Московский Государственный Университет имени
М.В.Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

Практикум осень 2022
Метрические алгоритмы классификации
Отчет KNN

Авдеев Роман Артемович
Группа 317

Октябрь, 2022

Оглавление

1 Введение

Классификация - это самая распространенная задача машинного обучения. Примерами задач классификации являются распознавание объектов, генерация текстов, подбор тематики текстов, идентификация объектов на изображениях, распознавание речи, машинный перевод и т.д.
Поэтому в рамках изучения алгоритмов машинного обучения была выполнена работа по реализации метода классификации KNN (K ближайших соседей, K-Nearest Neighbors). Это один из самых простых алгоритмов классификации, также иногда используемый в задачах регрессии.Выполненная работа посвящена исследованию возможностей и особенностей KNN, а также изучечению технологии работы с изображениями.

Задание состояло из двух частей:

1. Реализация метода классификации KNN на языке python:
   (a) классификатор
   (b) кросс-валидация
   (c) метрики
2. Проведение экспериментов на выданном датасете:
   (a) поиск оптимального алогритма поиска
   (b) поиск его лучшей конфигурации
   (c) аугментация данной выборки
   (d) оценка результатов на расширенном датасете

2 Пояснения к задаче

Некоторые формулы, используемые в решении:

- Евклидова метрика: $p(x, y) = ||x - y||^2 = \sum_{i=1}^{N}(x_i - y_i)^2$
- Косинусная метрика: $p(x, y) = 1 - \frac{(x,y)}{||x||||y||}$

Понятие взвешенного и невзвешенного алгоритмов:

Оригинальный алгоритм KNN:

$$g_c(z) = \sum_{i=1}^{k} [y_i^z = c] \ (1) \qquad y_p(z) = \arg\max_c g_c(z)$$

У оригинального алгоритма KNN есть один большой недостаток: он никак не учитывает расстояния до соседних объектов. Необходимо увеличивать вклад близких объектов и уменьшать вклад далёких. Можно заметить, что все индикаторы в формуле (1) учитываются в сумме с одинаковыми коэффициентами. Поэтому можно назначить этим индикаторам веса, которые тем больше, чем ближе объект к целевому. Таким образом, получаем взвешенный алгоритм KNN (weighted KNN):

$$y_p(z) = \arg\max_c \sum_{i=1}^{k} w_i[y_i^z = c] \text{ , где } w_i = \frac{1}{p(x_i,z)+\epsilon} \text{ . Полагаем } \epsilon = 10^{-5}$$

3 Список экспериментов

Все эксперименты проводились на датасете MNIST (Modified National Institute of Standards and Technology) - объёмная база данных образцов рукописного написания цифр. Использовалась версия MNIST-784 - имеет 784 признака. Данный датасет был разбит на обучающую (первые 60тыс. объектов) и тестовую (последние 10тыс. объектов) выборки.

3.1 Эксперимент 1

Задача данного эксперимента заключалась в поиске оптимального по времени алгоритма поиска 5 ближайших соседей для каждого объекта тестовой выборки по евклидовой метрике. Измерения проводились на 10, 20, 100 случайно выбранных признаках.

Алгоритмы поиска ближайших соседей:

- 'ball-tree'
- 'kd-tree'
- 'brute'
- 'my-own'

'ball-tree': стратегия, при использовании которой образовывается структура, состоящая из вложенных шаров. Каждая точка относится только

к одному шару. Все точки разбиты на непересекающиеся кластеры-шары. Эта структура образует двоичное дерево, в вершинах которого будут лежать центроиды, а кроной которого будут листья, содержащие точки пространства.

