Задание 1. Метрические алгоритмы классификации

Янаков Дмитрий, 317 группа

11 октября, 2021

1 Введение

В этом задании будет изучен метрический алгоритм классификации — «k ближайших соседей», а именно: проведены эксперименты на датасете изображений цифр MNIST, объём которого составляет 70 тыс. объектов. Для начала будет выбран самый быстрый алгоритм, затем, проведя более глубокое исследование с применением кросс-валидации, будут подобраны параметры для достижения наибольшей точности. И под конец мы рассмотрим влияние аугментации данных на работу алгоритма.

2 Пояснения к заданию

В приведённых ниже экспериментах будут использованы следующие метрики для алгоритмов классификации:

1. Евклидова: $\rho(x, y) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$
2. Косинусная: $\rho(x, y) = 1 - \frac{(x,y)}{||x||*||y||}$

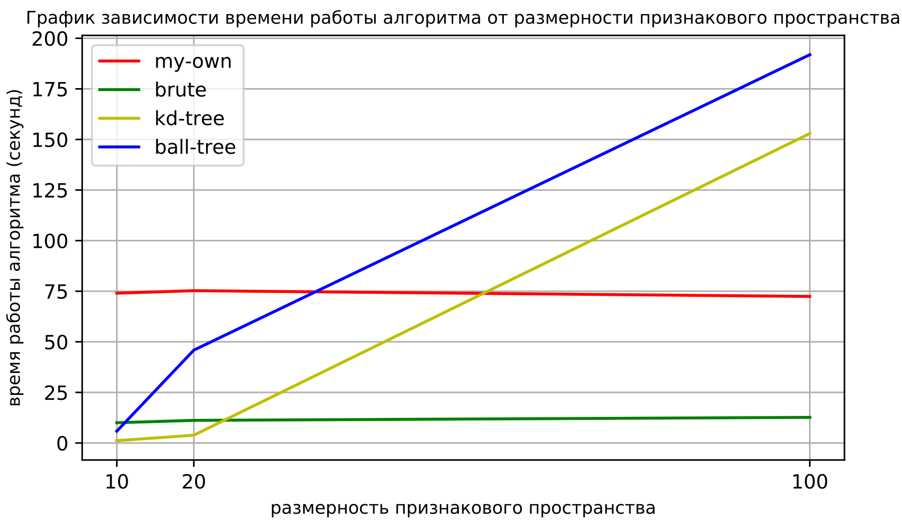
где d - размерность признакового пространства.

3 Эксперименты

В обучающей выборке будут первые **60** тыс. изображений, в тестовой — оставшиеся **10** тыс.

3.1 Эксперимент 1

Для начала следует выяснить, какой алгоритм классификации («brute», «kd-tree» , «ball-tree», «my-own») работает быстрее в различных ситуациях. Рассмотрим k=5 и выберем следующие размеронсти признакового пространства: 10, 20, 100. Метрику будем считать евклидовой.



Как видно, самым оптимальным по времени является алгоритм «brute». Однако, если рассматривать маленькую размерность признакового пространства, алгоритмы «kd-tree» и «ball-tree» работают быстрее, причем «kd-tree» является самым быстрым. С увеличением размерности, они показывают худший результат. Алгоритм «my-own» является также оптимальным, однако проигрывает «brute» по скорости.

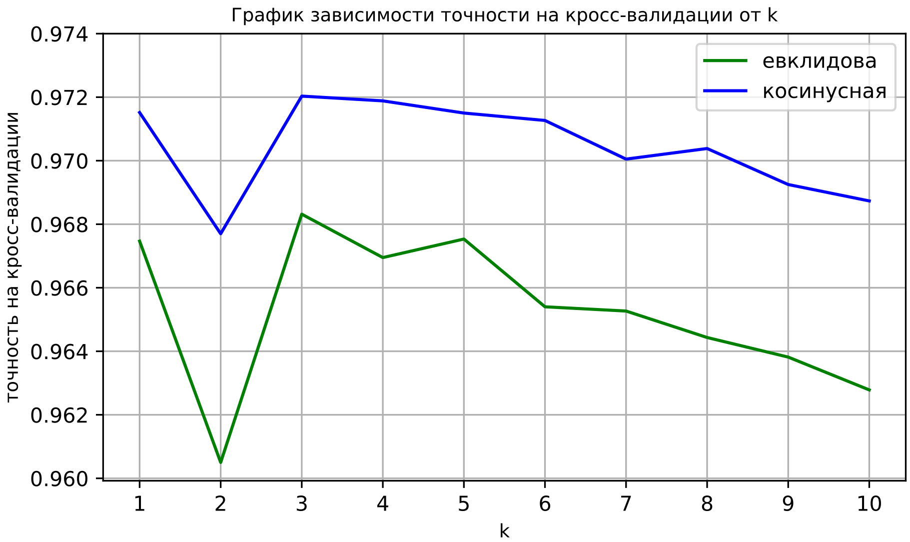
Далее будем использовать только алгоритм «brute», поскольку размерность признакового пространства в объектах = **784**.

