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

Отчет по исследованию KNN

Выполнил:

Студент 3 курса
Н. А. Панин

Москва
2022

1 Введение

В данной работе реализовывался и исследовался простейший метрический алгоритм классификации **метод k ближайших соседей (k nearest neighbors, KNN)**. Несмотря на свою тривиальность, данный метод успешно решает множество непростых задача, в том числе классификацию рукописных цифр. Именно поэтому в этом исследовании в качестве объектов классификации были взяты рукописные цифры из объемной базы данных MNIST.

В работе исследовалась зависимость времени поиска k ближайших соседей от размерности признакового пространства и от используемой стратегии поиска, рассматривался вопрос о наилучшем выборе гиперпараметров, сравнивался **взвешенный метод k ближайших соседей** с классическим KNN. Также был проведен анализ того, на каких цифрах алгоритм с наилучшим подбором гиперпараметров ошибался. В том числе были построены две новые модели классификации, основанные в одном случае на аугментации обучающей выборке, а в другом на аугментации тестовой выборке. Анализировалась обобщающая способность этих новых моделей с моделью без аугментации.

2 Список экспериментов

2.1 Эксперимент 1

2.1.1 Дизайн эксперимента

Заданы:

- $k = 5$
- Алгоритмы поиска ближайших соседей: *my_own, kd_tree (k-d-дерево), brute (brute force – полный перебор), ball_tree (дерево мячей)*
- Размерность признакового пространства[1]: 10, 20, 100

В первом эксперименте исследовался вопрос о времени работы алгоритма поиска k ближайших соседей в зависимости от размерности признакового пространства.

2.1.2 Результаты эксперимента

В ходе эксперимента было обнаружено, что в случае размерности 10 и 20 *kd_tree* лидирует среди остальных стратегий поиска. Действительно, в случае с размерностью 10 производительность *kd_tree* в 2 раза выше чем у *ball_tree* и почти в 10 раз выше чем у *my_own и brute*, а, если размерность равна 20, то выше в 1.5 - 2 раза чем у каждого из оставшихся алгоритмов. Впрочем уже при размерности признакового пространства равной 100 *my_own и brute* работают в ≈ 10 раз быстрее *ball_tree и kd_tree* (см. Рис.1). [2]

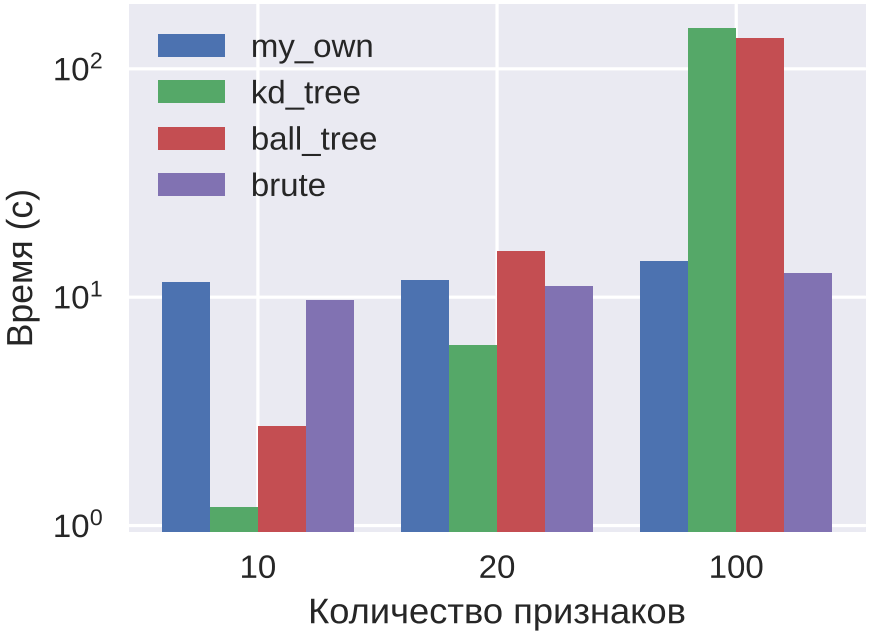
Причиной этому являются следующие факторы:

- Количество объектов N и количество признаков D
  - Сложность *brute* алгоритма равна $O(D * N)$
  - Сложность *kd_tree* алгоритма зависит от D. Для малых D ($\lessapprox 20$) равна $O(D * \log(N))$, поэтому она работает быстрее *brute*. Для

[1] В нашем случае это количество пикселей картинки

[2] *My_own и brute* работают примерно одинаково, но *brute* все же немного лучше

Рис. 1: Время работы алгоритма поиска ближайших соседей от количества признаков



большего количества признаков D сложность алгоритма становится $O(D * N)$, и более того в следствие накладных расходов из-за древовидной структуры время обработки запроса на выдачу ближайших соседей может стать значительно хуже, чем в *brute*

- Количество соседей k:
  - *brute* почти не зависит от k
  - *kd_tree* с увеличением k становиться медленнее

