Отчет по первому практическому заданию: метрические алгоритмы классификации

Выполнил студент 317 группы, Демьянов Иван

12 октября 2021 г.

Содержание

1 Введение

В данном практическом задании необходимо было реализовать три модуля, при помощи которых осуществляется решение задачи классификации методом k-ближайших соседей с разными параметрами.

Далее нужно было провести эксперименты, используя различные параметры, на датасете MNIST и составить отчёт.

2 Эксперименты

В этой работе планировалось провести 6 экспериментов. В начале каждого эксперимента будет присутствовать его краткое условие (постановка эксперимента). Далее будут описаны результаты эксперимента и вывод.

2.1 Эксперимент №1

В этом эксперименте необходимо было исследовать, какой алгоритм поиска ближайших соседей будет быстрее работать в различных ситуациях. Под различными ситуациями понимался выбор подмножества признаков размера 10, 20, 100, по которому будет считаться расстояние до данного объекта. Число соседей в данном эксперименте составляло 5.

В таблице 1 представлены результаты скорости работы алгоритмов на подмножествах признаков размера 10, 20, 100, соответственно.

стратегия \кол-во признаков	10	20	100
'my_own'	282.56	316.81	346.03
'brute'	14.60	15.79	17.03
'kd_tree'	2.62	16.89	154.25
'ball_tree'	12.21	41.74	155.46

Таблица 1: время (в сеукндах) работы алгоритмов

Вывод: по времени работы алгоритма 1-е место занимает стратегия 'brute'. В дальнейших экспериментах для экономии времени будет использоваться именно она.

2.2 Эксперимент №2

Во втором эксперименте требовалось оценить по кросс-валидации с 3 фолдами точность и время работы k ближайших соседей в зависимости от следующих факторов:

- (a) k от 1 до 10 (только влияние на точность).
- (b) Используется евклидова или косинусная метрика.

Результаты части (a) приведены в таблице 2. В итоге, самым точным оказался алгоритм, который находит три ближайших соседа для каждого объекта. Также алгоритм имеет неплохую стабильность точности при $k = 5$, поэтому далее буду предпочтительней использовать стратегию 'brute' с этими гиперпараметрами.

№ фолда \кол-во соседей	1	2	3	4	5	6	7	8	9	10
1	0.9689	0.9612	0.9695	0.9670	0.9681	0.9655	0.9652	0.9641	0.9637	0.9624
2	0.9667	0.9599	0.9682	0.9667	0.9673	0.9649	0.9650	0.9639	0.9632	0.9622
3	0.9667	0.9612	0.9671	0.9671	0.9672	0.9657	0.9656	0.9652	0.9645	0.9637

Таблица 2: точность алгоритма 'brute'

В части (b) считались точности стратегии 'brute', взятой с разными метриками, при $k = 5$. В результате косинусная метрика была точнее на всех 3-х фолдах(таблица 3), однако эта точность стоила больше времени (34.93 секунды, у евклидовой - 28.81).

№ фолда \метрика	'euclidean'	'cosine'
1	0.9681	0.9727
2	0.9673	0.9701
3	0.9672	0.9716

Таблица 3: точность алгоритма 'brute' при $k = 5$

2.3 Эксперимент №3

В эксперименте №3 предлагалось сравнить взвешенный метод k-ближайших соседей, где голос объекта равен $1/(distance + eps)$, где $eps = 10^{-5}$, с методом без весов при тех же фолдах и параметрах. В качестве параметров я взял евклидову метрику и $k = 5$.

№ фолда \взвешенный	-	+
1	0.9681	0.9692
2	0.9673	0.9682
3	0.9672	0.9687

Таблица 4: точность алгоритма для эксперимента №3

Результаты этого эксперимента приведены в таблице 4. Как и ожидалось, взвешенный метод является точнее, однако требует больше времени (для эксперимента при описанных параметрах дольше на полторы секунды).

