в тот же класс.

В стремлении к большей точности группировки точек описанные вычисления повторяются 10 (для несложных и малочисленных случаев 5) раз.

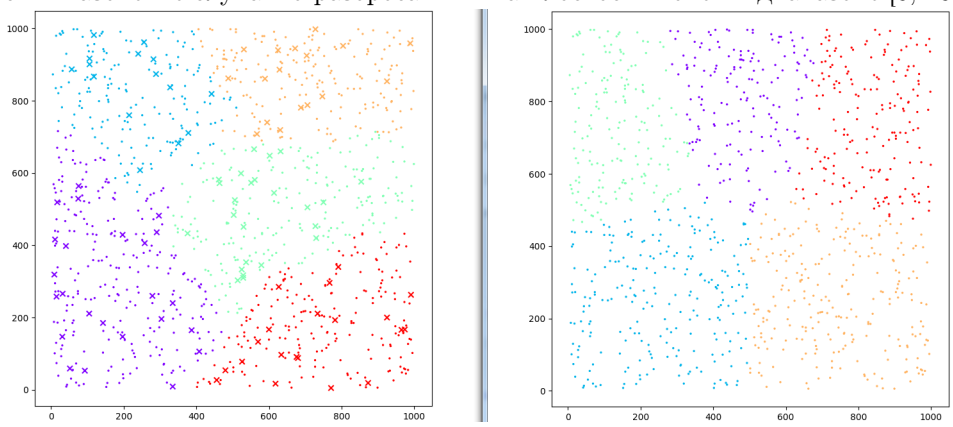
Все графики строятся с использованием библиотеки *matplotlib.pyplot*. На итоговых изображениях различные классы отрисованы различными цветами, точки тестовой выборки отмечены крестиками.

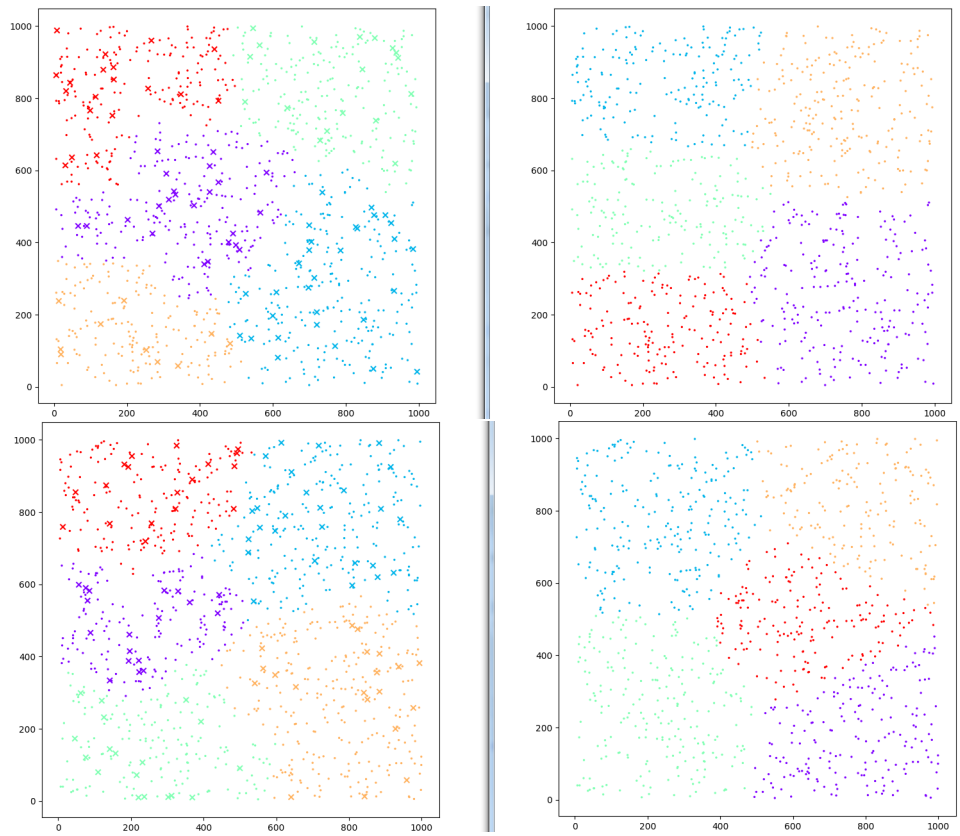
5 Результаты

Наборы точек, получаемые *pointGenerator.py*, успешно распределяются в классы, сопоставимые с принимаемыми за образец классами кластеризатора hierarchical, что дает повод считать код рабочим и для других, более сложных и многочисленных примеров. Корректность работы программы проверена на количестве 25, 100, 500 и 1000 точек и 2, 3, 4, 5, 7 и 10 классов.