2.1.3 Выводы из эксперимента

Таким образом, при больших k быстрее работает *brute*. При малой размерности D ($\lessapprox$20) лучше по производительности *kd_tree*, если размерность признакового пространства становится больше, то быстрее *brute*

2.2 Эксперимент 2

2.2.1 Дизайн эксперимента

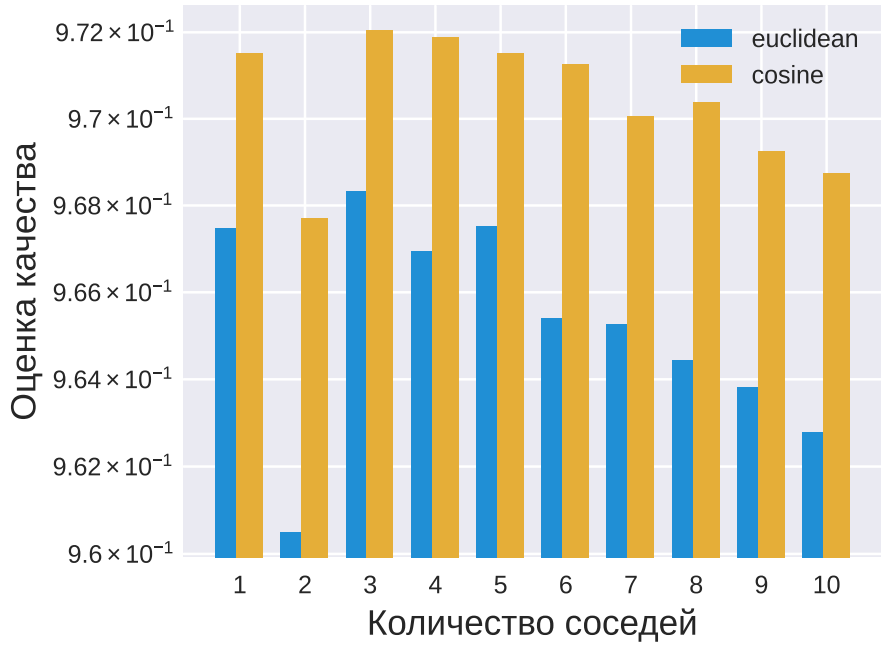
Заданы:

- $k = 1, 2, \ldots, 10$
- Метрика: *евклидова и косинусная*
- Веса: 1

Во втором эксперименте требовалось найти наилучшую метрику для определения степени близости объекта к тому или иному классу, найти оптимальный k. Также требовалось понять причину выбросов на графике зависимости оценки качества от количества соседей при их наличии. Для определения наилучших гиперпараметров была использована кросс-валидация с тремя фолдами для каждой комбинации гиперпараметров, на каждом фолде получали свою оценку качества алгоритма (*accuracy*) и в качестве итоговой оценки обобщающей способности алгоритма бралось среднее арифметическое. Далее среди всех комбинаций метрики и количества соседей выбирался вектор гиперпараметров с лучшим *accuracy*

2.2.2 Результаты эксперимента

Рис. 2: *accuracy* в зависимости от количества соседей



В результате эксперимента было обнаружено, что наилучшей метрикой при всех $k = 1, 2, \ldots, 10$ является косинусная, а наилучшим k при заданных выше параметрах модели будет 3 (см. Рис.2)

Причина, по которой евклидово расстояние хуже при классификации, в следующем. Если посмотреть на матрицу пикселей любой картинки, то в ней мощность множества нулей в среднем больше мощности ненулевых значений.[3] Довольно часто рукописные цифры отличаются именно поворотом на небольшие градусы. При этих поворотах часть ненулевых значений матрицы переходит в нулевые ячейки, а вместо прежних ненулевых значений в марицу впишутся нули. Тогда, учитывая еще и проклятие размерности расстояние объектов, относящихся к одному и тому же класса может быть слишком большим, чтобы отнести их к одному классу.

Также на Рис.2 виден выброс при k равном 2. При $k = 2$ существует два варианта, какие результаты могут быть кандидатами на верный класс: либо два разных класса одинаково хорошо подходят согласно KNN, либо лишь один класс хорошо подходит. Происходит именно первый вариант, потому что иначе колонка с $k = 1$ не была бы выше колонки с пометкой 2. Отсюда следует, что из-за равных весов модели приходится с равной вероятностью выбирать между двумя классами объектов быть может находящихся на разных расстояниях. Это плохо. Именно поэтому в третьем эксперименте при выборе весов, зависящих от расстояния, мы подобных сильных выбросов наблюдать не будем.