2.4 Эксперимент №4

В эксперименте №4 требовалось применить лучший по итогам предыдущих экспериментов алгоритм (strategy='brute', metric='cosine', $k = 3$) к исходной обучающей и тестовой выборке. Получились следующие точности:

- train predict accuracy: 1.0
- test predict accuracy: 0.9742

На обучающей выборке точность и вид матрицы ошибок (confusion matrix) очевидны. Получились следующие точности по кросс-валидации с 3 фолдами:

1. 0.9749
2. 0.9725
3. 0.9717

Из-за очевидности точности на обучающей выборки, нет смысла сравнить её с точностью по кросс-валидации с 3 фолдами. Точность на тестовой выборке приблизительно такая же как по кросс-валидации с 3 фолдами.

Точность лучших алгоритмов на данной выборке. На сайте MNIST можно найти список лучших результатов, достигнутых алгоритмами на этом наборе данных. Так, ансамбль из 35 сверточных нейронных сетей в 2012 году сумел получить всего 0.23% ошибок на наборе данных, что является очень хорошим результатом, вполне сравнимым с человеком. Параметры ансамбля: classifier - committee of 35 conv.net, 1-20-P-40-P150-10 (elastic distortions).

На обучающей выборке вид матрицы ошибок (confusion matrix) очевиден, поэтому целесообразнее перейти к исследованию матрицы ошибок для тестовой выборки (Рис. 1). По матрице ошибок можно определить, что наибольшее количество ошибок происходит при предсказании цифры '4'. 24 раза взвешенный алгоритм 'brute' при трёх ближайших соседях и косинусной метрике ошибочно предсказывал вместо неё цифру '9'. На (Рис. 2) продемонстрированы эти изображения. По ним можно увидеть общую черту: верхняя часть этих изображений имеет округлую форму, что для девяток более свойственно, чем для четверок.

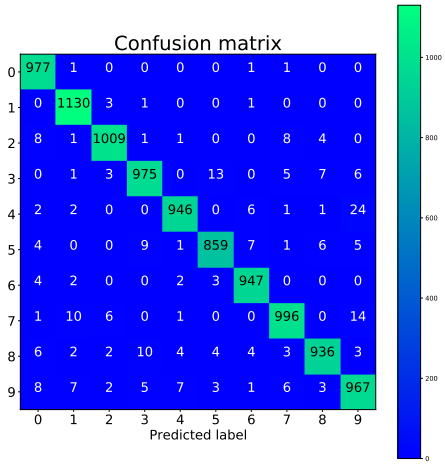

Рис. 1: Матрица ошибок для тестовой выборки

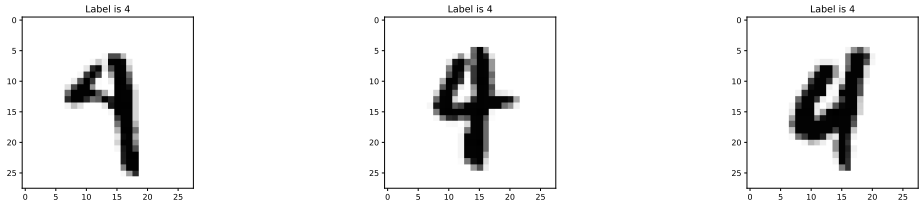

Рис. 2: 'Ложные девятки'

2.5 Эксперимент №5

В этом эксперименте требовалось произвести аугментацию тестовой выборки с помощью поворотов, смещений и применений гауссовского фильтра. Были протестированы по кросс-валидации с 3 фолдами следующие параметры преобразований:

- (a) Величина поворота: 5, 10, 15 (в каждую из двух сторон)
- (b) Величина смещения: 1, 2, 3 пикселя (по обоим из двух размерностей)
- (c) Дисперсия фильтра Гаусса: 0.5, 1, 1.5

В ходе эксперимента были выявлены самые удачные одиночные параметры аугментации:

- (a) Величина поворота: 5, поворот вправо
    1. 0.9981
    2. 0.9966
    3. 0.9943
- (b) Величина смещения: 1 (по обоим из двух размерностей)
    1. 0.9699
    2. 0.9781
    3. 0.971
- (c) Дисперсия фильтра Гаусса: 0.5

Дисперсия фильтра Гаусса: 0.5 дала точный результат на всех фолдах! Видимо, она устраняет проблему, описанную в эксперименте №4. Однако этот результат обусловлен еще и тем, что новые объекты с данным параметром очень похожи на старые, поэтому при выборе, вероятно, что в

числе ближайших соседей окажутся те же самые соседи и их параметризованные копии. Это может привести к переобучению при малом k.

