Отчет о практическом задании «Метрические алгоритмы классификации».

Практикум 317 группы, ММП ВМК МГУ.

Аристеев Олег Алексеевич.

Октябрь 2024.

Содержание

1 Вступление **2**

2 Эксперименты **2**

2.1 Исследование скорости алгоритма. 2

2.2 Оценка по кросс-валидации. 2

2.3 Влияние весов в алгоритме. 3

2.4 Применение лучшего алгоритма. 4

2.5 Аугментация обучающей выборки. 5

2.6 Преобразование объектов тестовой выборки. 7

3 Улучшения **8**

3.1 Комбинация предыдущих методов. 8

4 Выводы **9**

1 Вступление

Метрические алгоритмы классификации широко применяются при решении задач машинного обучения. В данной работе был рассмотрен один из наиболее интуитивных методов – алгоритм k-ближайших соседей (KNN). Он основан на гипотезе компактности: близкие объекты, как правило, лежат в одном классе, а объекты разных классов должны быть далеки в смысле той же метрики.. В рамках данной работы был рассмотрен датасет MNIST — коллекция изображений рукописных цифр, состоящая из 60000 обучающих и 10000 тестовых изображений.

Цель данного исследования — проанализировать эффективность алгоритма KNN на задаче распознавания рукописных цифр с использованием набора данных MNIST и изучить влияние различных параметров алгоритма.

Алгоритм KNN прост в реализации, но, тем не менее, показывает конкурентные результаты. Однако, важно заметить, что его качество напрямую зависит от удачного выбора гиперпараметров. Это будет продемонстрировано на экспериментах ниже.

2 Эксперименты

2.1 Исследование скорости алгоритма.

Для каждого из алгоритмов поиска ближайших соседей ‘kd_tree‘, ‘ball_tree‘, ‘brute‘ и ‘my_own‘ было исследовано время нахождения $k = 5$ ближайших соседей из обучающей выборки для каждого объекта тестовой выборки. Использовалась евклидова метрика. Близость между объектами вычислялась по подмножествам признаков длиной 10, 20 и 100. Набор этих признаков выбирался случайно и оставался фиксированным в течение эксперимента.

Полученные результаты отмечены в таблице 1. Обучающая выборка содержит 60 тысяч объектов, а тестовая выборка – 10 тысяч.

Стратегия	**Количество признаков**		
	10	**20**	**100**
my_own	7.23 с.	12.80 с.	80.57 с.
brute	9.50 с.	9.28 с.	10.06 с.
ball_tree	9.40 с.	36.75 с.	114.00 с.
kd_tree	2.72 с.	7.53 с.	98.49 с.

Таблица 1: Длительность работы KNN на разных наборах признаков и алгоритмах поиска соседей

Как видно, ‘brute‘ работает значительно быстрее остальных методов при достаточно большом количестве признаков. В дальнейших экспериментах будет использоваться именно он.

2.2 Оценка по кросс-валидации.

По кросс-валидации с 3 фолдами были измерены точность и время работы алгоритма k-ближайших соседей в зависимости от значения $k = 1, 2, \ldots, 10$ и выбора метрики: евклидовой или косинусной. Для каждого значения k было выбрано среднее значение accuracy и времени по всем разбиениям. Результаты отмечены на графиках (рис. 1).