2.2.3 Выводы из эксперимента

Из данного эксперимента и полученных результатов следует, что при оценке близости инвариантность одинаковых объектов при различных преобразованиях обучающей выборки[4] (повороты на малые углы, смещения и т.д.) и различие между объектами разных классов лучше всего оценивается косинусной метрикой. Также в этом эксперименте мы показали, что у классического KNN с равными весами есть проблемы, а именно при равных весах даже те соседи, которые, на самом деле, далеки от объекта учитываются одинаково с объектами близкими к объекту

2.3 Эксперимент 3

2.3.1 Дизайн эксперимента

Заданы:

[3] Это легко понять, хотя бы потому что само число занимает меньше места, чем вся картинка

[4] Под выборкой здесь подразумевается рукописный текст из MNIST.

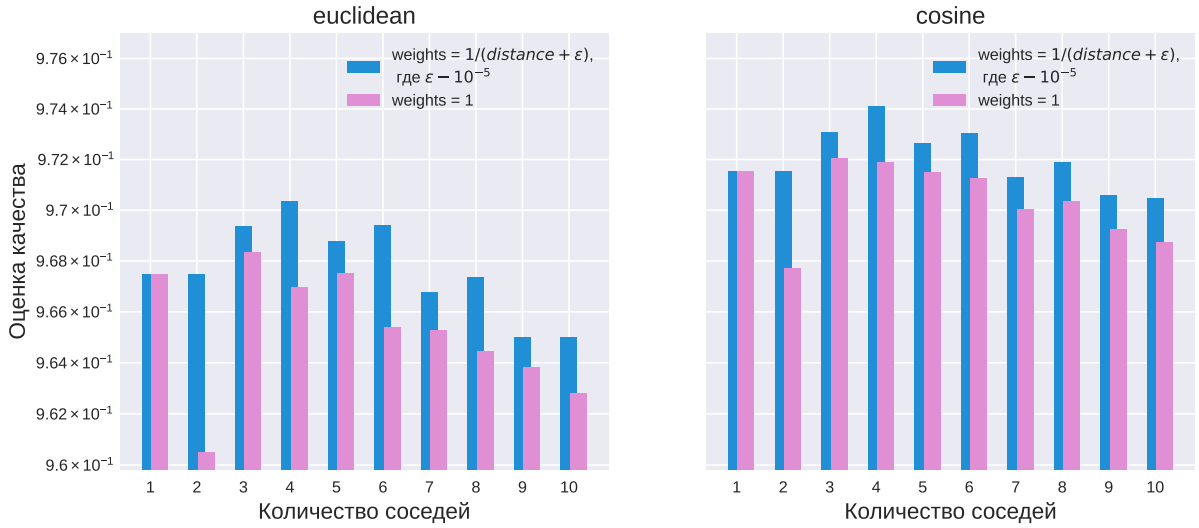
- $k = 1, 2, \ldots, 10$
- Метрика: *евклидова и косинусная*
- Веса: 1, $\frac{1}{(distance+\varepsilon)}$, *где* $\varepsilon = 10^{-5}$

В третьем эксперименте решается вопрос об улучшении классического KNN путем замены веса с постоянного значения на функцию, зависящую от расстояния рассматриваемого объекта до его соседа. Здесь также как и во втором эксперименте перебираются все возможные параметры и среди них выбирается наилучший по кросс-валидации с тремя фолдами.

2.3.2 Результаты эксперимента

В процессе исследования было выявлено отсутствие сильных выбросов при различных k. В том числе в случае с евклидовым расстоянием при $k = 2$ количество ошибок на $\approx$17.5% меньше у взвешенного алгоритма классификации чем у классического(это максимальное улучшение по всем k и метрикам). При любом k и при любой метрике *accuracy* был больше с весом равному $\frac{1}{(distance+\varepsilon)}$. Наилучшая оценка качества достигалась при $k = 4$. При этом по-прежнему косинусная метрика показывала самую высокую *accuracy* (см. Рис.3).

Рис. 3: Зависимость оценки качества, полученной с помощью кросс-валидации, от количества соседей



2.3.3 Выводы из эксперимента

Таким образом, благодаря использованию весов, зависящих от расстояния рассматриваемого объекта до его соседа можно получить значительно более точные ответы на тестовой выборке.

2.4 Эксперимент 4

2.4.1 Дизайн эксперимента

Заданы:

- $k = 4$
- Метрика: *косинусная*
- Алгоритм поиска: *brute*
- Веса: $\frac{1}{(distance+\varepsilon)}$, *где* $\varepsilon = 10^{-5}$

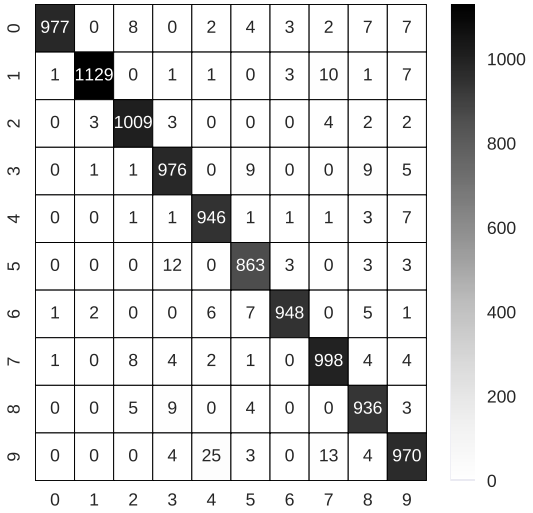
В ходе данного эксперимента использовались оптимальные параметры модели, указанные, выше которые были получены в результате предыдущих экспериментов. Далее считалась доля правильно предсказанных ответов *accuracy*. Она в свою очередь сравнивалась с точностью по кросс-валидации *cross_validation_accuracy*, а также с лучшими алгоритмами на данной выборке. Проведен анализ ошибок алгоритма при помощи матрицы ошибок.