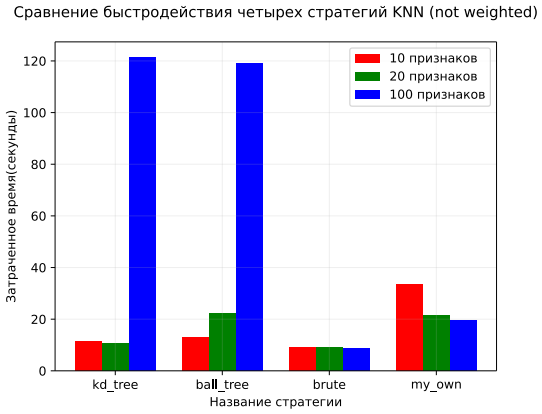
'kd-tree': kd-tree (k-мерное дерево) представляет собой иерархическое двоичное дерево. Данный алгоритм перестраивает весь набор данных в двоичную древовидную структуру, чтобы при предоставлении тестовых данных он выдавал результат путем обхода дерева, что занимает меньше времени, чем грубый поиск.

'brute': brute-force (грубая сила) - вычисление расстояний методом перебора между всеми парами точек в наборе данных. Алгоритм обеспечивает наиболее простую реализацию поиска ближайших соседей.

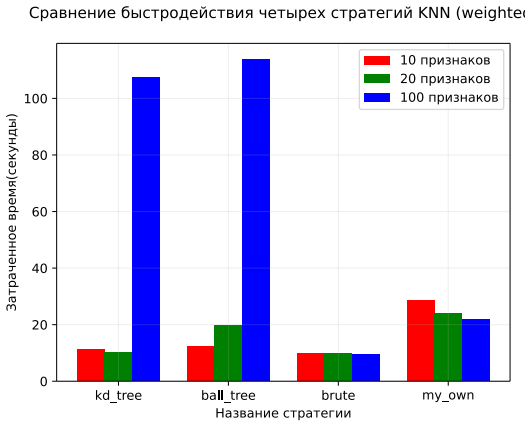
'my-own': собственная реализация алгоритма поиска ближайших соседей на основе евклидовой/косинусной метрики.

Результаты эксперимента:

Сначала были рассмотрены невзвешенные алгоритмы.
Как мы видим из графика, 'brute' оказался существенно быстрее конкурентов.



Затем для сравнения были взяты взвешенные алгоритмы с такими же параметрами:



И снова 'brute' получается быстрее на каждом количестве признаков. Значит, следующие эксперменты будем проводить именно на нем.

3.1.1 Выводы

Подумаем, чем может быть вызвана такая разница во времени: Производительность может сильно зависеть от характеристик данных. Например, равномерно ли распределены точки данных на каждой из картинок, каким образом они расположены. Также я узнал, что, например, kd-деревья не очень хорошо масштабируются с высокой размерностью. Так что ответ, похоже, связан с размерностью, точнее с 'проклятием размерности', т.е. с разреженностью данных при больших размерностях.

3.2 Эксперимент 2

Второй эксперимент был направлен на оценку точности и времени работы классификатора в зависимости от количества соседей и выбранной метрики; необходимо оценить по кросс-валидации с 3 фолдами точность и время в зависимости от k $k = 1, ..., 10$ и от метрики.

Введем понятие точности: accuracy — доля правильных ответов алгоритма.

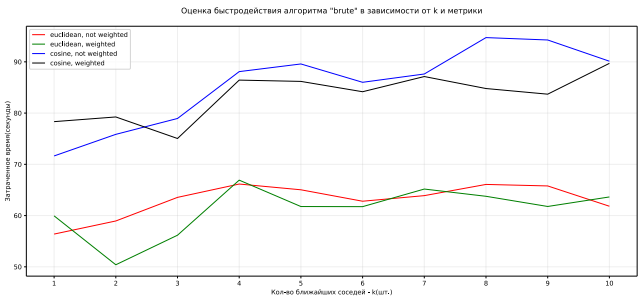
	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Здесь $\hat{y}$ — это ответ алгоритма на объекте, а y — истинная метка класса на этом объекте. Таким образом, ошибки классификации бывают двух видов: False Negative (FN) и False Positive (FP).

