Московский государственный университет
имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Метрические алгоритмы классификации

Практикум на ЭВМ

Выполнил:
Кадченко И. Е.

2022

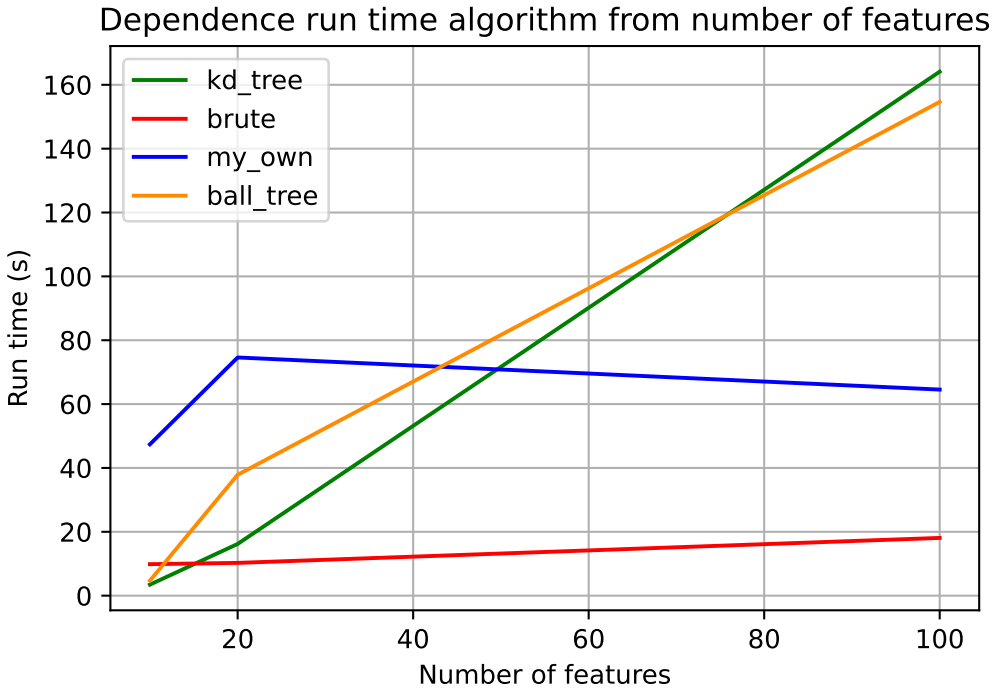
Содержание

Постановка задачи

Данный отчёт отражает проделанную мной работу в проведении экспериментов с датасетом изображений цифр MNIST на основе собственно реализованных метода $K-$ ближайших соседей, метрик и кросс-валидации.

Эксперимент №1

Стоит задача исследовать различные взятые из библиотеки алгоритмы $K-$ ближайших соседей, такие как 'kd_tree', 'ball_tree', 'brute' и собственно реализованный 'my_own' на быстродействие в различных ситуациях и выявить лучший в этом показателе. Рассмотрим работу каждого алгоритма в зависимости от количества выбранных случайным образом признаков. Размер подмножества положим равным 10, 20 и 100 соответственно.