2.4.2 Результаты эксперимента

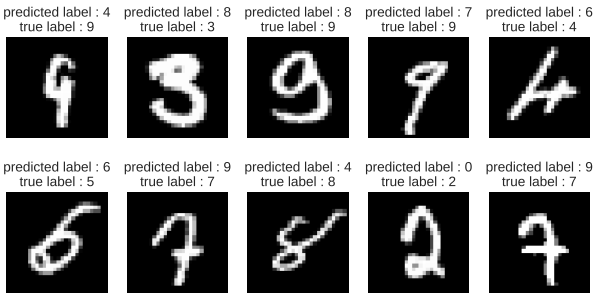
Accuracy оказался равен 0.9752, в то время как *cross_validation_accuracy* равен 0.9741. Данный результат говорит о том, что выбранная нами в предыдущих экспериментах модель поиска оптимальных гиперпараметров для KNN оказалась успешной, так как обобщающая способность (*accuracy*) алгоритма не меньше *cross_validation_accuracy*. Впрочем KNN не является лучшим решением для определения рукописных цифр из MNIST. В качестве такого алгоритма на данной выборке оказался алгоритм, указанный по этой ссылке. Его точность равна 0.9982. А это означает, что он на ≈92.8% делает меньше ошибок, чем KNN с рассматриваемыми параметрами.

Проанализируем с помощью матрицы ошибок, где наш алгоритм чаще всего ошибается (см. Рис.4a). На матрице ошибок видно, что алгоритм чаще всего ошибается в тех случаях, когда числа схожи между

собой по форме, так, например, в случае 6 цифра похожа по очертаниям на 5 (см. Рис.4b). Аналогичные рассуждения можно обобщить на все остальные ошибочно распознанные числа.



(a) Матрица ошибок



(b) Числа, на которых KNN ошибается

Рис. 4

2.4.3 Выводы из эксперимента

. Подытожив, сформулируем основные результаты этого эксперимента. По кросс-валидации с тремя фолдами можно с успехом подобрать оптимальные гиперпараметры для KNN. Более того, обобщающая способность на тестовой выше чем на кросс-валидации, это говорит о том, что никакого переобучения в нашем KNN нет. При рассмотрении матрицы ошибок было обнаружено, что цифры внешне похожие по начертанию часто приводят к ошибочной классификации. Это наводит на мысль, что, возможно, применив некоторые преобразования картинок мы сможем улучшить качество распознавания цифр

2.5 Эксперимент 5

2.5.1 Дизайн эксперимента

Заданы:

- $k = 4$
- Метрика: *косинусная*

- Алгоритм поиска: *brute*
- Веса: $\frac{1}{(distance+\varepsilon)}$, *где* $\varepsilon = 10^{-5}$
- Величина поворота: 5, 10, 15 (в каждую из двух сторон).
- Величина смещения: 1, 2, 3 пикселя (по каждой из двух размерностей).
- Дисперсия фильтра Гаусса: 0.5, 1, 1.5.
- Морфологические операции: эрозия, дилатация, открытие, закрытие с ядром 2

В данном эксперименте основной задачей было проверить как влияет на качество алгоритма добавление аугментированной выборки к обучающей. Для того, чтобы найти оптимальные параметры использовалась кросс-валидация с тремя фолдами. Непосредственно внутри кросс-валидации аугментированная обучающая выборка (обучающая именно для кросс-валидации, не для всей задачи в целом) добавлялась к обучающей и уже на новой обучающей выборке считался *accuracy*. Также, поскольку перебор всей сетки гиперпараметров (**Grid Search**) занял бы слишком много процессорного времени, то было решено делать следующее: поочередно брали какое-то преобразование и проходились по всему множеству его параметров и выбирали лучший, то есть по сути задача решалась "жадно".

При исследовании на матрицах ошибок к обучающей выборке добавлялась аугментированная, и с помощью новой обучающей выборки делали предсказание на тестовой. Далее это предсказание сравнивалось с предсказанием, полученным, когда обучающая выборка без аугментированной

2.5.2 Результаты эксперимента

В результате эксперимента лучшими параметрами преобразований были: сдвиг на (1, 0) пиксель (лучший сдвиг вправо по Ox), сдвиг на (-1, 0) пиксель (лучший сдвиг влево по Ox), сдвиг на (0, 1) пиксель (лучший сдвиг вправо по Oy), сдвиг на (0, -1) пиксель (лучший сдвиг влево по Oy), поворот против часовой стрелке на 10, поворот по часовой стрелке на 10, размытие по Гауссу с дисперсией фильтра 1.5, морфологическая операция дилатация.