Далее решил посмотреть точность кросс-валидации с 3 фолдами модели, которая обучится на выборке, состоящей из 60000 обычных объектов и 60000 объектов, на которые наложены лучшие параметры из прошлых опытов этого эксперимента. Получил при $k = 5$ следующую точность на фолдах: 0.99885 , 0.997625, 0.996975, при $k = 3$ – точное предсказание. Точность довольно хорошая, поэтому настало время испытать эту модель на тестовой выборке. В итоге испытания точность оказалась 0.9751, что хуже точности на фолдах, но точность предсказания, все же, немного возросла на тестовой выборке (в 4м эксперименте она составляла 0.9742).

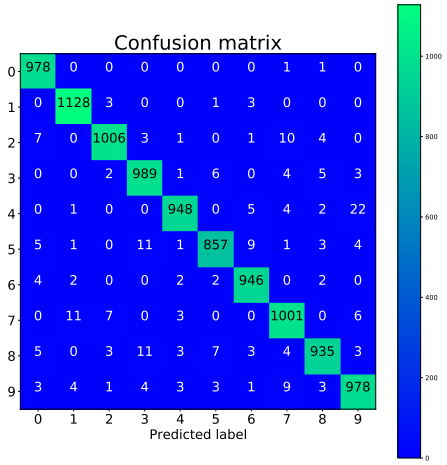

Рис. 3: Матрица ошибок при обновлённой обучающей выборке

В заключительном опыте данного эксперимента было решено обучить модель на выборке, состоящей из 60000 обычных объектов и 60000 объектов, на каждую треть которых наложен один лучший параметр. Точность предсказания улучшилась после такого предсказания и составила 0.9766. По виду матрицы ошибок для тестовой выборки при вышеописанной обучающей (Рис. 3) можно утверждать, что аугментация немного улучшила точность и убрала некоторые ошибочные раннее предсказания.

2.6 Эксперимент №6

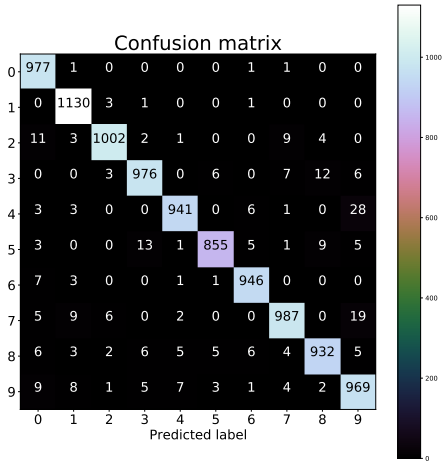

Рис. 4: Матрица ошибок при обновлённой тестовой выборке

В рамках данного эксперимента подразумевается обучение модели на оригинальном датасете, преобразование объектов тестовой выборки при помощи параметров из предыдущего эксперимен-

та, применение модели к преобразованным копиям изображения из тестовой выборки и получение результата путем голосования среди преобразованных объектов.

В первом опыте этого эксперимента было решено сделать 3 копии тестовой выборки и преобразовать каждый при помощи одного лучшего параметра из предыдущего эксперимента. Результат первого опыта: 0.9715. Точность немного хуже, чем при стандартном обучении. Матрица ошибок (Рис. 4).

Второй опыт аналогичен первому, только выбираются параметры, которые максимально из предложенных меняют копии. Результат второго опыта: 0.9258.

Качественное сравнение эксперимента №5 и эксперимента №6: очевидно, что в эксперименте №5 в модуле nearest_neighbors.py используется больше памяти для хранения обучающей выборки. Однако этот подход в ходе экспериментов точнее, чем менее затратный.

3 Общие выводы из работы

По итогам проведенных экспериментов, самым оптимальным в использовании является взвешенный метод 'brute' с косинусной метрикой и $k = 5$. В экспериментах 5-6 были предложены две разные стратегии аугментации данных, которые показали неплохую точность. Аугментация данных используется при устранении шумов у модели, что дает более точный прогноз.