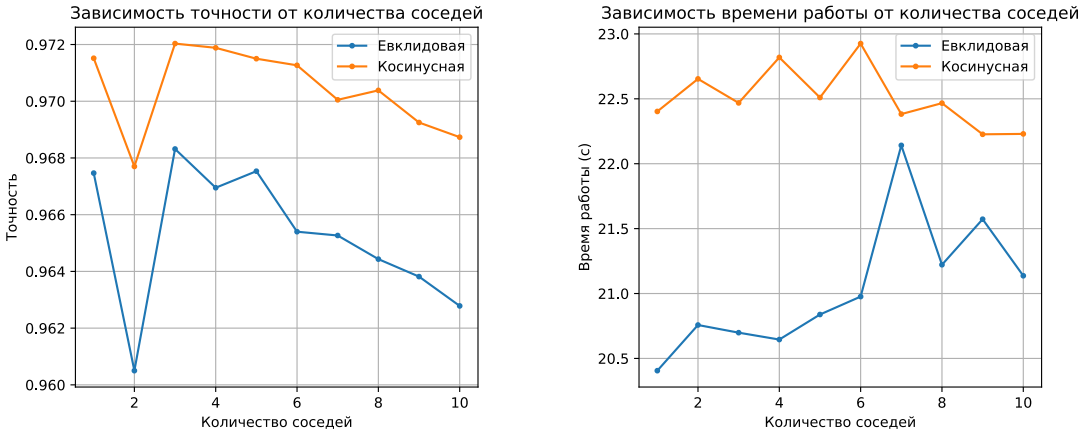

Рис. 1: Доля правильных ответов и время работы на невзвешенном KNN

Косинусная метрика показала себя лучше евклидовой. Дело в том, что она не учитывает длины входящих векторов, а только угол между ними. Небольшие изменения в отдельных пикселях могут существенно влиять на евклидово расстояние, но косинусная метрика их игнорирует, если они не меняют общее направления вектора.

Заметное падение наблюдается при $k = 2$. При таком значении возможна ситуация, когда два ближайших соседа принадлежат разным классам, и возникает ничья. Поскольку решение принимается голосованием без весов, то выбор соседа выбирается фактически случайно. Это может значительно ухудшить качество.

2.3 Влияние весов в алгоритме.

Для тех же параметров кросс-валидации были измерены точность и время работы при использовании взвешенного алгоритма KNN. Вес объекта вычислялся по формуле $1/(distance + \varepsilon)$ при $\varepsilon = 10^{-5}$. Полученные результаты отмечены на графиках (рис. 2).

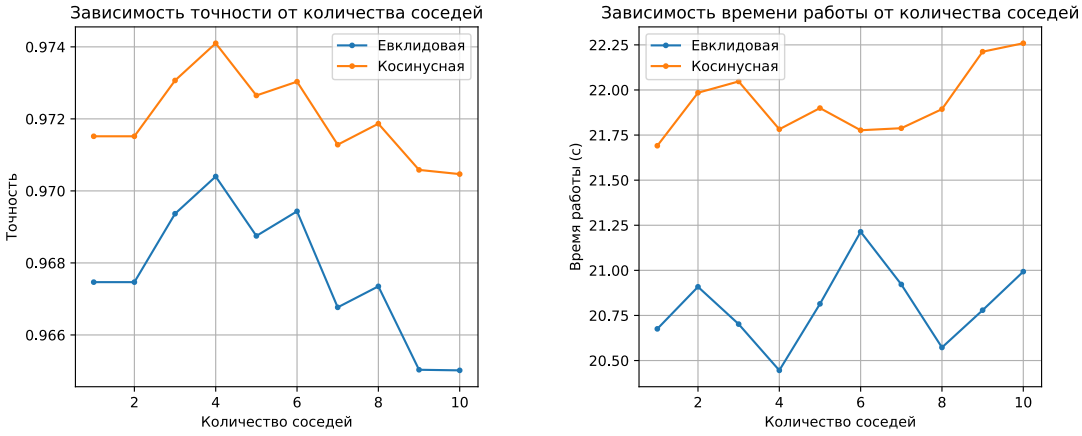

Рис. 2: Доля правильных ответов и время работы на взвешенном KNN

Качество оказалось в среднем лучше, чем при невзвешенном алгоритме. Кроме того, заметим, что при $k = 2$ больше не видно значительного падения, как в случае отсутствия весов. Это объясняется тем, что теперь выбор соседа происходит не случайно, а учитывается также вес голоса.

2.4 Применение лучшего алгоритма.

