

Задание 8. Кластеризация. Методы снижения размерности.

Частичное обучение.

Курс по методам машинного обучения, 2021-2022, Находнов Максим

1 Характеристики задания

- **Длительность:** 2 недели (до жесткого дедлайна)
- **Кросс-проверка:** 30 баллов; в течение 1 недели после жесткого дедлайна; нельзя сдавать после жесткого дедлайна
- **Юнит-тестирование:** 15 баллов; можно сдавать после жесткого дедлайна; публичная и приватная часть
- **Почта:** ml.cmc@mail.ru
- **Темы для писем на почту:** ВМК.ML[Задание 8][peer-review], ВМК.ML[Задание 8][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. Также будет предложено применить кластеризацию и снижение размерности в задачах Частичного Обучения (Semi-Supervised learning).

Оценка работы складывается из двух частей — **оценки за кросс-проверку (30 баллов)** и **оценки за юнит-тестирование трёх функций (15 баллов)**.

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

1. Функция для расчёта коэффициента силуэта
2. Функция для расчёта метрики B-Cubed
3. Функция для классификации по результатам кластеризации с использованием размеченных объектов

3 Кросс-проверка

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

Обратите внимание, что для ускорения выполнения работы в шаблоне решения прилагается файл `cifar10_deep_features.npy`. Не забудьте положить его в ту же директорию, что и сам ноутбук. Выполненный ноутбук `Clusterization.ipynb` необходимо сдать в тестирующую систему во вкладку `Кластеризация (ноутбук)`.

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. Локальные тесты

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

При тестировании баллы за каждую из функций начисляются независимо — 5 баллов за каждую задачу (4.5 балла за прохождение всех частных тестов и 0.5 балла за прохождение всех публичных тестов). Порядок тестов можно определить с помощью файла `run.py` из тестирующей системы.

Для проверки своего решения на открытых тестах необходимо скачать архив `public_tests.zip` и скрипт для запуска `run.py` из тестирующей системы, а затем расположить все файлы в соответствии с диаграммой 1.

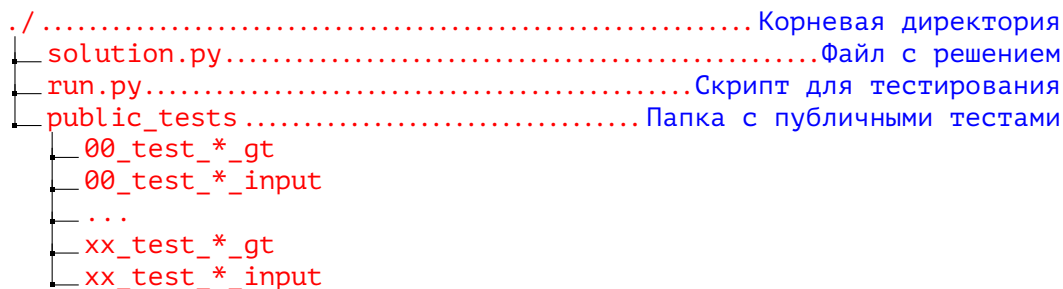


Рис. 1: Требуемая структура для локального тестирования на публичных тестах

Тестирование запускается следующей командой из корневой директории:

```
python ./run.py ./public_tests
# В случае успешного прохождения тестов вывод будет следующим:
>> Ok
>> ...
>> Ok
>> Mark: 1.5 1.500/1.500
```

5 Спецификация функций

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.

5.1 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 :

$$sil_i = \frac{d_i - s_i}{\max(d_i, s_i)}$$

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

$$S = \frac{1}{|X|} \sum_i sil_i$$

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

Реализуйте вычисление коэффициента силуэта для заданного разбиения. Шаблон функции представлен на листинге 2.

