Московский государственный университет имени М. В. Ломоносова.

Метрические алгоритмы классификации.

Чванкина Дарья
Октябрь 2021
Отчет по курс "Практикум на ЭВМ 2021/2022"

1 Введение.

В данной работе исследуются метрические алгоритмы классификации, используемые в машинном обучении. В ходе экспериментов использовались собственная реализация метода ближайших соседей, кросс-валидации и алгоритма вычисления евклидовой и косинусной метрики. Были проведены эксперименты на датасете MNIST, с применением, помимо собственной реализации, стратегий brute, kd и ball деревьев.

2 Эскперименты.

2.1 Эксперимент № 1.

Данный эксперимент показывает зависимость времени нахождения пяти ближайших соседей для различных стратегий от количества признаков объектов выборки.

Ниже представлены результаты в виде таблицы.

	10	20	100
my_own	49.24	69.47	90.50
brute	7.68	8.55	9.24
kd_tree	6.84	6.46	91.38
ball_tree	7.79	25.19	99.66

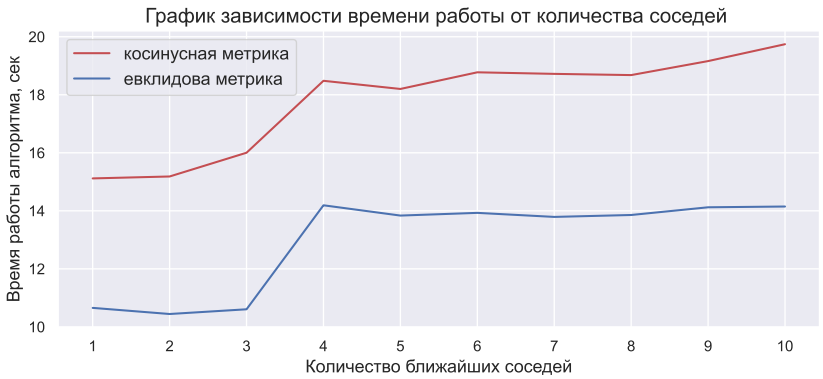
Таблица 1: Время поиска 5 ближайших соседей

С ростом количества признаков время работы каждого алгоритма растёт. При небольшом количестве признаков быстрее всего работает алгоритм с использованием kd деревьев. Но при росте размерности признакового пространства в 10 раз время работы алгоритмов kd и ball деревьев возрастает также в 10 раз. При этом время работы алгоритма brute выросло на пару секунд. Далее во всех экспериментах используется стратегия brute, как самый быстрый алгоритм.

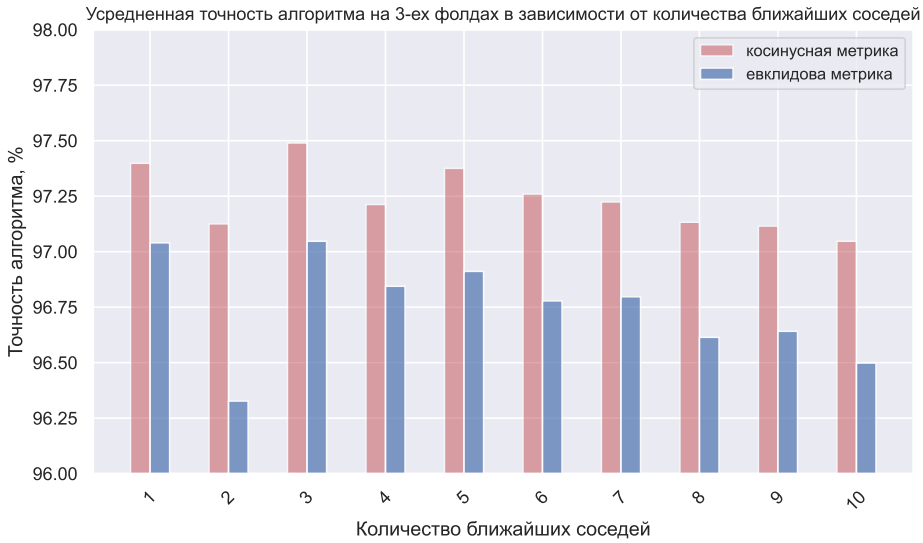
2.2 Эксперимент № 2.

