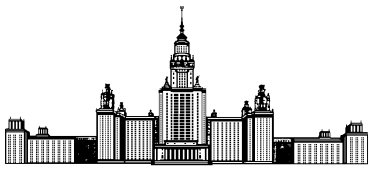
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

Отчет по первому заданию курса "Практикум на ЭВМ"

"Метрические алгоритмы классификации"

Выполнила:
студентка 3 курса 317 группы
Кривуля Полина Юрьевна

Москва, 2022

Содержание

1 Введение

Метод k-ближайших соседей (k Nearest Neighbors, или kNN) – популярный алгоритм классификации и регрессии, который используется в различных типах задач машинного обучения.

В случае использования метода для классификации объект присваивается тому классу, который является наиболее распространённым среди k соседей данного элемента, классы которых уже известны [1].

При взвешенном способе во внимание принимается не только количество попавших в область определённых классов, но и их удалённость от нового значения.

В наиболее общем виде алгоритм ближайших соседей есть

$$a(u) = arg\max_{y \in Y} \sum_{i=1}^{k} [x_{i;j} = y] w(i;u),$$

где $w(i;u)$ - заданная весовая функция [2].

В рамках данного исследования выполнена собственная реализация класса $KNNClassifier$, имеющего метод поиска k ближайших соседей и расстояния между объектом и его соседями, и метод predict, предсказывающий, к какому классу должен принадлежать данный объект. В классе реализован поиск соседей как с весами, так и без.

Также реализован модуль *cross_validation* с реализациями функций для применения кроссвалидации и модуль *distances* с реализацией функций для вычисления евклидова и косинусного расстояния.

В данном отчете проводятся эксперименты, позволяющие выяснить скорость и точность работы алгоритма на датасете MNIST в зависимости от различных параметров, а также исследуются возможные способы увеличения доли правильно предсказанных ответов.

2 Скорость работы алгоритмов в зависимости от числа признаков

2.1 Постановка задачи

Измерьте для каждого алгоритма (kd_tree, ball_tree, brute и my_own) поиска время нахождения 5 ближайших соседей для каждого объекта тестовой выборки по евклидовой метрике. Выберите подмножество признаков, по которому будет считаться расстояние, размера 10, 20, 100 (подмножество признаков выбирается один раз для всех объектов, случайно). Проверьте все алгоритмы поиска ближайших соседей, указанные в спецификации к заданию.

2.2 Эксперименты и результаты

С результатами данного эксперимента можно ознакомиться в таблице 1.

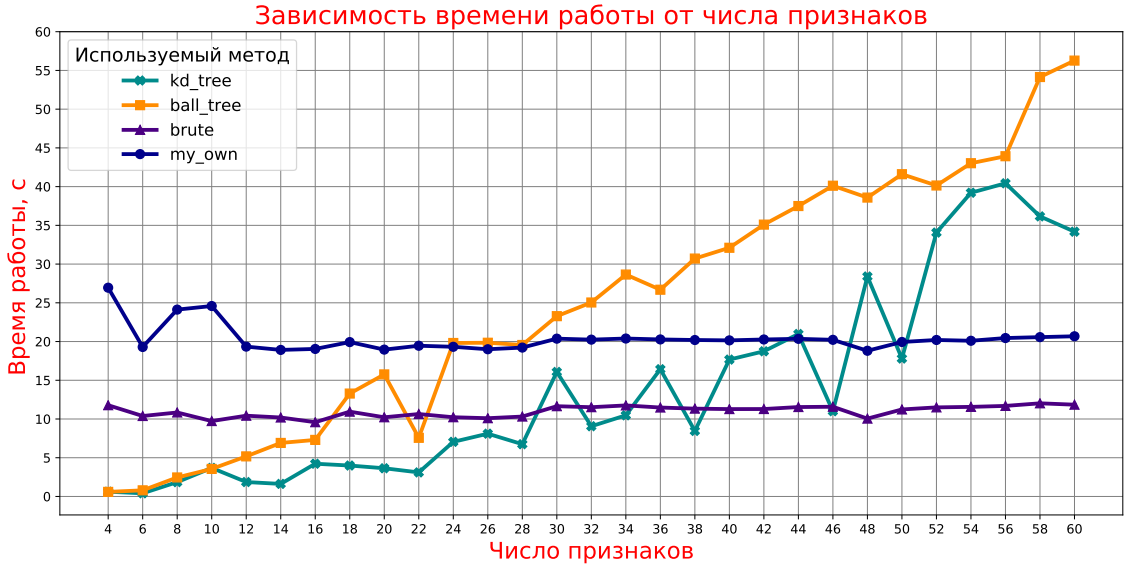
Число признаков \ Метод	kd_tree	ball_tree	brute	my_own
10	0.85 с	4.89 с	14.73 с	15.87 с
20	4.76 с	13.76 с	8.96 с	11.01 с
100	113.04 с	94.59 с	11.65 с	15.92 с

Таблица 1: Время поиска 5 ближайших соседей в зависимости от метода и числа признаков.

Возникают следующие предположения:

1. Методы kd_tree и ball_tree работают неэффективно по сравнению с другими методами при количестве признаков, большем 20
2. Время работы методов brute и my_own не зависит от числа признаков.

Чтобы проверить данные предположения, измерим время нахождения 5 ближайших соседей для каждого объекта тестовой выборки по евклидовой метрике для подмножеств признаков размера 4, 6, 8, ..., 60. Подмножество признаков также выбирается один раз для всех объектов, случайно. Получены следующие результаты:



Методы kd_tree и ball_tree работают быстрее методов brute и my_own, но только при размерности признакового пространства на порядок меньшем, чем представлено в датасете MNIST (в нем размерность равна 784). Поэтому использование данных алгоритмов в рамках дальнейшего решения задачи неэффективно.

Время работы методов brute и my_own, в целом, не зависит от числа признаков. На графике видны небольшие выбросы, которые можно связать со временем работы самой среды или "неудачным" выбором признаков, что подтверждается многократными запусками эксперимента. Алгоритмы устроены так, что время и работы практически не зависит от числа признаков, в большей степени оказывает влияние число соседей и количество объектов в выборке. Данный эксперимент подтвердил выдвинутые ранее предположения.

2.3 Выводы

Методы kd_tree и ball_tree работают эффективно при небольшом количестве признаков (<20). При большом количестве признаков методы работают в разы дольше, чем brute и my_own. Связано это с тем, что kd_tree и ball_tree хранят объекты в виде деревьев. При большой размерности пространства алгоритму приходится посещать больше ветвей дерева, чтобы найти ближайших соседей, и с ростом числа признаков сложность становится примерно такой же, как и в случае полного перебора [3]. Время работы методов brute и my_own не зависит от числа признаков.

Полученные данные согласуются с информацией, найденной в открытых источниках [4].

3 Кросс-валидация с 3 фолдами: точность и время работы

3.1 Постановка задачи

Оцените по кросс-валидации с 3 фолдами точность (долю правильно предсказанных ответов) и время работы k ближайших соседей в зависимости от следующих факторов:

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

Сравните взвешенный метод k ближайших соседей, где голос объекта равен $\frac{1}{distance+\epsilon}$, где $\epsilon = 10^{-5}$, с методом без весов при тех же фолдах и параметрах.

3.2 Эксперименты и результаты

Здесь и далее используем самый быстро работающий метод: **brute.** Запустим реализованную в рамках задания функцию *knn_cross_val_score* со списком значений числа соседей $k = [1, 2, ..., 10]$.

