

Задание 9. Кластеризация. Методы снижения размерности

Курс по методам машинного обучения, 2023-2024, Находнов Максим

1 Характеристики задания

- **Длительность:** 3 недели
- **Кросс-проверка:** 44 балла; в течение 1 недели после дедлайна; нельзя сдавать после жесткого дедлайна
- **Юнит-тестирование:** 16 баллов; можно сдавать после дедлайна со штрафом в 40%; публичная и приватная части; PER8
- **Почта:** ml.cmc@mail.ru
- **Темы для писем на почту:** ВМК.ML[Задание 9][peer-review], ВМК.ML[Задание 9][unit-tests]

Кросс-проверка: После окончания срока сдачи, у вас будет еще неделя на проверку решений как минимум **3х других студентов** — это **необходимое** условие для получения оценки за вашу работу. Если вы считаете, что вас оценили неправильно или есть какие-то вопросы, можете писать на почту с соответствующей темой письма

2 Описание задания

В данной работе вам предстоит познакомиться с методами машинного обучения без учителя — кластеризацией и алгоритмами снижения размерности.

В рамках юнит-тестирования Вам необходимо будет реализовать следующие две функции:

1. Функция для расчёта коэффициента силуэта
2. Функция для расчёта метрики B-Cubed

3 Кросс-проверка

- **Ссылка на задание:** [ссылка тут](#)
- **Тutorial по интерактивным графикам** [ссылка тут](#)

Подробное описание заданий для кросспроверки и соответствующая разбалловка находится в ноутбуке.

Обратите внимание, что для ускорения выполнения работы можно [скачать файл `cifar10_deep_features.npy`](#) (также скачивается автоматически в ноутбуке через gdown). Не забудьте положить его в ту же директорию, что и сам ноутбук. Выполненный ноутбук `Clusterization.ipynb` необходимо сдать в тестирующую систему во вкладку `Кластеризация (notebook)`.

Замечание: После отправки ноутбука убедитесь, что все графики сохранены корректно и правильно отображаются в системе.

Замечание: Перед сдачей проверьте, пожалуйста, что не оставили в ноутбуке где-либо свои ФИО, группу и так далее — кросс-рецензирование проводится анонимно.

4 Юнит-тестирование

Несколько важных замечаний:

Замечание: Запрещается пользоваться библиотеками, импорт которых не объявлен в файле с шаблонами функций.

Замечание: Задания, в которых есть решения, содержащие в каком-либо виде взлом тестов, дополнительные импорты и прочие нечестные приемы, будут автоматически оценены в 0 баллов без права пересдачи задания.

Замечание: Под циклами далее подразумеваются как явные Python-циклы (`for`, `while`, `list comprehension`, ...), так и неявное использование таких циклов внутри библиотек (`np.apply_along_axis` и подобные). В случае возникновения ошибки **Time limit** проверьте код на соответствие числа используемых циклов с требованиями к реализации.

Замечание: Для самопроверки доступны как публичные тесты, так и тесты внутри Jupyter Notebook (смотрите задания **1.c.1**, **1.c.3**).

4.1 Silhouette

Метрика **силуэт** является классическим представителем внутренних метрик кластеризации. Её суть заключается в оценке двух параметров, характеризующих выделенные кластеры — компактность и отделимость.

Положим, что C_i — номер кластера для объекта i .

s_i — компактность кластеризации объекта i определяется как среднее расстояние от него до всех объектов того же кластера:

$$s_i = \frac{1}{|\{j : C_j = C_i\}| - 1} \sum_{j: C_j = C_i} \|x_i - x_j\|$$

d_i — отделимость кластеризации объекта i определяется как среднее расстояние от него до всех объектов второго по близости кластера:

$$d_i = \min_{C: C \neq C_i} \frac{1}{|\{j : C_j = C\}|} \sum_{j: C_j = C} \|x_i - x_j\|$$

Тогда силуэт объекта i :

$$\text{sil}_i = \frac{d_i - s_i}{\max(d_i, s_i)}$$

И, наконец, коэффициент силуэта для выборки определяется как среднее силуэтов объектов:

$$S = \frac{1}{|X|} \sum_i \text{sil}_i$$

Если кластер состоит из одного объекта, то его силуэт равен нулю.