В этом эксперименте проводилась оценка времени работы алгоритма в зависимости от количества ближайших соседей и от используемой метрики.

Рис. 1:



При оценки времени работы алгоритма поиска ближайших соседей в зависимости от используемой метрики вычислялось время работы алгоритма при различном количестве ближайших соседей ($k = 1, 2, \ldots, 10$). Результаты показали (см. рис. 1), что при любом k метод ближайших соседей с косинусной метрикой работает дольше примерно на 5 секунд, чем с евклидовой, но точность алгоритма с косинусной метрикой больше(оценивалась с помощью кросс-валидации). Максимум точности алгоритма с косинусной метрикой достигает 0.974 при $k = 3$, также график зависимости от времени терпит крутой скачок вверх при том же k. Более высокая точность при использовании косинусной метрики говорит о том, что в данном наборе данных

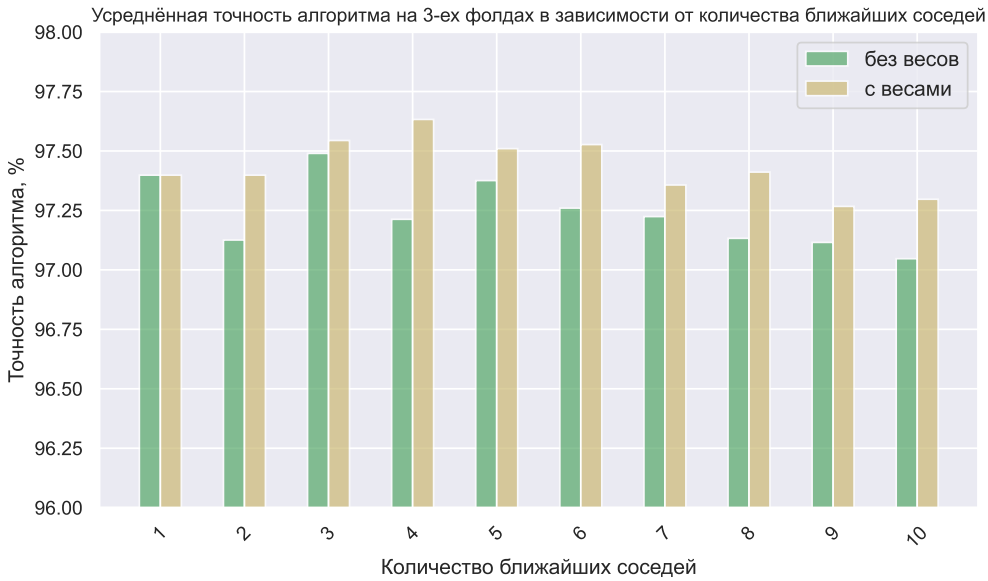


важен угол между векторами данных, а не точка в пространстве, которую они задают. Для большой точности алгоритма предпочтительнее использовать косинусную метрику, а количество ближайших соседей выбирать равным 3.

2.3 Эксперимент № 3.

