Московский Государственный Университет им. Ломоносова
Факультет вычислительной математики и кибернетики

Отчёт по заданию «Метрические алгоритмы классификации»

М. М. Шамшиев
317 группа
22 октября 2017

Аннотация

В данном отчете отражены основные результаты, полученные в ходе выполнения практического задания «Метрические алгоритмы классификации». Целью задания было написание собственной реализации алгоритма k-ближайших соседей (k-nearest neighbors algorithm, k-NN) и модуля с функциями для применения кросс-валидации. Также было необходимо провести ряд экспериментов на наборе данных MNIST [2] и сделать по его результатам соответствующие выводы.

Содержание

1 Исследование скорости работы различных алгоритмов поиска ближайших соседей

Исследование проводилось для четырех различных стратегий поиска ближайших соседей:

- my_own: собственная реализация, основанная на вычислении матрицы расстояний и полном переборе при поиске ближайших соседей.
- kd_tree, ball_tree, brute: взятые из библиотеки scikit-learn готовые реализации данных стратегий поиска.

	d=10	d=20	d=100
my_own	57.282	65.483	152.014
brute	13.611	13.906	17.358
kd_tree	1.878	3.169	135.483
ball_tree	7.944	23.812	166.647

Таблица 1: Время работы (с) в зависимости от размерности признакового пространства d

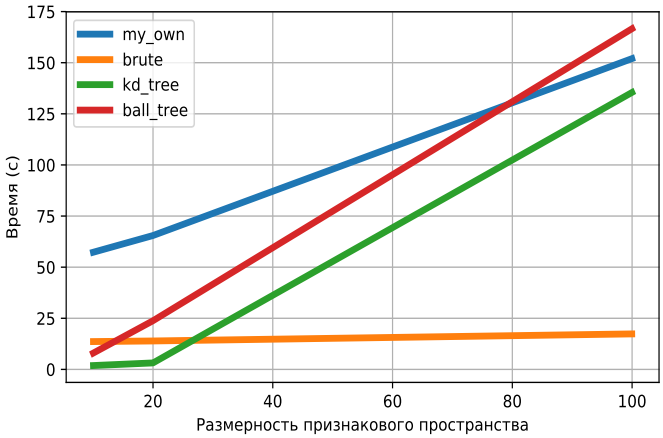

Рис. 1: График времени работы

В ходе исследования для выборки были выбраны подмножества признаков размерностей 10, 20 и 100, по которым считалось расстояние. Эксперимент проводился на модели, соответствующей невзвешенному алгоритму 5-ближайших соседей по евклидовой метрике.

Результаты отражены в Таб.1 , а их визуализацию можно видеть на Рис.1

Экспериментально показано, что стратегии «kd_tree» и «ball_tree», основанные на построении дополнительных структур данных для уменьшения количества вычислений при поиске ближайших соседей, очень быстры при малых размерностях признакового пространства, но с их увеличением эти стратегии становятся крайне неэффективными. Это явление можно объяснить так называемым «проклятием размерности».

Стратегия поиска «my_own» является самой времязатратной стратегией на малых размерностях (что, возможно, связано с не самой оптимальной её реализацией). Но необходимо отметить, что при увеличении количества признаков, по которым вычисляется расстояние, затраты времени данной стратегии растут медленней, чем соответствующие затраты времени стратегий «kd_tree» и «ball_tree», что отчетливо видно на Рис.1

Алгоритм «brute», основанный на вычислении всех расстояний и полном переборе, немного уступает по скорости стратегиям «kd_tree» и «ball_tree» на данных из признаковых пространств малой размерности, но уже при количестве признаков равном 100 выигрыш во времени является колоссальным.

В связи с необходимостью проведения экспериментов на наборе данных MNIST [2], где размерность признакового пространства составляет 784 признака, и руководствуясь результатами данного эксперимента, в дальнейшем при поиске ближайших соседей будем пользоваться стратегией «brute».