Далее продемонстрированы примеры работы программы. На каждом изображении в левой части находится пример разбиения, полученного нашей программой, в правой - пример разбиения иерархического кластеризатора, принятый за образец. Примечание к конкретному изображению находится под ним. Кластеры окрашиваются в соответствии с нумерацией, различной для двух распределений, потому идентифицировать кластер можно только по его составу.

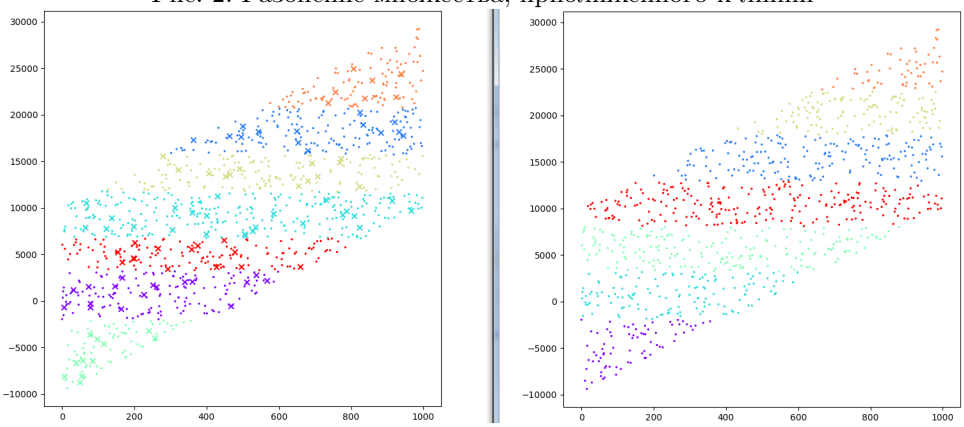
Рис. 1: Разбиение случайно разбросанных на плоскости точек в диапазоне [0; 1000]





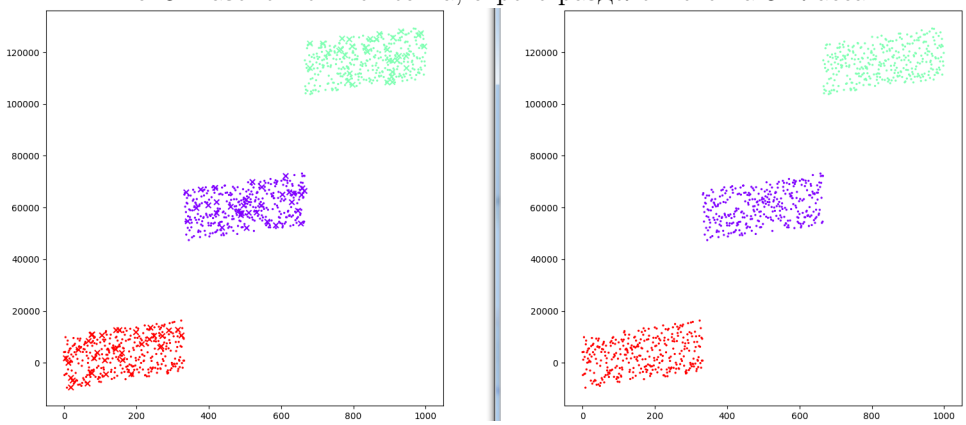
Здесь сложно сделать однозначные выводы по поводу правильности нашего классификатора, поскольку допустимо несколько различных вариантов разбиений, но каждая из демонстрируемых карт классов выглядит правдоподобно.

