Отчет по заданию 1

«Метрические алгоритмы классификации»

курс «Практикум на ЭВМ»
кафедра ММП ВМК МГУ

Михеев Борис, 317 группа

9 октября 2021 г.

Формулировка задания

В данной задаче требуется реализовать метод классификации k ближайших соседей, провести исследования качества осуществляемой им классификации и времени его работы в зависимости от гиперпараметров алгоритма, вида и размера входных данных, а также от стратегии поиска ближайших соседей. Также требуется реализовать кросс-валидацию. Для реализации используется язык `Python` и библиотеки `numpy, scikit-learn, scikit-image, pandas, matplotlib`. Исследования проводятся на наборе данных `MNIST`, содержащем изображения рукописных цифр. В данных 70000 объектов, которые разделяются на 10 классов - цифры от 0 до 9. Из них 60000 объектов отнесены к обучающей выборке, 10000 - к тестовой.

Исследование времени работы алгоритма

Исследуем время работы метода k ближайших соседей в зависимости от стратегии поиска ближайших соседей и количества признаков у объектов. Из стратегий рассматриваются:

- `my_own` - основана на нахождении попарных расстояний между объектами и поиске объектов, наименее удаленных от данного путем сортировки строк матрицы попарных расстояний.
- `brute` - также использует матрицы попарных расстояний, но с более эффективными методами поиска ближайших соседей (вероятно, используются эффективные способы поиска порядковых статистик без сортировки массивов)
- `kd_tree` - используется структура данных k-мерное дерево. В корне точки разделяются через гиперплоскость, то есть спуск от корня к листьям идет по некоторому условию, после определяется ближайший по расстоянию элемент. Далее от корня запускается алгоритм поиска ближайшего элемента в заданном диапазоне радиуса, равного найденному на предыдущем шаге расстоянию. Значение радиуса обновляется при нахождении более близкого элемента.
- `ball_tree` - используется структура `ball tree`. Вершины просматриваются от корня к листьям. Используется структура данных куча для хранения k ближайших точек. В вершине возможны три случая. Если расстояние от рассматриваемой точки до вершины

больше, чем у самой дальней точки из кучи, то куча не изменяется. Если вершина - лист, то просматриваются все точки в ней и куча обновляется соответствующим образом. Если вершина не листовая, то алгоритм вызывается рекурсивно для обоих поддеревьев, ищется поддерево с центром, ближайшим к рассматриваемой точке. Куча возвращается после завершения обоих вызовов и соответствующих изменений.

Для вычисления расстояний между объектами используется евклидова метрика. Множества признаков для рассмотрения выбираются случайным образом, один раз для всех объектов. Ищется k=5 ближайших соседей.

Стратегия	**Количество признаков**	**Время работы, сек**
`my_own`	10	76.11379
	20	82.52884
	100	250.88724
`brute`	10	10.78307
	20	12.30904
	100	14.42989
`kd_tree`	10	5.75891
	20	5.78681
	100	183.85219
`ball_tree`	10	8.85296
	20	25.21715
	100	200.53389

Таблица 1: Время работы различных стратегий, евклидова метрика, k=5

Дольше всего работает стратегия `my_own`, время ее работы увеличивается с ростом числа признаков. brute работает примерно одинаковое время, оно растет незначительно. Стратегия `kd_tree` при малом числе признаков работает несколько быстрее `ball_tree`, далее с ростом числа признаков разница во времени их работы становится более заметной, стретегия `kd_tree` оказывается быстрее. `kd_tree` и `ball_tree` являются самыми быстрыми при малом числе признаков, далее они работают дольше остальных стратегий.

Отметим, что `my_own` работает медленее `brute`, использующей тот же подход с испоьзованием матрицы попарных расстояний. Можно предположить, что это происходит вследствие использования более эффективных способов поиска первых k порядковых статистик по строкам в стратегии `brute`, сложность которых линейно зависит от k. В `my_own` же используется полная сортировка строк матрицы.

Также стоит отметить, что при большом количестве признаков время работы `kd_tree` и `ball_tree` становится большим и близким к времени работы стандартного алгоритма `my_own`. При этом при малом числе признаков они оказываются быстрее остальных стратегий.

