ВМК МГУ

Задание 10. Частичное Обучение. Методы снижения размерности.

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

1 Характеристики задания

- **Длительность:** 2 недели
- **Кросс-проверка:** 30 баллов; в течение 1 недели после дедлайна; нельзя сдавать после жесткого дедлайна
- **Юнит-тестирование:** 20 баллов; можно сдавать после дедлайна со штрафом в 40%; публичная и приватная части; PEP8
- **Почта:** ml.cmc@mail.ru
- **Темы для писем на почту:** ВМК.ML[Задание 10][peer-review], ВМК.ML[Задание 10][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 будет предложено применить кластеризацию и снижение размерности в задачах Частичного Обучения (Semi-Supervised learning).

В рамках юнит-тестирования Вам необходимо будет реализовать следующую функцию:

- Функция для классификации по результатам кластеризации с использованием размеченных объектов

3 Кросс-проверка

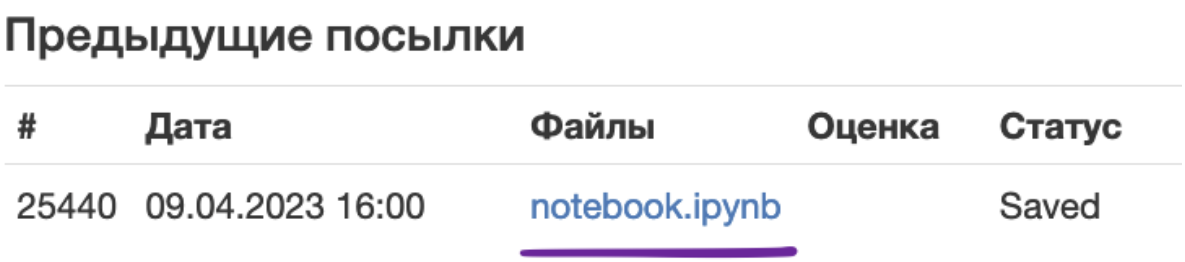
- **Ссылка на задание:** ссылка тут

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

Выполненный ноутбук `SemiSupervised.ipynb` необходимо сдать в тестирующую систему во вкладку `Частичное Обучение (notebook)`.

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

Замечание: После отправки ноутбука убедитесь, что все графики сохранены корректно и правильно отображаются в системе. Для этого достаточно открыть загруженный в систему файл и, просмотрев его, убедиться, что все сохранилось корректно



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 задание **1.1**).

4.1 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 1. При реализации разрешено использовать библиотечный класс `sklearn.cluster.KMeans`. Описание функций, формат входных и выходных данных, особенности реализации указаны в комментариях в листинге 1. 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 $n \leqslant 1000$, число кластеров $n_{clusters} \leqslant 100$, число подтестов в одном тесте $T \leqslant 200$
2. Число объектов $n \leqslant 10^7$, число кластеров $n_{clusters} \leqslant 2$, число подтестов в одном тесте $T = 1$

Ваша реализация должна удовлетворять следующим требованиям:

1. При вычислении не должно возникать warning, бесконечностей и nan-ов
2. Используйте не более одного цикла

```
class KMeansClassifier(sklearn.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 выборк
    :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. 1: Шаблон для реализации Semi-Supervised алгоритма с использованием KMeans

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 `Частичное Обучение (unit-tests)` одним файлом `solution.py`. Шаблон данного файла (`solution_template.zip/solution.py`) можно скачать в тестирующей системе.

Можно получить до 20 баллов: 4 балла за прохождение всех публичных тестов и 16 баллов за прохождение всех приватных тестов).

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

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

[1-8]: `public/kmeans_classifier`

[9-28]: `private/kmeans_classifier`

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 выбирали 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 19:22 cross_val.py 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 посылок 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
```