По результатам предыдущих экспериментов, алгоритм с наибольшей точностью на валидационной выборке (0.97545) использует поиск $k = 4$ ближайших соседей, косинусные расстояния и веса, определяемые формулой $1/(distance + \varepsilon)$ при $\varepsilon = 10^{-5}$.

Этот алгоритм на тестовой выборке имеет точность **0.9752**. В таблице 4 содержится полученная матрица ошибок.

Настоящее значение	**Предсказанное значение**									
	0	1	2	3	4	5	6	7	8	9
0	977	1	0	0	0	0	1	1	0	0
1	0	1129	3	1	0	0	2	0	0	0
2	8	0	1009	1	1	0	0	8	5	0
3	0	1	3	976	1	12	0	4	9	4
4	2	1	0	0	946	0	6	2	0	25
5	4	0	0	9	1	863	7	1	4	3
6	3	3	0	0	1	3	948	0	0	0
7	2	10	4	0	1	0	0	998	0	13
8	7	1	2	9	3	3	5	4	936	4
9	7	7	2	5	7	3	1	4	3	970

Таблица 2: Матрица ошибок

Видно, что на диагонали получается высокая точность. Это означает, что большая часть элементов правильно распознается алгоритмом. Но у модели есть некоторые типичные ошибки. Они случаются когда цифры написаны неаккуратно или нестандартно:

(a) Цифра 4 иногда классифицируется как 9 (25 случаев), что указывает на их схожесть в некоторых рукописных стилях.

(b) Цифра 3 ошибочно классифицируется как 8 (8 случаев) или 5 (5 случаев).

(c) Цифра 7 часто распознается как 1 (10 случаев).

(d) Цифра 8 часто распознается как 3 (9 случаев), и наоборот (9 случаев).

Рассмотрим некоторые случаи, когда алгоритм ошибочно распознавал цифру (рис-3).

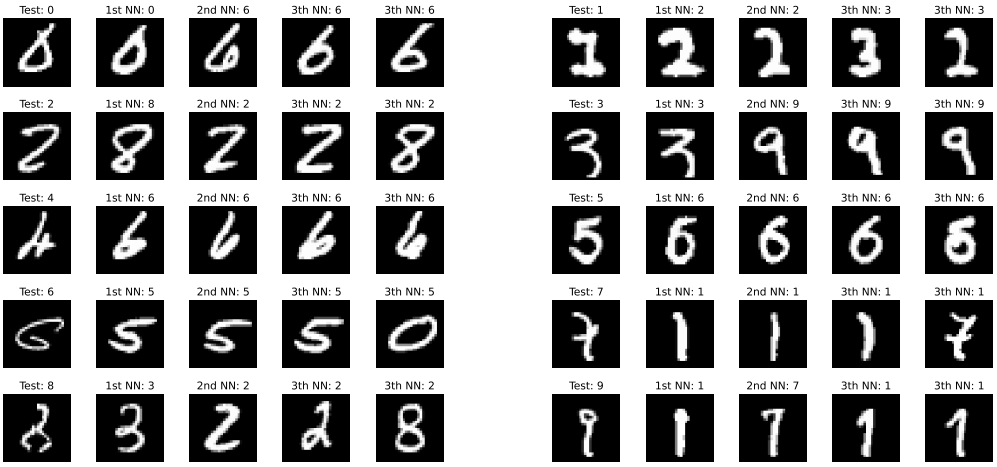

Рис. 3: Примеры ошибочных распознаваний

Видно, что на всех тестовых изображениях в этих случях цифра имеет большое число установленных в одном месте пикселей. Например, тест для цифры 2 и её первый сосед – цифра 8. Количество светлых пикселей у 2 почти образует подмножество светлый пикселей у 8, поэтому расстояние между ними оказалось таким маленьким. Аналогичные рассуждения можно провести и для других тестов.

В таком случае, может помочь применение методов предварительной обработки данных: сглаживания или нормализации изображений, чтобы убрать мелкие вариации в написании цифр.

2.5 Аугментация обучающей выборки.

