Метрические алгоритмы классификации. Отчет.

Пацация Александр, 317 группа

11 октября 2021 г.

1 Введение

В данном отчете описываются эксперименты и полученные результаты применительно к методу **k** ближайших соседей. Эсперименты проводятся над набором данных **MNIST**, представляющим из себя множество изображений рукописных цифр. Первые 60000 элементов используются как обучающая выборка, оставшиеся 10000 – как тестовая. Первым делом выбирается наиболее эффективный по времени алгоритм поиска соседей, затем подбираются гиперпараметры обеспечивающие наибольшую точнось метода на исходной выборке. Далее, исследуется влияние на точность аугментации обучающей и тестовой выборок.

2 Эксперименты

2.1 Первый эксперимент

В первом эксперименте сравнивается время поиска **k** соседей в зависимости от размерности признакового пространства при фиксированном значении $\mathbf{k} = 5$. Случайным образом выбирается подмножество из 10, 20, 100 признаков исходной обучающей выборки.

Сравнивается время работы следующих алгоритмов:

1. **my_own** – собственная реализация алгоритма **brute**.
2. **brute** – реализация алгоритма **brute** из библиотеки **scikit-learn**.
3. **kd_tree** – реализация алгоритма **kd_tree** из библиотеки **scikit-learn**.
4. **ball_tree** – реализация алгоритма **ball_tree** из библиотеки **scikit-learn**.

Как видно из гистограммы (рис. 1), при небольшом количестве признаков хороший результат показывают алгоритмы **kd_tree** и **ball_tree**. Однако при увеличении признакового пространства качество работы этих алгоритмов резко ухудшается. На пространстве из 100 признаков алгоритм **brute**, реализованный библиотекой **scikit-learn**, показывает наилучший результат, поэтому в следующих экспериментах используется именно он.

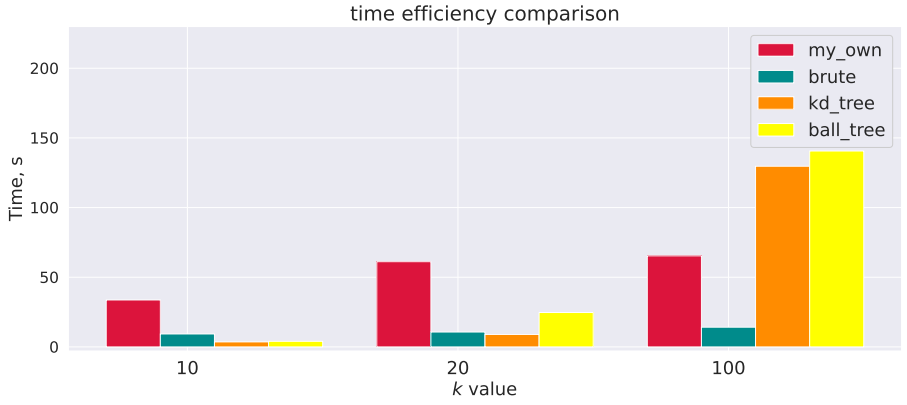


Рис. 1: Сравнение времени работы алгоритмов поиска соседей.

2.2 Второй эксперимент

Во втором эксперименте сравнивается точность и время выполнения метода **k** ближайших соседей в зависимости от значений **k** и используемой метрики. Точность проверятся по обучающей выборке с помощью кросс-валидации с 3 фолдами. Значения **k** рассматриваются в диапозоне от 1 до 10. Использьзуемые метрики:

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

Результаты эксперимента приведены на графике (рис. 2). При $\mathbf{k} = 2$ наблюдается резкий спад точности относительно других значений. Лучшая точность наблюдается при $\mathbf{k} = 3$ для обеих метрик. При последующем увеличении значения **k** качество алгоритма падает. Точность алгоритма при косинусной метрике лучше евклидовой на каждом фолде и при каждом значении **k**.