Реализуйте вычисление коэффициента силуэта для заданного разбиения. Шаблон функции представлен на листинге 1.

```
def silhouette_score(x, labels):  
    '''  
    :param np.ndarray x: Непустой двумерный массив векторов-признаков  
    :param np.ndarray labels: Непустой одномерный массив меток объектов  
    :return float: Коэффициент силуэта для выборки x с метками labels  
    '''  
  
    # Ваш код здесь: \(* o * l|l)/  
  
    return sil_score
```

Рис. 1: Шаблон для реализации подсчёта коэффициента силуэта

Входные данные тестов удовлетворяют одному из следующих ограничений:

1. Число объектов $n \leq 3000$, размерность пространства $d \leq 1200$
2. Число объектов $n \leq 5000$, размерность пространства $d = 1$

Ваша реализация должна удовлетворять следующим требованиям:

1. При вычислении не должно возникать warning, бесконечностей и nan-ов
2. Используйте не более одного цикла
3. Учтите, что метки кластеров могут идти не по порядку и принимать произвольные значения
4. Если в данных присутствует один кластер, то считайте что силуэт равен 0
5. Если $s_i = d_i = 0 \implies sil_i = 0$
6. Разрешено использовать `sklearn.metrics.pairwise_distances` и аналоги
7. Запрещено использовать любые библиотечные реализации коэффициента силуэта

4.2 B-Cubed

Пусть существует разметка (y_1, \dots, y_l) , не участвующая в обучении. Мы не использовали эту разметку в качестве дополнительного признака, так как нам не хочется мотивировать модель данным признаком. Тогда предлагается ввести оценку качества алгоритма кластеризации при помощи внешней разметки, саму же разметку тогда называют *gold standard*.

Один из вариантов учесть gold standard разметку — внешняя метрика B-Cubed. Данная метрика позволяет определять следующие особенности кластеризации:

1. **Гомогенность.** Базовое свойство разделения разных объектов в разные кластеры:

$$Q \left(\begin{array}{|c|c|c|} \hline & \diamond & \diamond \\ \hline \times & & \diamond \\ \hline \times & \times & \\ \hline \end{array} \right) < Q \left(\begin{array}{|c|c|c|} \hline & \diamond & \diamond \\ \hline \times & & \diamond \\ \hline \times & \times & \\ \hline \end{array} \right)$$

2. **Полнота.** Один кластер не должен дробиться на несколько маленьких:

$$Q \left(\begin{array}{|c|c|c|} \hline & \times & \times \\ \hline \times & & \times \\ \hline \times & \times & \\ \hline \end{array} \right) < Q \left(\begin{array}{|c|c|c|} \hline & \times & \times \\ \hline \times & & \times \\ \hline \times & \times & \\ \hline \end{array} \right)$$

3. **Rag-bag.** Весь мусор должен быть в одном "мусорном" кластере, чтобы остальные кластеры были "чистыми":

$$Q \left(\begin{array}{|c|c|c|} \hline \times & \times & \bullet \circ \\ \hline \times & \times & \triangleright \star \\ \hline \times & * & \odot \square \\ \hline \end{array} \right) < Q \left(\begin{array}{|c|c|c|} \hline \times & \times & \bullet \circ \\ \hline \times & \times & \triangleright \star \\ \hline \times & * & \odot \square \\ \hline \end{array} \right)$$

4. **Cluster size vs. quantity.** Лучше испортить один кластер с целью улучшить качество множества других:

$$Q \left(\begin{array}{|c|c|c|} \hline \times & \circ & \circ \\ \hline \times & \star & \star \\ \hline \times & \triangleright & \triangleright \\ \hline \times & \odot & \odot \\ \hline \end{array} \right) < Q \left(\begin{array}{|c|c|c|} \hline \times & \circ & \circ \\ \hline \times & \star & \star \\ \hline \times & \triangleright & \triangleright \\ \hline \times & \odot & \odot \\ \hline \end{array} \right)$$

Пусть $L(x)$ — gold standard, $C(x)$ — номер кластера, выдаваемый рассматриваемым алгоритмом. Рассмотрим несколько величин:

$$\text{Correctness}(x, x') = \begin{cases} 1, & C(x) = C(x') \wedge L(x) = L(x') \\ 0, & \text{иначе} \end{cases}$$

$$\text{Precision-BCubed} = \text{Avg}_x \text{ Avg}_{x': C(x)=C(x')} \text{Correctness}(x, x')$$

$$\text{Recall-BCubed} = \text{Avg}_x \text{ Avg}_{x': L(x)=L(x')} \text{Correctness}(x, x')$$

Тогда,

$$\text{B-Cubed} = F_1 = 2 \frac{\text{Precision-BCubed} \times \text{Recall-BCubed}}{\text{Precision-BCubed} + \text{Recall-BCubed}}$$