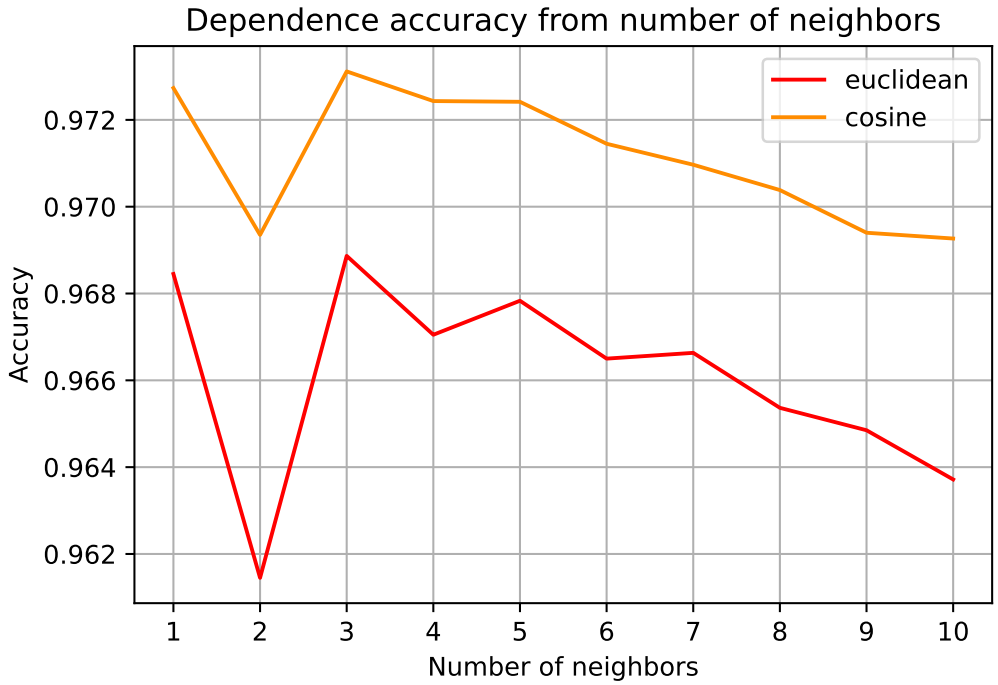


Наилучшим образом себя показал алгоритм 'brute', продемонстрировав наименьшее возрастание времени с увелечением числа признаков. Далее по быстродействию идёт собственная реализация 'my_own', показывая схожую асимптотику, но, тем не менее, уступая 'brute'. Алгоритмы 'ball_tree' и 'kd_tree' также между собой имеют схожую линейную асимптотику, причём при маленьком количестве признаков 'kd_tree' работает быстрее, но с увечелением числа признаков (более 80) начинает уступать 'ball_tree'. Можно заметить, что с увелечением количества признаков при маленьком абсолютном значении (вблизи 20 признаков), зависимость имеет неоднородный характер - наблюдаем проклятье размерности (curse of dimensionality).

Далее в экспериментах будет использоваться алгоритм 'brute', если не оговорено иное.

Эксперимент №2

В данном эксперименте требуется установить зависимость точности работы алгоритма от количества соседей k и используемой метрики, евклидовой или косинусная, а также определить влияние метрики на время выполнения. Проведём исследование по кросс-валидации с 3 фолдами, рассматривая количество соседей k в диапазоне от 1 до 10. Результат отражён на графике ниже.



Показывает худшую точность при любом количестве ближайших соседей в среднем на константу алгоритм с евклидовой метрикой, чем с косинусной. Объяснить это можно тем, что в задаче классификации изображений (как и в задаче классификации текстов) более значимым критерием является схожая ориентированность объектов друг к другу, а не еклидово расстояние между ними. Алгоритм с косинусной метрикой работает медленее чем с евклидовой, показывая время полного цикла кросс-валидации $942s$ против $182s$.

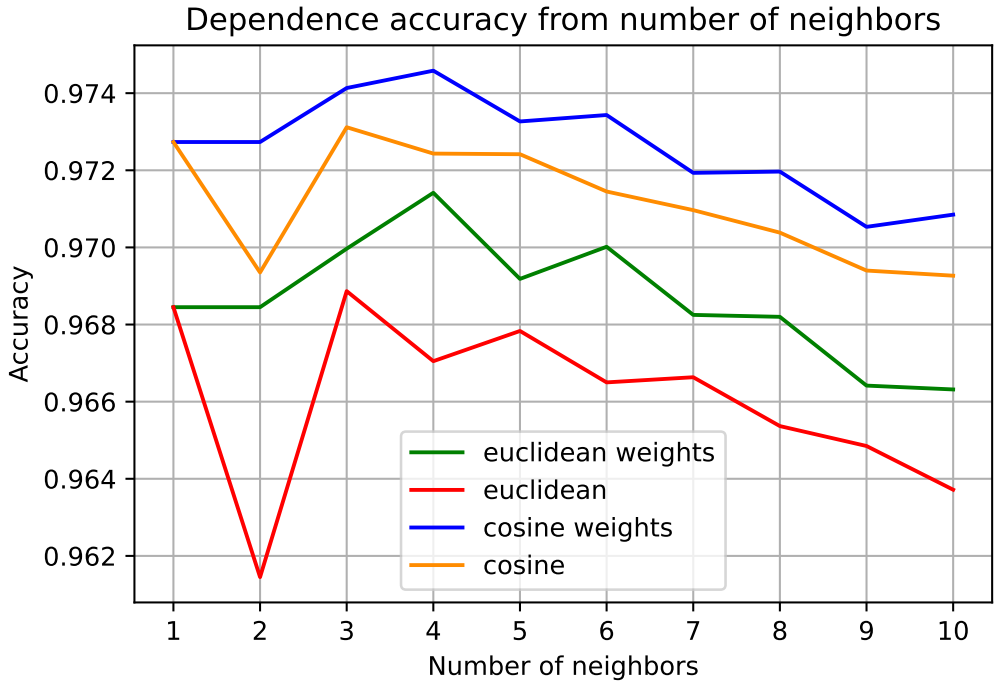
Оба алгоритма показали лучший результат при $k = 3$. При $k > 3$ точность обоих алгоритмов с увелечнием k снижается, а при $k = 2$ происходит резкое падение качества. Это связано с тем, что алгоритмы при различных двух ближайших соседях каждый раз будут выбирать первый объект, чтобы устранить неопределённость. Вследствие чего в примерно половине случаев объект классифицируется неправильно.