2 Оценка точности и времени работы алгоритма в зависимости от количества соседей и используемой метрики

При написании класса KNNClassifier помимо методов, указанных в требованиях к реализации (см. [1]), был добавлен метод

$$predict\_using\_distance(self, index\_matrix, dist\_matrix),$$

принимающий на вход матрицы индексов и расстояний от объектов тестовой выборки до их k-ближайших соседей. Это было сделано с целью осуществления эффективного подбора параметров для алгоритма k-NN: например, при необходимости узнать качество работы алгоритма для различных m значений параметра k: $k_1 \leqslant k_2 \leqslant \ldots \leqslant k_m$, нужно единожды найти k_m ближайших соседей, а далее лишь выбирать из найденных нужное количество.

В данном эксперименте измерение точности и времени работы алгоритма проводилось для всевозможных комбинаций параметров количества соседей (принимало значения от 1 до 10) и используемой метрики (евклидова или косинусная), поскольку в дальнейшем будет необходимо выбрать наилучшую модель, что можно осуществить только перебрав всевозможные различные значения параметров модели.

Средние точности (доли правильно предсказанных ответов), указанные в Таб. 2, были вычислены по кросс-валидации с тремя папками, проведенной на обучающей выборке (первые 60000 объектов датасета MNIST).

Невзвешенный		
	Евклидова	Косинусная
k=1	0.970	0.972
k=2	0.965	0.967
k=3	0.970	0.974
k=4	0.968	0.972
k=5	0.969	0.973
k=6	0.967	0.972
k=7	0.967	0.972
k=8	0.966	0.970
k=9	0.966	0.970
k=10	0.965	0.969

Таблица 2: Средняя точность метода без весов на кросс-валидации

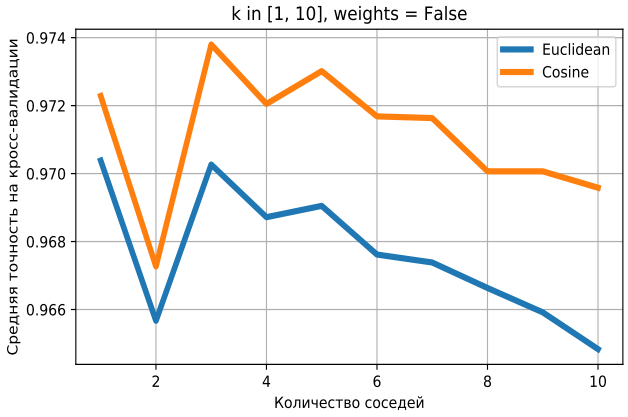

Рис. 2: График точности метода

Эксперимент проводился для невзвешенного алгоритма k-ближайших соседей. Из Таб. 2 и Рис. 2 видно, что максимальная средняя точность для обеих метрик достигается при $k = 3$ ближайших соседях, а разница между ними составляет 0.4%.

Также необходимо заметить, что, например, для евклидовой метрики разница между наилучшей и наихудшей точностью в зависимости от k не превосходит 0.5%, что может свидетельствовать об устойчивости метода на наборе данных MNIST к выбору параметра k. Из Рис. 2 отчетливо видно, что средняя точность при использовании косинусной метрики выше, чем при использовании евклидовой.

Время работы кросс-валидации по евклидовой метрике составило 2 минуты 2 секунды. С другой стороны, при тех же параметрах кросс-валидации по косинусной метрике время работы составило 2 минуты 25 секунд. Разницу во времени можно объяснить необходимостью применения операции деления при подсчёте косинусного расстояния (1), являющейся более времязатратной по сравнению с другими арифметическими операциями.

3 Сравнение взвешенного метода k-ближайших соседей с методом без весов

В данном эксперименте вес объекта x (один из k-ближайших соседей) равнялся $\omega = 1 \Big/ (\rho(x,y) + \varepsilon)$, где $\rho(x,y)$ - расстояние до объекта тестовой выборки y, $\varepsilon = 10^{-5}$.

Косинусное расстояние для объектов $x, y \in \mathbb{R}^d$ определяется формулой

$$\rho(x,y) = 1 - \frac{\sum_{s=1}^{d} x^s y^s}{\sqrt{\sum_{s=1}^{d} (x^s)^2} \sqrt{\sum_{s=1}^{d} (y^s)^2}} \tag{1}$$

С целью оптимизации вычисления расстояний две суммы, стоящие в знаменателе, вычислялись для каждого объекта единожды, а не пересчитывались отдельно по каждой паре объектов.