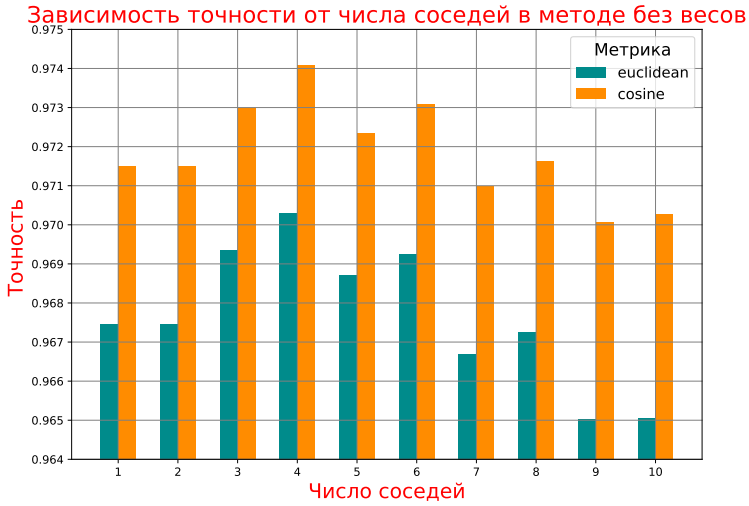
Для используемых метрик без весов получены значения времени работы, представленные в таблице 2.

Используемая метрика	Евклидова	Косинусная
Время работы, с	135.9	122.6

Таблица 2: Время работы метода без весов.

Здесь и далее долю правильно предсказанных ответов будем называть точностью.

Получены значения точности, представленные на диаграмме ниже.



Косинусная метрика показала точность лучше, чем евклидова. Это можно объяснить тем, что косинусная метрика определяет степень сходства между объектами, и потому она менее чувствительна к сдвигам и растяжениям. Также она сработала незначительно (на 10%) быстрее.

Заметим, что при $k = 2$ точность такая же, как и при $k = 1$. Это происходит из-за особенностей реализации функции *knn_cross_val_score*: из-за линейности алгоритма ответы при каждом k неявно зависят от ответов на предыдущих значениях, а в случае ситуации неоднозначного выбора класса для объекта, выбирается не случайный класс, а первый добавленный. То есть для $k = 2$ получается, что класс всегда будет выбран как для $k = 1$. Неслучайностью выбора в случае неоднозначного класса можно объяснить и то, что при нечетных k результат хуже, чем на предыдущих четных (кроме $k = 3$, что уже объяснено). При $k > 4$ точность начинает падать. Это объясняется тем, что в данной выборке объектов классов всего 10, поэтому при значениях k, близких к 10, алгоритм чрезмерно устойчив к шумовым выбросам и вырождается в константу [2].

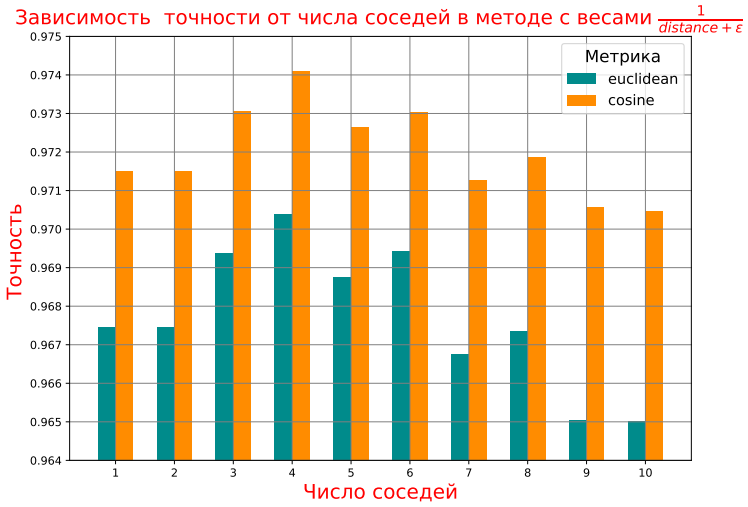
Для используемых метрик с весами с голосом объекта $\frac{1}{distance+\epsilon}$, где $\epsilon = 10^{-5}$, получены значения времени работы, представленные в таблице 3.

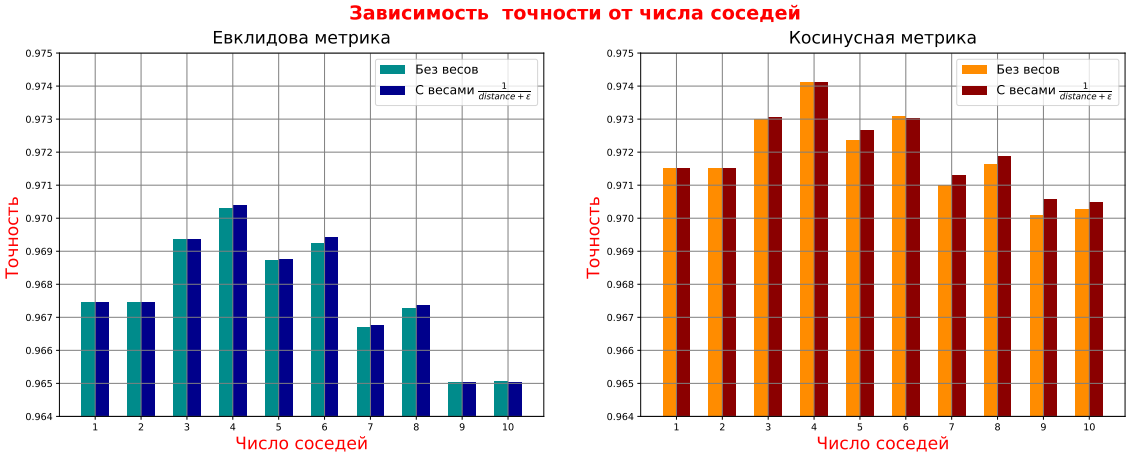
Используемая метрика	Евклидова	Косинусная
Время работы, с	129.9	123.1

Таблица 3: Время работы метода с весами.

Время работы незначительно уменьшилось, так как при взвешенном методе возникает меньше неоднозначных ситуаций.

Получены значения точности, представленные на диаграммах ниже.





При $k = 2$, как и в методе без весов, результат такой же, что и при $k = 1$. Аналогично методу без весов, это связано с линейностью алгоритма (а также с особенностью используемой функции веса, описанной ниже). Ближайший сосед (с бóльшим весом) учтен при $k = 1$, поэтому ответы при $k = 2$ совпадают с предыдущими.

Используемая функция веса объекта такова, что чем меньше расстояние, тем больше вес, следовательно, чем меньше номер соседа (номера упорядочены по удаленности от объекта, на котором ищется ответ), тем большее влияние он оказывает на ответ. Поэтому при $k = 1$ ответ будет таким, какой он есть на первом соседе вне зависимости от факта взвешенности метода, и при $k = 1, 2$ результат одинаковый в методе без весов и с весами. Метод с весами позволяет избежать возникающих неоднозначных ситуаций, и в большинстве случаев точность повышается.

3.3 Выводы

При числе соседей $k = 1, 2$ ответы совпадают в обоих методах. Косинусная метрика показала результат лучше, чем евклидова, а максимальная точность достигается при $k = 4$ косинусной метрики (одинаковая в методе с весами и без). При значениях k, близких к 10, точность падает, так как алгоритм становится чрезмерно устойчив к шумовым выбросам и вырождается в константу [2].