Далее будем задавать параметр k равным 3 и использовать косинусное расстояние, если не оговорено иное.

Эксперимент №3

Определим зависимость точности алгоритма от того факта, используются ли веса $\frac{1}{distance+\varepsilon}$, где $\varepsilon = 10^{-5}$. Рассотрим используемый в предыдущем экспе-

рименте алгоритм как с косинусной, так и с евклидовой метриками по кросс-валидации с тремя фолдами. Результат приведён на графике:



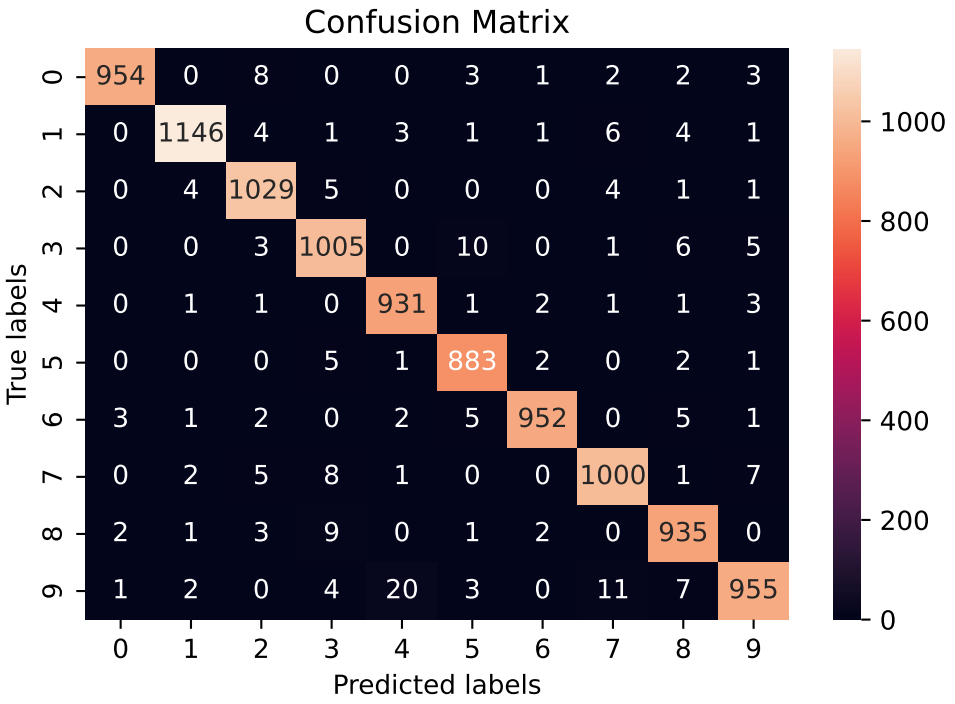
Независимо от используемой метрики, вариация алгоритма с весами оказывается лучше для всех значений k, за исключением $k = 1$, когда наличие весов не имеет значения. Также использование весов сглаживает упомянутый в предыдущем эксперименте выброс при $k = 2$, так как приоритет в данном случае будет уже отдаваться соседу с наибольшим весом.