Реализуйте вычисление метрики B-Cubed. Шаблон функции представлен на листинге 2.

```
def bcubed_score(true_labels, predicted_labels):
    """
    :param np.ndarray true_labels: Непустой одномерный массив меток объектов
    :param np.ndarray predicted_labels: Непустой одномерный массив меток объектов
    :return float: B-Cubed для объектов с истинными метками true_labels и
        предсказанными метками predicted_labels
    """

    # Ваш код здесь: \(* o * l|l)/

    return score
```

Рис. 2: Шаблон для реализации подсчёта метрики B-Cubed

Входные данные тестов удовлетворяют одному из следующих ограничений:

1. Число объектов $n \leq 1000$, число подтестов в одном тесте $T \leq 70$

При реализации обратите внимание на следующие пункты:

1. При вычислении не должно возникать warning, бесконечностей и nan-ов
2. Использование циклов запрещено
3. Обратите внимание на параметр `where` у функций-агрегаторов в `numpy` (`numpy` $\geq 1.20.0$)
4. Запрещено использовать любые библиотечные реализации B-Cubed

5 Тестирование

В cv-gml можно скачать все файлы, необходимые для тестирования, одним архивом. Для этого просто скачайте zip-архив во вкладке **шаблон решения** соответствующего задания и разархивируйте его. Далее следуйте инструкциям по запуску тестирования.

5.1 Онлайн тестирование

Решение задач на юнит-тестирование сдаётся во вкладку `Кластеризация (unit-tests)` одним файлом `solution.py`. Шаблон данного файла (`solution_template.zip/solution.py`) можно скачать в тестирующей системе.

При тестировании баллы за каждую из функций начисляются независимо — 8 баллов за каждую задачу (7 баллов за прохождение всех частных тестов и 1 балл за прохождение всех публичных тестов).

Из-за особенностей тестирующей системы явного разделения тестов на публичные и частные, а также между отдельными подзадачами отсутствует.

Соответствие между номером теста и задачей можно установить из следующего списка:

[1-7]: `public/bcubed_score`

[8-15]: `public/silhouette_score`

[16-33]: `private/bcubed_score`

[34-52]: `private/silhouette_score`

Тестирование запускается следующей командой из корневой директории:

```
python ./run.py ./public_tests
# В случае успешного прохождения тестов вывод будет следующим:
>> Ok
>> ...
>> Ok
>> Mark: 2.0 2.000/2.000
```

6 Стиль программирования

При выполнении задач типа unit-tests, ML-задания вам необходимо будет соблюдать определенный стиль программирования (codestyle). В данном случае мы выбрали PEP8 как один из популярных стилей для языка Python. Зачем мы это вводим? Хорошая читаемость кода – не менее важный параметр, чем работоспособность кода :) Единый стиль позволяет быстрее понимать код сокомандников (в командных проектах, например), упрощает понимание кода (как другим, так и вам). Также, привыкнув к какому-либо стилю программирования, вам будет проще переориентироваться на другой.

Полезные при изучении PEP8 ссылки, если что-то непонятно, дополнительный материал можно найти самостоятельно в интернете:

- [Официальный сайт PEP8, на английском](#)
- [Небольшое руководство по основам на русском](#)

Требования к PEP8 мы вводим только для заданий с авто-тестами, требований к такому же оформлению ноутбуков нет. Но улучшение качества кода в соответствии с PEP8 в них приветствуется!

Внимание!!! В проверяющей системе, при несоответствии прикрепляемого кода PEP8, будет высвечиваться вердикт `Preprocessing failed`. Более подробно посмотреть на ошибки можно, нажав на них:

12.10.2022 cross_val.py
19:22 scalers.py

Preprocessing failed

Результат

Время
работы в
секундах

Preprocessing failed: Runtime error

```
Traceback (most recent call last):
  File "pre.py", line 39, in <module>
    raise RuntimeError(err_message)
RuntimeError: Found 6 errors or warnings in submission.
Detailed info:
scalers.py:6:65: W291 trailing whitespace
scalers.py:17:73: W291 trailing whitespace
scalers.py:31:13: E128 continuation line under-indented for visual indent
scalers.py:38:56: W291 trailing whitespace
scalers.py:44:43: W291 trailing whitespace
scalers.py:80:33: E131 continuation line unaligned for hanging indent
```

Также посылки, упавшие по code style, считаются за попытку сдачи и идут в счет общего количества посылков за день.

Проверить стиль программирования локально можно при помощи утилиты `pycodestyle` (в окружении, которое вы ставили, эта утилита уже есть) с параметром максимальной длины строки (мы используем 160 вместо дефолтных 79):

```
pycodestyle --max-line-length=160 your_file_with_functions.py
```