Задание 1. Метрические алгоритмы классификации

Сумина Евгения, практикум 317 группы

2020

Введение

В задании требовалось написать на языке Python собственные реализации метода ближайших соседей и кросс-валидации и оформить их в виде модулей. Затем провести с датасетом изображений цифр MNIST следующие эксперименты:

1. Исследовать скорость работы различных алгоритмов поиска.
2. Оценить точность и время работы алгоритма в зависимости от некоторых факторов.
3. Сравнить взвешенный метод k-ближайших соседей с методом без весов.
4. Проанализировать, в каком случае возникают ошибки.
5. Рассмотреть различные подходы аугументации выборки как возможность повышения точности прогноза.

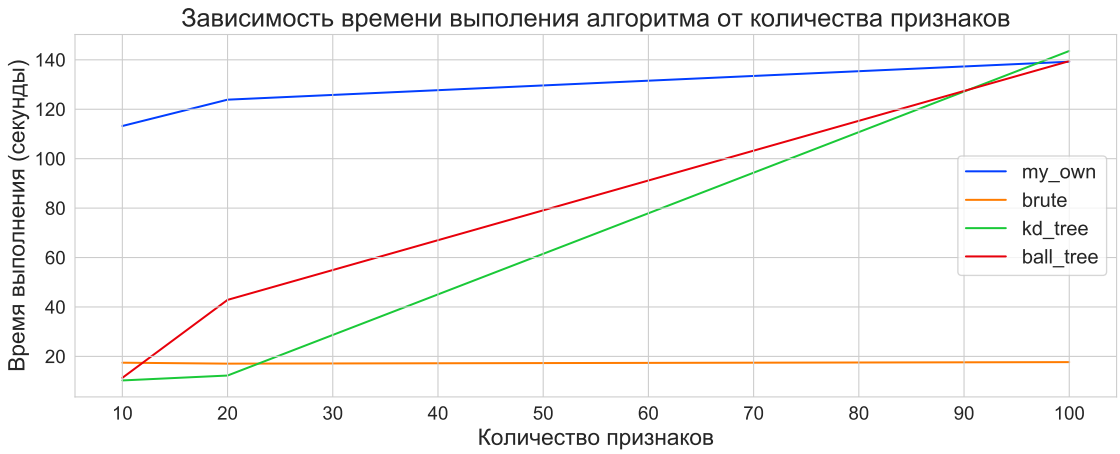
Первый эксперимент

Сравнительный анализ проводился для четырех алгоритмов: my_own (собственная реализация метода), brute, Kd-tree и Ball-tree. Из всего множества признаков были выбраны 10, 20 и 100 случайных, для которых строилась тестовая выборка. На графике представлены полученные данные.

Заметим, что brute в среднем работает быстрее в признаковом пространстве большей размерности. Низкая скорость вычислений при использовании Kd-tree и Ball-tree связаны с проклятьем размерности: Kd-tree становится медленнее, чем brute, если количество объектов в выборке N и число признаков k связаны соотношением:

$$N < 2^k$$

Kd-tree разбивает объекты по каждому измерению. В случаях, когда выполнено неравенство, объекты в выборке заканчиваются раньше, чем выполнится разбиение