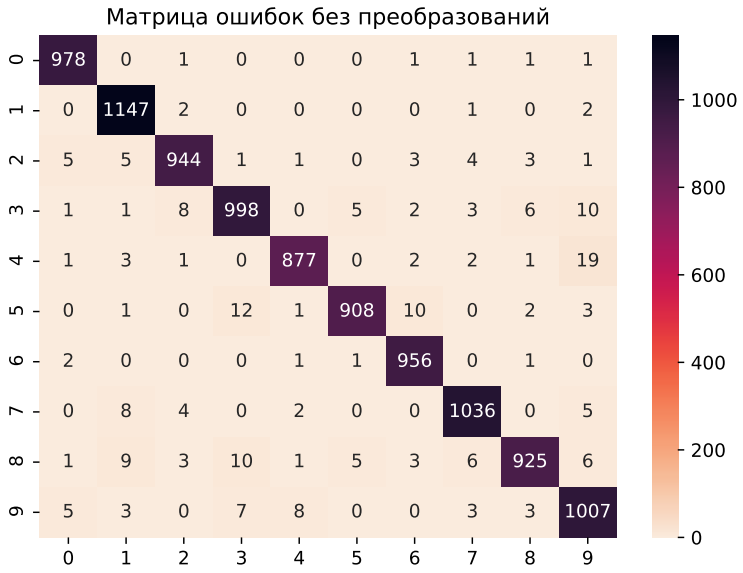
В данном экспермиенте сравниваются два подхода использования алгоритма — с весами и без весов. Оценка проводится по кросс-валидации с использованием косинусной метрики с одинаковыми параметрами и фолдами. Каждый объект входил в сумму со своим весом равным $1/(distance+\epsilon)$, где $\epsilon = 10^{-5}$. Максимальная точность равна 0.976 при $k = 4$. Алгоритм с весами даёт более точный результат, так как учитывает на каком расстоянии находится объект обучающей выборки.

Рис. 2:



2.4 Эксперимент № 4.

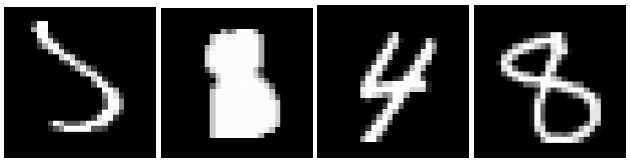
Наиболее точным алгоритмом для данной выборки среди рассмотренных является взвешенный алгоритм с косинусной метрикой и с параметром k равным 3. Точность такого алгоритма равна 0.977. Самый точный алгоритм классификации в мире для набора данных MNIST достигает точности 0.998.



То есть метрический алгоритм ошибается на 2 процента больше при классификации изображений, чем самый точный алгоритм. Проанализируем на каких объектах ошибается алгоритм. Для этого построим матрицу ошибок.

Больше всего алгоритм ошибается, классифицируя цифру четыре как девять. Также он путает цифры 5 и 3, 8 и 3, 5 и 6. Некоторые из этих изображений классифицировать сложно и человеку. Но некоторые алгоритм плохо классифицирует из-за того, что изображение повернуто на некоторый угол или сдвинуто немного вправо или влево.

Рис. 3: На этих данных алгоритм ошибался



2.5 Эксперимент № 5.

Для повышения точности алгоритма размножим выборку поворотами на 5, 10 и 15 градусов в обе стороны. При сравнении использовалась точность на кросс-валидации с тремя фолдами и с параметром k равным 3. Поворот на 5 градусов увеличивает точность на 2% в любую из сторон. Поворот на 10 градусов увеличивает точность всего на 1 процент, а при повороте больше, чем на 10 градусов, точность алгоритма начинает падать.

Таблица 2: Преобразования выборки

Таблица 3: Точность при повороте на угол

градусы	0°	±5°	±10°	±15°
фолд 1	0.976	0.996	0.986	0.975
фолд 2	0.977	0.994	0.985	0.979
фолд 3	0.976	0.992	0.982	0.974