Таким образом, далее будет использоваться стретегия `brute` как наиболее оптимальная.

Исследование точности алгоритма в зависимости от числа ближайших соседей и метрики

Оценим по кросс-валидации с разбиением на 3 фолда точность и время работы алгоритма в зависимости от числа ближайших соседей k и используемой при расчетах метрики. Рассмотрим значения k в диапазоне от 1 до 10, в качестве метрик рассмотрим евклидову и косинусную. Качество работы оценим по метрике `accuracy` - доле правильно предсказанных меток класса.

Число ближайших соседей	**Метрика**	**Точность на 1 фолде**	**Точность на 2 фолде**	**Точность на 3 фолде**	**Среднее значение точности**
1	евклидова	0.96895	0.96675	0.9667	0.96746
	косинусная	0.9733	0.9708	0.97045	0.97152
2	евклидова	0.9603	0.95995	0.96125	0.9605
	косинусная	0.96895	0.9664	0.96775	0.9677
3	евклидова	0.96955	0.96825	0.96715	0.96832
	косинусная	0.97365	0.97155	0.9709	0.97203
4	евклидова	0.96705	0.9667	0.9671	0.96695
	косинусная	0.9725	0.97095	0.9722	0.97188
5	евклидова	0.9681	0.9673	0.9672	0.96753
	косинусная	0.97275	0.97015	0.9716	0.9715
6	евклидова	0.96555	0.96495	0.9657	0.96539
	косинусная	0.97195	0.9697	0.97215	0.97126
7	евклидова	0.9652	0.965	0.9656	0.96527
	косинусная	0.97125	0.96815	0.97075	0.97005
8	евклидова	0.96415	0.9639	0.96525	0.96443
	косинусная	0.97125	0.96835	0.97155	0.97038
9	евклидова	0.9637	0.9632	0.96455	0.96382
	косинусная	0.97005	0.967	0.9707	0.96925
10	евклидова	0.96245	0.9622	0.9637	0.96278
	косинусная	0.96935	0.96615	0.9707	0.96873

Таблица 2: Точность классификации для различных метрик и значений k на кросс-валидации с 3 фолдами

Наилучшие значения `accuracy` достигаются при k=3. При использовании косинусной меры расстояния значения `accuracy` несколько выше. Быстрее всего алгоритм работает для евклидовой метрики, несколько дольше для косинусной (188.8 и 192.8 сек. соответственно).

Сравнение стандартного и взвешенного методов

Далее рассмотрим взвешенный метод k ближайших соседей. В нем учитываются расстояния до конкретных объектов при прогнозе, метки класса объектов берутся с коэффициентом, обратно пропорцильным расстоянию до них. Качество классификации и время работы для него оценим также на кросс-валидации с 3 фолдами и с теми же наборами параметров.

Число ближайших соседей	**Метрика**	**Точность на 1 фолде**	**Точность на 2 фолде**	**Точность на 3 фолде**	**Среднее значение точности**
1	евклидова	0.96895	0.96675	0.9667	0.96746
	косинусная	0.9733	0.9708	0.97045	0.97152
2	евклидова	0.96895	0.96675	0.9667	0.96746
	косинусная	0.9733	0.9708	0.97045	0.97152
3	евклидова	0.97075	0.9691	0.96825	0.96936
	косинусная	0.97495	0.9725	0.97175	0.97306
4	евклидова	0.9713	0.9698	0.9701	0.9704
	косинусная	0.97545	0.9732	0.97365	0.9741
5	евклидова	0.96925	0.96825	0.96875	0.96875
	косинусная	0.974	0.97095	0.973	0.97265
6	евклидова	0.9704	0.96845	0.96945	0.96943
	косинусная	0.97465	0.9714	0.97305	0.97303
7	евклидова	0.96735	0.96595	0.967	0.96676
	косинусная	0.9726	0.9693	0.97195	0.97128
8	евклидова	0.9676	0.96665	0.9678	0.96735
	косинусная	0.97295	0.97045	0.9722	0.97187
9	евклидова	0.9651	0.9643	0.9657	0.96503
	косинусная	0.97165	0.96865	0.97145	0.97059
10	евклидова	0.9647	0.96465	0.9657	0.96501
	косинусная	0.97145	0.96855	0.9714	0.97047

Таблица 3: Точность классификации для различных метрик и значений k на кросс-валидации с 3 фолдами

Взвешенный метод в целом дает лучшие показатели, чем невзвешенный. Оптимальным для данного метода числом соседей является k=4.