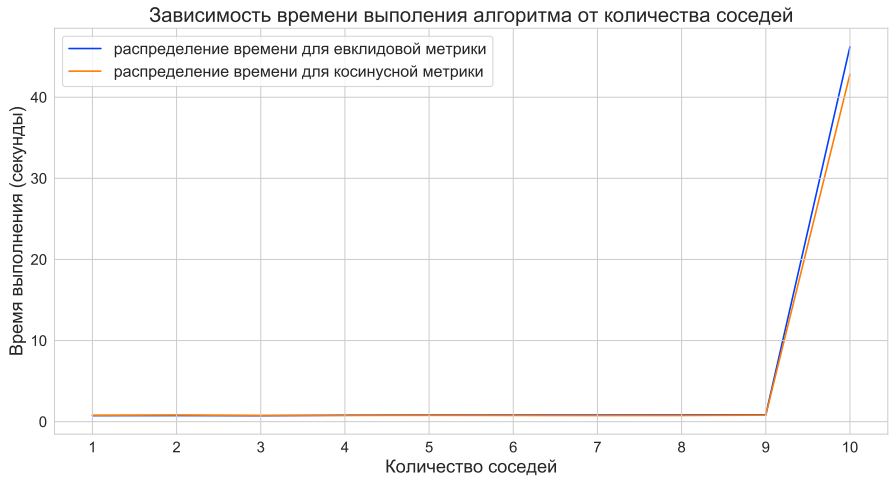


по всем измерениям. В результате для измерений, по которым не было выполнено разбиение, расстояние ищется случайно, что вызывает дополнительные накладные расходы. Для Ball-tree ситуация аналогична.

My_own изначально работает медленнее остальных алгоритмов, т.к. в нем отсутствует оптимизация, доступная методам из библиотеки sklearn. Однако его сложность близка к линейной, поэтому, начиная с пространства определенной размерности, его использование становится более оправданным, чем использование Kd-tree и Ball-tree.

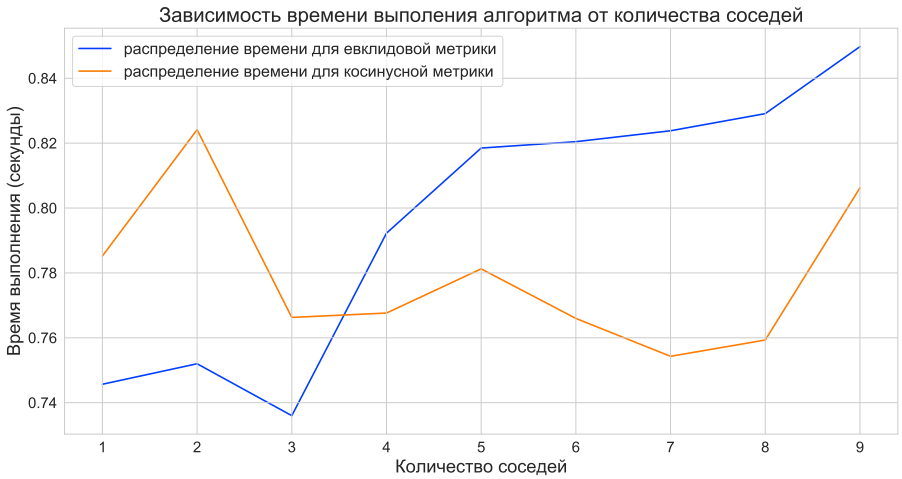
В дальнейших экспериментах используется brute, т.к. время его вычислений минимально.

Второй эксперимент



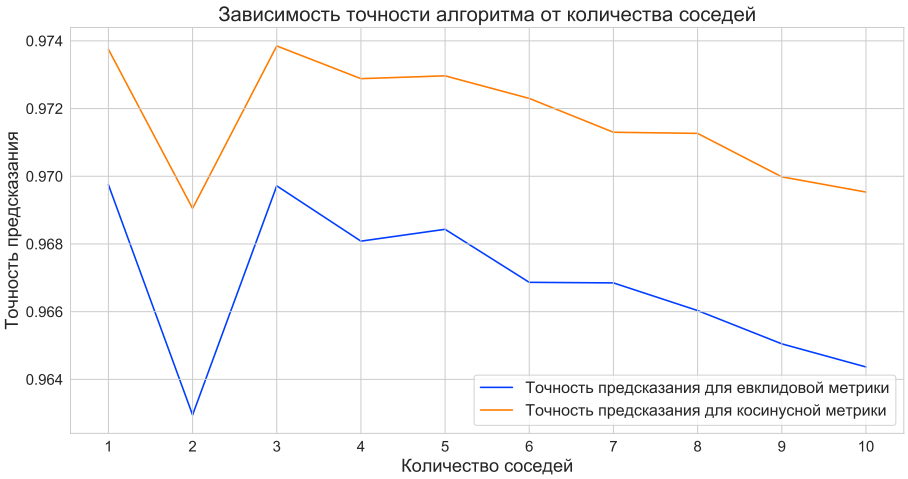
Проанализируем время работы k ближайших соседей в зависимости от выбранной метрики. Заметим, что в этом эксперименте все анализируемые значения берутся усредненными по трем фолдам. Построенный график вызывает определенный