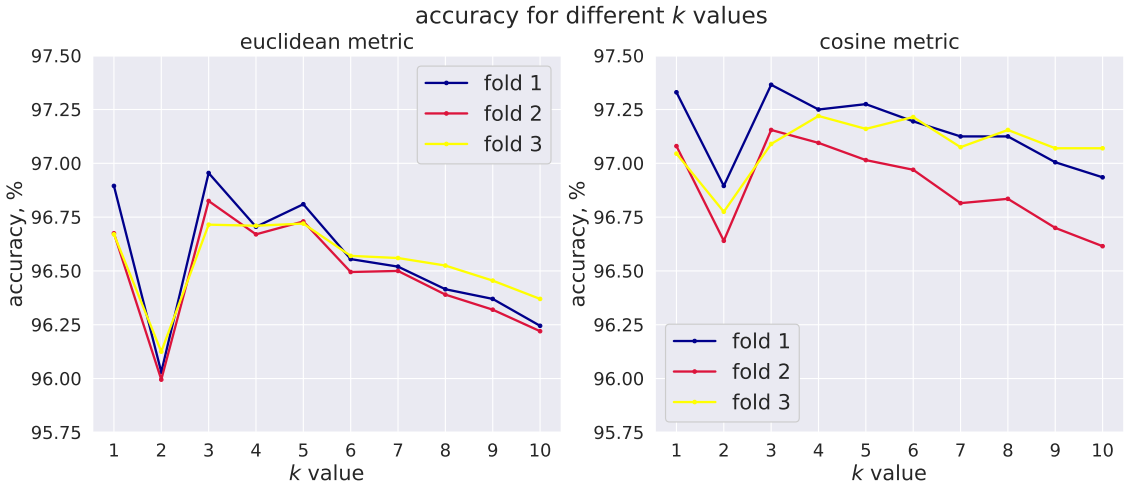


Рис. 2: зависимость точности алгоритма от значения **k** на каждом фолде.

Время выполнения кросс-валидации метода, использующего косинусную метрику и метода, использующего евклидову примерно одинаковое: **126** и **132** секунд соответственно.

2.3 Третий эксперимент

В данном эксперименте сравниваются обычный метод **k** ближайших соседей для косинусной и евклидовой метрик со взвешенным методом. Голоса объектов равны $\frac{1}{distance+\varepsilon}$, где $\varepsilon = 10^{-5}$, *distance* - расстояние между объектом тестовой выборки и обучающей. Диапазон k остается неизменным. Точность проверяется кросс-валидацией с 3 фолдами.

Из графика (рис. 3) видно, что взешенный алгоритм показывает значительно более точный результат при $\mathbf{k} = 2$, в особеености для евклидовой метрики. В таблице (Таблица. 1) представлена средняя по 3 фолдам точность взвешенного алгоритма ближайших соседей. Взвешенный алгоритм, так же, как и обычный, работает точнее с косинусной метрикой при всех **k**. Лучшая точность наблюдается при $\mathbf{k} = 4$.

metric	**k=1**	**k=2**	**k=3**	**k=4**	**k=5**	**k=6**	**k=7**	**k=8**	**k=9**	**k=10**
euclidean	0.9674	0.9674	0.9693	0.9704	0.9687	0.9694	0.9667	0.9673	0.9650	0.9650
cosine	0.9715	0.9715	0.9730	**0.9741**	0.9726	0.9730	0.9712	0.9718	0.9705	0.9704

Таблица 1: средняя точность взешенных алгоритмов по кросс-валидации при различных **k**

По результатам первых трех экспериментов лучшую точность показал взвешенный алгоритм с использованием косинусной метрики при $\mathbf{k} = 4$. В дальнейшем используются именно эти гиперпараметры.