4 Применение лучшего алгоритма к исходной обучающей и тестовой выборке

4.1 Постановка задачи

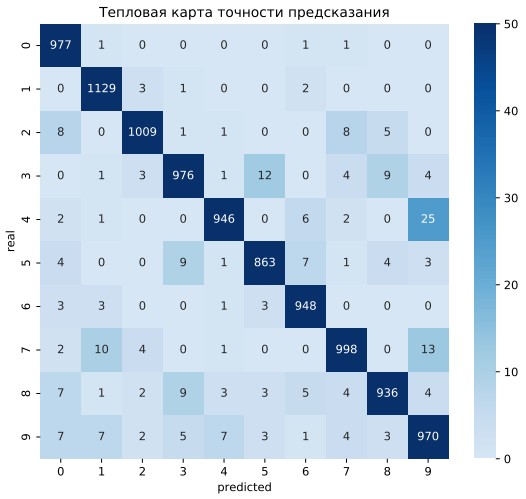
Примените лучший алгоритм к исходной обучающей и тестовой выборке. Подсчитайте точность. Сравните с точностью по кросс-валидации. Сравните с указанной в интернете точностью лучших алгоритмов на данной выборке. Выполните анализ ошибок. Для этого необходимо построить и проанализировать матрицу ошибок (confusion matrix). Также визуализируйте несколько объектов из тестовой выборки, на которых были допущены ошибки. Проанализируйте и укажите их общие черты.

4.2 Эксперименты и результаты

По результатам предыдущего эксперимента лучшим алгоритмом вышла косинусная метрика для $k = 4$ с весами и без. Так как метод с весами показал себя не хуже, чем метод без весов, используем далее его. Ранее было выяснено, что быстрее остальных методов работает brute. Поэтому в этом и следующих экспериментах используем косинусную метрику, число соседей $k = 4$, взвешенный метод brute. Функция весов остается прежней.

При использовании данного алгоритма получена точность (доля правильно правильно предсказанных ответов) на тестовой выборке 0.9752. При этом лучшая (среднняя по фолдам) точность, полученная по кросс-валидации с 3 фолдами, равна 0.9741. Ее более низкое значение обусловлено тем, что по кросс-валидации обучение происходит на меньшем числе объектов: на 40 тыс., а тестирование на 20 тыс., в то время как размер всей обучающей выборки равен 60 тыс. объектов, а тестовой - 10 тыс.. Лучшая точность на данной выборке, представленная на Kaggle, равна 0.9998 [4].

Визуализация матрицы ошибок в данном эксперименте представлена ниже.



Чаще всего неправильно распознавалась "9": в 25 случаях реальная "4" была распознана алгоритмом как "9", и в 39 случаях реальная "9" распознана неправильно. Ниже представлены оба случая, причем справа над каждом объектом написано, какая именно цифра была предсказана.

Заметим, что в первом случае большинство неправильно предсказанных "4" написаны с "замыканием" сверху (как печатная, а не письменная версия цифры), и поэтому они становятся похожими на "9". Во втором случае часть "9" написаны так, что и человек может неправильно их распознать. Если такие объекты попадают в обучающую выборку, то это оказывает негативное влияние на дальнейшее распознавание алгоритмом правильно написанных цифр.

Ниже представлены 50 случайно выбранных из неправильно распознанных цифр. Многие из этих цифр написаны непонятно, поэтому сверху каждой написано, какая это на самом деле цифра ("Real") и как эту цифру распознал алгоритм ("Pred").

Помимо непонятно написанных цифр (которые являются неинформативными объектами и/или выбросами), алгоритм часто неправильно распознает нечеткие и повернутые цифры. Данную проблему можно решать, в частности, с помощью аугментации.

4.3 Выводы

Доля правильно предсказанных данным алгоритмом ответов на тестовой выборке на 0.0246 меньше, чем лучшая на Kaggle. Алгоритм неправильно распознает в целом для человека непонятно написанные цифры (возможно, выбросы), а также нечеткие, повернутые и сдвинутые.

5 Аугментация обучающей выборки

5.1 Постановка задачи

Выполните аугментацию обучающей выборки. Для этого нужно размножить ее с помощью поворотов, смещений, морфологических операций и применений гауссовского фильтра. Разрешается использовать библиотеки для работы с изображениями. Подберите по кросс-валидации с 3 фолдами параметры преобразований. Рассмотрите следующие параметры для преобразований и их комбинации:

- 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.

5.2 Эксперименты и результаты

В данном эксперименте возникла необходимость немного изменить функцию $knn\_cross\_val\_score$ и сделать в ней случайный выбор индексов (иначе при некоторых параметрах преобразования точность выдавалась как равная 1).

Напомним, что исходная точность по кросс-валидации с 3 фолдами была равна 0.9741.

- Величина поворота: 5, 10, 15 (в каждую из двух сторон)

  При реализации данной аугментации размер обучающей выборки увеличивается в 3 раза: помимо самой выборки к ней добавляется она же, повернутая на ϕ° в одну сторону и на ϕ° в другую.

  В результате по кросс-валидации с 3 фолдами получены результаты, представленные в таблице 4.

Величина поворота	5°	10°	15°
Точность	0.9935	0.9910	0.9879

Таблица 4: Точность в зависимости от величины поворота.

  Визуализации матриц ошибок для каждого случая представлены в приложении (5° - 1(a), 10° - 1(b), 15° - 1(c)).

  При повороте на 5° неправильно предсказанных ответов стало количественно меньше, особенно в случае с цифрой "9". При повороте на 10° и 15°, наоборот, ошибок в этом случае становится больше (особенно когда реальная цифра "4" распознается как "9") и добавляются другие ошибки. Но данные преобразования помогают исправить обозначенную ранее ошибку распознавания повернутого объекта.

- Величина смещения: 1, 2, 3 пикселя (по каждой из двух размерностей)

  При реализации данной аугментации размер обучающей выборки увеличивается в 3 раза: помимо самой выборки к ней добавляется она же, смещенная на α пикселей в одну сторону и на α в другую. Сторона смещения подбиралась как дающая лучшую точность на значении $\alpha = 1$, что представлено в таблице 5.

Пара векторов смещений	Полученная точность
[1, 1], [-1, -1]	0.9776
[1, 0], [-1, 0]	0.9842
[1, 0], [0, -1]	0.9842
[0, 1], [0, -1]	0.9840
[1, 0], [0, 1]	0.9850

Таблица 5: Точность в зависимости направлений смещения на 1 пиксель.

Лучшая точность была достигнута на последней паре векторов. Поэтому данная пара и будет использоваться в смещениях на 2 и 3 пикселя. Полученные значения представлены в таблице 6.

Величина смещения, пикселей	1	2	3
Точность	0.9850	0.9766	0.9758

Таблица 6: Точность в зависимости от величины сдвига.

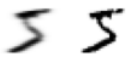
Визуализации матриц ошибок для каждого случая величины сдвига (по паре векторов смещения [1, 0], [0, 1]) представлены в приложении (смещение на 1 пиксель - 1(d), на 2 - 1(e), на 3 - 1(f)).

Данные преобразования помогают исправить обозначенную ранее ошибку распознавания сдвинутого объекта, причем лучшую точность дает смещение на 1 пиксель, а оставшиеся смещения количественно добавляют новые ошибки.

- Дисперсия фильтра Гаусса: 0.5, 1, 1.5

  При реализации данной аугментации размер обучающей выборки увеличивается в 2 раза: помимо самой выборки к ней добавляется она же, преобразованная с использованием фильтра Гаусса с заданной дисперсией.

  Дисперсия 1.5 сильно портит объекты, что представлено ниже.