Рис. 2: Разбиение множества, приближенного к линии

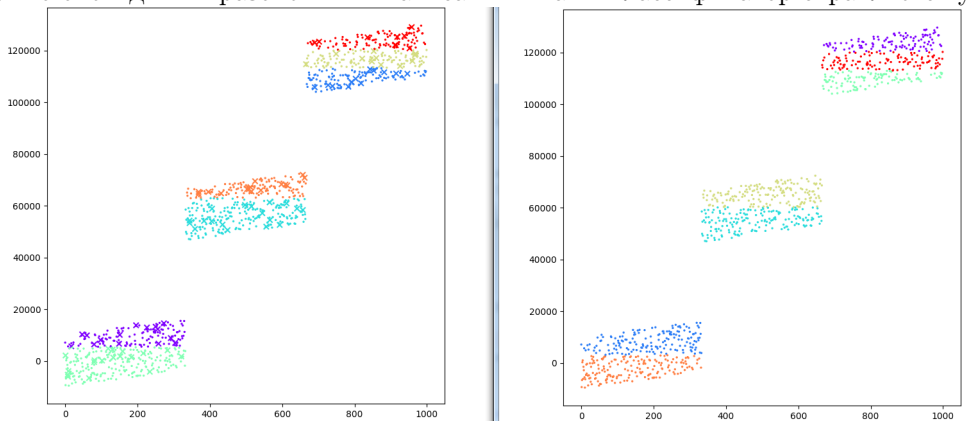


Разбиения имеют одинаковую структуру, однако несколько отличаются качественно. Схожесть результатов обусловлена использованием одного метода распределения точек тестовой выборки, дающего практически идентичные результаты на имеющих похожее расположение группах точек. Ввиду автоматического растяжения осей координат графики несколько искажены: на самом деле классы имеют приближенную к квадратной форму.

Рис. 3: Разбиение множества, строго разделенного на 3 класса



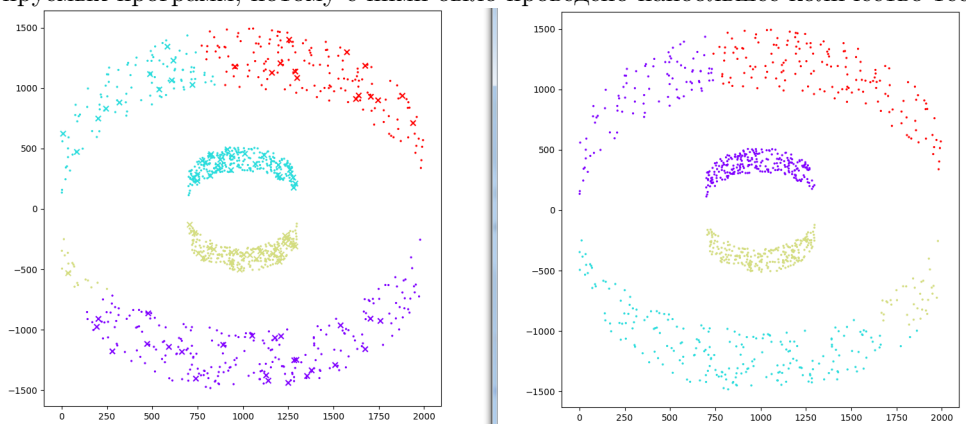
С визуально очевидными разбиениями написанный нами классификатор справляется успешно.



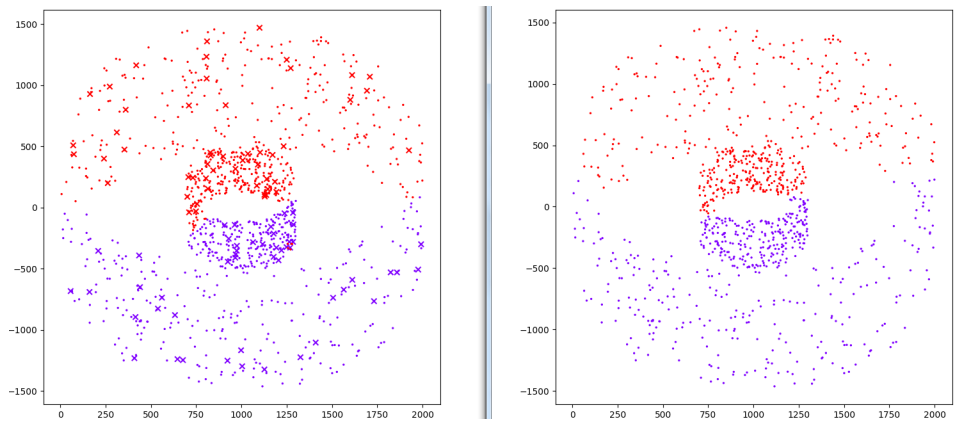
В более вариативных ситуациях распределения отличаются, но, тем не менее, имеют схожие паттерны по указанной выше причине.

