ВМК

Задание 2. Метрические алгоритмы классификации

Практикум 317 группы, 2015

Начало выполнения задания: 8 октября 2016 года.
Срок сдачи: **23 октября 2016 года, 23:59.**

Формулировка задания

Данное задание направлено на ознакомление с метрическими алгоритмами классификации, а также методами работы с текстовой информацией и изображениями. В задании необходимо:

1. Написать на языке Python функции для собственной реализации метода ближайших соседей и кросс-валидации. Прототипы функций должны строго соответствовать прототипам, описанным в спецификации ниже. При написании необходимо пользоваться стандартными средствами языка Python, библиотеками numpy и matplotlib. Библиотеками scipy и scikit-learn пользоваться запрещается, если это не обговорено отдельно в пункте задания.
2. Провести описанные ниже эксперименты с датасетами MNIST и 20 newsgroups.
3. Написать отчёт в IPython Notebook. Ячейки с кодом должны быть прокомментированы с помощью `markdown` ячеек. За отсутствие анализа экспериментов и выводов балл будет снижаться! В начале отчёта обязательно укажите используемую версию Python.

Список экспериментов

1. Загрузить датасет MNIST при помощи функции `sklearn.datasets.fetch_mldata("MNIST original")`.
2. Разбить датасет на обучающую выборку (первые 60 тыс. объектов) и тестовую выборку (10 тыс. последних объектов). Ответы на тестовой выборке не следует использовать ни в каких экспериментах, кроме финального.
3. Визуализировать по 5 случайных объектов из каждого из 10 классов. Воспользуйтесь методами `np.reshape`, `pyplot.subplot`, и `pyplot.imshow` с параметром `cmap="Greys"`. Также можно убрать оси координат при помощи команды `pyplot.axis("off")`.
4. Исследовать, какой алгоритм поиска ближайших соседей будет быстрее работать в различных ситуациях. Для каждого объекта тестовой выборки найти 5 его ближайших соседей в обучающей (при этом ответы на тестовой выборке не используются!) для евклидовой метрики. Для выборки нужно выбрать подмножество признаков, по которому будет считаться расстояние, размера 10, 20, 100 (подмножество признаков выбирается один раз для всех объектов, случайно). Необходимо проверить все алгоритмы поиска ближайших соседей, указанные в спецификации к заданию.

   Замечание. Для оценки времени долго работающих функций можно пользоваться либо функциями из модуля `time`, либо magic-командой `%time`, которая запускает код лишь один раз.
5. Оценить по кросс-валидации с 3 фолдами точность (долю правильно предсказанных ответов) и скорость метода k ближайших соседей в зависимости от следующих факторов:
   (a) k от 1 до 10 (только влияние на точность).
   (b) Используется евклидова или косинусная метрика.
6. Сравнить взвешенный метод k ближайших соседей, где голос объекта равен $1/(distance + \varepsilon)$, где ε — малое число, с методом без весов при тех же фолдах и параметрах.
7. Применить лучший алгоритм к исходной обучающей и тестовой выборке. Подсчитать точность. Сравнить с точностью по кросс-валидации. Сравнить с указанной в интернете точностью лучших алгоритмов на данной выборке.
8. Визуализировать несколько объектов из тестовой выборки, на которых были допущены ошибки. Проанализировать и указать их общие черты. Построить и проанализировать матрицу ошибок (confusion matrix). Можно воспользоваться функцией `sklearn.metrics.confusion_matrix`.

9. Загрузить обучающую выборку датасета 20 newsgroups при помощи метода `sklearn.datasets.fetch_20newsgroups`. Убрать все заголовки, подписи и цитаты, используя аргумент `remove`.

   Замечание. Метод `sklearn.datasets.fetch_20newsgroups` может работать некорректно, если датасет уже содержится на диске.

10. Перевести во всех документах все буквы в нижний регистр. Заменить во всех документах символы, не являющиеся буквами и цифрами, на пробелы.

    Замечание. Полезные функции: `str.lower`, `str.isalnum`. Полезный модуль: `re`.

11. Разбить каждый документ на термы по пробельным символам (пробелы, переносы строки).

    Замечание. Полезные функции: `str.split`, `re.split`.