  Для фильтра Гаусса с дисперсией 0.5 и 1 по кросс-валидации с 3 фолдами получены результаты, представленные в таблице 7.

Дисперсия фильтра Гаусса	0.5	1
Точность	0.9926	0.9912

Таблица 7: Точность в зависимости от дисперсии фильтра Гаусса.

Оба фильтра дают хорошие значения точности. Визуализации матриц ошибок для каждого случая представлены в приложении (с дисперсией 0,5 - 1(k), с дисперсией 1 - 1(l)). Данные преобразования помогают исправить обозначенную ранее ошибку распознавания размытого объекта.

- Морфологические операции: эрозия, дилатация, открытие, закрытие с ядром 2

  При реализации данной аугментации размер обучающей выборки увеличивается в 2 раза: помимо самой выборки к ней добавляется она же, преобразованная с использованием морфологической операции.

Морфологические операции	Эрозия	Дилатация	Открытие	Закрытие
Точность	0.9863	0.9876	0.9823	0.9819

Таблица 8: Точность в зависимости от используемой морфологической операции.

Визуализации матриц ошибок для каждого случая представлены в приложении (эрозия - 1(j), дилатация - 1(i), открытие - 1(g), закрытие - 1(h)). При сравнении полученных матриц ошибок с изначальной, можно заметить, что данные преобразования не дают заметного повышения точности. Скорее всего, на кросс-валидации точность повысилась больше за счет увеличения выборки, чем за счет данных преобразований.

- Комбинация различных преобразований

  Для комбинации различных преобразований реализована функция augm, которая в зависимости от параметров, поступающих на вход, преобразовывает выборку следующим образом: случайно выбирается поворот в диапазоне $[-5°; 5°]$, и/или случайным образом сдвигает на -1, 0, 1 пиксель по обеим размерностям, и/или случайным образом решает, преобразовывать ли с помощью фильтра Гаусса с дисперсией 0.5, и/или случайно выбирает морфологическую операцию. Все перечисленные действия применяются к каждому объекту.

  0. Увеличим исходную выборку в 3 раза: преобразуем ее с помощью поворота на 5° в обе стороны и фильтра Гаусса с дисперсией 0.5.

     Таким образом получена точность 0.9935, что не отличается от полученной при данном повороте без фильтра Гаусса (см. таблица 4). Матрица ошибок представлена в приложении (1(m))

  1. Случайный выбор параметров 1

     Увеличим исходную выборку в 3 раза. Одну копию исходной выборки преобразуем с помощью функции augm для случайного поворота и фильтра Гаусса, на второй с помощью функции augm сделаем все случайные преобразования. Получена точность 0.9828, что хуже предыдущего результата. Матрица ошибок представлена в приложении (1(n)).

  2. Случайный выбор параметров 2

     Увеличим исходную выборку в 3 раза. Одну копию исходной выборки преобразуем с помощью функции augm для случайного поворота и фильтра Гаусса, на второй с помощью функции augm сделаем оставшиеся случайные преобразования (сдвиг на 1 пиксель, морфологические операции). Получена точность 0.9827. Матрица ошибок представлена в приложении (1(o)).

  3. Случайный выбор параметров 3

     Увеличим исходную выборку в 3 раза. Одну копию исходной выборки преобразуем с помощью функции augm для случайного поворота и фильтра Гаусса, на второй с помощью функции augm сделаем случайный поворот. Получена точность 0.9880. Матрица ошибок представлена в приложении (1(p))

  4. Выбор параметров 4

     Увеличим исходную выборку в 4 раза. Одну копию исходной выборки преобразуем с помощью функции augm для всех случайных параметров, вторую преобразуем с помощью (не случайного) поворота на 5°, третью с помощью поворота на $-5°$. Также ко второй и третьей выборке применим фильтр Гаусса с дисперсией 0.5. Получена точность по кросс-валидации с 3 фолдами 0.9944, что лучше всего опробованного ранее. Матрица ошибок представлена в приложении (1(q)). Увеличились все правильные предсказания, кроме "6" (числа на диагонали матрицы), то есть, увеличилось число правильно предсказанных ответов на тестовой выборке. В случае "6" добавились 3 случая неправильного предсказания, что по отношению к правильно распознанной цифрой "6" (945 случаев) незначительно.

     При использовании данной аугментированной обучающей выборки, доля правильно предсказанных ответов на тестовой выборке увеличивается с 0.9752 до 0.9809.

5.3 Выводы

С помощью аугментации обучающей выборки в экспериментах получено увеличение точности по кросс-валидации. Исходная точность по кросс-валидации с 3 фолдами была равна 0.9741, а с помощью увеличения выборки в 4 раза и последующих преобразований было достигнуто значение 0.9944.

6 Преобразование тестовой выборки

6.1 Постановка задачи

Реализуйте описанный выше алгоритм, основанный на преобразовании объектов тестовой выборки. Проверьте то же самое множество параметров, что и в предыдущем пункте. Проанализируйте как изменилась матрица ошибок, какие ошибки алгоритма помогает исправить каждое преобразование. Качественно сравните два подхода (5 и 6 пункты) между собой.

6.2 Эксперименты и результаты

Напомним, что исходная точность, посчитанная как доля правильно предсказанных ответов на тестовой выборке, равна 0.9752.

- Величина поворота: 5, 10, 15 (в каждую из двух сторон)

  Случайным образом выбирается, в какую именно сторону поворачивать объект. В таблице 9 представлено, как преобразование тестовой выборки влияет на долю полученных правильных ответов (на преобразованной выборке).

Величина поворота	5°	10°	15°
Точность	0.9723	0.9669	0.9421

Таблица 9: Точность в зависимости от величины поворота.

  Визуализации матриц ошибок для каждого случая представлены в приложении (5° - 2(a), 10° - 2(b), 15° - 2(c)).

  Голосование среди полученных ответов на исходной выборке и на одном повороте бессмысленно, более подробно было описано в пункте 3.2.

  При использовании всех трех преобразований и голосовании среди них и полученных ответов на исходной выборке, результат хуже, чем для просто поворота на 5°: получена точность 0.9661. Были проверены всевозможные комбинации преобразований и голосования среди полученных результатов и единственное улучшение было достигнуто при следующем варианте:

  Принятие решения путем голосования среди ответов на тестовой выборке, повернутой на 5°, повернутой на $-5°$ и исходной. Таким образом была достигнута точность 0.9763. Матрица ошибок представлена в приложении (2(g)).

- Величина смещения: 1, 2, 3 пикселя (по каждой из двух размерностей)

  Голосование среди полученных ответов на исходной выборке, на смещении на α пикселей по вектору $[\alpha; 0]$, на смещении на α пикселей по вектору $[0; \alpha]$. Полученные результаты представлены в таблице 10

Величина смещения	1	2	3
Доля правильных ответов	0.9712	0.9455	0.8681

Таблица 10: Точность в зависимости от величины смещения.

Визуализации матриц ошибок для каждого случая представлены в приложении (1 - 2(d), 2 - 2(e), 3 - 2(f)).

Были проверены всевозможные комбинации преобразований и голосования среди полученных результатов и улучшение было достигнуто при следующем варианте:

Голосование среди полученных ответов на исходной выборке и всевозможных смещениях на 1. Получена точность 0.9761. Матрица ошибок представлена в приложении (2(h)).

- Дисперсия фильтра Гаусса: 0.5, 1, 1.5

  В таблице 11 представлено, как данные преобразования тестовой выборки влияют на долю полученных правильных ответов (на преобразованной выборке). При этом известно, что фильтр Гаусса с дисперсией 1.5 уже достаточно сильно портит объекты.