Эксперимент №4

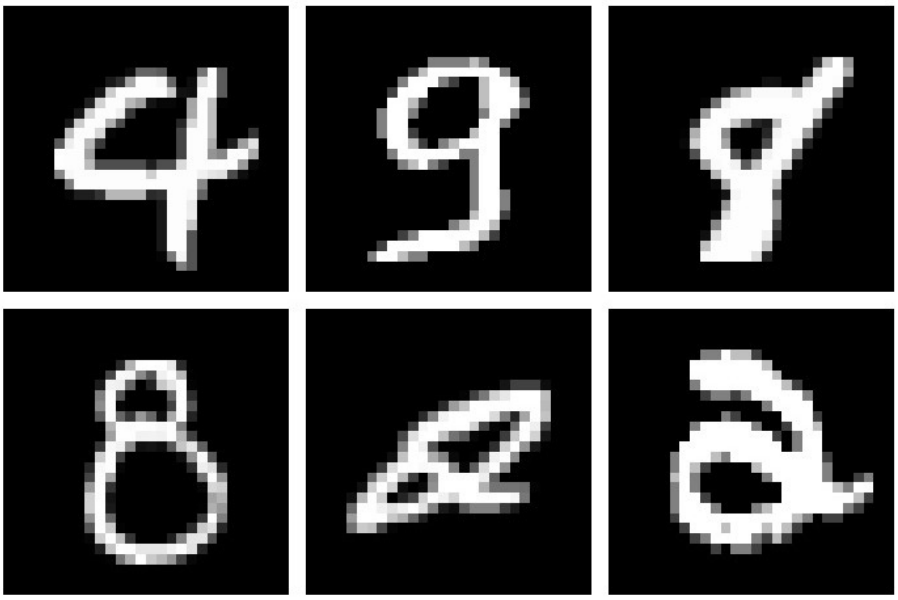
Проверим точность лучшего на основе предыдущих экспериментов алгоритма и сравним с точностью по кросс-валидации. Практическим путём было выявлено, что лучшим оказался алгоритм 'brute' с косинусной метрикой, наличием весов и тремя ближайшими соседями. Точность с помощью него на нашем датасете равна 0.9790. Точность на кросс-валидации:

Фолд 1: 0.9755
Фолд 2: 0.9756
Фолд 3: 0.9723

средняя точность на кросс-валидации - 0.9745. Точность алгоритма на обучающей и тестовой выборках близка к кросс-валидационной, что означает хорошую обученность модели. Свёрточные сети, например, с использованием библиотеки Keras, дают точность более 0.99. Наша модель сильно приблизилась к этому значению, показав хороший результат. Посмотрим, где наша модель совершала ошибки:



Больше всего модель ошибалась, путая 9 с 4. В два раза меньше - путая ту же 9 с 7 или 3 с 5. Посмотрим на объекты, на которых были сделаны ошибки:



С тремя из шести картинок у человека возникут сложности с определением цифры, и даже с высокой долей вероятности он совершит ошибку, поэтому эти неточности нашей модели могут быть оправданы. Три другие должны распознаваться, так как они кажутся различимыми. Поэтому попытаемся усовершенствовать нашу модель - следующий эксперимент об этом.