	Взвешенный	
	Евклидова	Косинусная
k=1	0.969	0.974
k=2	0.969	0.974
k=3	0.971	0.975
k=4	0.971	0.976
k=5	0.970	0.975
k=6	0.970	0.975
k=7	0.968	0.973
k=8	0.968	0.973
k=9	0.966	0.972
k=10	0.967	0.972

Таблица 3: Средняя точность взвешенного метода на кросс-валидации

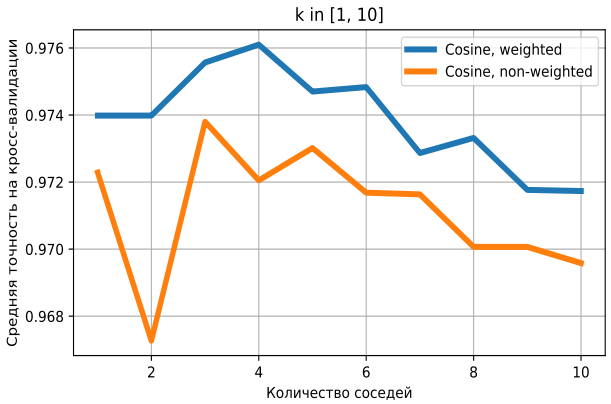

Рис. 3: Сравнение точности взвешенного и невзвешенного методов по косинусной метрике

Средние значения точности, полученные по кросс-валидации с тремя папками взвешенного метода k-ближайших соседей, приведены в Таб. 3. Анализируя полученные результаты, можно заметить, что косинусная метрика вновь показывает себя более подходящей для решения данной задачи классификации.

Для взвешенного метода лучшие точности по обеим метрикам достигаются при $k = 4$ ближайших соседях, а их разница вновь составила 0.5%. Также, сравнивая полученные результаты с результатами невзвешенного метода (Таб. 2), можно сделать вывод, что взвешенный метод дает более высокую точность на кросс-валидации. График сравнения точностей взвешенного и невзвешенного метода по лучшим метрикам (в обоих случаях косинусная) изображен на Рис. 3.

Время работы кросс-валидации взвешенного метода k ближайших соседей по евклидовой метрике составило 2 минуты 40 секунд, а по косинусной - 2 минуты 59 секунд. Таким образом, время работы взвешенного метода превосходит время работы метода без весов, что естественно объясняется необходимостью вычисления матрицы весов.

4 Применение лучшего алгоритма

На основе результатов проведенных экспериментов можно сделать вывод, что наилучшей является модель, соответствующая взвешенному методу 4 ближайших соседей по косинусной метрике. В дальнейшем все эксперименты будут проводиться именно для этой модели.

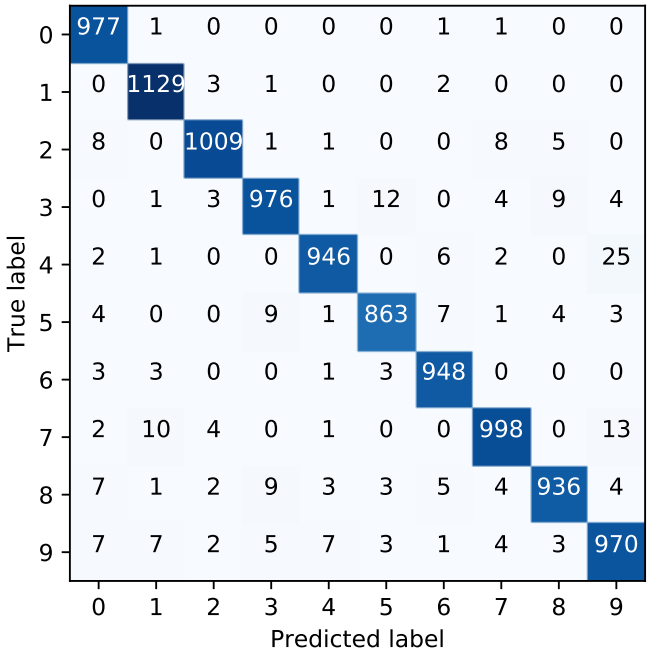

Рис. 4: Матрица ошибок алгоритма на тестовой выборке

(a) Цифра «4», распознанная как «9»

