

Курс по методам машинного обучения

Практическое задание № 4

Линейные модели регрессии и классификации

Джеффри Хинтон, Ян Лекун, Йошуа Бенжио

1 Описание задания

В данном задании требуется реализовать несколько классов для предварительной обработки категориальных признаков с целью повышения качества линейной модели. Далее, необходимо воспользоваться свежереализованными классами для предобработки и ответить на вопросы в прилагаемом ноутбуке. Старайтесь отвечать развёрнуто и обосновано. Там где необходимо, напишите код, подтверждающий ваши размышления, и затежайте формулы.

Подробное описание того, что нужно реализовать приведено в следующей главе. Также, вы можете загрузить пример решения из системы и посмотреть на пример использования в публичных тестах.

Для **запуска публичных тестов**, необходимо скачать файл `run.py` и архив с публичными тестами. Архив нужно разархивировать в директорию `tests`, которая должна находиться там же, где и `run.py`. Запуск тестов производится одной из четырёх команд `"python3 run.py onehot"` `"python3 run.py weights"` `"python3 run.py counters"` `"python3 run.py foldcounters"` для тестирования one-hot кодирования, теоретического задания, счётчиков и счётчиков с фолдами соответственно.

Реализованные классы с предобработкой необходимо сохранить в файле Task.py и загрузить в систему. При запуске тестов Task.py должен лежать рядом с run.py. При этом запрещается использовать все библиотеки, кроме тех, которые импортируются в шаблоне Task.py. За эту часть задания можно получить 20 баллов. Напомним, что такой вид заданий можно сдавать после дедлайна со штрафом 40%.

Выполненный ноутбук с ответами также загружается в систему и оценивается по системе кросс-рецензирования. При этом, для выполнения ноутбука вам потребуются данные, которые также можно скачать в системе проверки заданий во вкладке "Linear models (unit-tests) "Дополнительные файлы". Сохраните скачанный csv файл в той же директории, что и ноутбук. За эту часть задания можно получить 25 баллов. Данный вид задания принимается **только до дедлайна**. 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, можете писать на почту ml.ctmc@mail.ru с темой письма ВМК.ML[Задание 4][peer-review] с просьбой перепроверить оценивание задания.

Важно: *перед сдачей проверьте, пожалуйста, что не оставили в ноутбуке где-либо свои ФИО, группу и т.д. — кросс-рецензирование проводится анонимно.*

Важно: *запрещается пользоваться библиотеками, импорт которых не объявлен в файлах с шаблонами функций.*

Важно: *задания, в которых есть решения, содержащие в каком-либо виде взлом тестов, дополнительные импорты и прочие нечест-*

ные приемы, будут автоматически оценены в 0 баллов без права пересдачи задания.

Важно: перед сдачей задания на кросс-проверку не забудьте перезапустить ноутбук: *Kernel -> Restart & Run All*.

Успехов!

2 Что нужно реализовать

2.1 One-hot преобразование

Класс `MyOneHotEncoder` реализует one-hot кодирование признаков. Пусть i -ый признак принимает значения $\{a_1, a_2, \dots, a_n\}$. Тогда значение a_j должно быть преобразовано в вектор длины n , у которого все компоненты кроме j -ой нулевые, а j -ая компонента равна единице. При этом, неоднозначность с номерами значений признаков разрешается следующим образом: меньший номер достаётся меньшему значению (для строк имеется ввиду лексикографический порядок).

В классе `MyOneHotEncoder` требуется реализовать методы `fit` и `transform`. Метод `fit` принимает на вход `pandas.DataFrame` размера $n_{\text{object}} \times n_{\text{features}}$ — обучающая выборка с категориальными признаками. После вызова метода `fit` объект класса `OneHotEncoder` должен запомнить всю необходимую информацию для one-hot преобразования признаков.

Метод `transform` (который и осуществляет напрямую one-hot кодирование) также принимает на вход `pandas.DataFrame` размера $n_{\text{object}} \times n_{\text{features}}$, где n_{features} совпадает с таковым у метода `fit`. Воз-

вращает метод `transform numpy.array` размера $n_{\text{object}} \times (|f_1| + \dots + |f_{n_{\text{features}}}|)$, где $|f_i|$ — количество уникальных значений i -го признака. Для устранения неопределённости будем считать, что при $i < j$, соответствующие i -ому признаку $|f_i|$ бинарных признаков идут раньше, чем соответствующие j -ому признаку $|f_j|$ бинарных признаков. Отдельно обрабатывать ситуацию, когда данные, попавшие в метод `transform`, содержат значения признаков, не встречавшиеся в обучающей выборке, **НЕ ТРЕБУЕТСЯ**.