Рис. 4: Разбиение множества, визуально представляющего собой вложенные эллипсы

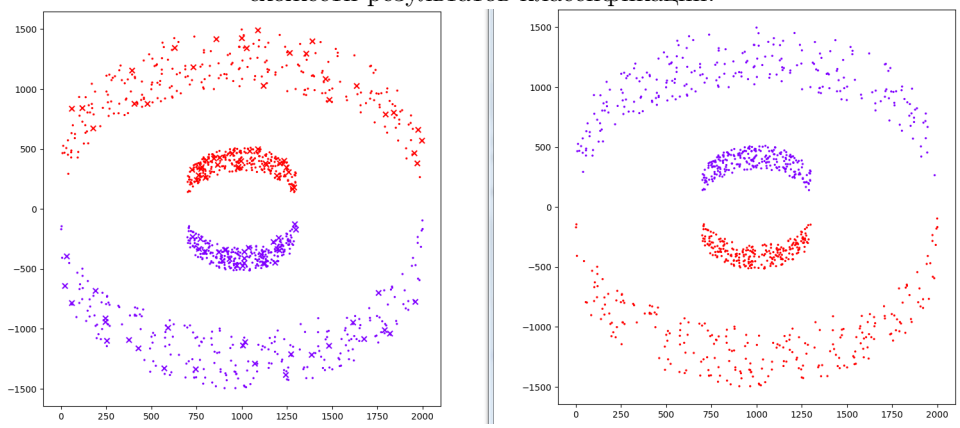
Именно на таких множествах наблюдаются наибольшие ошибки кластеризаторов и различия в работе тестируемых программ, потому с ними было проведено наибольшее количество тестов.



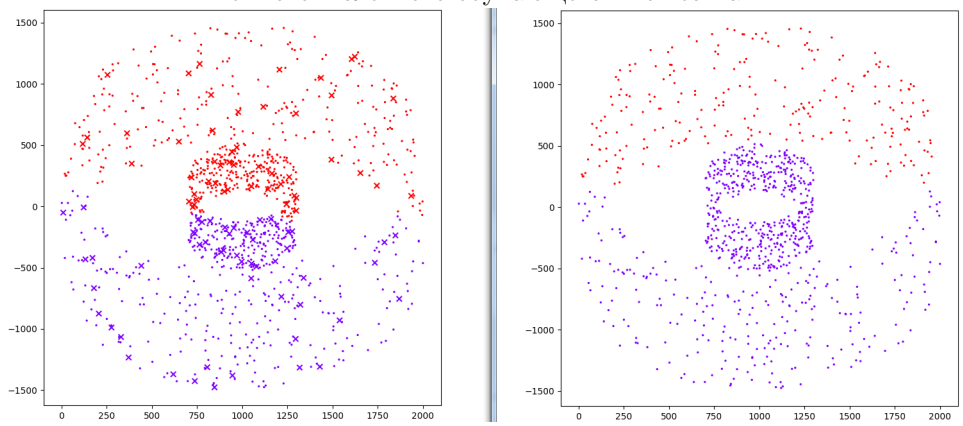
В данном случае в качестве кластеризатора тестовой выборки и образцового примера был использован метод K-Means, что привело к, с одной стороны, практически полному совпадению рисунка двух классов и, с другой, к зеркальной симметрии рисунков других классов. В целом, K-Means показывал менее интуитивно правдивые результаты на примерах генерации эллипсов и потому был заменен методом Hierarchical несмотря на более высокую скорость работы по сравнению с ним.