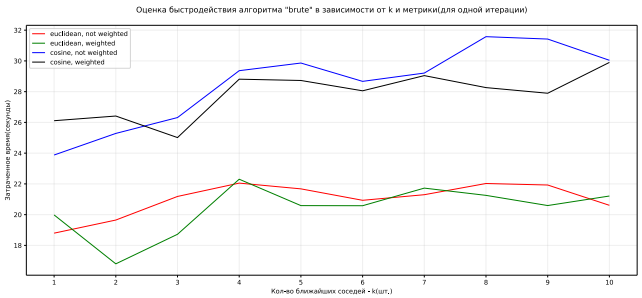
$accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

3.2.1 Оценка времени работы

Итак, в этом эксперименте было замерено время работы каждой конфигурации 'brute':



А время одной итерации:



Получили:
Средняя разница времени работы косинусной и евклидовой метрик при взвешенном алгоритме: 7.447810300191245 (сек.)

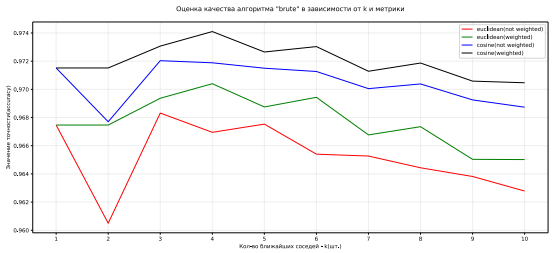
Средняя разница времени работы косинусной и евклидовой метрик при невзвешенном алгоритме: 7.547190340360006 (сек.)

Неожиданный результат, учитывая, что косинусное расстояние обычно работает быстрее. Но, полагаю, есть объяснение, почему в этом эксперименте получилось по-другому. Я думаю, дело в том, что размерность 28x28 не дает нам истинное быстродействие косинусной меры, которое раскроется на больших размерностях. В текущей ситуации если у нас есть два вектора размерности n, то при вычислении евклидового расстояния, мы сделаем n вычитаний, n возведений в квадрат, а затем вычислим корень. А при косинусной метрике мы сделаем n умножений, $(n-1)$ сложений (для скалярного произведения), затем **дважды** посчитаем норму: n возведений в квадрат, $(n-1)$ сложение и 1 корень. И затем еще одно деление. Наверное, именно поэтому в нашем эксперименте конкретно на этих данных, в данной конфигурации

модели и при текущей реализации метрик мы получаем, что косинусная метрика работает дольше евклидовой.

3.2.2 Оценка точности

В этой части Эксперимента 2 была измерена точность взвешенных и невзвешенных алогритмов на разных метриках:



Заметим, что лучшее качество достигается при $k = 3$ для невзвешенных алгоритмов на косинусной и евклидовой метриках, и при $k = 4$ для взвешенных алгоритмов на косинусной и евклидовой метриках.

Сравнив косинусную и евклидову метрики, получаем, что косинусная метрика дает прирост качества:

- При взвешенном алгоритме - 0.0043 (в среднем)
- При невзвешенном алгоритме - 0.0052 (в среднем)

Видим из графика, что при $k = 2$ невзвешенные алгоритмы сильно теряют в качестве. При $k = 3$ и $k = 4$ достигаются лучшие значения, далее качество медленно убывает с ростом количества соседей(k).

3.3 Эксперимент 3

В эксперименте 3 была поставлена задача сравнить взвешенный и невзвешенный алгоритмы при таких же фолдах и параметрах, как в Эксперименте 2.

В предыдущем эксперименте мы уже проводили анализ взвешенных/невзвешенных алгоритмов при использовании каждой из метрик.
Косинусная метрика показала более высокое качество, поэтому в данном эксперименте подробнее рассмотрем алгоритм с применением именно этой метрики. Оценка проводится по кросс-валидации на 3 фолдах.

Средняя разница времени работы косинусной метрики при взвешенном и невзвешенном алгоритмах: 0.738 (сек.). Можно считать, что алгоритмы

работают за одинаковое время.

Средняя разница качества косинусной метрики при взвешенном и невзвешенном алгоритмах: 0.00158 - взвешенный дает лучшее качество.

3.4 Эксперимент 4

В эксперименте 4 к обучающей и тестовой выборке был применен оптимальный алгоритм, полученный в Экспериментах 1-3; проведено сравнение с точностью по кросс-валидации; выполнен анализ ошибок.

Лучший алгоритм: взвешенный brute, косинусое расстояние, $k = 4$ ближайших соседа.