3.2 Эксперимент 2

Теперь, когда мы определились с алгоритмом, исследуем вопрос зависимости точности алгоритма по кросс-валидации (на 3 фолдах) от параметров k и метрики. А также сравним время работы кросс-валидации в случае разных метрик.

Как упоминалось раньше, метрики будут следующие: "евклидова"и "косинусная". Параметр k будем рассматривать в диапазоне от 1 до 10.

Результаты эксперимента приведены ниже:



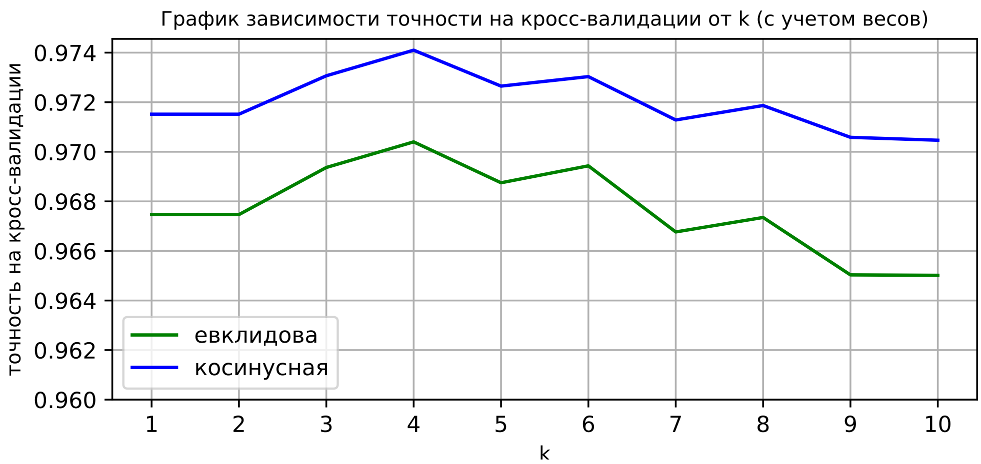
Можно заметить, что косинусная метрика при любых k показывает лучшую точность, чем евклидова. Однако общее время работы кросс-валидации на параметрах k для обоих метрик приблизительно равно: **260** секунд — евклидова и **257** секунд — косинусная.

3.3 Эксперимент 3

Все производимые ранее подсчёты выполнялись без учёта весов. Добавим веса к алгоритмам и выполним новые подсчёты при тех же параметрах, как в эксперименте 2.

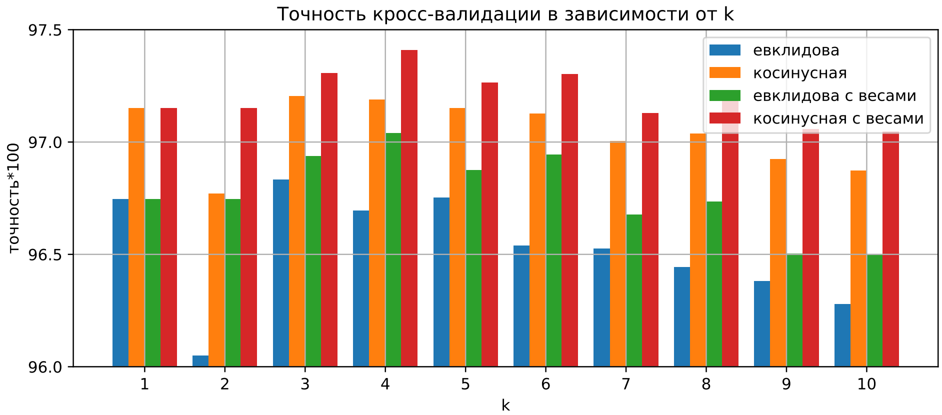
Будем считать, что вес объекта равен: $w = \frac{1}{distance + 10^{-5}}$

Получим следующий результат:



Ситуация аналогична второму эксперименту: косинусная метрика показала себя лучше при всех значениях k. Время работы осталось неизменно.