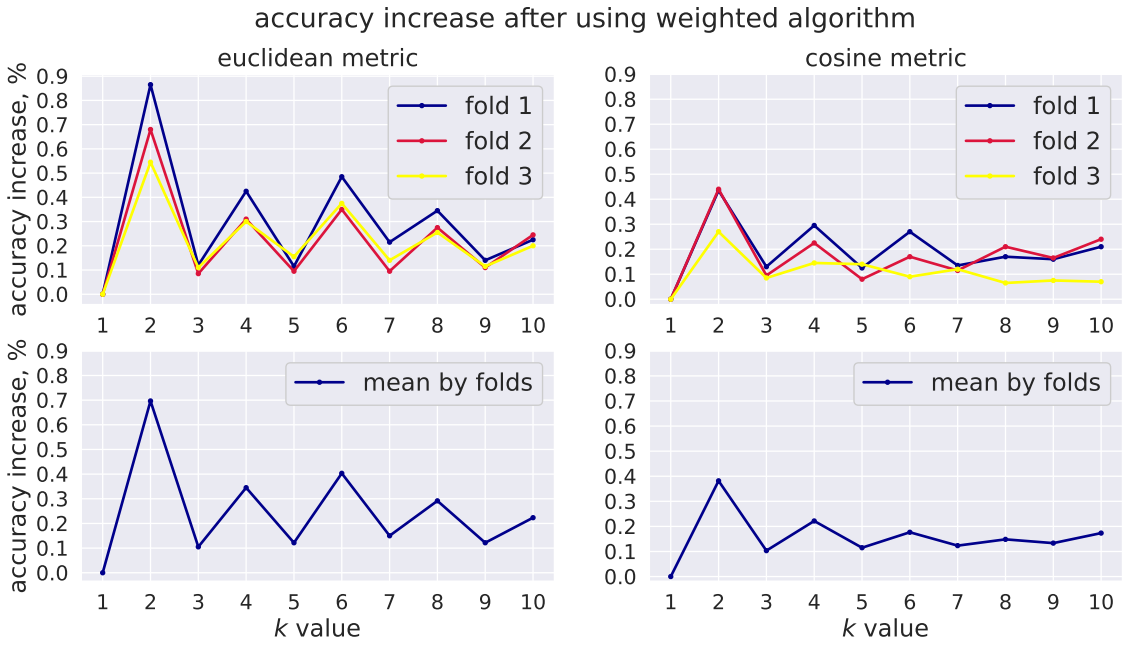


Рис. 3: процентное улучшение работы алгоритма при взвешенном методе.

2.4 Четвертый эксперимент

В данном эксперименте, алгоритм с гиперпараметрами, отобранными по результатам первых трех экспериментов, применяются к исходной обучающей и тестовой выборке. Анализируется точность по сравнению с кросс-валидацией и лучшими алгоритмами на базе данных **MNIST**.

На графике (рис. 4) сравнивается точность алгоритма на тестовой выборке со средней по каждому фолду точностью на кросс-валидации. Алгоритм показывает лучший результат при $\mathbf{k} = 4$ как на тестовой выборке, так и на кросс-валидации. Лучшая точность – **97.52%** на тестовой выборке.

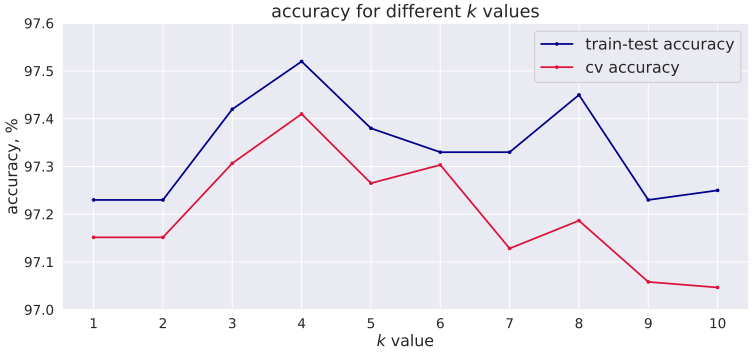


Рис. 4: сравнение точности алгоритма на кросс-валидации и тестовой выборке.

В таблице (Таблица 2) приведено сравнение точности алгоритма ближайших соседей, полученной в реузльтате экспериментов, с точностью лучших алгоритмов на данной выборке, где:

- **KNN** – k nearest neighbors. Алгоритм ближайших соседей с гиперпараметрами, отобранными в ходе экспериментов.
- **CNN** – convolutional neural network. Алгоритм глубокого обучения.
- **SVM** – support vector machine. Алгоритм обучения с учителем.
- **RMDL** – random multimodel deep learning. Алгоритм глубокого обучения.

Построив матрицу ошибок (Рис. 5), можно проанализировать на каких объектах, алгоритм ближайших соседей допускает ошибки.

model	KNN	CNN	SVM	RMDL
accuracy	0.9752	0.9829	0.9823	0.9982

Таблица 2: сравнение различных алгоритмов на тестовой выборке.

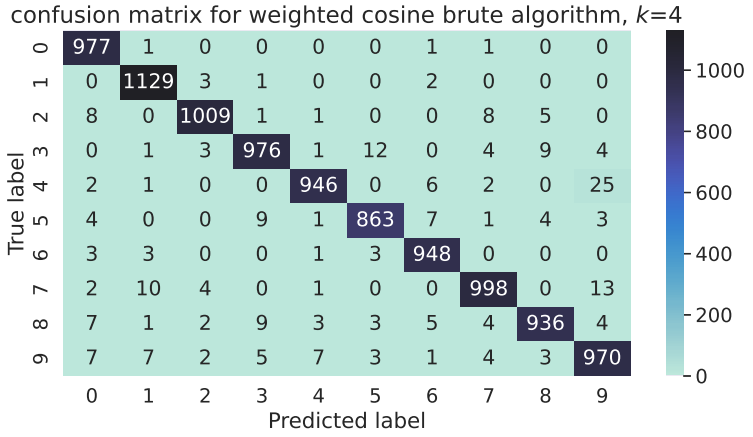


Рис. 5: матрица ошибок.

