

Задание 6. Метод опорных векторов (SVM)

Курс по методам машинного обучения, 2023-2024, Николай Скачков

1 Характеристики задания

- **Длительность:** 2 недели
- **Кросс-проверка:** 20 баллов + 2 бонусных балла; в течение 1 недели после дедлайна; нельзя сдавать после жесткого дедлайна
- **ML-задание:** 10 баллов; Можно сдавать после дедлайна со штрафом в 40%; публичная и приватная части
- **Почта:** ml.cmc@mail.ru
- **Темы для писем на почту:** ВМК.ML[Задание 6][peer-review], ВМК.ML[Задание 6][ml-task]

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 данном задании Вам предстоит изучить работу SVM и исследовать свойства этого семейства алгоритмов. Итоговым решением задания является ноутбук с экспериментами, требования к которым описаны в шаблоне ноутбука, и который будет проверяться на кросс-проверке. Также в задании присутствует контекст (ML-решение), на котором Вам предстоит обучить модель и показать качество на тесте выше указанного порога.

2.1 Постановка задачи SVM

В самом простом случае SVM задаётся следующими условиями: мы хотим найти гиперплоскость, верно разделяющую объекты двух классов таким образом, чтобы полоса, определяемая этой прямой (под полосой подразумевается множество гиперплоскостей, параллельных данной, каждая из которых безошибочно разделяет два класса), была наибольшей ширины. Признаковые описания N объектов будут описываться векторами x_i , а метка класса обозначаться $y_i \in \{-1, 1\}$.

Нетрудно показать, что максимизация ширины эквивалентна $\frac{2}{\|w\|^2} \rightarrow \max_w$. Где w — нормаль, задающая искомую гиперплоскость. Если b — сдвиг, то получается условная задача оптимизации следующего вида:

$$\begin{cases} \|w\|^2 \rightarrow \min_w \\ y_i(w^T x_i - b) \geq 1, i \in (1, N) \end{cases} \quad (1)$$

Для этой задачи можно выписать двойственную и решать её (если Вы не проходили что это, можете почитать в интернете). Двойственная для этой задачи выглядит так:

$$\begin{cases} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j (x_i, x_j) - \sum_{i=1}^N \lambda_i \rightarrow \min_{\lambda} \\ \lambda_i \geq 0, i \in (1, N) \\ \sum_{i=1}^N y_i \lambda_i = 0 \end{cases} \quad (2)$$

В системе (2) обучаемые переменные λ — показывают «опорность» объектов. Смысл этого Вы сможете лучше понять, визуализировав опорные объекты при выполнении задания.

К сожалению системы (1) и (2) не имеют решения в случае линейной неразделимости выборки. Эта проблема решается добавлением дополнительных переменных-штрафов, однако выписывать систему для этого случая не будем. Решаемая система для нелинейного случая выглядит так:

$$\begin{cases} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j (x_i, x_j) - \sum_{i=1}^N \lambda_i \longrightarrow \min_{\lambda} \\ 0 \leq \lambda_i \leq C, i \in (1, N) \\ \sum_{i=1}^N y_i \lambda_i = 0 \end{cases} \quad (3)$$

где C — некий гиперпараметр, который нужно подбирать под каждую задачу.

Самое большое достоинство SVM — это возможность обобщить модель для нелинейного случая. Для этого делается следующий логический переход. Пусть есть некоторое преобразование $\psi : X \rightarrow H$. Где X — пространство входных объектов, H — некоторое бесконечномерное пространство. Тогда в системе (1) мы не сможем посчитать произведение $w^T \psi(x_i)$, и получим бесконечное множество параметров.

Зато в системе (2) у нас изменится только скалярное произведение объектов (x_i, x_j) . Причем нам не нужно знать, как выглядит само преобразование ψ . Достаточно только задать вид желаемого скалярного произведения, которое будем обозначать так $(x_i, x_j)_{\psi}$.

Не будем углубляться в то, какие требования нужно предъявить к функции $\text{kernel}(\cdot, \cdot)$, чтобы для него существовало ψ , для которого $\text{kernel}(\cdot, \cdot) = (\cdot, \cdot)_{\psi}$ (такие преобразования называются ядрами), сразу приведем несколько примеров:

1. $\text{kernel}(x, x') = (x^T x')^d$ — полиномиальное ядро степени d ;
2. $\text{kernel}(x, x') = e^{-\gamma \|x - x'\|^2}$ — rbf ядро.

Все эти ядра реализованы, Вам нужно будет только исследовать их работу.