Выполним аугментацию обучающей выборки. Для этого рассмотрим следующие преобразования:

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

(b) Величина сдвига: 1, 2, 3 пикселя (по каждой из двух размерностей)

(c) Дисперсия фильтра Гаусса: 0.5, 1, 1.5

(d) Морфологические операции: эрозия, дилатация, открытие, закрытие с ядром 2

В данном эксперименте есть всего $7 \cdot 9 \cdot 3 \cdot 15 = 2835$ возможных комбинаций параметров аугментации. С учетом того, что время выполнения одного эксперимента с $k = 4$, косинусной метрикой и использованием весов занимает в среднем 22 секунды, получим, что суммарное время выполнение можно оценить, как:

$$2835 \cdot 3 \cdot 22 \text{ с} = 187110 \text{ с} \sim 52 \text{ часа}$$

Поэтому, чтобы значительно ускорить процесс вычислений, воспользуемся жадной стратегией подбора параметров: на каждом этапе (a)–(d) будем выбирать лучший параметр по валидационной выборке и фиксировать его до конца эксперимента.

(a) Была исследована зависимость точности на валидацинной выборке от величины поворота. Полученные данные отмечены в таблице 3. Оптимальным значением получилось -5.

Угол поворота (градусы)	-15	-10	**-5**	5	10	15
Точность	0.97184	0.97286	**0.973583**	0.97306	0.97328	0.97258

Таблица 3: Зависимость точности на валидационной выборке от угла поворота

(b) Следующим этапом была найдена оптимальная величина смещения при фиксированном в прошлом пункте повороте -5. Полученные значения отмечены в таблице 5.

Сдвиг по x	**Сдвиг по y**		
	1	2	3
1	**0.97155**	0.97065	0.97084
2	0.97067	0.97103	0.97093
3	0.97116	0.97107	0.97111

Таблица 4: Зависимость точности на валидационной выборке от дисперсии фильтра Гаусса

(c) Затем была найдена оптимальная дисперсия в фильтре Гаусса 5. Оптимальное значение $\sigma = 0.5$, и действительно, при больших значениях дисперсии некоторые изображения становятся плохо различимыми (рис-4).

Дисперсия	**0.5**	1.0	1.5
Точность	**0.9708**	0.970767	0.9688

Таблица 5: Зависимость точности на валидационной выборке от дисперсии фильтра Гаусса

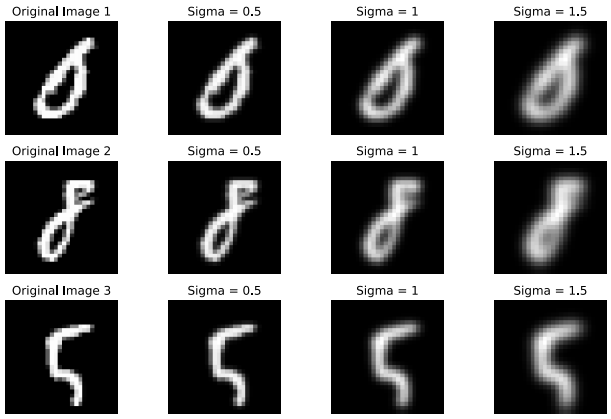

Рис. 4: Качество изображения в зависимости от дисперсии фильтра Гаусса

(d) Следующий этап – поиск оптимального набора морфологических операций 6. По валидационной выборке получился набор «dilate» и «open», но поскольку алгоритм показал плохую точность на тесте с этими параметрами, было принято решение выбрать вторую по качеству на валидационной выборке – только «dilate».