Можно заметить, что ошибаясь на цифре 4, алгоритм чаще всего относит объект к классу 9. Рассмотрев примеры объектов, на которых была допущена данная ошибка (Рис. 6), можно заметить, что они очень похожы на цифру 9.

Рис. 6: примеры объектов на которых была допущена ошибка.

Рассмотрим также некоторые объекты из класса девяток, на которых алгоритм выдал неправильный ответ (Рис. 7). Цифра часто бывает похожа на другую, поэтому алгоритм допускает ошибки. В частности объект (a) был отнесен в класс единиц, (b) - в класс восьмерок, (c) и (d) - в класс 4, (e) - в класс семерок. Стоит отметить, что объект (d) сложно отличить от четверки даже человеку.

(a) (b) (c) (d) (e)

Рис. 7: примеры цифр 9 на которых алгоритм допустил ошибку.

2.5 Пятый эксперимент

В данном эксперименте проводится аугментация обучающей выборки с использованием следующих преобразований:

- поворот изображения (на 5, 10, 15 градусов в оба направления).
- смещение изображения (на 1, 2, 3 пикселя по обеим осям).
- применение фильтра Гаусса (с дисперсией 0.5, 1.0, 1.5).

Чтобы выбрать самые оптимальные параметры аугиментации, для всевозможных параметров вычисляется точность по кросс-валидации в комбинации с исходной обучающей выборкой. Сохраняется параметр, при котором преобразвоание дает лучший показатель на всех фолдах. Чтобы избежать переобучения, на вход кросс-валидации подается матрица $\mathbf{X}$, получаемая следующим образом:

- Четные (и нулевой) элементы матрицы $\mathbf{X}$ равны элементам матрицы $\mathbf{X_1}$ в том же порядке.
- Нечетные элементы матрицы $\mathbf{X}$ равны элементам матрицы $\mathbf{X_2}$ в том же порядке.

$\mathbf{X_1}$, $\mathbf{X_2}$ - исходная и преобразованная обучающие выборки соответственно. Данный подход был достигнут совместно с Владимиром Хисматуллиным.

В результате, количетво строк новой обучающей выборки будет составлять $4*n$, где n - размер исходной обучающей выборки. По итогам вычислений были выбраны следующие параметры преобразований:

- Поворот на 10 градусов по часовой стрелке.
- Смещение на 1 пиксель по каждой из осей.
- Фильтр Гаусса с дисперсией $\sigma = 1.0$.

Резльтаты вычислений представлены в таблицах 3 и 4.

σ	0.5	1.0	1.5
fold 1	0.9768	**0.9793**	0.9763
fold 2	0.9740	**0.9775**	0.9753
fold 3	0.9741	**0.9769**	0.9744

a)

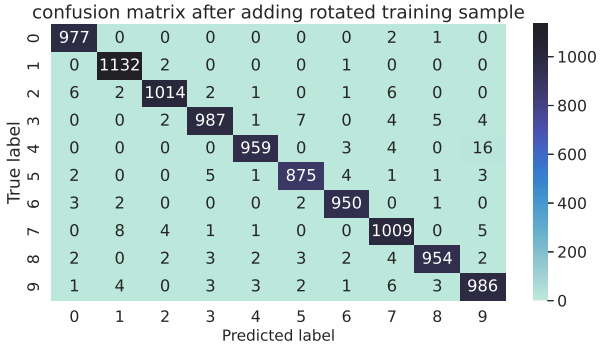
degree	-15	-10	-5	5	10	15
fold 1	0.9769	0.9785	0.9784	0.9782	**0.9783**	0.9776
fold 2	0.9751	0.9755	0.9760	0.9763	**0.9768**	0.9760
fold 3	0.9748	0.97585	0.9759	0.9767	**0.9766**	0.9758

b)