2.2 Счётчики

Класс `Counters` реализует другой способ кодирования категориальных признаков. Пусть i -ый признак принимает значения $\{a_1, a_2, \dots, a_n\}$. Тогда признак a_j должен быть преобразован в вектор длины 3. Первая компонента данного вектора представляет из себя среднее значение целевой переменной для объектов, у которых i -ый признак принимает значение a_j :

$$\text{successes} = \frac{\sum_{k=1}^{n_{\text{objects}}} y_k \mathbb{I}[x_k^i = a_j]}{\sum_{k=1}^{n_{\text{objects}}} \mathbb{I}[x_k^i = a_j]}$$

Здесь \mathbb{I} означает индикаторную функцию: $\mathbb{I}[\zeta] = 1$, если выражение ζ истинно, и $\mathbb{I}[\zeta] = 0$, если выражение ζ ложно.

Вторая компонента — это доля объектов, у которых i -ый признак принимает значение a_j :

$$\text{counters} = \frac{\sum_{k=1}^{n_{\text{objects}}} \mathbb{I}[x_k^i = a_j]}{n_{\text{objects}}}$$

Третья компонента — сглаженное отношение между этими величинами:

$$\text{relation} = \frac{\text{successes} + a}{\text{counters} + b}, \quad a \geq 0, \quad b \geq 0$$

В классе `SimpleCounterEncoder` необходимо реализовать метод `fit`, аналогичный одноимённому методу класса `OneHotEncoder`. Отличие в том, что помимо обучающих объектов метод `fit` принимает на вход соответствующий вектор со значениями целевой переменной в формате `pandas.Series` (при правильной реализации форматы `numpy.array` и `pandas.DataFrame` также подойдут).

Метод `transform` в классе `counters` получает на вход `pandas.DataFrame` размера $n_{\text{object}} \times n_{\text{features}}$, где n_{features} совпадает с таковым у метода `fit`. Возвращает метод `transform` `numpy.array` размера $n_{\text{object}} \times 3 \cdot n_{\text{features}}$. Также у данного метода будут параметры a и b по умолчанию равные 10^{-5} .

Одним из недостатков кодирования при помощи счётчиков является риск утечки значения целевой переменной для данного объекта. На практике это может приводить к переобучению. Для того, чтобы это предотвратить обучающая выборка разбивается на k подмножеств, и величины `successes`, `counters` и `relation` считаются по оставшимся $k - 1$ подмножествам (как в кросс-валидации).

Данную стратегию реализует класс `FoldCounters`, в котором опять нужно реализовать методы `fit` и `transform`. Также обратите внимание, что в метод `__init__` подаётся на вход параметр `n_folds` — число подмножеств, которое необходимо использовать в методе `__init__`. Метод `fit` устроен также, как и в `Counters`, за исключени-

ем параметра `seed`. Для разбиения обучающей выборки на фолды необходимо использовать вспомогательную функцию `group_k_fold`. Данное разбиение необходимо запомнить в методе `fit`! Функция `group_k_fold` генерирует случайное разбиение, поэтому не забываем прокидывать параметр `seed`.

Метод `transform` в классе `FoldCounters` получает на вход `pandas.DataFrame` размера $n_{\text{object}} \times n_{\text{features}}$, где n_{object} и n_{features} совпадают с таковыми у метода `fit` (`fit` и `transform` применяются к одной выборке). Возвращает метод `transform` `numpy.array` размера $n_{\text{object}} \times 3 \cdot n_{\text{features}}$. Также у данного метода будут параметры `a` и `b` по умолчанию равные 10^{-5} .

Отдельно рассматривать ситуацию, когда в одно или несколько разбиений не попали все значения какого-то признака **НЕ ТРЕБУЕТСЯ**.

2.3 Теоретический вопрос

Рассмотрим следующую задачу. Величина x принимает значения из конечного множества X . Целевая переменная y принимает значения $\{0, 1\}$. То есть можно сказать, что у нас бинарная классификация с одним категориальным признаком. К величине x применяется one-hot преобразование, в результате которого возникает величина x' (это уже вектор из нулей и единиц). Прогноз $p(x)$ целевой переменной y равен: $p(x) = \langle w, x' \rangle$, где w — обучаемые веса. Обратите внимание, что здесь нет свободного члена, и нет сигмоиды. Обучение весов w осуществляется минимизацией логистической функции потерь (logloss) по весам w : $L = - \sum_{i=1}^{n_{\text{objects}}} y_i \cdot$

$\log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))$. Как будут выглядеть оптимальные веса?

В качестве ответа на вопрос, требуется написать функцию `weights` (не используя специализированных библиотек для линейной классификации). Данная функция принимает на вход параметры x и y : оба `numpy.array` размера n_{objects} — это обучающая выборка. Возвращает функция список с оптимальными значениями весов. Порядок весов в списке определяется аналогично one-hot кодированию.