Теперь сравним оба метода: без весов и с весами.



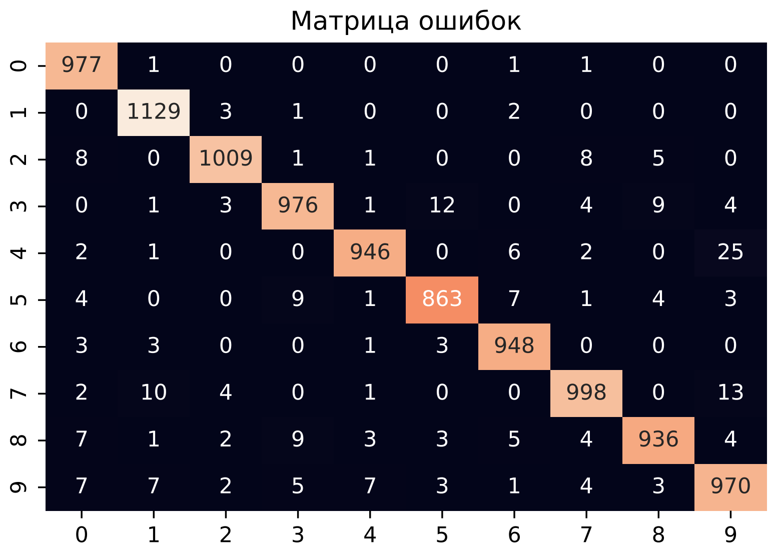
Как видно, косинусная метрика с весами показывает лучшую точность при всех значениях k (наилучшая достигается при k=4). Отметим также тот факт, что просто косинусная метрика лучше евклидовой с весами.

3.4 Эксперимент 4

Итак, мы нашли наилучший алгоритм: «brute», k = 4, косинусная метрика с весами.

Применим его к исходной обучающей и тестовой выборке и получим следующую точность: **97.52**. Она будет больше точности, получаемой при кросс-валидации: **97.41**, но меньше точности, получаемой при использовании лучших алгоритмов на данной выборке: **99.82**.

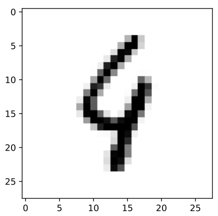
Построим теперь матрицу ошибок (по определению, a_{ij} элемент этой матрицы равен количеству объектов, принадлежащих классу i, но классифицированных классом j) для полученного результата:



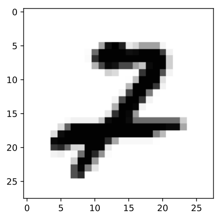
Исходя из неё можно сделать много выводов. Некоторые из них:

1. Чаще всего происходит неправильное распознавание цифры 4. Объектам из тестовой выборки ставится в соответствие цифра 9.
2. Меньше всех ошибок в распознавании у цифры 0.
3. Цифре 9, как и цифре 8 может быть поставлена в соотвествие любая из цифр.
4. Цифра 7 не может быть спутана с цифрой 3, однако обратное имеет место быть.

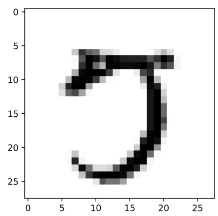
Рассмотрим на примерах неправильно классифицированные цифры и укажем их общие черты:



(4 -> 9)



(7 -> 2)



(5 -> 7)

Скобка вида (a -> b) означает, что цифре **a** алгоритм классификации поставил в соответствие цифру **b**.

Можно заметить, что ошибки в основном связаны с тем, что классифицируемая цифра очень близка к результативной цифре. Действительно, на первом рисунке цифре 4 немного не хватает дорисовать линию сверху до полной цифры 9. На втором, если дорисовать нижний левый угол у цифры 7, то мы получим цифру 2.

С другой стороны, ошибки могут мыть связаны и с неточным изображением цифры, как на примере третьего рисунка, где верхняя часть цифры 5 «прилипла» к ее нижней части.

3.5 Эксперимент 5

Мы получили итоговую точность — **97.52**. Но можно ли её улучшить? Попробуем ответить на этот вопрос, используя аугментацию данных, а именно: размножим обучающую выборку с помощью поворотов, смещений и применений гауссовского фильтра.

Возьмём следующие величины для каждого из преобразований:

- Поворот: 5, 10, 15 (в каждую из двух сторон).
- Сдвиг: 1, 2, 3 пикселя (по каждой из двух размерностей)
- Дисперсия фильтра Гаусса: 0.5, 1.0, 1.5.