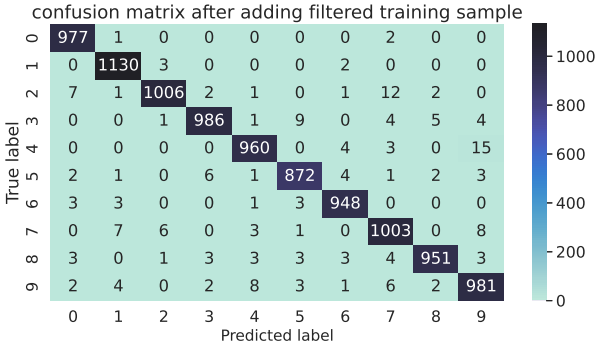
Таблица 3: (a) - результат кросс-валидации при различных значених дисперсии фильтра Гаусса. (b) - результат кросс-валидации при различных параметрах поворота.

shift	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,3)	(3,1)	(3,2)	(3,3)
fold 1	**0.9760**	0.9759	0.9756	0.9757	0.9756	0.9753	0.9756	0.9756	0.9756
fold 2	**0.9746**	0.9736	0.9733	0.9735	0.9735	0.9734	0.9733	0.9732	0.9732
fold 3	**0.9747**	0.9738	0.9735	0.9738	0.9737	0.9736	0.9737	0.9735	0.9735

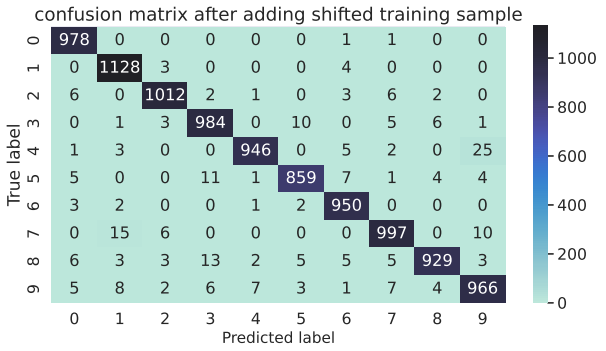
Таблица 4: результат кросс-валидации при некоторых параметрах сдвига. в скобках представлена пара чисел: сдвиг по вертикальной и горизонтальной осям соответственно.



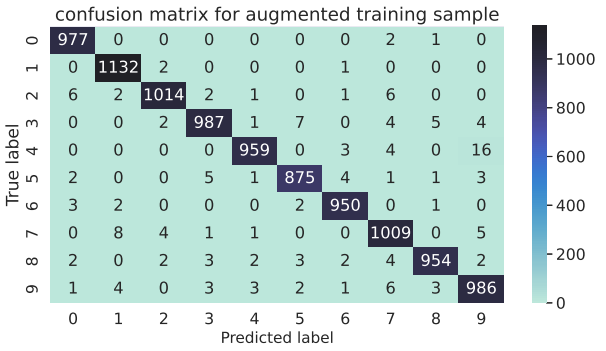
(a)



(b)



(c)



(d)

Рис. 8: (a) – матрица ошибок после добавления повернутой обучающей выборки. (b) – матрица ошибок после добавления обучающей выборки с примененным фильтром Гаусса. (c) – матрица ошибок после добавления сдвинутой обучающей выборки. (d) – результирующая матрица ошибок.

Результирующая точность на тестовой выборке - $\mathbf{98,43}\%$. Матрицы ошибок представлены на Рис. 8. Видно, что поворот на 10 градусов исправил значительное количество ошибок, в особенности на цифрах 9 и 8. Применение фильтра, помимо исправления изрядной доли ошибок на 8 и 9, исправило довольно много ошибок на цифре 4. Однако можно заметить, что количество ошибок на цифре 2 увеличилось. Смещение исправляет заметное количество неправильных ответов на цифре 3, но на цифрах 5, 8, 9 наблюдается увеличение неправильно классифицированных объектов.

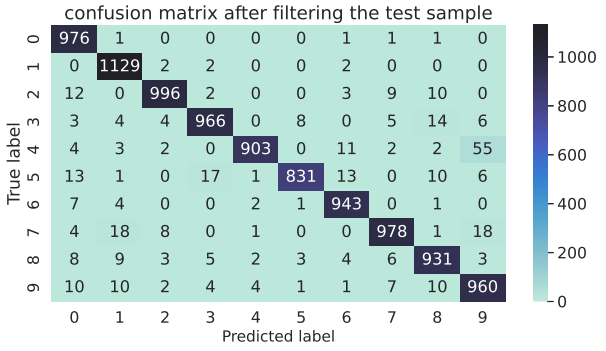
Алгоритму удалось правильно классифицировать объекты (a)-(d)(рис. 7), на которых он допускал ошибки обучаясь на исходной выборке. Объект (e) по-прежнему классифицируется неверно. Из приведенных в пример четверок, была испралена только вторая, на остальных по-прежнему допускается ошибка.