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 $d = 1$

```
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. 2: Шаблон для реализации подсчёта коэффициента силуэта

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

5.2 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:

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

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

$$Q \left(\begin{array}{ccc} & \times & \times \\ \times & & \times \\ \times & \times & \end{array} \right) < Q \left(\begin{array}{ccc} & \times & \times \\ \times & & \times \\ \times & \times & \end{array} \right)$$

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

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

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

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

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 3.

Входные данные тестов удовлетворяют одному из следующих ограничений:

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

```
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. 3: Шаблон для реализации подсчёта метрики B-Cubed

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

5.3 KMeansClassifier

Рассмотрим задачу Semi-Supervised learning для задачи классификации. В таком случае метки правильных классов известны только для части объектов. Будем считать, что метки для неразмеченных объектов равны -1 .

Предлагается следующий способ построения модели для решения задачи классификации: на первом шаге используется алгоритм кластеризации для определения групп похожих объектов. Затем, каждому кластеру назначается класс в соответствии с размеченной частью выборки.

Реализуйте данный алгоритм. Шаблон класса представлен на листинге 4. Обратите внимание, что автоматическое тестирование применяется только к функции `KMeansClassifier._best_fit_classification`. Работа остальных методов класса будет проверяться на кросс-проверке.

Входные данные тестов удовлетворяют одному из следующих ограничений:

1. Число объектов $n \leq 1000$, число кластеров $n_{\text{clusters}} \leq 100$, число подтестов в одном тесте $T \leq 200$
2. Число объектов $n \leq 10^7$, число кластеров $n_{\text{clusters}} \leq 2$, число подтестов в одном тесте $T = 1$

```
class KMeansClassifier(sklern.base.BaseEstimator):
    def __init__(self, n_clusters):
        """
        :param int n_clusters: Число кластеров которых нужно выделить
            в обучающей выборке с помощью алгоритма кластеризации
        """
        super().__init__()
        self.n_clusters = n_clusters

    # Ваш код здесь: \(* o * l|l)/
```

```

def fit(self, data, labels):
    """
        Функция обучает кластеризатор KMeans с заданным числом кластеров, а затем с помощью
        self._best_fit_classification восстанавливает разметку объектов

        :param np.ndarray data: Непустой двумерный массив векторов-признаков объектов обучающей выборки
        :param np.ndarray labels: Непустой одномерный массив. Разметка обучающей выборки.
            Неразмеченные объекты имеют метку -1. Размеченные объекты могут иметь произвольную
            неотрицательную метку. Существует хотя бы один размеченный объект
        :return KMeansClassifier
    """
    # Ваш код здесь: \(* o * l|l)/

    return self

def _best_fit_classification(self, cluster_labels, true_labels):
    """
        :param np.ndarray cluster_labels: Непустой одномерный массив. Предсказанные метки кластеров.
            Содержит элементы в диапазоне [0, ..., n_clusters - 1]
        :param np.ndarray true_labels: Непустой одномерный массив. Частичная разметка выборки.
            Неразмеченные объекты имеют метку -1. Размеченные объекты могут иметь
            произвольную неотрицательную метку.
            Существует хотя бы один размеченный объект
        :return
            np.ndarray mapping: Соответствие между номерами кластеров и номерами классов в выборке,
                то есть mapping[idx] -- номер класса для кластера idx
            np.ndarray predicted_labels: Предсказанные в соответствии с mapping метки объектов

            Соответствие между номером кластера и меткой класса определяется как номер класса
            с максимальным числом объектов внутри этого кластера.
            * Если есть несколько классов с числом объектов, равным максимальному,
                то выбирается метка с наименьшим номером.
            * Если кластер не содержит размеченных объектов, то выбирается номер класса
                с максимальным числом элементов в выборке.
            * Если же и таких классов несколько, то также выбирается класс с наименьшим номером
    """
    # Ваш код здесь: \(* o * l|l)/

    return mapping, predicted_labels

def predict(self, data):
    """
        Функция выполняет предсказание меток класса для объектов, поданных на вход.
        Предсказание происходит в два этапа
        1. Определение меток кластеров для новых объектов
        2. Преобразование меток кластеров в метки классов с помощью выученного преобразования
        :param np.ndarray data: Непустой двумерный массив векторов-признаков объектов
        :return np.ndarray: Предсказанные метки класса
    """
    # Ваш код здесь: \(* o * l|l)/

    return predictions

```

Рис. 4: Шаблон для реализации Semi-Supervised алгоритма с использованием KMeans