С помощью кросс-валидации (по 3 фолдам) подберём оптимальные параметры для каждого из преобразований:

Поворот	+5	+10	+15	-5	-10	-15
Точность	0.9957	0.9836	0.9743	0.9956	0.9830	0.9735

Сдвиг	[1,0]	[2,0]	[3,0]	[0,1]	[0,2]	[0,3]
Точность	0.9830	0.9701	0.9690	0.9832	0.9701	0.9689

Дисперсия фильтра Гаусса	0.5	1.0	1.5
Точность	1.0	0.9912	0.9798

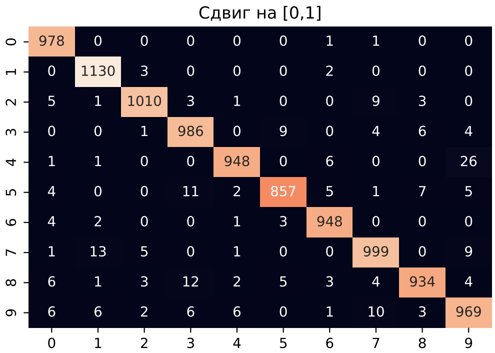
Оптимальным значением параметра сдвига является сдвиг по второй размерности на 1 пиксель.

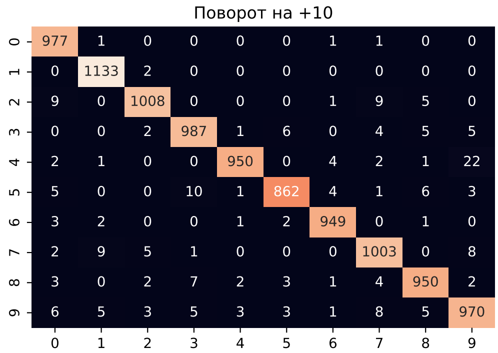
В значениях точности при использовании поворота есть два значения очень близкие к единице: они достигаются при повороте на +5 и -5. Возможно это переобучение? Сравним точность классификации в таких случаях. Действительно, точность классификации при добавлении к оригинальной выборке выборки полученной поворотом на +5 — **97.6**, -5 — **97.45**, а при +10 (следующем значении, на котором достигается максимальная точность) — **97.89**. Таким образом, оптимальным значением параметра поворота является поворот на +10

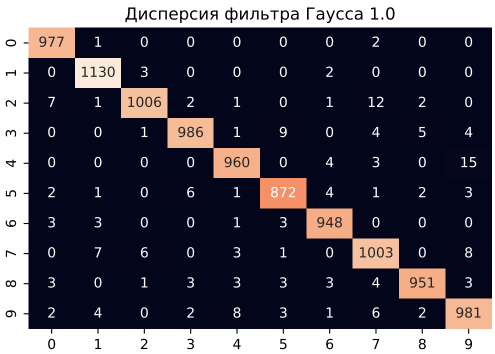
При использовании дисперсии фильтра Гаусса 0.5 и 1.0 точность, как и в предыдущем случае близка к 1 (при 0.5 она равна 1). Исследуем этот случай на переобучение. Точность классификации при 0.5 — **97.58**, при 1.0 — **98.14**, а при 1.5 — **97.82**. То есть в случае фильтра

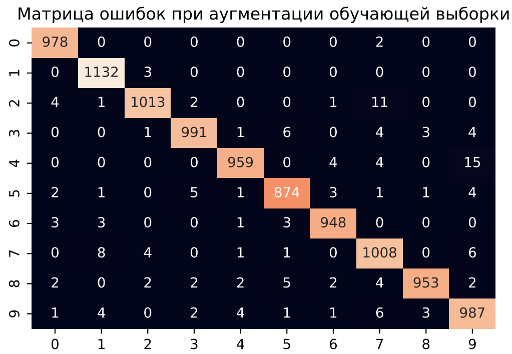
Гаусса мы получили, что при 0.5 получается переобучение, а при 1.0 уже нет (хоть значение точности на кросс-валидации и близко к единице). Следовательно, оптимальным значением параметра дисперсии фильтра Гаусса является дисперсия 1.0.

Рассмотрим, как изменились матрицы ошибок при каждом из преобразований и общая матрица ошибок при обучении модели на выборке из $4 * 70000 = 280.000$ объектах по сравнению с изначальной матрицей ошибок.