12. Преобразовать датасет в разреженную матрицу `scipy.sparse.csr_matrix`, где значение x в позиции (i, j) означает, что в документе i слово j встретилось x раз. Необходимо воспользоваться наиболее эффективным конструктором `csr_matrix((data, indices, indptr), shape=(M, N))`.

    Замечание. Не забудьте указать параметр `shape`, так как число используемых термов в тестовой выборке может отличаться от числа термов в обучении.

13. Произвести tf-idf преобразование датасета при помощи `sklearn.feature_extraction.text.TfidfTransformer`. Используйте параметры по умолчанию.

14. Оценить точность (долю правильно предсказанных ответов) и скорость метода k ближайших соседей при помощи кросс-валидации с 3 фолдами. Исследуйте на одних и тех же фолдах влияние следующих факторов:

    (a) k от 1 до 10 (только влияние на точность).

    (b) Используется евклидова или косинусная метрика.

    (c) Используется ли преобразование tf-idf.

    (d) Используются взвешенный или невзвешенный метод k ближайших соседей.

15. Загрузите тестовую выборку (параметр `subset` метода `fetch_20newsgroups`). Примените лучший алгоритм к тестовой выборке. Сравнить точность с полученной по кросс-валидации.

16. Вывести несколько документов из тестовой выборки, на которых были допущены ошибки. Проанализировать их. Построить и проанализировать матрицу ошибок (confusion matrix). Можно воспользоваться функцией `sklearn.metrics.confusion_matrix`.

17. Сделать выводы: в каких случаях следует пользоваться каким алгоритмом, какие параметры важно оптимизировать, какие ошибки допускают алгоритмы.

18. Бонусное задание: попробуйте улучшить точность метода k ближайших соседей на тестовой выборке датасета MNIST. Размер бонуса будет зависеть от достигнутой точности:

    +0.2 при точности $\geqslant 0.98$

    +0.4 при точности $\geqslant 0.985$

    +0.8 при точности $\geqslant 0.99$

    Замечание. Для улучшения результата можно попробовать преобразовать исходное признаковое пространство, изменить метрику, использовать несколько алгоритмов k ближайших соседей, соединённых в ансамбль.

Требования к реализации

Замечание 1. Далее под выборкой объектов будем понимать `numpy array` размера $N \times D$ или разреженную матрицу `scipy.sparse.csr_matrix` того же размера, под ответами для объектов выборки будем понимать `numpy array` размера N, где N — количество объектов в выборке, D — размер признакового пространства.
Замечание 2. Для всех функций можно задать аргументы по умолчанию, которые будут удобны вам в вашем эксперименте.
Среди предоставленных файлов должны быть следующие модули и функции в них:

1. Модуль `nearest_neighbors`, содержащий собственную реализацию метода ближайших соседей.

   Класс `KNN_classifier`

   В классе обязательно должны присутствовать атрибуты:

   - `k` — число ближайших соседей в алгоритме ближайших соседей
   - `strategy` — алгоритм поиска ближайших соседей. Может принимать следующие значения:
     - `'my_own'` — собственная реализация (например, на основе кода подсчёта евклидова расстояния между двумя множествами точек из задания №1)
     - `'brute'` — использование `sklearn.neighbors.NearestNeighbors(algorithm='brute')`
     - `'kd_tree'` — использование `sklearn.neighbors.NearestNeighbors(algorithm='kd_tree')`
     - `'ball_tree'` — использование `sklearn.neighbors.NearestNeighbors(algorithm='ball_tree')`

     Замечание. Для некоторых алгоритмов поиска ближайших соседей вам потребуется хранить обучающую выборку и ответы на ней. Некоторые алгоритмы не требуют хранения выборки, но требуют хранения дополнительной информации о её структуре.
   - `metric` — название метрики, по которой считается расстояние между объектами. Может принимать следующие значения:
     - `'euclidean'` — евклидова метрика
     - `'cosine'` — косинусная метрика
   - `weights` — переменная типа `bool`. Значение `True` означает, что нужно использовать взвешенный метод k ближайших соседей. Во взвешенном методе ближайших соседей голос одного объекта равен $1/(distance + \varepsilon)$, где $\varepsilon — 1e - 5$
   - `test_block_size` — размер блока данных для тестовой выборки

     Замечание. При поиске k ближайших соседей некоторые методы строят в памяти матрицу попарных расстояний обучающей выборки и тестовой выборки. Рекомендуется написать функцию, которая ищет ближайших соседей блоками, то есть делает запросы ближайших соседей для первых N тестовых объектов, затем для следующих N, и так далее, и в конце объединяет полученные результаты.

