ВМК

Задание 3. Основы ООП в Python

Практикум 317 группы 2017-2018, осенний семестр

Начало выполнения задания: 22 сентября 2017 года.
Срок сдачи: **29 сентября 2017 года, 23:59.**

Решение каждой задачи должно быть описано в модуле `task_<номер задачи>.py`.

В систему anytask необходимо сдать `zip` архив, содержащий решения задач, называющийся `contest3_<фамилия студента>_<имя студента>.zip`.

Задания проверяются с помощью автоматических тестов. Количество баллов за задачу зависит от количества пройденных тестов. Задание считается засчитанным, если хотя бы одна задача проходит больше 25% тестов.

1. Написать собственный класс `CooSparseMatrix` для разреженной двумерной матрицы с координатным форматом хранения. При таком формате в памяти хранятся только ненулевые элементы матрицы и их координаты (номера строк и столбцов). Запрещается использовать готовые реализации разреженных матриц.

   Класс должен включать следующие методы:

   - `__init__(self, dense_matrix, shape=None)` — создание разреженной матрицы из плотной
     - `dense_matrix` — двумерный `numpy array`
     - `shape` — размер разреженной матрицы, если `None`, то совпадает с размером поданного массив. Если размер противоречит поданному массиву, необходимо выбросить исключение `TypeError`
   - `add_element(self, value, coords)` — присваивание элементу с координатами `coords` значение `value`. Метод возвращает `None`.
     - `value` — `float`
     - `coords` — кортеж из двух `int`. Если координаты выходят за shape матрицы, или элемент с такими же координатами уже содержится в матрице, необходимо выбросить исключение `TypeError`
   - `__add__(self, coo_matrix)` — сложение двух разреженных матриц. Если размеры матриц не совпадают, необходимо выбросить исключение `TypeError`. Метод возвращает `CooSparseMatrix`.
     - `coo_matrix` — объект типа `CooSparseMatrix`
   - `__mul__(self, value)` — умножение матрицы на число. Метод возвращает `CooSparseMatrix`.
     - `value` — `float`
   - `__getitem__(self, i)` — получение `i`-ой строки разреженной матрицы в формате `numpy array`. Если `i` выходит за shape, необходимо выбросить исключение `TypeError`. Метод возвращает одномерный `numpy array`.
     - `i` — `int`
   - `toarray(self)` — преобразование матрицы в `numpy array`. Метод возвращает двумерный `numpy array`.

   Замечание 1. При реализации может быть полезно использовать функцию `zip`

   Замечание 2. Реализовать хранение элементов можно с помощью различных структур данных. Например, можно использовать два списка, словарь или расширение словаря `defaultdict` из библиотеки `collections`

   Замечание 3. После выполнение операции, изменяющей значения элементов матрицы, необходимо удалять из памяти нулевы элементы.

   Замечание 4. `__add__` и `__mul__` не должны изменять элементы исходной матрицы

2. Написать класс `MulticlassStrategy`, позволяющий обобщить алгоритм, реализующий бинарную классификацию, на многоклассовый случай.

   Известно, что класс <Бинарный классификатор> имеет следующие методы:

   - `fit(X, y)` — метод, производящий обучение алгоритма
     - `X` — выборка, `numpy array` размера $N_{train} \times D$
     - `y` — вектор ответов, `numpy array` состоящий из 1 и -1, размера N_{train}

- `predict_proba(X)` — метод, выдающий для каждого объекта из выборки X вероятности принадлежностям классам
  - `X` — выборка, `numpy array` размера $\texttt{N}_{\texttt{test}} \times \texttt{D}$

Пусть решается задача классификации на k классов. Один из подходов к решению заключается в сведении этой задачи к нескольким задачам бинарной классификации. Пусть `y` — вектор ответов, каждый класс кодируется числом из множества $\{0, 1, \dots k-1\}$.

В классе `MulticlassStrategy` необходимо реализовать следующие методы:

- `__init__(self, classifier, **kwargs)` — инициализация классификатора
  - `classifier` — базовый бинарный классификатор, удовлетворяющий условиям выше
  - `**kwargs` — гиперпараметры классификатора
- `fit(self, X, y)` — обучение классификатора по методу один против всех.
  - `X` — выборка, `numpy array` размера $\texttt{N}_{\texttt{train}} \times \texttt{D}$
  - `y` — вектор ответов, `numpy array`, состоящий из $0, 1, \dots k-1$, размера $\texttt{N}_{\texttt{train}}$
- `predict(self, X)` — получить предсказания классификатора по методу один против всех. Метод должен вернуть одномерный `numpy array`, состоящий из меток классов
  - `X` — выборка, `numpy array` размера $\texttt{N}_{\texttt{train}} \times \texttt{D}$

Ликбез: метод один против всех для многоклассовой классификации

Пусть дана обучающая выборка $X = (x_i, y_i)_{i=1}^l$, где $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{Y} = \{1, \dots, K\}$. Задачу многоклассовой классификации можно свести к набору бинарных задач.

Один против всех (one-vs-rest): Обучается K классификаторов $a_1(x), \dots, a_K(x)$. Алгоритм $a_j(x)$ обучается по выборке X_j:

$$X_j = (x_i, 2\mathbb{I}[y_i = j] - 1)_{i=1}^l$$

Каждый $a_j(x)$ выдает вероятность принадлежности объекта к классу j. Итоговый классификатор будет выдавать класс, соответствующий самому уверенному алгоритму:

$$a(x) = \arg \max_{j \in \{1,\dots,k\}} a_j(x)$$