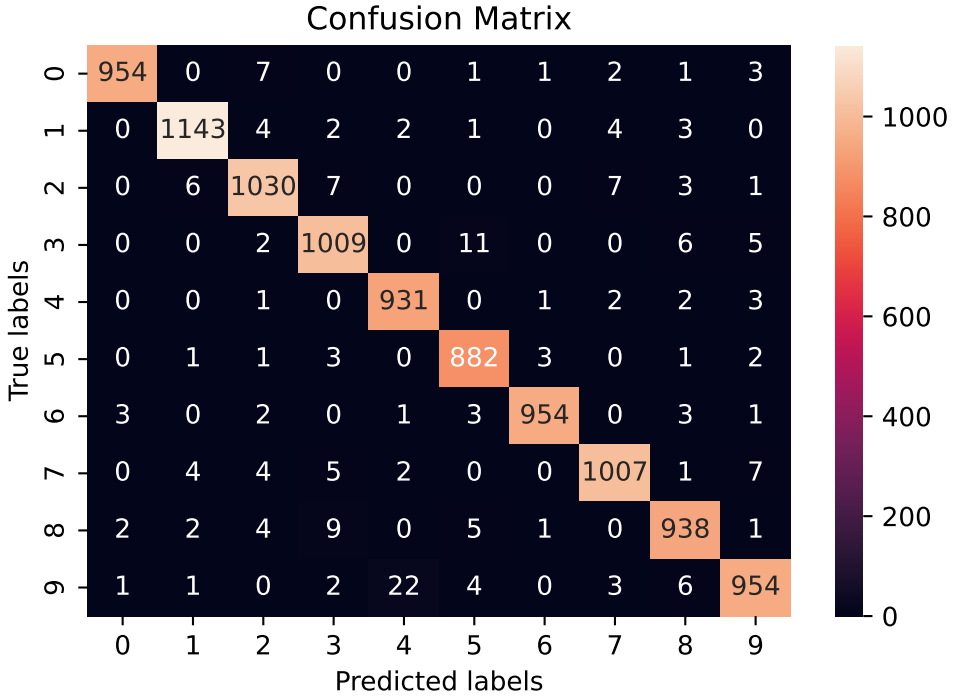
Эксперимент №5

Стоит задача осуществить аугментацию обучающей выборки с целью анализа точности и изменений матрицы ошибок. Осуществлять аугментацию будем при помощи поворотов, сдвигов, фильтров Гаусса и морфологических операций.

Результаты преобразований-поворотов по кросс-валидации с 3-мя фолдами:

Преобразование	фолд 1	фолд 2	фолд 3
5° против часовой стрелки	0.9764	0.9762	0.9778
5° по часовой стрелке	0.9743	0.9768	0.9763
10° против часовой стрелки	0.9764	0.9774	0.9784
10° по часовой стрелке	0.9739	0.9758	0.9754
15° против часовой стрелки	0.9757	0.9770	0.9777
15° по часовой стрелке	0.9726	0.9748	0.9742

Каждое из преобразований немного повысило точность, показываемую алгоритмом в эксперименте № 4. Самую значительную разницу показал поворот на 10° против часовой стрелки. Рассмотрим для него матрицу ошибок.



Ощутимое изменение аугментация с поворотом даёт для классификации 7, модель стала почти в 4 раза реже путать 9 с 7. В остальном изменения незначительные как в сторону улучшения, так и ухудшения.

Следущий тип преобразований - сдвиг изображения. Что получили:

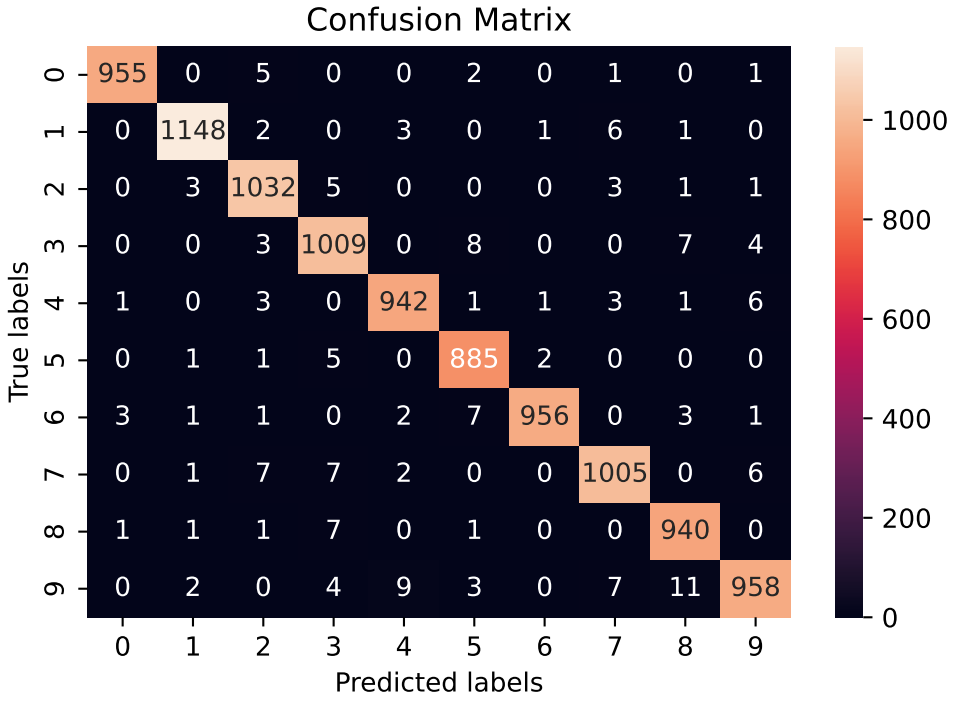
Преобразование	фолд 1	фолд 2	фолд 3
1 пиксель вправо	0.9750	0.9768	0.9760
2 пикселя вправо	0.9751	0.9750	0.9749
3 пикселя вправо	0.9742	0.9741	0.9738
1 пиксель вверх	0.9762	0.9759	0.9760
2 пикселя вверх	0.9752	0.9743	0.9743
3 пикселя вверх	0.9741	0.9735	0.9736