Дисперсия фильтра Гаусса	0.5	1	1.5
Доля правильных ответов	0.9742	0.9711	0.9623

Таблица 11: Точность в зависимости от дисперсии фильтра Гаусса.

Визуализации матриц ошибок для каждого случая представлены в приложении (дисперсия 0.5 - 2(i), дисперсия 1 - 2(j), дисперсия 1.5 - 2(k)).

Были реализованы различные комбинации преобразований из данного пункта и принятие решения путем голосования, но улучшение по сравнению с исходной точностью достигнуто не было. Например, голосованием среди преобразования с дисперсией 0.5, с дисперсией 1 и ответов на исходной тестовой выборке, была получена точность 0.9743 При этом просто на исходной достигается точность 0.9752.

- Морфологические операции: эрозия, дилатация, открытие, закрытие с ядром 2

  В таблице 12 представлено, как данные преобразования тестовой выборки влияет на долю полученных правильных ответов (на преобразованной выборке).

Морфологические операции	Эрозия	Дилатация	Открытие	Закрытие
Доля правильных ответов	0.9665	0.9638	0.9475	0.9481

Таблица 12: Точность в зависимости от морфологической операции

Визуализации матриц ошибок для каждого случая представлены в приложении (эрозия - 2(l), дилатация - 2(m), открытие - 2(n), закрытие - 2(o)).

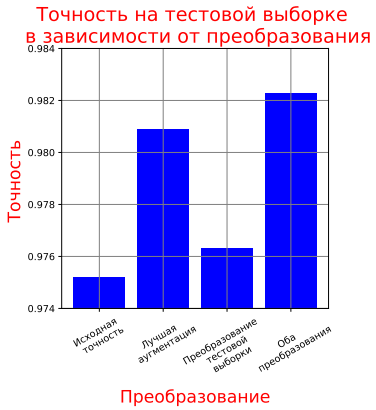
Данные преобразования дают точность хуже, чем все предыдущие и по матрицам ошибок видно, что ошибки появляются практически в одних и тех же местах, поэтому голосование среди них бесполезно.

- Обобщение

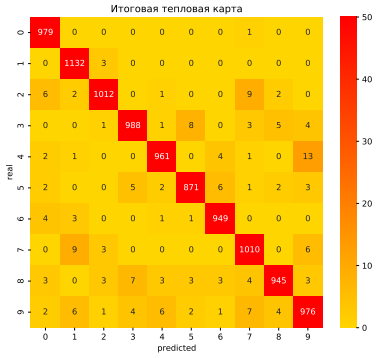
  Если сделать голосование среди лучших полученных результатов (в экспериментах 1 и 2 данного пункта), то точность будет как в эксперименте 1 (0.9763).

- Обобщение с аугментацией

  Совместим аугментацию, на которой был достигнут лучший результат (комбинация преобразований 4) и обобщение из предыдущего подпункта. Таким способом получена точность 0.9823. Напомним, что исходная точность на тестовой выборке (не по кросс-валидации) была равна 0.9752, а доля правильно предсказанных ответов на аугментированной обучающей выборке - 0.9809. Более наглядно это отображено на диаграмме ниже.



А также полученный результат визуализирован с помощью матрицы ошибок ниже.



Можно заметить, что теперь нет цифр, которые заметно чаще распознаются неправильно (как было ранее с "4" и "9").

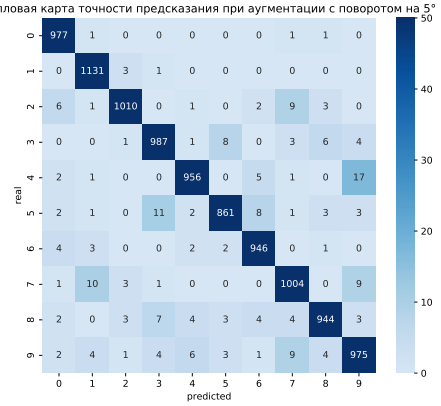
6.3 Выводы

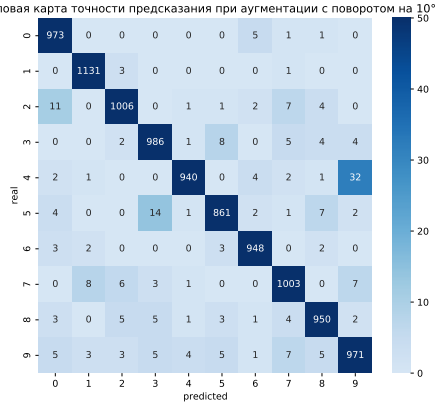
При аугментации точность при почти любых преобразованиях повышается, прежде всего за счет увеличения обучающей выборки, но при этом значительно увеличивается время выполнения программы. На обучающей выборке исходного объема (60 тыс. изображений) время поиска ответов на стандартной тестовой выборке равно 50 секундам. При увеличении выборки в 4 раза время увеличивается до 183 секунд. При преобразовании объектов тестовой выборки улучшение достигается только в некоторых случаях путем комбинаций различных преобразований и принятия решения голосованием, но программа выполняется быстро, так как объем обучающей выборки не изменяется.

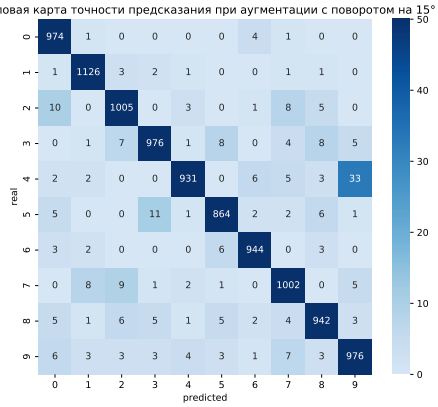
7 Выводы

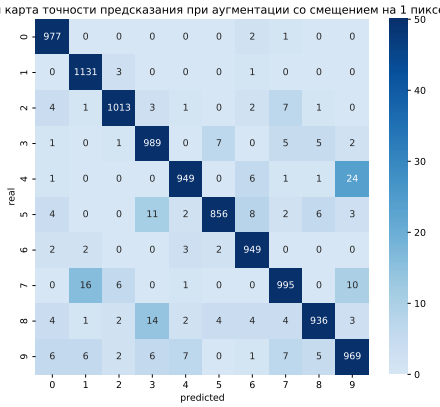
В данном исследовании лучшее значение доли правильно предсказанных ответов на тестовой выборке равно 0.9823. Такая точность достигнута при использовании метода brute с весами с голосом объекта $\frac{1}{distance+\epsilon}$, где $\epsilon = 10^{-5}$, и числом соседей k=4, а также совместном преобразовании объектов обучающей и тестовой выборки. При преобразовании обучающей выборки исходная выборка была увеличена в 4 раза и первая копия была преобразована почти случайным образом, вторая с помощью (не случайного) поворота на 5°, третья с помощью поворота на $-5°$. Также ко второй и третьей выборке был применен фильтр Гаусса с дисперсией 0.5. Преобразование тестовой выборки было выполнено несколько раз путем сдвига во всевозможные стороны на 1 пиксель, поворота в обе стороны на 5°, а решение было принято среди полученных ответов путем голосования. Точность метода без преобразований была равна 0.9752.