При сдвиге вправо по оси x на (1, 0) (6) получаем, что лишь у тройки лучше предсказывается результат (на 10 ошибок меньше), в остальных же случаях есть даже небольшие ухудшения (в среднем на ≈1 ошибку больше). При сдвиге влево по оси x на (-1, 0) (7) почти на всех есть улучшение качества(на ≈3 ошибки меньше в среднем для всех), в особенности для 9 и 8. При сдвиге вправо по оси y на (0, 1) (8) качество не изменилось. При сдвиге влево по оси y на (0, -1) (9) обобщающая способность не изменялась, а местами даже ухудшалась. При повороте на 10 градусов против часовой стрелки (10) качество почти не менялось. При повороте на 10 градусов по часовой стрелки (11) точность алгоритма возросла практически для всех цифр (в среднем на ≈4 ошибки меньше). После размытия обучающей выборки (12) точность алгоритма также возросла для всех цифр(на ≈6 ошибок меньше). И наконец в случае дилатации (13) точность для большей части цифр не изменяется, тем не менее для двойки получилось допустить на 14 ошибок меньше.

2.5.3 Выводы из эксперимента

Таким образом, наиболее удачными преобразованиями являются:

- Сдвиг влево по оси x на (-1, 0)
- Поворот на 10 градусов по часовой стрелке
- Гауссовское размытие с дисперсией фильтра 1.5
- Дилатация с ядром размера (2, 2)

2.6 Эксперимент 6

2.6.1 Дизайн эксперимента

Заданы:

- $k = 4$
- Метрика: *косинусная*
- Алгоритм поиска: *brute*
- Веса: $\frac{1}{(distance+\varepsilon)}$, *где* $\varepsilon = 10^{-5}$
- Величина поворота: 5, 10, 15 (в каждую из двух сторон).

- Величина смещения: 1, 2, 3 пикселя (по каждой из двух размерностей).
- Дисперсия фильтра Гаусса: 0.5, 1, 1.5.
- Морфологические операции: эрозия, дилатация, открытие, закрытие с ядром 2

В этом эксперименте необходимо было реализовать новую модель классификации. В отличие от 5 эксперимента, для этого вместо аугментации обучающей выборки и добавлении ее в исходную обучающую, тем самым получая другую модель, здесь использовалась аугментация тестовой выборки и голосование. Смысл заключался в следующем: сначала предсказывали ответ по не аугментированной тестовой выборки, потом по аугментированной соответствующим преобразованием(см. выше) и в конце бралась мода от ответов[5], которая выступала в качестве итогового предсказания.

Лучшие параметры преобразований определялись так: фиксировался некоторый произвольный набор параметров по всем преобразованиям из множества возможных параметров, далее проходились по всем преобразованиям поочередно и выбирали наилучший параметр для текущего преобразования (по *accuracy*), после этого лучший параметр фиксировался и использовался для определения лучших параметров следующих преобразований. Получился итерационный алгоритм нахождения лучших параметров преобразований.

Следующим этапом эксперимента было исследование ошибок алгоритма. Было решено использовать также голосование, голосование проводилось только по исходной тестовой выборке и по аугментированной по какому-то одному преобразованию, то есть голосование проводилось только по двум ответам. Далее строилась матрица ошибок и она сравнивалась с матрицами ошибок из 4 и 5 эксперимента.