(b) Цифра «3», распознанная как «5»

Рис. 5: Неправильно распознанные объекты

Средняя точность этой модели на кросс-валидации составила 0.976 (Рис. 3). Применив указанную модель к исходной обучающей и тестовой (последние 10000 объектов датасета MNIST) выборке, получаем точность, равную 0.9752. Полученный результат согласуется с точностью, вычисленной на кросс-валидации. На сайте [2] указана точность для алгоритма k-NN по евклидовой метрике без предобработки

изображений, составляющая 0.9717. Результатов для косинусной метрики не приведено.

Построив и проанализировав матрицу ошибок, которая была визуализирована с помощью [3] и приведена на Рис. 4, можно сделать вывод, что наибольшее количество ошибок допускалось при классификации изображений, отвечающим цифрам:

- «9»: 39 ошибок, причем ошибки были допущены примерно «равномерно» по всем классам. Например, по 7 раз алгоритм ошибался, распознавая цифры «0», «1» и «4», что можно связать с существованием множества различных способов рукописного написания цифры «9».
- «8»: 38 ошибок, причем ошибки вновь допущены примерно «равномерно».
- «4»: 36 ошибок, причем подавляющее число раз (25) алгоритм предполагал, что изображена цифра «9». На Рис. 5 (a) можно видеть, что, действительно, в объекте, на котором была допущена ошибка, даже с точки зрения человека нелегко распознать правильную цифру.
- «3»: 34 ошибки, из которых 12 раз алгоритм классифицировал написанную цифру как «5». Посмотрев на некоторые ошибочно классифицированные объекты, можно заметить, что, действительно, некоторые цифры «3» являются «недописанными» (например, отсутствует верхняя черточка), что и объясняет схожесть написанного с цифрой «5». Один из таких объектов приведен на Рис. 5 (b).

Несомненно, как общую черту почти всех неправильно распознанных объектов можно выделить их неразборчивое написание и, зачастую, несоответствие понятиям того, как должна выглядеть та или иная рукописная цифра.

5 Преобразование объектов обучающей выборки для улучшения качества алгоритма

Целью данного эксперимента было увеличение точности алгоритма путем расширения обучающей выборки с помощью элементарных преобразований изображений. Необходимо отметить, что при применении всех последующих элементарных преобразований обучающая выборка размножалась симметрично. Например, при повороте изображения на n градусов из одного объекта обучающей выборки получалось два новых – повернутое изображение на n градусов по часовой стрелке и на n градусов против.

Для проведения эксперимента в модуль cross-validation была добавлена функция $knn\_cross\_val\_score\_aug$, которая на каждой итерации кросс-валидации размножала папки, выступающие в роли обучающей выборки, с помощью элементарного преобразования, переданного ей в качестве аргумента. Также стоит отметить, что во время выполнения функция не хранит все новые обучающие выборки, а лишь вычисляет по отдельности ближайших соседей для каждой из них и затем объединяет полученные результаты. Это позволяет использовать данную функцию даже на компьютерах с относительно небольшим объемом оперативной памяти.

Обучающая выборка размножалась с помощью следующих элементарных преобразований изображений:

5.1 Поворот изображения на 5, 10, 15 градусов в каждую из сторон

При данных преобразованиях новая обучающая выборка получалась в три раза больше исходной: к каждому «старому» объекту добавлялось два новых. Результаты кросс-валидации с тремя папками приведены на Таб. 4. Отсюда видно, что лучшая точность достигается при использовании поворота изображения на 10 градусов. При этом по сравнению с лучшим результатом, показанным этой же моделью, на выборке без преобразований, точность возрасла на 0.5%.