Морфологические операции	Точность
erode	0.9620416
dilate	**0.9715750**
open	0.9690166
close	0.9706083
erode + dilate	0.9690166
erode + open	0.949575
erode + close	0.9623833
dilate + open	*0.9717250*
dilate + close	0.9694666
open + close	0.9692833
erode + dilate + open	0.96954166
erode + dilate + close	0.96928333
erode + open + close	0.94952500
dilate + open + close	0.96968333
erode + dilate + open + close	0.969475

Таблица 6: Зависимость точности на валидационной выборке от выбора морфологических операций

Таким образом, оптимальном набором преобразований получилась следующая комбинация: (a) величина поворота -5, (b) смещения на 1 и 1 по X и Y, (c) дисперсия в фильтре Гаусса 0.5 и (d) использование дилатации. Таким образом, получили новую обучающую выборку с 120 000 объектами, состоящую из оригинального датасета и датасета, полученного путем преобразования каждого оригинального объекта.

Обучив метод KNN с фиксированными ранее параметрами на оригинальной тестовой выборке, получили качество **0.9755** (без преобразований точность была 0.9752). Так выглядит новая матрица ошибок 7.

Настоящее значение	**Предсказанное значение**									
	0	1	2	3	4	5	6	7	8	9
0	977	0	0	0	0	0	2	1	0	0
1	0	1128	3	0	0	0	4	0	0	0
2	3	1	1015	2	1	0	1	6	3	0
3	0	1	3	990	0	9	0	4	2	1
4	0	3	0	0	949	0	5	2	0	23
5	4	0	0	9	2	859	9	1	4	4
6	5	2	0	0	1	2	948	0	0	0
7	0	16	5	1	2	0	0	991	0	13
8	6	2	3	15	3	4	6	3	929	3
9	2	7	3	6	9	1	0	8	4	969

Таблица 7: Матрица ошибок после преобразований обучающей выборки

Как видно, заметно улучшилось качество для цифр 2 и 3, но снизилась точность распознавания для, например, цифр 7 и 8. В среднем, цифра 3 стала гораздо ошибочно распознаваться как 8 и 9. Но 8 стала чаще распознаваться как 3. Это происходит из-за того, что при использование фильтра Гаусса размывает изображение, и цифра 8, написанная неаккуратно, может стать более похожей на цифру 3. Кроме того, цифра 7 стала чаще ошибочно распознаваться как цифра 1. Эта ошибка, вероятно, связана с поворотом: отсутсвие ‘черточки‘ при написании 7 может сделать её похожей на 1 при повороте.

2.6 Преобразование объектов тестовой выборки.

В этом эксперименте обучающая выборка осталась оригинальной (60000 элементов), а для проведения голосования по объектам тестовой выборки, она была увеличена в пять раз: хранился оригинальный датасет и 4 датасета, для которых был применен только один из типов преобразования (a)–(d). Таким образом, для предсказания класса конкретного элемента, проводилось голосование по результатом на 5 тестовых наборах. На этом эксперименте точность получилась равной **0.9741**, что хуже, чем в предыдущих экспериментах. Матрица ошибок в таком случае имеет следующий вид 8:

Настоящее значение	**Предсказанное значение**									
	0	1	2	3	4	5	6	7	8	9
0	978	1	0	0	0	0	0	1	0	0
1	0	1132	2	1	0	0	0	0	0	0
2	14	0	1003	0	1	0	0	9	5	0
3	0	1	2	985	1	5	0	4	8	4
4	2	2	0	0	946	0	6	2	0	24
5	5	0	0	15	1	854	5	1	7	4
6	4	3	0	0	1	2	948	0	0	0
7	4	10	6	0	1	1	0	997	0	9
8	7	2	2	11	3	1	5	4	935	4
9	9	8	2	4	7	4	1	7	4	963

Таблица 8: Матрица ошибок после преобразований тестовой выборки (1)

Видно, что появились нетипичные ошибки для предыдущих экспериментов, например, цифры 2, 9 стала часто распознаваться как ноль.