В случае ядрового преобразования, решаемая система выглядит так:

$$\begin{cases} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j \text{kernel}(x_i, x_j) - \sum_{i=1}^N \lambda_i \longrightarrow \min_{\lambda} \\ 0 \leq \lambda_i \leq C, i \in (1, N) \\ \sum_{i=1}^N y_i \lambda_i = 0 \end{cases} \quad (4)$$

Вся данная информация дана скорее для расширения кругозора, однако может сильно помочь с объяснением результатов экспериментов, предложенных в ноутбуке.

3 Кросс-проверка

- Ссылка на задание: [ссылка тут](#)

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 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	cross_val.py	Preprocessing failed	
19:22	scalers.py	# Результат	Время работы в секундах
<div> <div>Preprocessing failed: Runtime error</div> <div> <pre> 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 </pre> </div> </div>			

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 вместе дефолтных 79):

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

5 ML-задание

Важно: В этом задании есть относительно новая форма сдачи для вас — **контекст (ML-решение)**. Решение по структуре и тестированию аналогично решениям для заданий типа unit-test. Только в случае ML-решения вы пишете, как правило, какую-то функцию, в которой обучается ML-модель и возвращаются предсказания на тестовом множестве. А система проверяет качество (метрику качества) вашего решения на заданных данных (как открытых, так и закрытых) и по заданным порогам выставляет оценку. Более подробно про каждое такое задание будет описано в pdf.

В файле `svm_solution.py` (можно найти в шаблоне решения) из вкладки «Основы SVM (ML)» необходимо написать функцию, в которой обучается модель по входной выборке (она в системе такая же как у Вас в приложении) и делается предсказание для тестовой выборки. Важно чтобы выход функции был **в таком же формате**, как и в шаблоне (как и вход). Оцениваться ваша модель будет по следующим критериям (все ДО – включительно):

На закрытом тесте:

1. если ассигасу на тесте не больше 80%, то 0 баллов
2. если ассигасу на тесте от 80% до 90%, то 2 баллов.
3. если ассигасу на тесте от 90% до 92%, то 5 баллов.
4. если ассигасу на тесте больше 92%, то 8 баллов.

На открытом тесте:

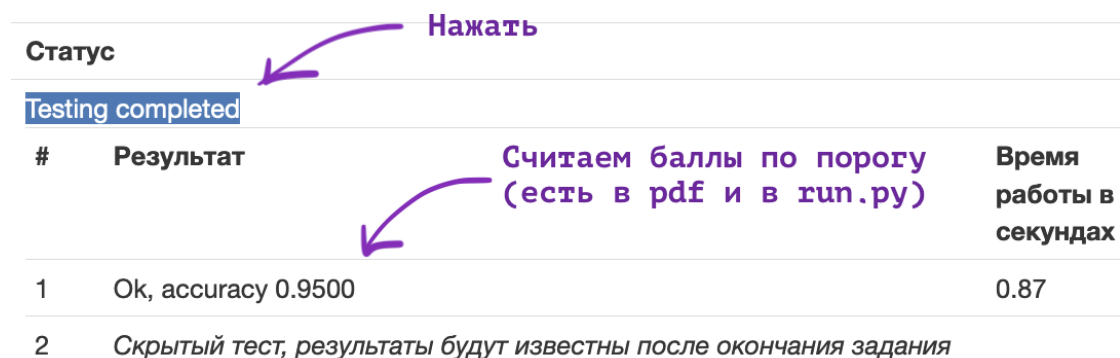
1. если ассигасу на тесте не больше 60%, то 0 баллов

2. если ассигасу на тесте от 60% до 80%, то 1 балл.
3. если ассигасу на тесте больше 80%, то 2 баллов.

В систему сдается написанный вами файл `svm_solution.py`, в котором реализована функция. Число попыток сдачи ограничено пятью в день, так что оценивайте качество Вашей модели на кросс-валидации в ноутбуке.

В данном задании два теста (два среза данных): публичный, который оценивается из **2 баллов**, и приватный (недоступен вам), оцениваемый из **8 баллов**. За публичный тест Вы можете получить гарантированные баллы уже до жёсткого дедлайна, а баллы за приватный тест вам откроются после дедлайна.

В системе, из-за наличия закрытого теста, вы не сможете увидеть свои баллы до дедлайна даже за открытый тест. Но вы сможете посчитать баллы за открытый тест способом, приведенным на картинке ниже:



Статус			
Testing completed			
#	Результат	Считаем баллы по порогу (есть в pdf и в run.py)	Время работы в секундах
1	Ok, accuracy 0.9500		0.87
2	Скрытый тест, результаты будут известны после окончания задания		

Важно: В этом задании в файле с шаблоном `svm_solution.py` **МОЖНО** использовать дополнительные импорты (например, для обработки признаков), однако модель, которую вы обучаете, должна быть из семейства моделей *SVM*. В ноутбуке можно использовать дополнительные импорты.

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

6 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.

Запуск тестов производится командой

```
python run.py ./public_tests
```