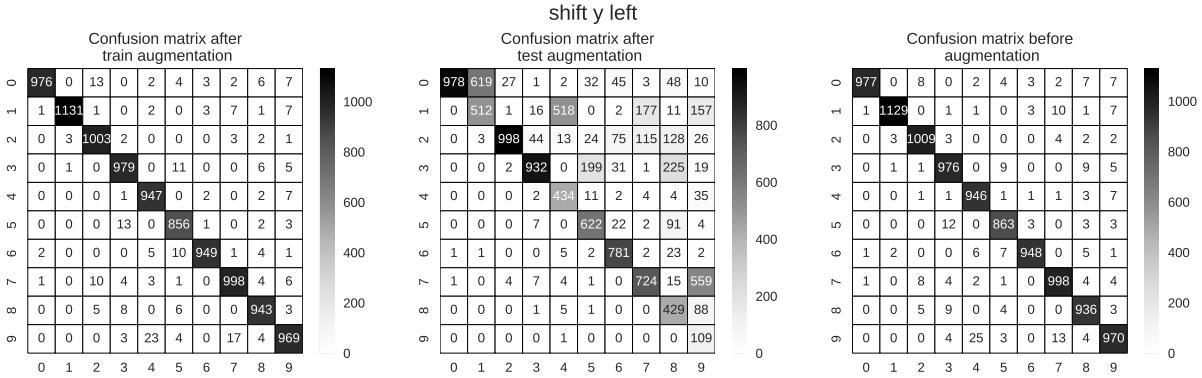
2.6.2 Результаты эксперимента

Какое бы преобразование в отдельности не было выбрано при использовании аугментации тестовой выборки, в каждой матрице число ошибок для каждой цифры лишь увеличивалось относительно ошибок в матрицах ошибок из 4 и 5 эксперимента. Здесь приведем пример лишь одного преобразования (сдвиг влево), которое показывает насколько преобразование в отдельности от других дает плохое предсказание (см. Рис.5)

[5]У нас это 9 векторов-ответов

Тем не менее, если аугментировать тестовую выборку по всем преобразованиям с найденными оптимальными параметрами и далее использовать голосование, то $accuracy \approx 0.9783$, что выше чем без аугментации (в 4 эксперименте мы получили $accuracy \approx 0.9752$). Таким образом, алгоритм на 12.5% делает меньше ошибок.

Рис. 5



2.6.3 Выводы из эксперимента

Аугментировать тестовую выборку по всем преобразованиям с оптимальными параметрами и далее использовать голосование является хорошей альтернативой KNN, имеющего в качестве обучающей выборки исходную обучающую выборку и ее измененные версии, а также неплохой альтернативой KNN, не использующего аугментацию , и причин этому несколько:

- Используется в $\lceil D/N \rceil$ раз меньше памяти, чем для KNN, в котором обучающая выборка объединяется с измененными версиями себя, где D - размерность признакового пространства, а N - количество преобразований.

- Точность при достаточно большом количестве преобразований выше чем у KNN, не использующего аугментация

3 Общие выводы

В рамках проведенного исследования были достигнуты поставленные цели и решены сформулированные в начале исследования задачи. Особенно хотелось бы выделить следующие результаты:

- Размерность признакового пространства влияет на быстродействие алгоритма поиска ближайших соседей. Как правило, чем меньше признаков, тем быстрее *kd_tree* и чем больше признаков тем быстрее *brute*.

- Задача выбора гиперпараметров для KNN (метрика, количество соседей, веса) хорошо решается кросс-валидацией всего с тремя фолдами

- Для сравнения изображений рукописных текстов косинусное расстояние лучше евклидового расстояния оценивается степень близость объектов

- В методе k ближайших соседей играют большую роль веса, назначенные каждому соседу. Оказалось, что результаты классификации становятся точнее при выборе весов, зависящих от расстояния от объекта до соседей

- Модель, построенная на добавлении к обучающей выборке аугментированную, и модель, построенная при помощи аугментации тестовой выборки и голосования, показывают прирост в точности классификации, однако преимуществом второго метода над первым является меньший объем используемой в предсказании памяти компьютера