2.6 Шестой эксперимент

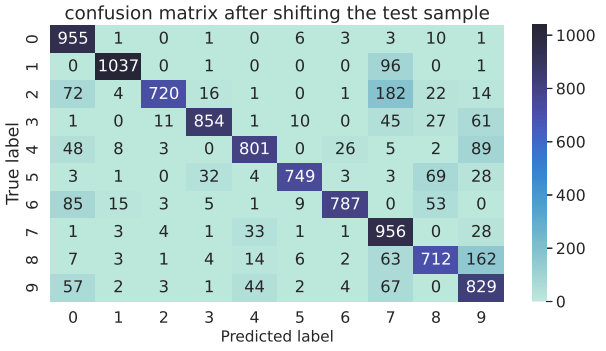
В данном эксперименте подбираются параметры аугментации для тестовой выборки. Для каждого преобразования считается вектор откликов. Далее, для каждой их комбинации считается результирующий целевой вектор путем голосования. В результате перебора были подобраны следующие параметры:

- Фильтр Гаусса с дисперсией $\sigma = 1.0$.
- Сдвиг на 2 пикселя по вертикальной оси.
- Поворот на 5 градусов по часовой стрелке.

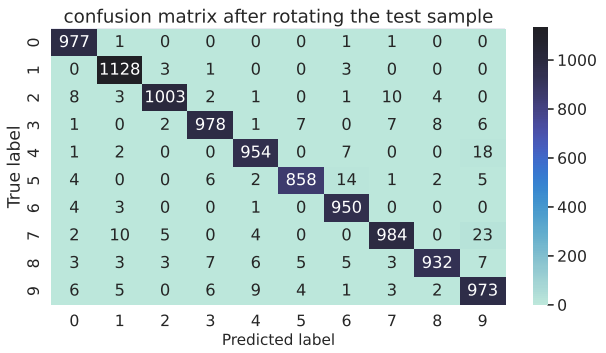
Итоговая точность – $\mathbf{97,59}\%$. Удалось достичь улучшения всего на $\mathbf{0,07}\%$ по сравнению с результатом на исходной тестовой выборке.



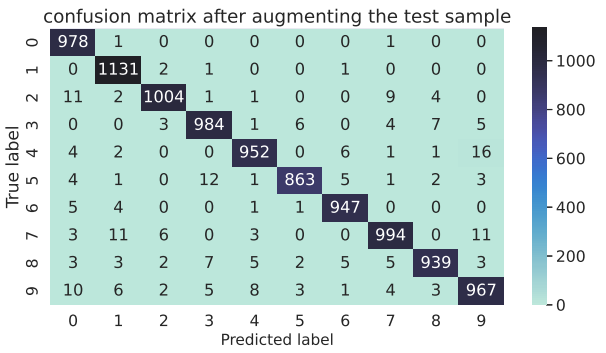
(a)



(b)



(c)



(d)

Рис. 9: (a) – матрица ошибок после применения фильтра Гаусса к тестовой выборке. (b) – матрица ошибок после сдвига тестовой выборки. (c) – матрица ошибок после поворота тестовой выборки. (d) – результирующая матрица ошибок.

Если сравнивать матрицу ошибок после применения фильтра Гаусса (рис. 9 (a)) с исходной (рис. 5) можно заметить, что применение фильтра исправляет ошибки на объектах класса 2, в то время как на исходной тестовой выборке алгоритм классифицирует их как 4 и 6 , а также ошибки на объектах класса 4, которые алгоритм отнес к классу 5. Сдвиг исправляет некоторые ошибки на цифре 5, на которых выдавались классы 0 и 6. Поворот исправляет несколько ошибок на цифре 4, в особенности на объектах, которые до этого относились к 9. Также исправляется некоторое количество ошибок на цифрах 6 и 9.

Подход, использованный в пятом эксперименте показал значительно более хорошие результаты чем в данном. Подход, основанный на преобразовании обучающей выборки точнее на **0.84%**. Однако объем вычислений при аугментации обучающей выборки значительно превышает объем вычислений при аугментации тестовой выборки.

3 Выводы

По итогам шести экспериментов удалось подобрать гиперпараметры, обеспечивающие наилучший результат на исходной обучающей выборке – **97.52%**. После размножения выборки с использованием сдвигов, поворотов и фильтра Гаусса была достигнута точность **98.43%**. При преобразовании тестовой выборки и проведения голосования среди преобразованных объектов значительных улучшений получить не удалось, итоговая точность – **97.59%**.