Анализируя матрицу ошибок (Рис. 6), полученную после обучения алгоритма на преобразованной обучающей выборке и предсказании на исходной тестовой, можно сказать, что данное преобразование положительно сказалось на классификации почти всех цифр. Так, например, удалось уменьшить число ошибок при классификации цифры «5» с 29 до 18, цифры «8» с 38 до 27, а также было совершено на 10 ошибок меньше при распознавании каждой из цифр «3» и «7».

Поворот	
$\alpha = 5°$	0.9808
$\alpha = 10°$	0.9810
$\alpha = 15°$	0.9793

Таблица 4: Средняя точность на кросс-валидации в зависимости от угла поворота α

Сдвиг	
$p = 1$	0.9817
$p = 2$	0.9785
$p = 3$	0.9764

Таблица 5: Средняя точность на кросс-валидации в зависимости от величины сдвига p

Фильтр Гаусса	
$\sigma^2 = 0.5$	0.9734
$\sigma^2 = 1$	0.9803
$\sigma^2 = 1.5$	0.9753

Таблица 6: Средняя точность на кросс-валидации в зависимости дисперсии фильтра

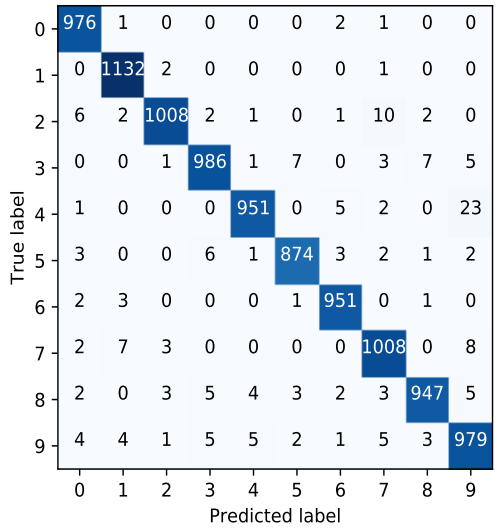

Рис. 6: Матрица ошибок алгоритма при преобразовании обучающей выборки поворотом на 10°

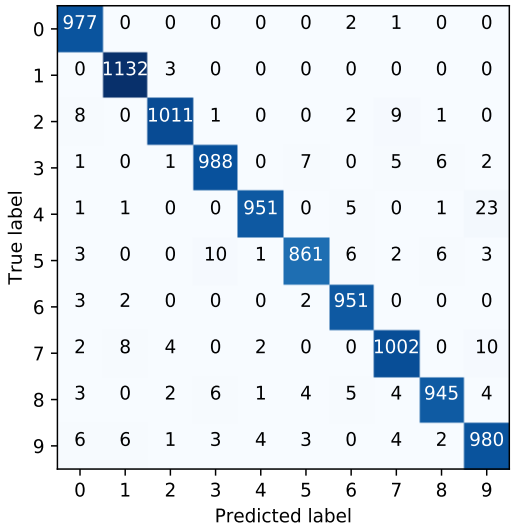

Рис. 7: Матрица ошибок алгоритма при преобразовании обучающей выборки сдвигами на 1 пиксель

5.2 Смещение изображения на 1, 2, 3 пикселя в каждую из сторон

В данном случае новая обучающая выборка была уже в 5 раз больше исходной. Из Таб. 5 можно видеть, что лучшая точность по кросс-валидации, равная 0.9817,

достигается при смещении изображения на 1 пиксель в каждую из сторон. Эта точность превышает лучший результат модели на непреобразованных данных на 0.57%. Посмотрев на матрицу ошибок (Рис. 7), можно заметить, что алгоритм стал значительно лучше распознавать цифру «3» (вместо 34 ошибок - 22), а также при классификации цифры «9» количество ошибок сократилось с 39 до 29.

5.3 Применение фильтра Гаусса

Посмотрев на Таб. 6, можно сказать, что лучший результат по кросс-валидации достигается при использовании фильтра Гаусса с дисперсией $\sigma^2 = 1$. Анализируя матрицу ошибок (Рис. 8), можно заметить, что алгоритм допустил на 15 ошибок меньше при распознавании цифры «8» и на 14 ошибок меньше для цифры «4».