Таким образом, в результате экспериментов и оценки на кросс-валидации были найдены оптимальные параметры алгоритма. Далее будем использовать взвешенный метод со стратегией `brute`, косинусной метрикой и k=4 как наилучший.

Применение к тестовой выборке и оценка качества

Применим алгоритм к тестовой выборке, оценим его точность. Полученное значение `accuracy` равно 0.9752. Это значение несколько выше, чем полученное на кросс-валидации, которое равно 0.97408. Причиной этого может быть то, что качество оценок все же зависит от фолдов, и возможно в какой то момент в валидационную выборку попадали объекты, наиболее отличающиеся от объектов обучающей выборки.

Данный результат является одним из лучших среди различных алгоритмов, примененных к датасету MNIST [1]. Алгоритм `knn` и его вариации в среднем дают точность от 0.95 до 0.9937, линейные классификаторы имеют точность около 0.88, алгоритм градиентного бустинга - 0.9913, метод опорных векторов - 0.9944, глубокие нейронные сети - 0.9965, сверточные нейронные сети - до 0.9979. Таким образом, можно судить о достаточно высокой точности данного метода на данном наборе данных.

Построим матрицу ошибок и проанализируем ее. По строкам отмечены истинные метки классов, по столбцам - предсказанные. В каждой ячейке `[i, j]` содержится число объектов класса `i`, отнесенных алгоритмом к классу `j`.