Модель в такой конфигурации показала точность: 0.9752

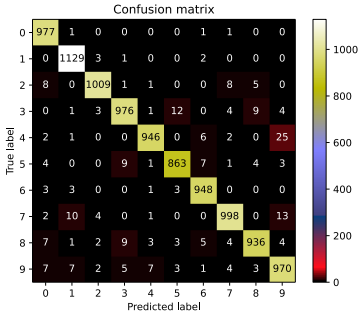
Точность по кросс-валидации для 3 фолдов: 0.9741

Лучшее значение точности, которое мне удалось найти для данного датасета, составляет 0.987. Оно достигается при использовании нейронной сети.

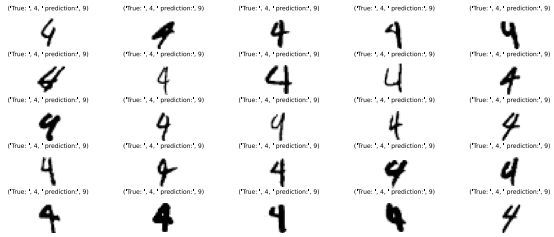
Также при применении классификатора RandomForest видел значение немного хуже полученного в данном эксперименте: $accuracy = 0.9705$

3.4.1 Анализ ошибок:

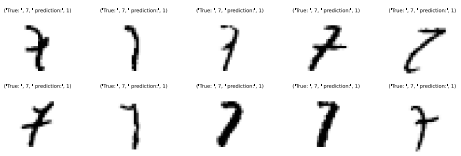
Построена confusion matrix:



Как мы видим, самой частой ошибкой является неверное предсказание цифры 4, алгоритм ее путает с цифрой 9:



Думаю, ошибки здесь вызваны тем, что в обучающей выборке часто встречалось написание 9 без закругления внизу(просто вертикальная черта), тогда если у 4 нет четкого разделения ее верхних линий, то она становится похожа на девятку. Также здесь на ухудшение распознаваемости влияет толщина линий.
Посмотрим на ошибки с цифрами 7 и 1:



Считаю, что довольно большое количество ошибок в этом случае (10 ошибок) получается из-за отсутствия "черточки" у семерки, что делает ее похожей на единицу. Также, как мы видим, что у цифр, написанных более толстым шрифтом, эта "черточка" почти сливается с самой цифрой, что тоже делает ее близкой к единице.

3.5 Эксперимент 5

В данном эксперименте была проведена аугментация обучающей выборки, а также подобраны по кросс-валидации с 3 фолдами параметры преобразований. Для этого использовались библиотеки scipy.ndimage, skimage и cv2.

Аугментация (augmentation, увеличение) данных для обучения. Аугментацию данных применяют в машинном обучении. Под аугментацией понимается увеличение выборки данных для обучения через модификацию этих данных и создание на этой основе дополнительных.

Рассмотренный способы аугментации:

- Поворот изображения (величина поворота 5, 10, 15 в каждую из двух сторон

- Смещение изображения (1, 2, 3 пикселя по каждой из двух размерностей)
- Применение фильтра Гаусса (дисперсия 0.5, 1, 1.5)
- Морфологические операции
  - эрозия
  - дилатация
  - открытие
  - закрытие

3.5.1 Поворот изображения

По кросс-валидации измерил качество при поворотах обучающей выборки на -15, -10, -5, 5, 10, 15. Ниже представлены усредненные значения точности:

Table 1: Повороты изображения

-15	-10	-5	5	10	15
0.9506	0.9492	0.9501	0.9512	0.9543	0.9533

3.5.2 Смещение изображения

Смещение изображение было доступно в 49 вариантах: от $(-3,-3),(-3,-2)$ ... до $(3,2),(3,3)$
Выпишем смещения с наилучшими показателями точности:

Table 2: Смещения изображения

$(0,-1)$	$(0,-2)$	$(0,2)$	$(-2,1)$	$(-2,0)$	$(0,-3)$
0.9460	0.9459	0.9448	0.9440	0.9432	0.9432

3.5.3 Применение фильтра Гаусса

Значения точности при использовании каждой из дисперсий:

Table 3: Фильтр Гаусса

0.5	1	1.5
0.9759	0.9759	0.9760

3.5.4 Эрозия

Усредненная по фолдам точность при использовании эрозии: 0.9498

3.5.5 Дилатация

Усредненная по фолдам точность при использовании дилатации: 0.9524

3.5.6 Открытие

Усредненная по фолдам точность при использовании открытия: 0.97599

3.5.7 Закрытие

Усредненная по фолдам точность при использовании закрытия: 0.97596

3.5.8 Вывод

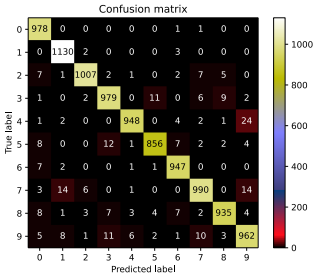
После проведенных преобразований и измерения качества по кросс-валидации видим, что наиболее оптимальной комбинацией является:

- поворот на 10
- смещение на $(0, -1)$
- фильтр Гаусса с дисперсией 1.5
- открытие
- закрытие

Выберем из этого списка пробразования, показавшие прирост точности: фильтр Гаусса, открытие, закрытие.

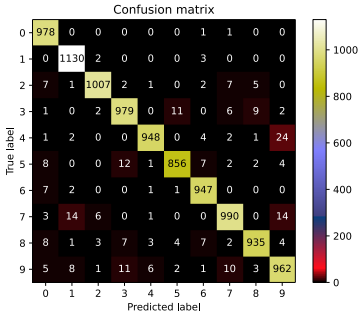
Аугментировали при помощи указанных преобразований, обучили модель.

После применения открытия получаем $accuracy = 0.9732$
Confusion matrix:



Видим, что в некоторых случаях ошибок стало меньше (ошибки для 4и9, 7и1, 5и3), но в некоторых наоборот увеличилось(ошибки 9и7, 3и5).

После применения закрытия получаем точно такую же $accuracy = 0.9732$. Видим, что как и при кросс-валидации, открытие и закрытие дали одинаковые результаты.
Confusion matrix:



После применения фильтра Гаусса с дисперсией 1.5: $accuracy = 0.976$

Значит, аугментация обучающей выборки немного (около 0.0008), но повысила точность предсказания. Это связано с тем, что мы улучшаем способность алгоритма воспринимать нечетко написанные цифры, даем большее разнообразие искажений, которые модель должна корректно различать.

3.6 Эксперимент 6

В данном эксперименте была проведена аугментация тестовой выборки, проверены те же преобраования при тех же условиях, что и в предыдущем эксперименте.

Проведя те же измерения, выберем тоже открытие, закрытие и фильтр Гаусса, тк они показывают лучшую точность (см. Table 3 и пункты 3.5.5, 3.5.6).

После применения фильтра Гаусса с дисперсией 1.5 $accuracy = 0.9665$

После применения открытия к тестовой выборке $accuracy = 0.915$.

То есть видим, что при аугментации тестовой выборки качество падает. Думаю, это связано с тем, что количество тестовых измененных данных увеличивается, а значит растет число "мало знакомых" или "незнакомых" для алгоритма цифр: размытых, растушеванных, заушмленных или наоборот "слишком чистых". Алгоритм обучался на одинаковых по своей специфике данных, поэтому различать и правильно относить к классам нетипичные написания цифр ему сложнее.

4 Общие выводы

В ходе работы над данным заданием был реализован свой способ нахождения ближайших соседей, использованы существующие решения для сравнения времени и точности моделей. Была написана и оттестирована кросс-валидации с разными параметрами. Проведен анализ взвешенных/невзвешенных алгоритмов, а также использованных метрик.

Получили, что оптимальным алгоритмом является поиск стратегией 'brute' с использованием весов на косинусной метрике с числом соседей $k = 4$.

Изучены методы работы с изображениями в библиотеках scipy, skimage, cv2; с помощью этих преобразований проведена аугментация обучающей и тестовой выборок. Выяснили, что в нашем случае наибольших прирост точности дали Фильтр Гаусса и морфологические операции(открытие, закрытие).