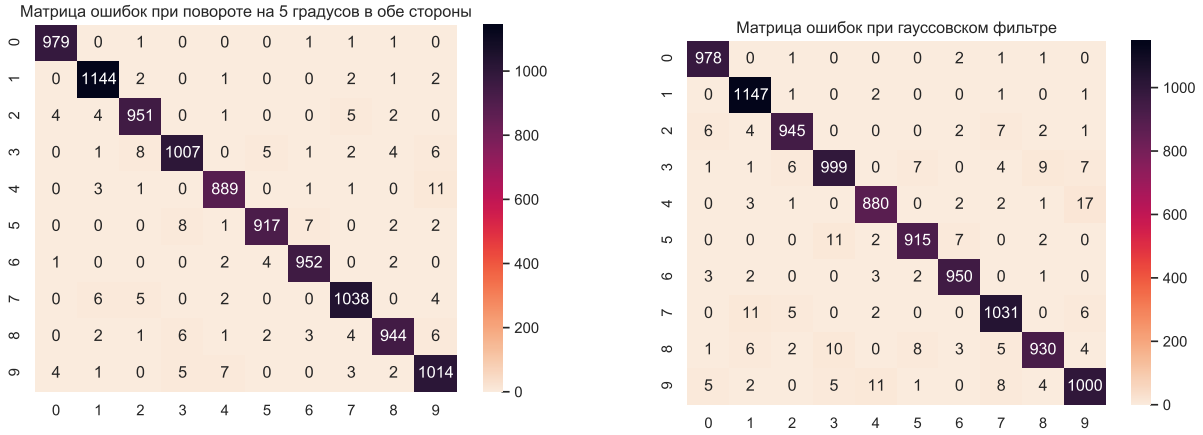
Таблица 4: точность при сдвиге

сдвиг(пиксели)	0	1	2	3
фолд 1	0.976	0.981	0.967	0.969
фолд 2	0.977	0.972	0.967	0.968
фолд 3	0.976	0.953	0.951	0.951

При обучении на выборке, в которой присутствуют поворот на 5 градусов в обе стороны, алгоритм показал точность на валидационной выборке 0.984. (При повороте тольво в одну сторону - 0.981). Алгоритм стал лучше классифицировать циры, так как теперь он распознаёт повёрнутую цифру.

При свдиге больше, чем на 1 пиксель точность на кросс валидации уменьшается. При свдиге на 1 пиксель по разным осям точность алгоритма заметно не улучшается.

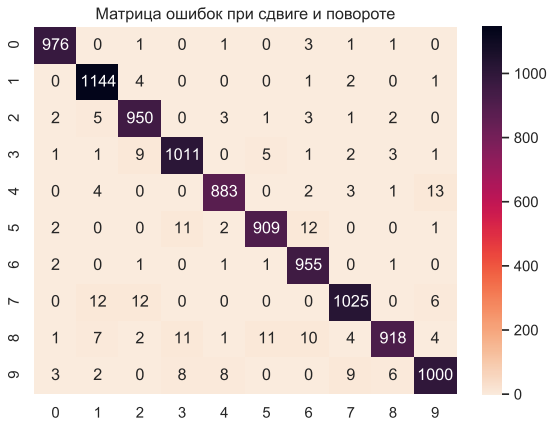
Рис. 4: Матрицы ошибок



Гауссовский фильтр разумно применять при дисперсии равной 0.5, при больших значений дисперсии картинка сильно портится. Точность на валидационной выборки при таком размножении обучающей выборки равна 0.977. Точность алгоритма без преобразований такая же. Матрица ошибок почти не изменилась.

Рис. 5: Применение гауссовского фильтра при $\sigma = 0.5, 1.0, 1.5$

При повороте выборки на 5 градусов и свдиге на 1 пиксель по одной оси точность на тестовой выборки составила 0.997, что не отличается от точности алгоритма без рзмножения выборки. Матрица ошибок стала менее однородной. Появилось больше классов, на которых алгоритм ошибся больше 10 раз.

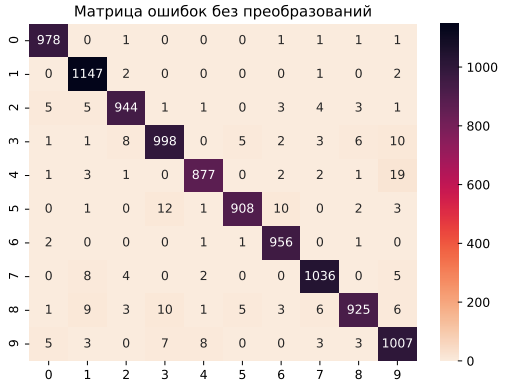


2.6 Эксперимент № 6.