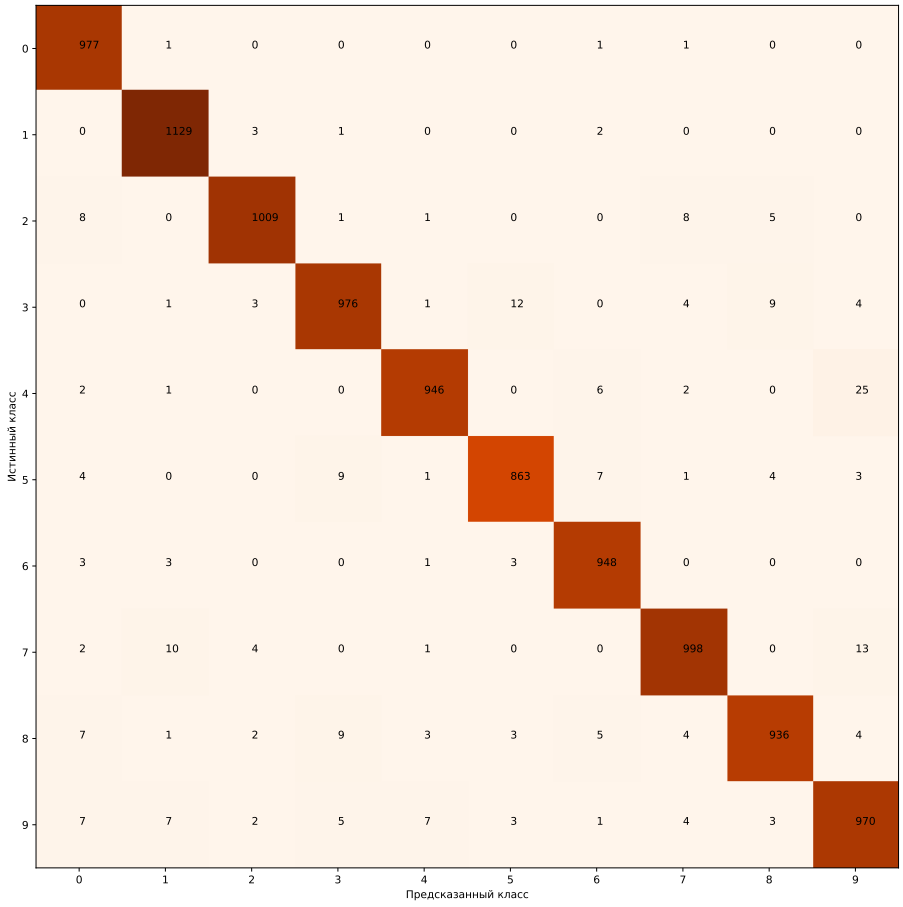

Рис. 1: Матрица ошибок, допущенных на тестовой выборке

Большинство объектов классифицировано верно, ложно классифицированных объектов мало, для каждого из классов количество неверных прогнозов незначительно по сравнению с размером самого класса. Это свидетельствует о хорошей точности алгоритма. Больше всего ошибок допущено на цифрах, схожих друг с другом, имеющих схожие характерные черты в

зависимости от написания, например, 1 и 7, 2 и 7, 4 и 9, 7 и 9, 5 и 3 и т. д.

Для наглядности рассмотрим некоторые объекты, на которых алгоритм допустил ошибку:

Рис. 2: Цифры 7, определенные алгоритмом как 1

Рис. 3: Цифры 7, определенные алгоритмом как 9

Рис. 4: Цифры 5, определенные алгоритмом как 3

Рис. 5: Цифры 4, определенные алгоритмом как 9

Аугментация обучающей выборки

Размножим обучающую выборку при помощи поворотов, сдвигов и использования гауссовского фильтра (размытия), их комбинаций с целью повышения качества алгоритма, ведь в действительности цифры могут быть написаны по-разному, а их изображения и фотографии могут иметь различное качество. Заметим, что такие преобразования как отражение, поворот на 180 градусов и прочие преобразования подобия не могут быть использованы, так как цифры характеризует их конкретное положение (к примеру, перевернутую 6 нельзя отличить от 9 и т. д.). Будем рассматривать повороты на 5, 10, 15 градусов по и против часовой стрелки, сдвиги на 1, 2, 3 пикселя по вертикали и горизонтали, применение фильтра Гаусса для `sigma` 0.5, 1, 1.5 и их комбинации. Размер обучающей выборки - 70000*784, количество всевозможных сочетаний параметров аугментаций - 648. Таким образом, перебрать все сочетания параметров, применить их ко всей обучающей выборке и оценить качество по кросс-валидации является вычислительно сложной задачей, требующей больших затрат времени. Поступим следующим образом. Будем случайно выбирать лишь некоторые сочетания параметров преобразований и применять к части обучающей выборки. Индексы для ее объектов также выберем случайно. К фрагменту исходной выборки добавим преобразованные версии так, чтобы получить датасет того же размера, что и исходный, и объекты в нем перемешаем. Подберем оптимальные параметры преобразований по кросс-валидации с 3 фолдами. В таблице приведем лишь некоторые из рассмотренных наборов значений параметров.

Угол поворота	**Сдвиг по горизонтали**	**Сдвиг по вертикали**	**Параметр фильтра Гаусса**	**Точность классификации**
5	1	-3	0.5	0.9882
10	-2	-2	1	0.9817
15	2	1	0.5	0.98062
-10	2	3	1.5	0.97675
15	-3	2	1	0.97773
-10	-3	-1	1	0.98248
-15	1	2	1	0.9775
-15	3	-1	1.5	0.97273
-15	-3	1	1.5	0.97227
15	1	-1	0.5	0.983
5	-1	2	0.5	0.98843
5	2	-1	1.5	0.98007
5	-2	-2	1.5	0.97955
10	-1	2	1.5	0.97725
10	-2	-1	0.5	0.98502

Таблица 4: Точность классификации при различных наборах параметров аугментации обучающей выборки

В целом результат работы алгоритма стал лучше. Выберем лучший набор параметров среди протестированных: (5, -1, 2, 0.5). Применим алгоритм с соответствующими параметрами на тестовой выборке и оценим качество. Получим значение `accuracy` 0.9777, что свидетельствует о повышении качества классификации при использовании аугментации обучающей выборки. Проанализируем матрицу ошибок:

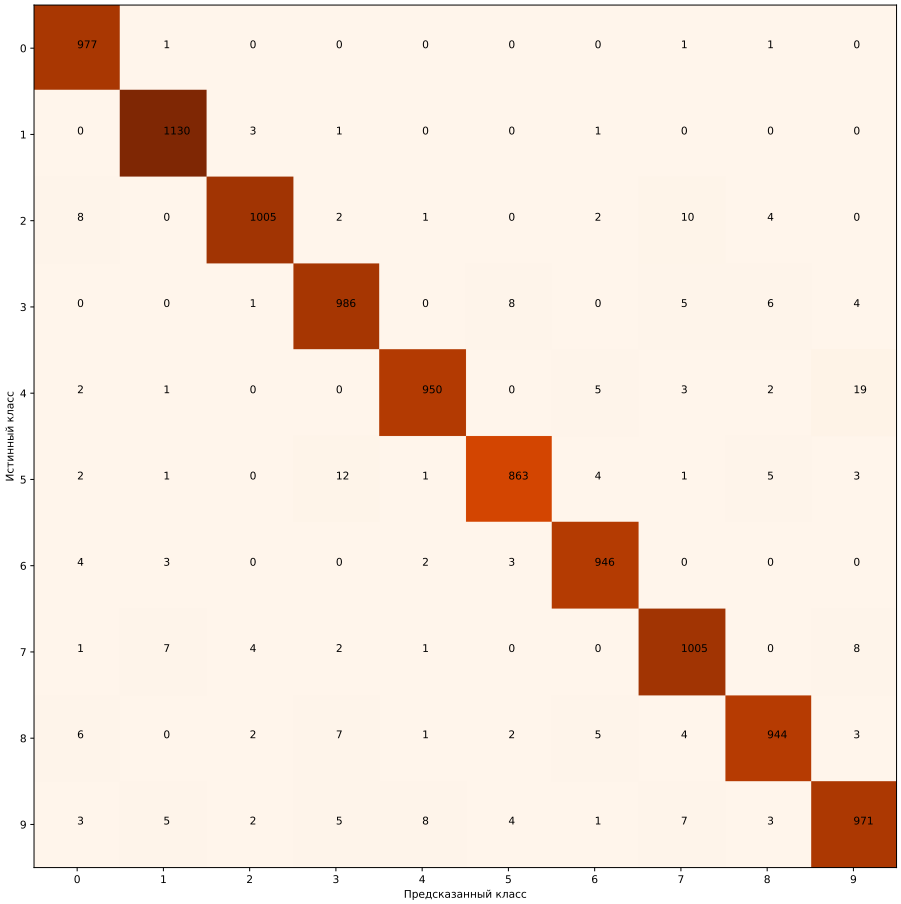

Рис. 6: Матрица ошибок, допущенных на тестовой выборке при использовании аугментации обучающей выборки

Большинство объектов все так же классифицировано верно, и количество ошибок на схожих цифрах уменьшилось. Можно предположить, что наиболее значительными преобразованиями являются поворот и размытие гауссовским фильтром, так как изображения цифр зависят от качества и ракурса съемки, а также почерка конкретного человека.

Аугментация тестовой выборки

Теперь применим те же преобразования к тестовой выборке, а обучать модель будем на исходной обучающей выборке. Каждый из объектов будет представлен также своими преобразованными версиями, а прогноз для исходного объекта будет получаться в результате голосования среди преобразованных объектов, т. е. будет браться самый часто встречающийся прогноз среди результатов работы для исходного и преобразованных объектов. Будем ис-

пользовать те же параметры аугментации, что и при преобразовании тестовой выборки как оптимальные. Получим значение `accuracy` 0.9755, что свидетельствует о незначительном улучшении результата в сравнении с алгоритмом, обученным и оцененным на исходных выборках. Рассмотрим матрицу ошибок:

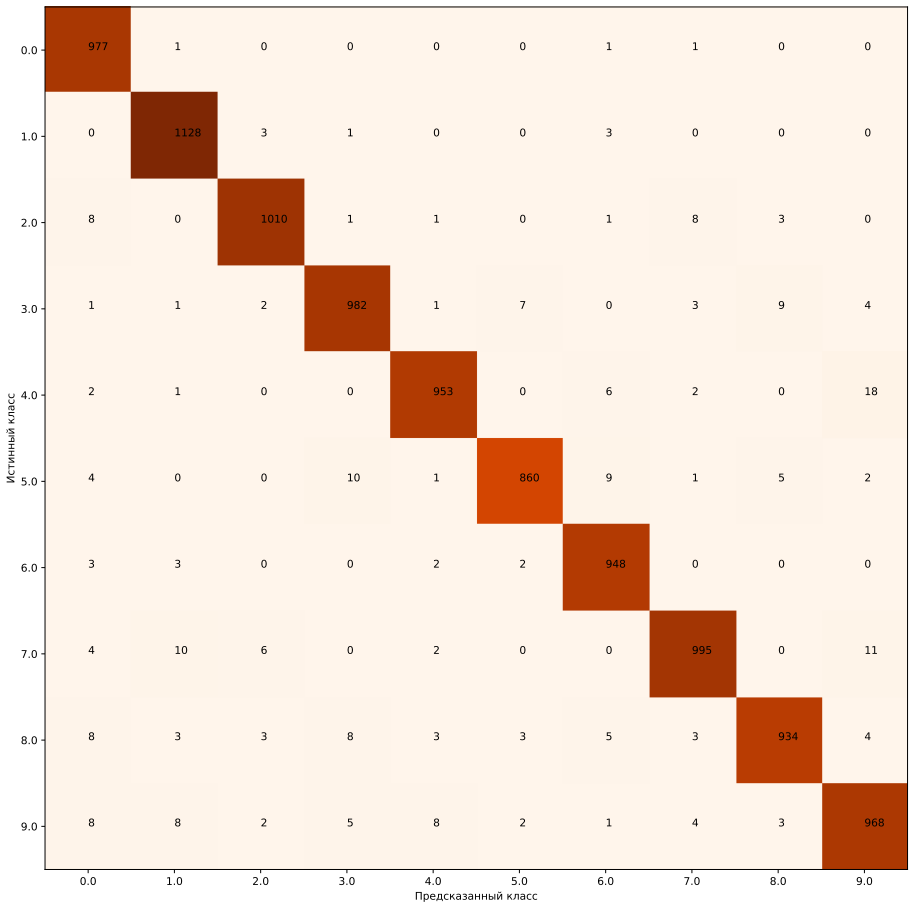

Рис. 7: Матрица ошибок, допущенных на тестовой выборке при использовании аугментации тестовой выборки

По прежнему объекты классифицированы в большей части верно, чуть меньше ошибок допущено на схожих изображениях.

Естественно, еще лучшие результаты при аугментации обучающей и тестовой выборки можно было бы получить, проведя полный перебор параметров с использованием полной выборки и найдя оптимальные.

Сравнив преобразование обучающей и тестовых выборок, можно заметить, что преобразование тренировочной выборки дало лучший результат. В целом целесообразно проводить

перебор и остальных параметров алгоритма, а также применять аугментацию как к обучающей, так и к тестовой выборке. Это может быть полезно, когда обучающая выборка содержит множество различных представителей классов, и их количество сбалансированно, как в случае датасета `MNIST`. Аугментация же тестовых объектов даст преимущество в случае использования алгоритма на зашумленных и некачественных изображениях, что согласуется с реальностью.

Вывод

В работе было проведено исследование метода k ближайших соседей классификации, его различных версий и модификаций, оценено качество и скорость его работы в зависимости от гиперпараметров и входных данных, используемых метрик, а также была рассмотрена идея аугментации обучающей и тестовой выборок как способ улучшения результата работы, оценено ее влияние на качество классификации. В целом алгоритм показывает высокие значения метрики `accuracy`, что говорит о достаточно высоком качестве его работы, однако он является вычислительно трудоемким при большом размере входных данных. Наилучшие результаты на данных `MNIST` алгоритм показывает при использовании стратеги `brute`, косинусной метрики и коэффициентов с учетом расстояний до объектов (взвешенный метод), и эти результаты являются одними из лучших среди различных алгоритмов на данном наборе данных. Также была показана эффективность идеи аугментации обучающей и тестовой выборок, примененной даже лишь к некоторому неполному набору параметров и неполной выборке. Однако подбор параметров аугментации и обучение на расширенной исходной выборке требуют множества вычислений и больших затрат времени.

Список литературы

[1] Yann LeCun, Corinna Cortes, Christopher J.C. Burges. (1999). The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.