   Описание методов:

   (a) `__init__(self, k, strategy, metric, weights, test_block_size)`

   Описание параметров:

   - Все параметры аналогичны атрибутам класса

   Конструктор класса.

   (b) `fit(self, X, y)`

   Описание параметров:

   - `X` — обучающая выборка объектов
   - `y` — ответы объектов на обучающей выборке

   Метод производит обучение алгоритма с учётом стратегии указанной в параметре класса `strategy`.

   (c) `find_kneighbors(self, X, return_distance)`

   Описание параметров:

   - `X` — выборка объектов
   - `return_distance` — переменная типа `bool`

   Метод возвращает `tuple` из двух `numpy array` размера (`X.shape[0], k`). `[i, j]` элемент первого массива должен быть равен индексу `j`-ого ближайшего соседа из обучающей выборки для объекта с индексом `i`. `[i, j]` элемент второго массива должен быть равен расстоянию от `i`-го объекта, до его `j`-го ближайшего соседа.

   Если `return_distance=False`, возвращается только первый из указанных массивов. Метод должен использовать стратегию поиска указанную в параметре класса `strategy`.

   (d) `predict(self, X)`

   Описание параметров:

   - `X` — тестовая выборка объектов

   Метод должен вернуть одномерный `numpy array` размера `X.shape[0]`, состоящий из предсказаний алгоритма (меток классов) для объектов тестовой выборки.

2. Модуль `cross_validation` с реализациями функций для применения кросс-валидации:

   (a) `kfold(n, n_folds)`

   Описание параметров:

   - `n` — количество объектов в выборке
   - `n_folds` — количество фолдов на которые делится выборка

   Функция реализует генерацию индексов обучающей и валидационной выборки для кросс-валидации с `n_folds` фолдами. Функция возвращает список длины `n_folds`, каждый элемент списка — кортеж из двух одномерных `numpy array`. Первый массив содержит индексы обучающей подвыборки, а второй валидационной.

   (b) `knn_cross_val_score(X, y, k_list, score, cv, weights, metric, strategy, test_block_size)`

   Описание параметров:

   - `X` — обучающая выборка
   - `y` — ответы объектов на обучающей выборке
   - `k_list` — список из проверяемых значений для числа ближайших соседей, числа в списке упорядочены по возрастанию
   - `score` — название метрики, по которой оценивается качество алгоритма. Обязательно должна быть реализована метрика 'accuracy' (доля правильно предсказанных ответов)
   - `cv` — список из кортежей, содержащих индексы обучающей и валидационной выборки — выход функций `kfold` или `stratified_kfold`. Если параметр не задан, необходимо внутри функции реализовать генерацию индексов с помощью функции `kfold`
   - `metric` — название метрики, по которой считается расстояние между объектами
   - `weights` — переменная типа `bool`. Значение `True` означает, что нужно использовать взвешенный метод k ближайших соседей
   - `strategy` — алгоритм поиска ближайших соседей
   - `test_block_size` — размер блока данных для тестовой выборки
     Замечание. Параметры `metric`, `weights`, `strategy`, `test_block_size` соответствуют аналогичным параметрам для класса `KNN_classifier`.

   Функция для измерения метрики качества `score` алгоритма ближайших соседей, реализованного через класс `KNN_classifier` с параметрами `metric` и `weights` на кросс-валидации, заданной списком индексов `cv` для обучающей выборки `X`, ответов на ней `y`. Оценку качества метода ближайших соседей нужно рассчитать для нескольких значений k: $[k_1, \ldots, k_n]$, $k_1 < k_2 < \cdots < k_n$, заданных в `k_list`. Сложность алгоритма для одного объекта из валидационной выборки должна иметь порядок $O(k_n)$.

   Функция должна возвращать словарь, где ключами являются значения k, а элементами — `numpy array` размера `len(cv)` с качеством на каждом фолде.

   Замечание. Для тестирования алгоритма удобно использовать функцию `cross_val_score` из библиотеки `scikit-learn`.