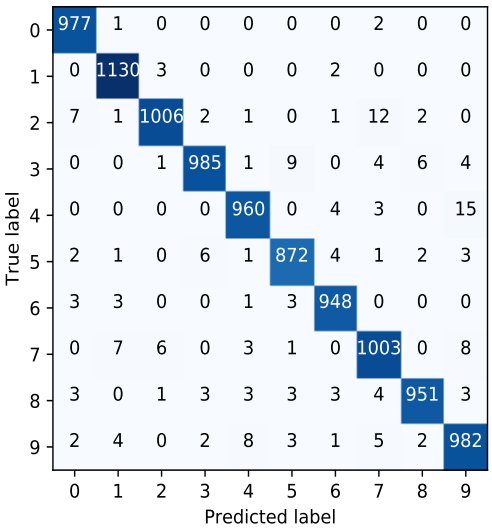

Рис. 8: Матрица ошибок алгоритма при преобразовании обучающей выборки фильтром Гаусса с дисперсией $\sigma^2 = 1$

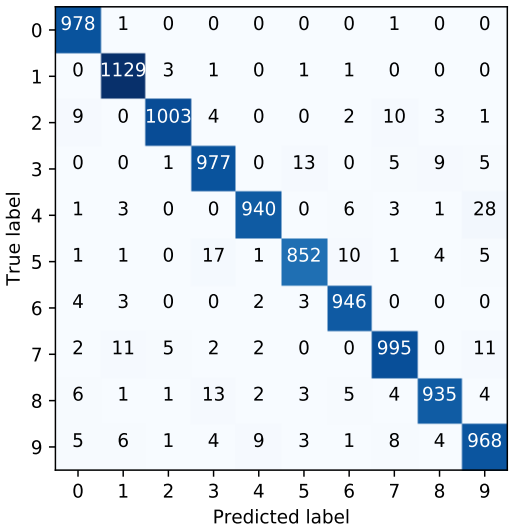

Рис. 9: Матрица ошибок алгоритма на тестовой выборке, преобразованной поворотами на 10°

5.4 Финальное предсказание

Предположив, что комбинация лучших преобразований, выявленных в пунктах 5.1-5.3, даст увеличение точности алгоритма, к исходной обучающей выборке были применены 2 поворота на 10°, 4 сдвига на 1 пиксель (в каждую из сторон) и фильтр Гаусса с дисперсией $\sigma^2 = 1$. Таким образом, новая обучающая выборка получилась в 8 раз больше исходной. Точность предсказания для тестовой выборки оказалась равной 0.985, что является улучшением по сравнению с результатом предсказания на непреобразованных данных на 0.98%.

6 Преобразование объектов тестовой выборки

Целью данного эксперимента было увеличение точности алгоритма, но, в отличии от предыдущего, уже путем расширения тестовой выборки. Также, как и в

предыдущих пунктах, при применении всех последующих элементарных преобразований обучающая выборка размножалась симметрично.

Для проведения эксперимента в модуль cross-validation была добавлена функция *knn_cross_val_score_aug_test*, которая на каждой итерации кросс-валидации размножала папки, выступающие в роли тестовой выборки, с помощью элементарного преобразования, переданного ей в качестве аргумента.

Поворот	
$\alpha = 5°$	0.9781
$\alpha = 10°$	0.9786
$\alpha = 15°$	0.9761

Таблица 7: Средняя точность при преобразовании тестовой выборки в зависимости от угла поворота α

Сдвиг	
$p = 1$	0.9814
$p = 2$	0.9780
$p = 3$	0.9762

Таблица 8: Средняя точность при преобразовании тестовой выборки в зависимости от величины сдвига p (пиксели)

Фильтр Гаусса	
$\sigma^2 = 0.5$	0.9733
$\sigma^2 = 1$	0.9623
$\sigma^2 = 1.5$	0.9498

Таблица 9: Средняя точность при преобразовании тестовой выборки в зависимости дисперсии фильтра Гаусса σ^2

В полной аналогии с предыдущими экспериментами были проведены исследования влияния различных преобразований тестовой выборки на точность предсказания. Результаты кросс-валидаций отображены в Таб. 7 - Таб. 9.