Преобразование сдвигом не оказало значительного влияния на работу нашей модели, оставив точность на том же уровне. Сдвиги на 1 пиксель как вправо, так и вверх повышают точность предсказания, тогда как сдвиги на 2 и 3 пикселя приводят уже к падению качества.

Исследуем результаты применения фильтра Гаусса:

Преобразование	фолд 1	фолд 2	фолд 3
Дисперсия 0.5	0.9767	0.9729	0.9776
Дисперсия 1	0.9800	0.9772	0.9804
Дисперсия 1.5	0.9808	0.9773	0.9803

Фильтр Гаусса повысил точность предсказания значительнее всего среди упомянутых преобразований. Качество повышается с увеличением дисперсии. Рассмотрим таблицу ошибок для преобразования с дисперсией 1.5:

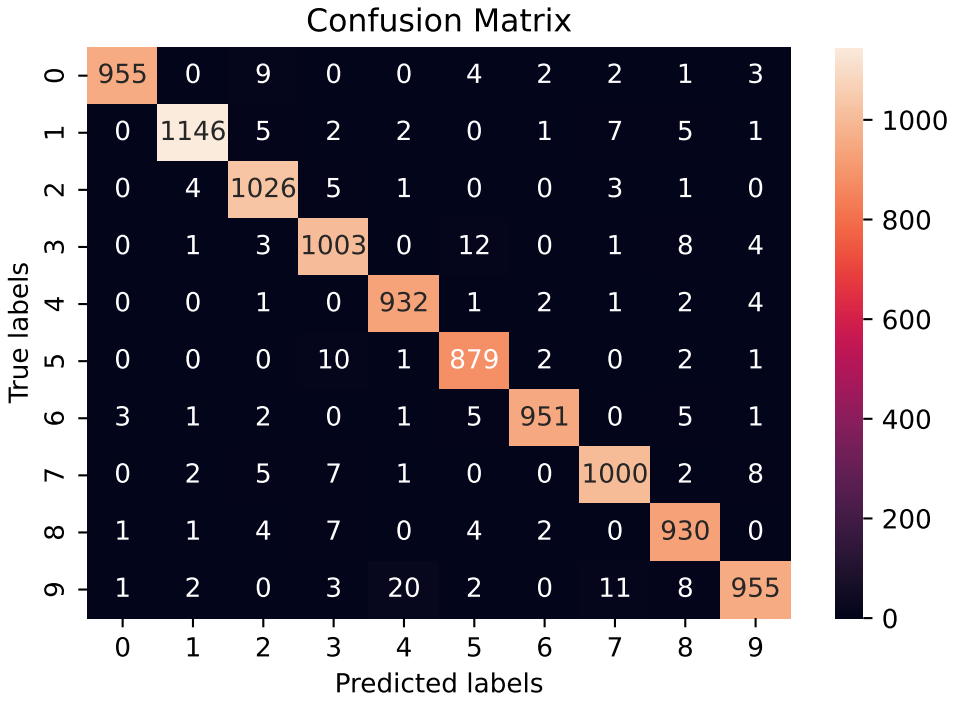


Фильтр Гаусса повысил точность идентифицирования для каждой предсказываемой цифры. Так, для цифры 4 - в два раза сократились неправильные ответы при её распознавании как 7 или 9. Цифра 9 так же стала в два раза реже интерпретироваться как 4. Последняя ошибка являлась самой частой в предыдущих экспериментах, фильтр Гаусса понизил частоту её появления.