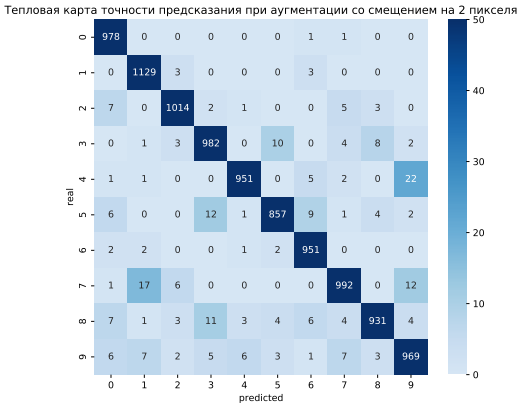
Приложение

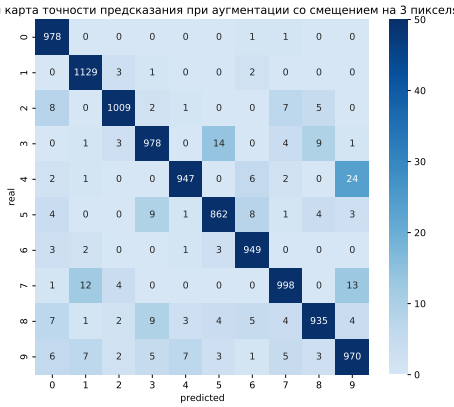

Тепловая карта точности предсказания при аугментации с поворотом на 5°
real
predicted


Тепловая карта точности предсказания при аугментации с поворотом на 10°
real
predicted

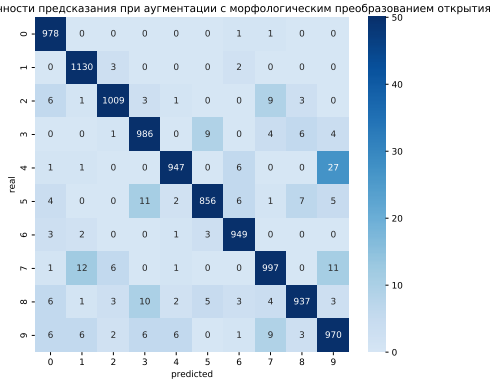

Тепловая карта точности предсказания при аугментации с поворотом на 15°
real
predicted


Тепловая карта точности предсказания при аугментации со смещением на 1 пиксель
real
predicted

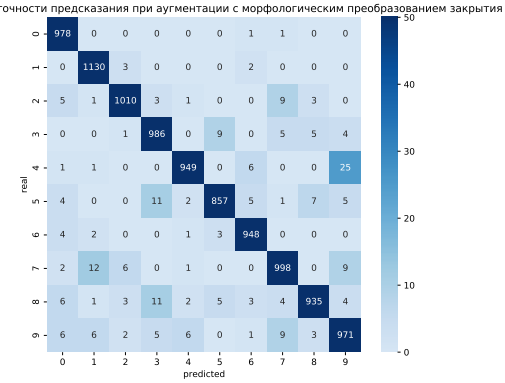

Тепловая карта точности предсказания при аугментации со смещением на 2 пикселя
real
predicted


Тепловая карта точности предсказания при аугментации со смещением на 3 пикселя
real
predicted

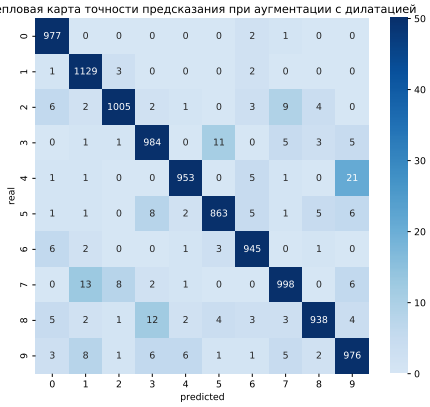
Тепловая карта точности предсказания при аугментации с морфологическим преобразованием открытия



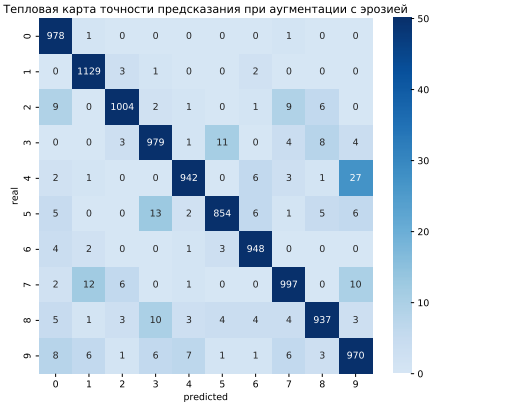
Тепловая карта точности предсказания при аугментации с морфологическим преобразованием закрытия



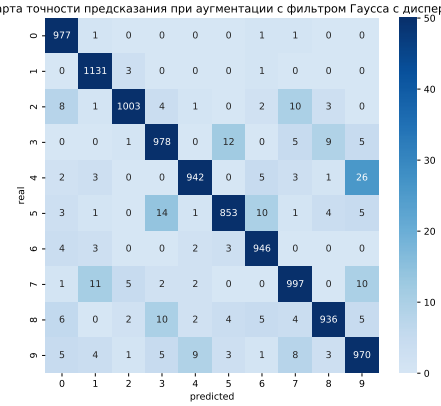
Тепловая карта точности предсказания при аугментации с дилатацией



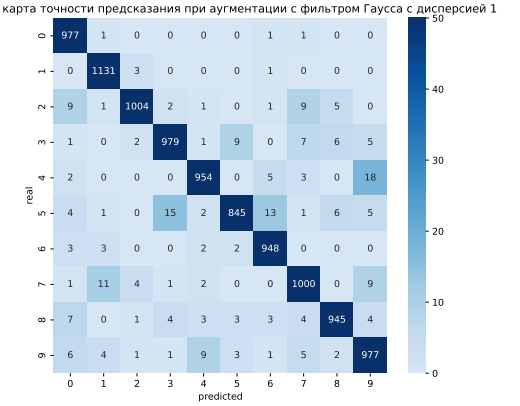
Тепловая карта точности предсказания при аугментации с эрозией

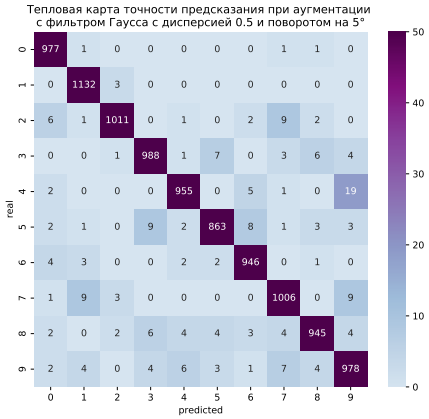


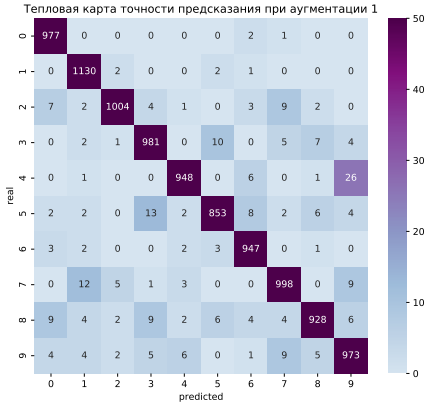
Тепловая карта точности предсказания при аугментации с фильтром Гаусса с дисперсией 0.5