4 Аппендикс

Рис. 6: Сдвиг на (1, 0) пиксель

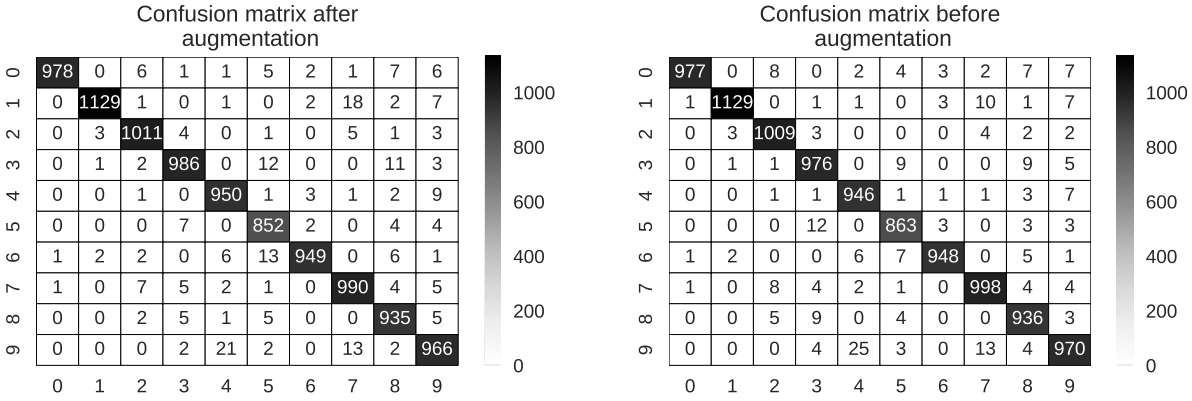


Рис. 7: Сдвиг на (-1, 0) пиксель

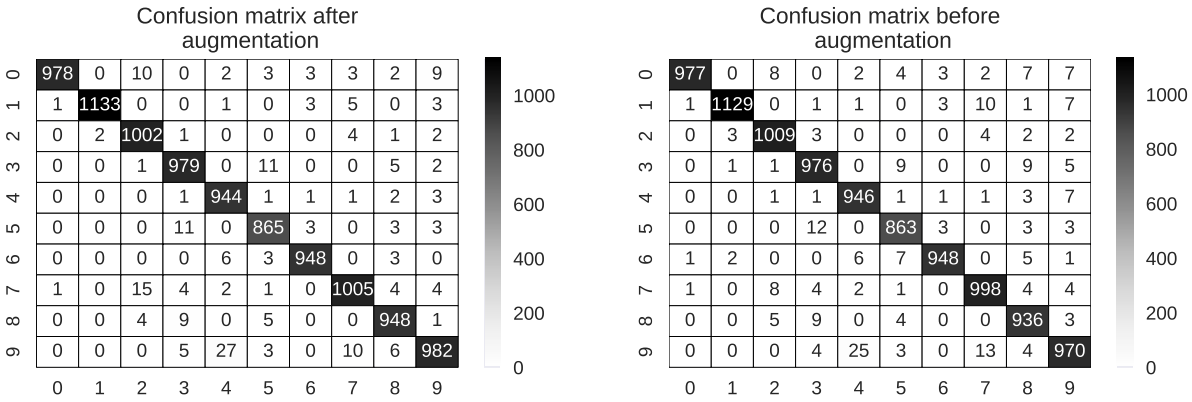


Рис. 8: Сдвиг на (0, 1) пиксель

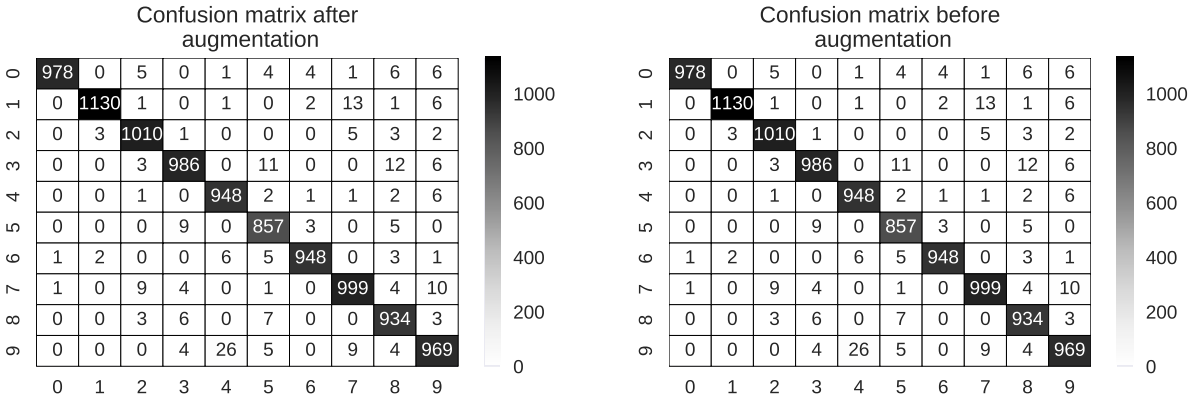


Рис. 9: Сдвиг на (0, -1) пиксель