Рассмотрим такие морфологические преобразования, как дилатация, эрозия, открытие и закрытие. Применим их к обучающей выборке и посмотрим на результат:

Преобразование	фолд 1	фолд 2	фолд 3
Дилатация	0.9738	0.9743	0.9779
Эрозия	0.9742	0.9752	0.9766
Открытие	0.9717	0.9727	0.9755
Закрытие	0.9734	0.9727	0.9762

В целом все морфологические операции понизили точность предсказания. Сильнее всего понизило точность открытие. Посмотрим на матрицу ошибок для этого преобразования:



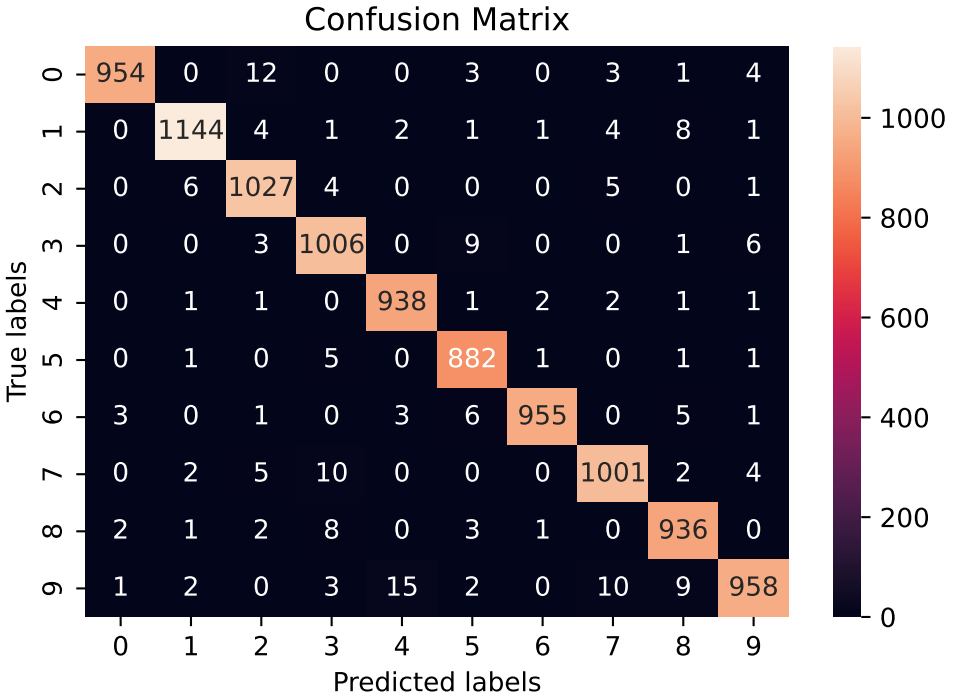
Нет цифр, которые бы распознавались лучше с применением открытия - результат либо нулевой, либо отрицательный. Так, в два раза увеличилось количество ошибок при распознавании 5 как 3, также алгоритм чаще стал принимать разные цифры за цифру 8.

Обобщая можно сказать, что нельзя дать однозначного ответа, даст ли применение различных преобразований аугментации улучшений в точности - нужно смотреть в каждом конкретном случае. Для нашей модели наилучшим образом себя показал фильтр Гаусса с дисперсией 1.5, повысив среднюю точность по кросс-валидации с 3 фолдами с 0.9745 до 0.9795 - повышение на 0.005.

Эксперимент №6

Теперь требуется выяснить, как будет вести себя модель, если применить аугментацию к тестовой выборке. Будем проверять то же множество парамет-

ров, что и в предыдущем эксперименте. Преобразуем тестовую выборку для каждого параметра. Получив предсказание для каждой такой выборки, конечное предсказание алгоритма найдём путём голосования среди преобразованных объектов. Матрица ошибок тогда будет выглядеть следующим образом:



Алгоритм улучшил точность предсказания с 0.9790 до 0.9801. Алгоритм перестал ошибаться в некоторых случаях, например не путает 4 с 1, 4 с 5, и наоборот стал ошибаться там, где раньше этого не делал: 4 с 0, 8 с 9. Модель, можно сказать, не потерпела явных изменений. Это объясняется тем, что расширение данных используется для увеличения размера обучающей выборки и получения большего количества различных изображений - модель становится более обобщённой и надёжной. Тестовый же набор данных должен быть приближен к реальным данным, а аугментация может только способствовать появлению ошибок.