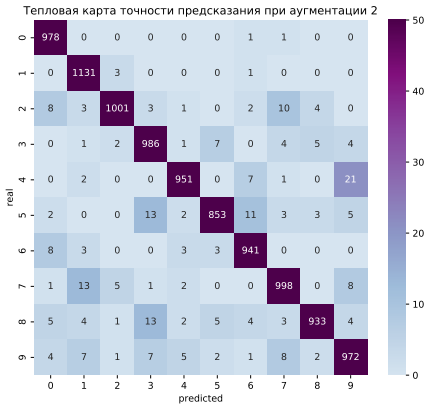


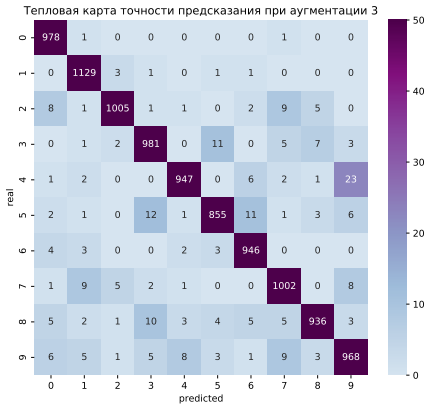
Тепловая карта точности предсказания при аугментации с фильтром Гаусса с дисперсией 1

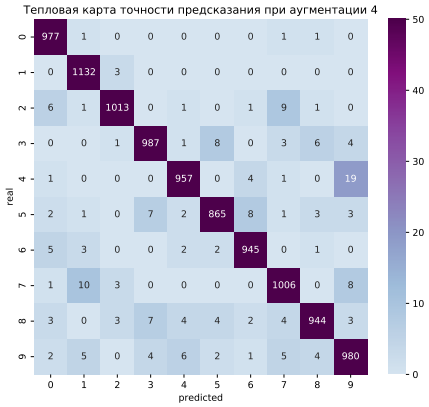


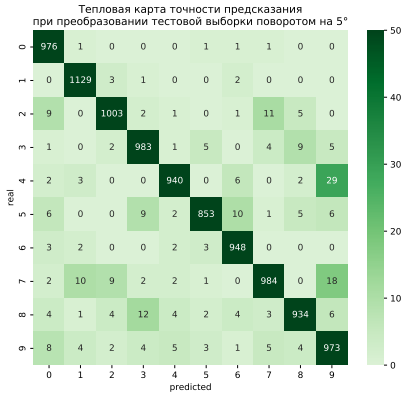

Тепловая карта точности предсказания при аугментации с фильтром Гаусса с дисперсией 0.5 и поворотом на 5°
real
predicted

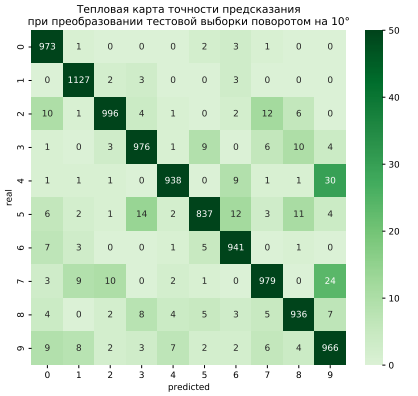

Тепловая карта точности предсказания при аугментации 1
real
predicted

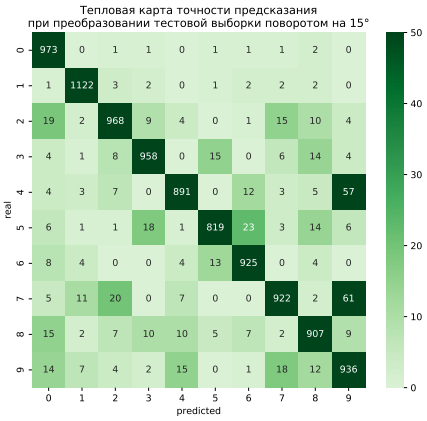

Тепловая карта точности предсказания при аугментации 2
real
predicted

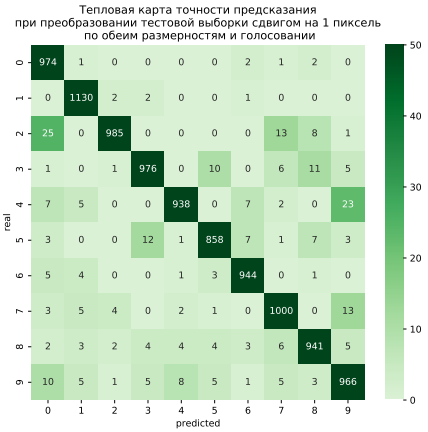

Тепловая карта точности предсказания при аугментации 3
real
predicted

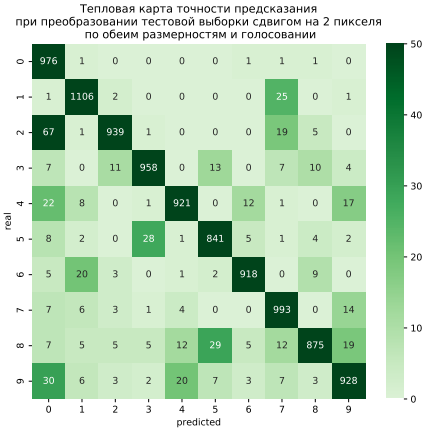

Тепловая карта точности предсказания при аугментации 4
real
predicted

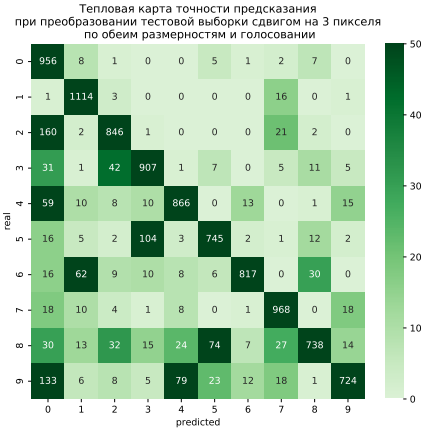

Тепловая карта точности предсказания
при преобразовании тестовой выборки поворотом на 5°
real
predicted

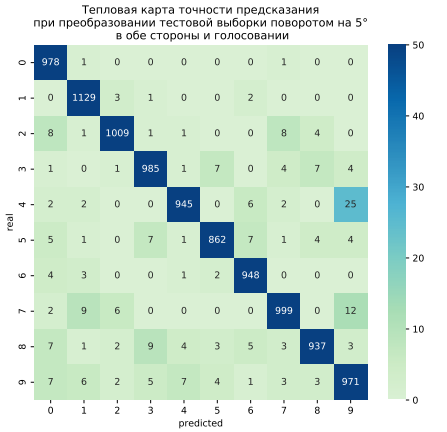

Тепловая карта точности предсказания
при преобразовании тестовой выборки поворотом на 10°
real
predicted

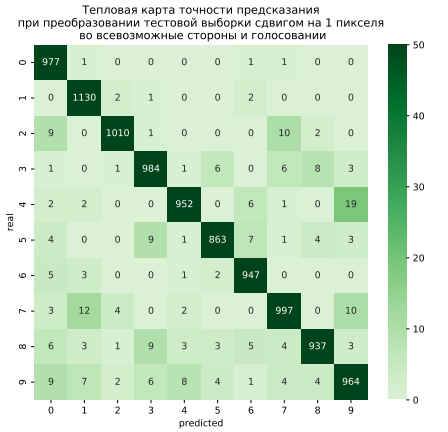

Тепловая карта точности предсказания
при преобразовании тестовой выборки поворотом на 15°
real
predicted