Согласно полученным результатам, лучшими параметрами для поворота и сдвига по-прежнему являются 10° и 1 пиксель соответственно. А вот результаты кросс-валидации при использовании фильтра Гаусса значительно уступают точности, полученной на кросс-валидации без преобразования данных. Этот факт даёт основание не включать фильтр Гаусса в комбинацию преобразований для финального предсказания. Матрицы ошибок изображены на Рис. 9 - Рис. 11.

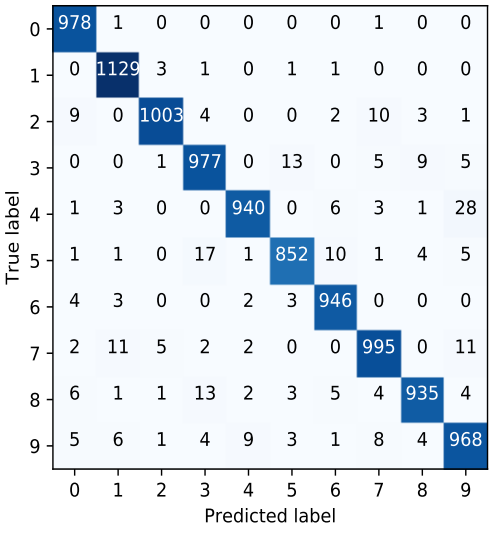

Рис. 10: Матрица ошибок алгоритма на тестовой выборке, преобразованной сдвигом на 1 пиксель

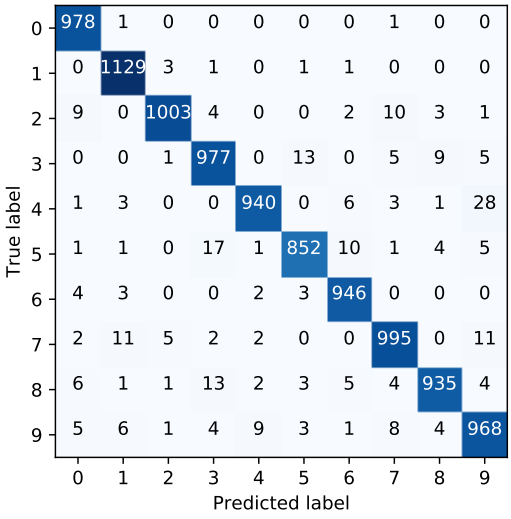

Рис. 11: Матрица ошибок алгоритма на тестовой выборке, преобразованной фильтром Гаусса с дисперсией $\sigma^2 = 0.5$

Также в предположении, что комбинация лучших преобразований даст увеличение точности алгоритма, к исходной тестовой выборке были применены 2 поворота

на 10° и 4 сдвига на 1 пиксель (в каждую из сторон). В результате такой предобработки тестовой выборки точность предсказания составила 0.9821, что также является улучшением по сравнению с результатом предсказания на непреобразованных данных на 0.69%.

Сравнивая различные способы предобработки данных, описанных в пунктах 5-6, можно сделать некоторые выводы:

- Подход, основанный на преобразовании тестовых данных, является более эффективным с точки зрения требуемого количества памяти, поскольку обычно размер тестовой выборки в несколько раз меньше, чем размер обучающей.
- С другой стороны, на практике подход, основанный на преобразовании обучающей выборки, обеспечил больший прирост точности.
- Также подход, основанный на преобразовании обучающей выборки, показался автору отчета интуитивно более понятным и более правильным с точки зрения концепции метода k-NN.

7 Заключение

Таким образом, в ходе выполнения практического задания «Метрические алгоритмы классификации» была написана собственная реализация метода k-ближайших соседей, модуль с функциями для применения кросс-валидации, а также проведены эксперименты, помогающие лучше понять данный алгоритм. Немаловажным результатом выполнения данного задания является приобретение навыков по работе с системой компьютерной вёрстки LaTeX.

Список литературы

[1] Задание 1. Метрические алгоритмы классификации

[2] The MNIST database of handwritten digits

[3] Визуализация матрицы ошибок

[4] Воронцов К. В., LaTeXв примерах, 2005