Разберемся с тем, какие ошибки алгоритма помогает исправить каждое из преобразований. Некоторые из них:

1. Сдвиг:
   - Помогает лучше распознавать цифру 3. Число правильно распознанных троек увеличилось на 10 штук.
   - Цифра 4 больше не путается с цифрой 7.
2. Поворот:
   - Помогает лучше распознавать цифру 3 и цифру 8. Число правильно распознанных троек и восьмерок увеличилось на 11 и 14 соответственно.
   - Цифра 1 больше не путается ни с кем, кроме цифры 2.
3. Дисперсия фильтра Гаусса:
   - Лучше распознает следующие цифры: 3, 4, 8, 9.
   - Меньше путает цифру 4 с цифрой 9.
   - Больше не путает цифру 9 с цифрой 2.

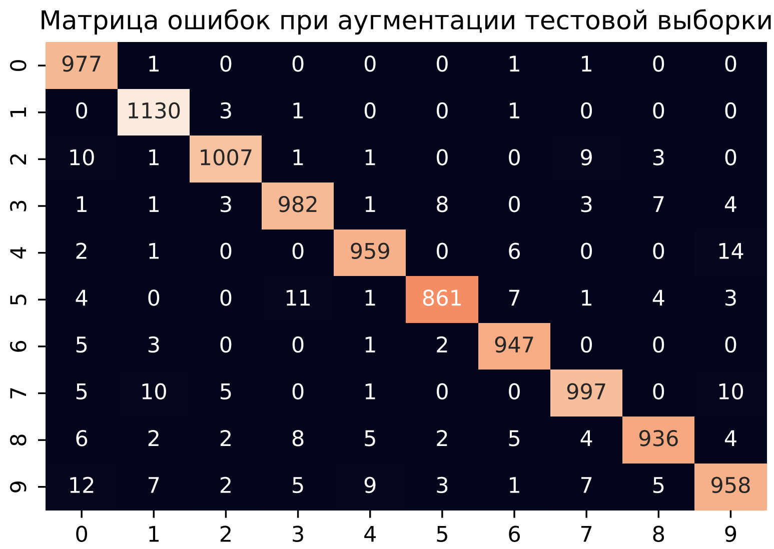
Конечно, с приходом улучшений могут возникать и некоторые ухудшения, но поскольку общая точность возрастает (для сдвига - **97.59**, для поворота - **97.89**, для фильтра дисперсии Гаусса - **98.14** и для общей аугментации - **98.43**), это даёт основание использовать аугментацию обучающей выборки.

3.6 Эксперимент 6

Попробуем теперь размножить тестовую выборку, используя аугментацию аналогично эксперименту 5, и исследовать вопрос повышения точности классификации. Итоговый результат будет получаться путём голосования среди преобразованных объектов.

Проверяя то же самое множество параметров, находим оптимальные параметры, при которых точность алгоритма немного увеличивается — **97.54**. Она достигается при свдиге на 2 пикселя по 2 размерности, повороте на +10 и дисперсии фильтра Гаусса 0.5.

Матрица ошибок будет иметь следующий вид:



Видно, что сильных улучшений по сранению с изначальной матрицей ошибок мы не получили. К примеру, заметным стало улучшение распознования цифры 4 и 3, однако хуже стала распознаваться цифра 9.

3.6.1 Сравнение подходов

Качественно сравнивая два подхода (аугментация обучающей и тестовой выборки), можно отметить следующее:

- При аугментации обучающей выборки можно достичь большего относительного прироста точности классификации.
- При аугментации обучающей выборки мы обучаем модель на увеличенном объеме данных, а при аугментации тестовой выборки мы обучаемся на оригинальной выборке один раз, но предсказание происходит несколько раз и результат получаем путём голосования.
- Подбор параметров при аугментации обучающей выборки на кросс-валидации происходит лучше, чем при аугментации тестовой выборки.
- Объем вычислений при аугментации обучающей выборки гораздо больше, чем при аугментации тестовой выборки.

3.7 Выводы

Итак, мы рассмотрели датасет изображений MNIST и провели эксперименты над ним для выявления наилучшего алгоритма для классификации. Изначально мы определили самый быстрый алгоритм, потом подобрали параметры для достижения большей точности, а под конец использовали аугментацию данных для еще большего повышения точности. Причем, как мы отметили, точность, которую мы достигли — далеко не предел. В ходе экспериментов активно использовалась кросс-валидация для исключения параметров, которые не давали наилучшего прироста точности. В заключение были отмечены плюсы и минусы используемых подходов, а также то, какие ошибки они помогают исправить.