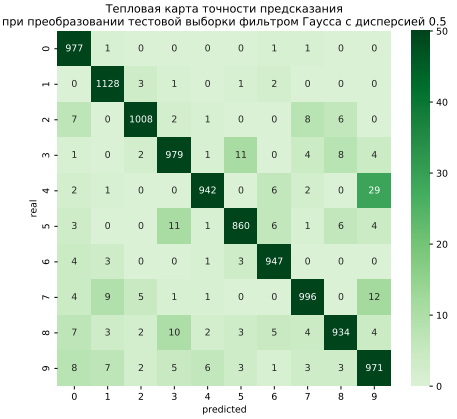

Тепловая карта точности предсказания
при преобразовании тестовой выборки сдвигом на 1 пиксель
по обеим размерностям и голосовании
real
predicted

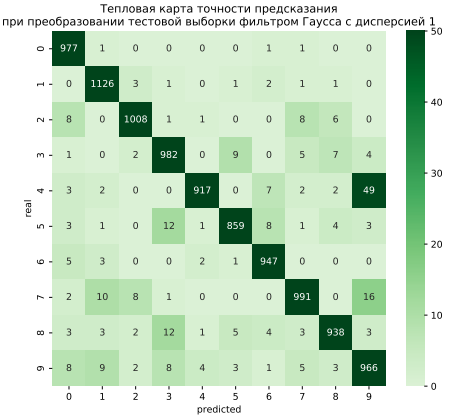

Тепловая карта точности предсказания
при преобразовании тестовой выборки сдвигом на 2 пикселя
по обеим размерностям и голосовании
real
predicted

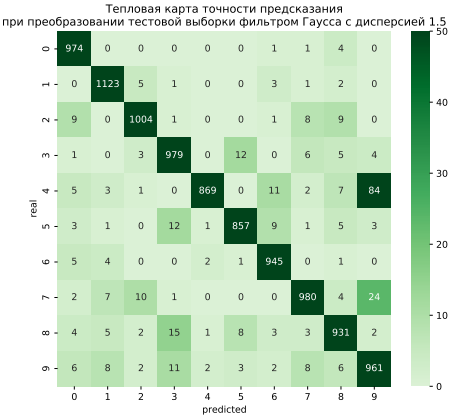

Тепловая карта точности предсказания
при преобразовании тестовой выборки сдвигом на 3 пикселя
по обеим размерностям и голосовании
real
predicted

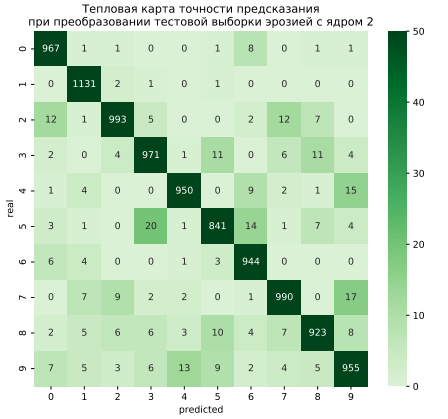

Тепловая карта точности предсказания
при преобразовании тестовой выборки поворотом на 5°
в обе стороны и голосовании
real
predicted

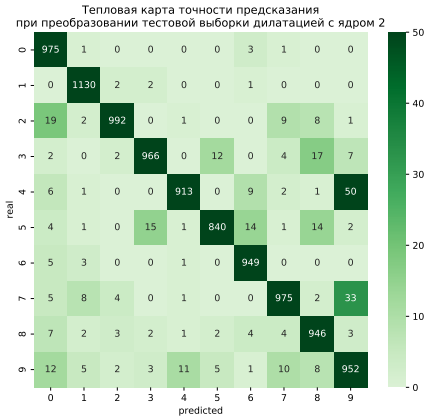

Тепловая карта точности предсказания
при преобразовании тестовой выборки сдвигом на 1 пикселя
во всевозможные стороны и голосовании
real
predicted

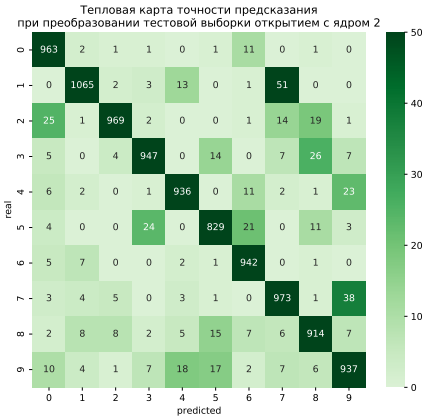

Тепловая карта точности предсказания
при преобразовании тестовой выборки фильтром Гаусса с дисперсией 0.5
real
predicted

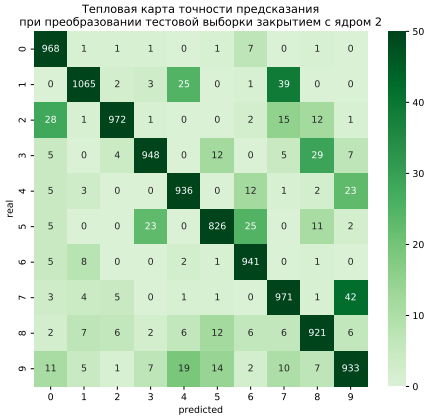

Тепловая карта точности предсказания
при преобразовании тестовой выборки фильтром Гаусса с дисперсией 1
real
predicted


Тепловая карта точности предсказания
при преобразовании тестовой выборки фильтром Гаусса с дисперсией 1.5
real
predicted


Тепловая карта точности предсказания
при преобразовании тестовой выборки эрозией с ядром 2
real
predicted


Тепловая карта точности предсказания
при преобразовании тестовой выборки дилатацией с ядром 2
real
predicted
0 1 2 3 4 5 6 7 8 9
975 1 0 0 0 0 3 1 0 0
0 1130 2 2 0 0 1 0 0 0
19 2 992 0 1 0 0 9 8 1
2 0 2 966 0 12 0 4 17 7
6 1 0 0 913 0 9 2 1 50
4 1 0 15 1 840 14 1 14 2
5 3 0 0 1 0 949 0 0 0
5 8 4 0 1 0 0 975 2 33
7 2 3 2 1 2 4 4 946 3
12 5 2 3 11 5 1 10 8 952
0 10 20 30 40 50


Тепловая карта точности предсказания
при преобразовании тестовой выборки открытием с ядром 2
real
predicted
0 1 2 3 4 5 6 7 8 9
963 2 1 1 0 1 11 0 1 0
0 1065 2 3 13 0 1 51 0 0
25 1 969 2 0 0 1 14 19 1
5 0 4 947 0 14 0 7 26 7
6 2 0 1 936 0 11 2 1 23
4 0 0 24 0 829 21 0 11 3
5 7 0 0 2 1 942 0 1 0
3 4 5 0 3 1 0 973 1 38
2 8 8 2 5 15 7 6 914 7
10 4 1 7 18 17 2 7 6 937
0 10 20 30 40 50


Тепловая карта точности предсказания
при преобразовании тестовой выборки закрытием с ядром 2
real
predicted
0 1 2 3 4 5 6 7 8 9
968 1 1 1 0 1 7 0 1 0
0 1065 2 3 25 0 1 39 0 0
28 1 972 1 0 0 2 15 12 1
5 0 4 948 0 12 0 5 29 7
5 3 0 0 936 0 12 1 2 23
5 0 0 23 0 826 25 0 11 2
5 8 0 0 2 1 941 0 1 0
3 4 5 0 1 1 0 971 1 42
2 7 6 2 6 12 6 6 921 6
11 5 1 7 19 14 2 10 7 933
0 10 20 30 40 50

Список используемой литературы

[1] Википедия - метод К ближайших соседей

[2] Machinelearning - метод ближайшего соседа

[3] Ml-handbook

[4] Russianblogs

[5] MNIST Perfect 100% using kNN