Вероятно, дело оказалось в том, что тестовая выборка, полученная преобразованием-сдвигом, имела низкую точность распознавания (0.8797), поэтому было принято решение убрать её из процедуры голосования, и тогда точность оказалась равной **0.9754**, что лишь немного уступает качеству на предыдущем эксперименте. Таблица ошибок имеет следующий вид 9

Настоящее значение	**Предсказанное значение**									
	0	1	2	3	4	5	6	7	8	9
0	977	0	0	0	0	0	2	1	0	0
1	1	1127	3	0	0	0	4	0	0	0
2	4	1	1016	1	1	0	1	5	3	0
3	1	1	2	991	0	8	0	4	2	1
4	0	3	0	0	951	0	5	2	0	21
5	4	0	0	10	2	856	9	1	6	4
6	5	2	0	0	1	2	948	0	0	0
7	0	17	7	0	2	0	0	991	0	11
8	6	2	4	16	3	4	6	2	928	3
9	3	8	3	6	8	0	0	8	4	969

Таблица 9: Матрица ошибок после преобразований тестовой выборки (2)

Действительно, после такого исправления, значительно снизилилось количество нетипичных ошибок из предыдущего опыта. Но при этом матрица оказалась очень похожей на матрицу 7 из предыдущего эксперимента.

Хоть этот подход показал себя достаточно похоже на предыдущих в рамках этой модели, можно провести качественное сравнение этих двух подходов.

К преимуществам аугментации обучающей выборки можно отнести то, что в обучающей выборки появляется большое количество уникальных объектов, а это может повысить устойчивость к шумам и небольшим вариациям в данных. Недостатком такого подхода является увеличение длительности работы из-за большого размера обучающей выборки и риск переобучения: аугментация может сильно исказить входные данные. К преимуществам аугментации тестовой выборки относится устойчивость к случайным ошибкам, поскольку для предсказание используется не одна реализация объекта, а его вариации. Недостатком может служить то, что мы не используем возможные преимущества аугментацией обучающей выборки.

3 Улучшения

3.1 Комбинация предыдущих методов.

Когда обучение происходило по оригинальной обучающей выборке, показатель точности не улучшился. Поэтому было принято решение попробовать провести обучение не по оригинальной, а по аугментированной обучающей выборке (120000 элементов). Процедура голосования была такой же, как в шестом эксперименте. В этом случае, точность оказалась **0.9768**.

Далее, перебрав различные комбинации параметров распознавания, была получена точность **0.9783**: голосование проводилось по 6 презентациям тестовой выборки: пяти из предыдущего пункта с весом 1, и одним новым, полученным применением сразу всех типов преобразования с весом 2. В этом случае матрица ошибки выглядит следующим образом (табл. 10):

Настоящее значение	**Предсказанное значение**									
	0	1	2	3	4	5	6	7	8	9
0	979	0	0	0	0	0	0	1	0	0
1	0	1132	3	0	0	0	0	0	0	0
2	7	2	1015	0	0	0	0	6	2	0
3	0	0	2	993	1	6	0	4	2	2
4	1	1	0	0	956	0	4	1	0	19
5	3	0	0	13	2	861	7	1	1	4
6	4	3	0	0	1	2	948	0	0	0
7	1	14	6	1	1	0	0	998	0	7
8	5	1	3	16	3	3	5	4	931	3
9	5	7	2	7	7	1	0	6	4	970

Таблица 10: Матрица ошибок (комбинированный метод)

Видно значительное улучшение качества по сравнению с предыдущими моделями. Цифры 1 и 3, например, распознаются с очень хорошей точностью (1 и 3 ошибки соответственно). И несмотря на то, что в то же время количество ошибок, при распознавании 8 и 9 достаточно велико – 43 и 39, это все равно строго меньше, чем в предыдущих методах.

4 Выводы

В результате использования метода k-ближайших соседей (KNN) для задачи распознавания рукописных цифр наилучшие показатели были достигнуты при использовании косинусной метрики и взвешивания ближайших соседей.

Аугментация обучающей и тестовой выборки также сыграла роль в улучшении результатов. Добавление дополнительных вариаций изображений цифр (например, повороты, сдвиги, масштабирование) позволило модели лучше обобщать и распознавать различные неточности в написании цифр.