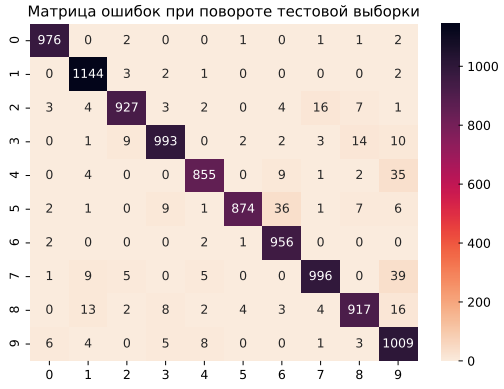
Оценим точность алгоритмов при тех же преобразований(поворот, свиг, гауссовский фильтр) тестовой выборки. Для каждого преобразования в качестве результата выдавался класс, который выбирался путём голосования.

При преобразованиях тестовой выборки матрица ошибок заметно ухудшилась, точность алгоритма понизилась. При сдвиге тестовой выборки там, где алгоритм неправильно классифицровал, появилось ещё больше ошибок. Матрицы ошибок стали неоднородными. Например, при сдвиге тестовой выборки алгоритм ни разу не классифицировал цифру 3, как цифру 6, но классифицировал цифру 4 как цифру 9 104 раза.

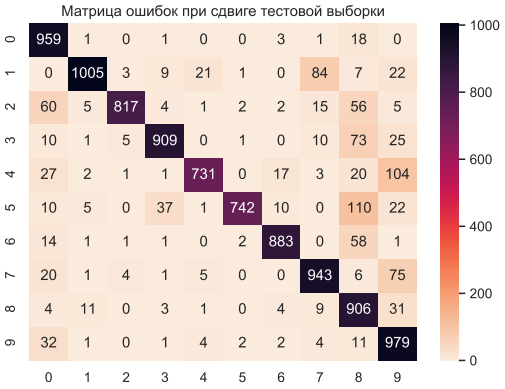
Точность алгоритма при всех этих преобразованиях упала более, чем на 1%, ошибок стало больше. При неизменении обучающей выборки и изменении тестовой, точность алгоритма всегда будет ухудшаться. При размножении обучающей выборки путем небольших изменений точность будет расти, но если изменения слишком большие, то алгоритм будет выдавать меньшую точность.



a)



b)



c)

d)

Рис. 6:

При подходе, который применялся в эксперементе 5, можно повысить точность алгоритма, улучшить качество классификации. При изменении тестовой выборки и выборе класса путем голосования классификатор ошибается чаще. Для повышения точности алгоритма лучше применять преобразования к обучающей выборке.

3 Заключение

В экспериментах использовалось несколько методов нахождения ближайших соседей — brute, kd и ball деревья, а также собственная реализация. При маленькой размерности признакого пространства быстрее всхе работает метод с применением kd деревьев, но при большом количестве признаков лучше всего использовать

метод brute. Для датасета изображения цифр MNIST лучше использовать косинусную метрику, чем евклидову, но алгоритм будет работать на ней дольше. Использование весов улучшает классификацию. Также было найдено оптимальное k равное 4, при котором точность алгоритма наиболее высокая. Можно размножать обучающую выборку поворотами на несколько градусов для повышения точности. Преобразования сдвигом или гауссовским фильтром не улучшило классификацию. Любые преобразования с тестовой выборкой и выбором класса с помошью голосования привели к ухудшению точности.

4 Список литературы

1. https://stackoverflow.com/questions/43864855
2. https://www.machinelearningmastery.ru/importance-of-distance-metrics
3. https://ru.wikipedia.org/wiki/MNIST
4. https://scikit-learn.org/stable/modules/neighbors.htmlnearest-neighbor-algorithms