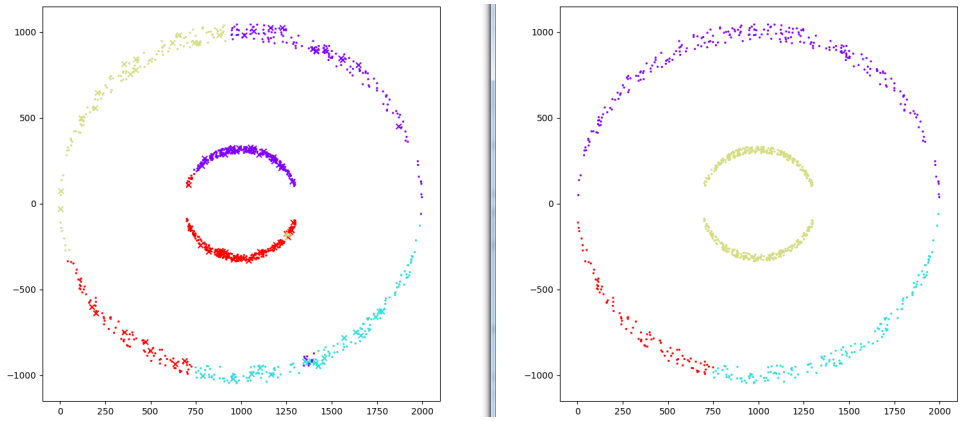
В большинстве случаев разбиения совпадают с точностью 90-95%, что в основном обусловлено постановкой в приоритет при вычислении класса конкретной точки принадлежности к классу ближайшей к рассматриваемой точки из тестовой выборки. Поскольку для разбиения на классы тестовой выборки и образцового примера используется один метод, на практике эти разбиения очень схожи, что ведет к схожести результатов классификации.



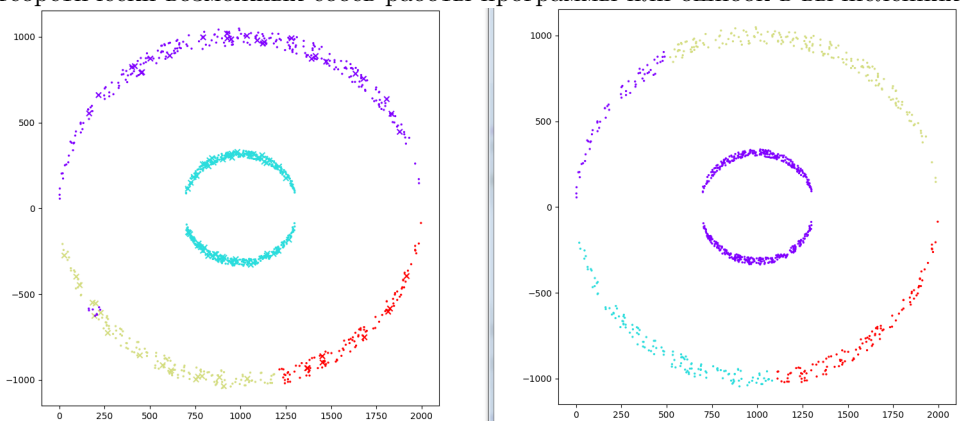
Отдельные случаи демонстируют совпадение с точностью 100%. Данный пример был сохранен из серии тестов с K-Means-кластеризатором как наиболее явный. Для Hierarchical-кластеризатора характерна небольшая погрешность на крайних точках эллипсов - следствие рандомизированного подбора относительно немногочисленного обучающего множества.



Данный пример демонстрирует результат применения разных кластеризаторов к тестовой выборке - был использован K-Means - и образцовому разбиению - был использован Hierarchical. Поскольку точки нижнего внешнего полуэллипса ближе подходят к внутреннему эллипсу, обе половины которого соединены, результат принимаемой за образец иерархической кластеризации закономерен. K-Means, кластеризовавший точки тестового множества, представил характерное для себя на данном примере разбиение, что и обусловило различие итоговых картин.



Наименьшее зафиксированное совпадение имеет точность $\sim 35\%$ и представляет собой случайно появившуюся и также случайно исчезнувшую уникальную загадку. Возможно, это был результат редких, но теоретически возможных сбоев работы программы или ошибок в вычислениях.



Редкие и наиболее ценные примеры классификации демонстрируют качественное превосходство результатов работы написанного нами классификатора над результатами применения встроенных методов Python. В данном случае в обоих разбиениях использовался метод Hierarchical, показывающий себя надежным и эффективным в разбиении слева, однако по неизвестной нам причине дающий ошибку в кластеризации справа.

Итого, были приведены и описаны примеры работы нашего алгоритма, сопоставленные с примерами работы классических кластеризаторов, реализованных с помощью встроенных методов Python. В среднем, доля совпадения составила $\sim 85\%$ на разбиениях, правильность которых могла быть оценена интуитивно.

Тактическая концепция	Стельмах Татьяна, Евдокимов Данил, Владимирова Элина, Назаров Максим
Подбор данных	Владимирова Элина
Код и проверка корректности его работы	Владимирова Элина, Евдокимов Данил
Отчет	Владимирова Элина, Стельмах Татьяна