AIMasters

Практическое задание 1
Классификация изображений цифр метрическими методами

курс «Машинное обучение 1», 2022

Спецификация

Замечание Далее под выборкой объектов будем понимать `np.ndarray` размера $N \times D$, под ответами для объектов выборки будем понимать `np.ndarray` размера N, где N — количество объектов в выборке, D — размер признакового пространства.

Среди предоставленных файлов должны быть следующие модули и функции в них:

1. Модуль `knn.distances` с реализацией функции для вычисления расстояния:

   (a) `euclidean_distance(X, Y)`

   Описание параметров:

   - `X` — `np.ndarray` размера $N \times D$;
   - `Y` — `np.ndarray` размера $M \times D$.

   Функция возвращает `np.ndarray` размера $N \times M$, каждый элемент которого — евклидово расстояние между соответствующей парой векторов из массивов X и Y.

   (b) `cosine_distance(X, Y)`

   Описание параметров:

   - `X` — `np.ndarray` размера $N \times D$;
   - `Y` — `np.ndarray` размера $M \times D$.

   Функция возвращает `np.ndarray` размера $N \times M$, каждый элемент которого — косинусное расстояние между соответствующей парой векторов из массивов X и Y.

2. Модуль `knn.nearest_neighbors`, содержащий собственную реализацию поиска ближайших соседей.

   Класс `NearestNeighborsFinder`

   Описание методов:

   (a) `__init__(self, n_neighbors, metric="euclidean")` — конструктор (инициализатор) класса.

   - `n_neighbors` — число K ближайших соседей в алгоритме ближайших соседей;
   - `metric` — название метрики, по которой считается расстояние между объектами. Может принимать следующие значения:
     - `'euclidean'` — евклидова метрика;
     - `'cosine'` — косинусная метрика.

   (b) `fit(self, X, y=None)`

   - `X` — обучающая выборка размера N;
   - `y` — ничего не означающий аргумент, нужен для поддержания единого интерфейса (см., например, `sklearn.neighbors.NearestNeighbors.fit`).

   Метод производит обучение модели (т.к. мы используем стратегию полного перебора это означает, что алгоритм просто запоминает всю обучающую выборку), возвращает `self`.

   (c) `kneighbors(self, X, return_distance=False)`

   - `X` — тестовая выборка размера M;
   - `return_distance` — булев флаг, нужно ли вернуть расстояния для объектов.

   Метод производит поиск ближайших соседей. В случае `return_distance=True` возвращает кортеж (`distances`, `indices`) из двух `np.ndarray` размера $M \times K$, где

   - `distances[i, j]` — расстояние от `i`-го объекта, до его `j`-го ближайшего соседа;
   - `indices[i, j]` — индекс ближайшего соседа из обучающей выборки до объекта с индексом `i`.

   Если `return_distance=False`, возвращается только второй из указанных массивов.

3. Модуль `knn.classification`, содержащий собственную реализацию классификатора на основе метода ближайших соседей.

   Класс `KNNClassifier`

   Описание методов:

   (a) `__init__(self, n_neighbors, algorithm='my_own', metric='euclidean', weights='uniform')` — конструктор (инициализатор) класса.

   - `n_neighbors` — число ближайших соседей в алгоритме ближайших соседей K;
   - `algorithm` — алгоритм поиска ближайших соседей. Может принимать следующие значения:
     - `'my_own'` — использование `knn.nearest_neighbors.NearestNeighborsFinder()`;
     - `'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')`.
   - `metric` — название метрики, по которой считается расстояние между объектами;
   - `weights` — алгоритм взвешивания. Может принимать следующие значения:
     - `'uniform'` — обычный метод ближайшего соседа, где вес каждого объекта равен 1;
     - `'distance'` — взвешенный метод ближайшего соседа, где вес каждого объекта равен

     $$weight = 1/(distance + \varepsilon)$$

     где $\varepsilon = 10^{-5}$ (см. `KNNClassifier.EPS`).

   (b) `fit(self, X, y=None)`

   - `X` — обучающая выборка размера N;
   - `y` — ответы объектов на обучающей выборке.

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

   (c) `kneighbors(self, X, return_distance=False)`

   - `X` — тестовая выборка размера M;
   - `return_distance` — булев флаг, нужно ли вернуть расстояния для объектов.

   Логика метода аналогичена методу `knn.nearest_neighbors.NearestNeighborsFinder.kneighbors`.

   (d) `predict(self, X)`

   - `X` — тестовая выборка размера M.

   Метод должен вернуть одномерный `np.ndarray` размера M, состоящий из предсказаний алгоритма (меток классов) для объектов тестовой выборки.

   (e) `_predict_precomputed(self, indices, distances)`

   - `indices` — массив индексов, `np.ndarray` размера $M \times K$;
   - `distances` — массив расстояний, `np.ndarray` размера $M \times K$;

   Вспомогательный метод, который должен вернуть одномерный `np.ndarray` размера M, состоящий из предсказаний алгоритма (меток классов) для объектов тестовой выборки по заданным массивам расстояний и индексов.

   Класс `BatchedMixin` — миксин для реализации метода ближайшего соседа с использованием батчей.

   Описание методов:

   (a) `kneighbors(self, X, return_distance=False)`

   - `X` — тестовая выборка размера M;
   - `return_distance` — булев флаг, нужно ли вернуть расстояния для объектов.

   Логика метода аналогичена методу `knn.nearest_neighbors.NearestNeighborsFinder.kneighbors`.

   (b) `set_batch_size(self, batch_size)`

   - `batch_size` — размер батча.

   Метод устанавливает используемый размер батча.

   Класс `BatchedKNNClassifier` — реализация классификатора на основе метода ближайших соседей с использованием батчей. Класс является наследником классов `BatchedMixin` и `KNNClassifier`, поэтому содержит все методы из этих классов.

4. Модуль `knn.model_selection` с реализациями функций для применения кросс-валидации:

   (a) `knn_cross_val_score(X, y, k_list, scoring, cv=None, **kwargs)`

   Описание параметров:

   - `X` — обучающая выборка;
   - `y` — ответы объектов на обучающей выборке;
   - `k_list` — список из проверяемых значений для числа ближайших соседей;
   - `scoring` — название метрики, по которой оценивается качество алгоритма. Обязательно должна быть реализована метрика 'accuracy' (доля правильно предсказанных ответов);
   - `cv` — класс, реализующий интерфейс `sklearn.model_selection.BaseCrossValidator` для кросс-валидации, например, класс `sklearn.model_selection.KFold`.
   - `**kwargs` — параметры конструктора класса `knn.classifier.KNNClassifier`.

   Функция должна возвращать словарь, где ключами являются значения K из `k_list`, а элементами — `np.ndarray` размера `len(cv)` с качеством на каждом фолде.