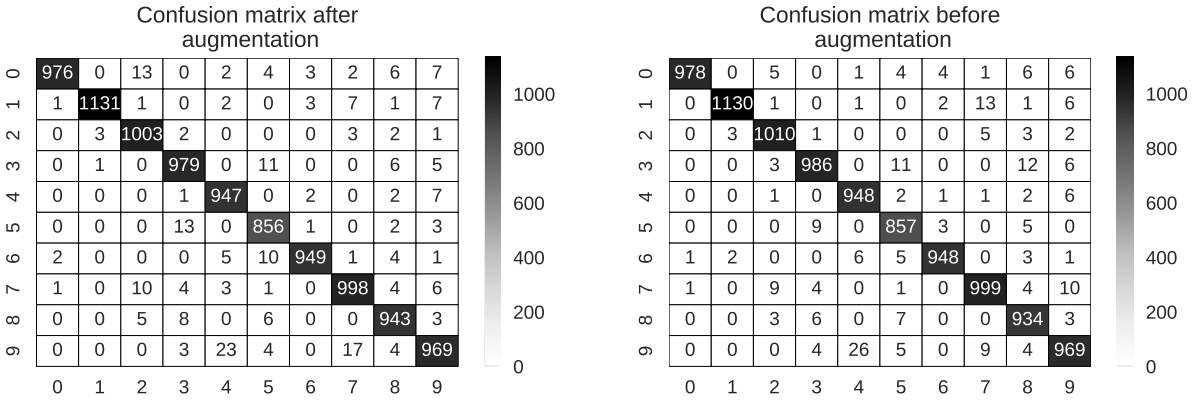


Рис. 10: Поворот на -10

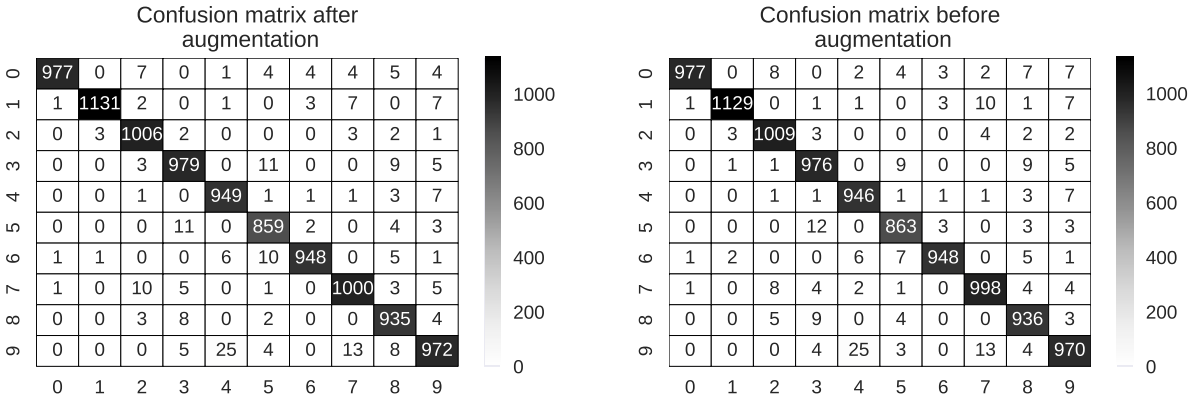


Рис. 11: Поворот на 10

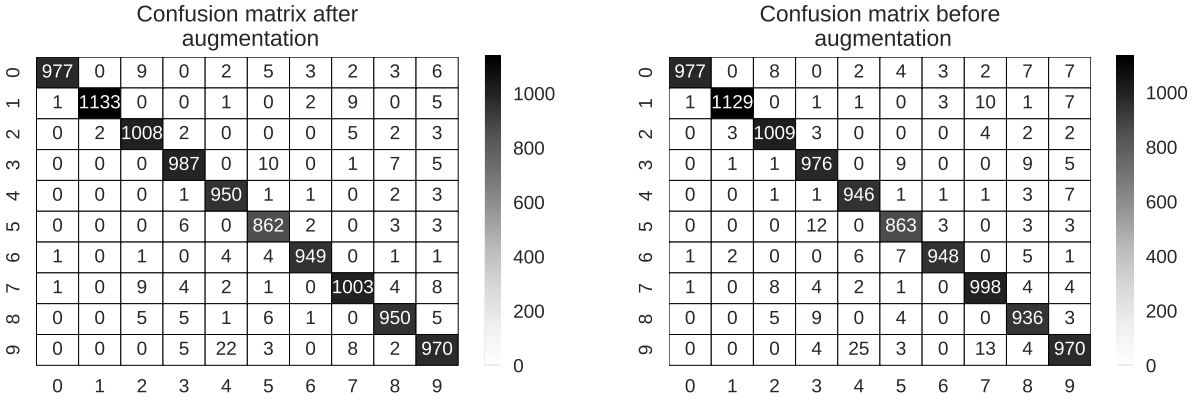


Рис. 12: Размытие по Гауссу с лисперсией фильтра 1.5

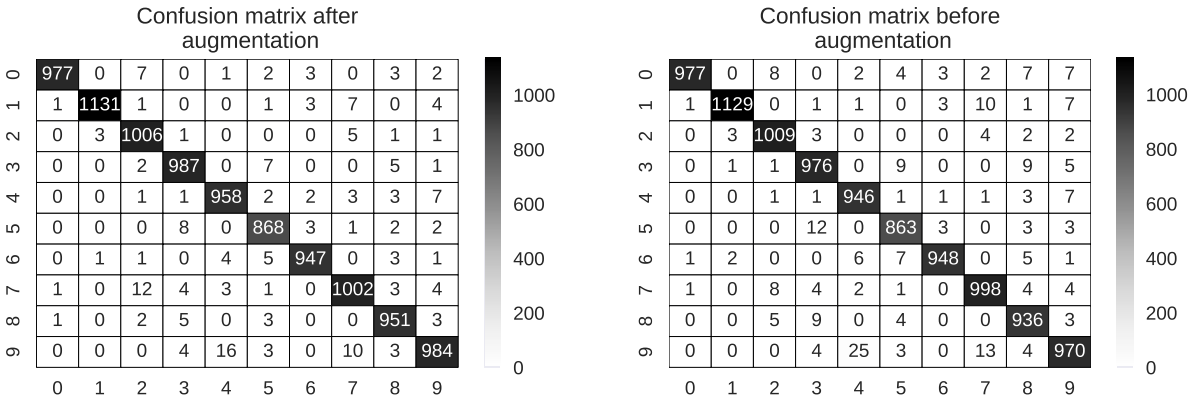


Рис. 13: Дилатация