вопрос: почему время выполнения алгоритма настолько велико для десяти соседей, хотя для остальных вариантов оно держится примерно на одинаковом уровне? Однако это легко объясняется использованным подходом: для экономии времени сначала матрица попарных расстояний строится для масимально возможного числа соседей, и все последующие расстояния выбираются уже из нее. Поэтому, чтобы детальнее разобраться во времени работы алгоритма, уберем из рассмотрения случай десяти соседей.



Теперь можно заметить, что косинусная метрика в среднем позволяет производить вычисления быстрее, чем евклидова. Однако все-таки разброс показателей не сильно отличается.

Теперь проанализируем зависимость точности алгоритма от количества соседей и выбранной метрики. Можно заметить, что показатели различаются примерно на одну и ту же величину. При этом точность алгоритма, в котором используется косинусная метрика, всюду выше, чем точность алгоритма с евклидовой метрикой.

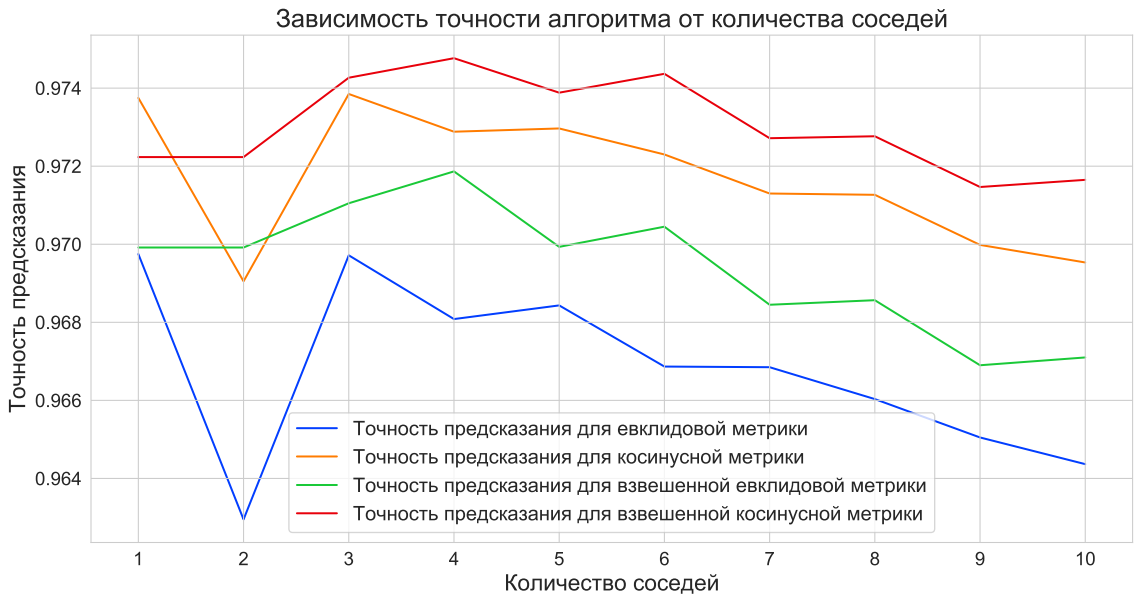


Третий эксперимент

Сравним взвешенный метод k ближайших соседей с методом без весов при тех же фолдах и параметрах. Точность предсказания взвешенного метода получается выше. Это можно объяснить выбранным голосом объекта:

$$w = 1/(distance + \epsilon)$$

Получается, что соседи, расположенные ближе к объекту, имеют больший голос, чем удаленные на большое расстояние. Наибольшая точность достигается для взвешенного метода с четырьмя соседями.



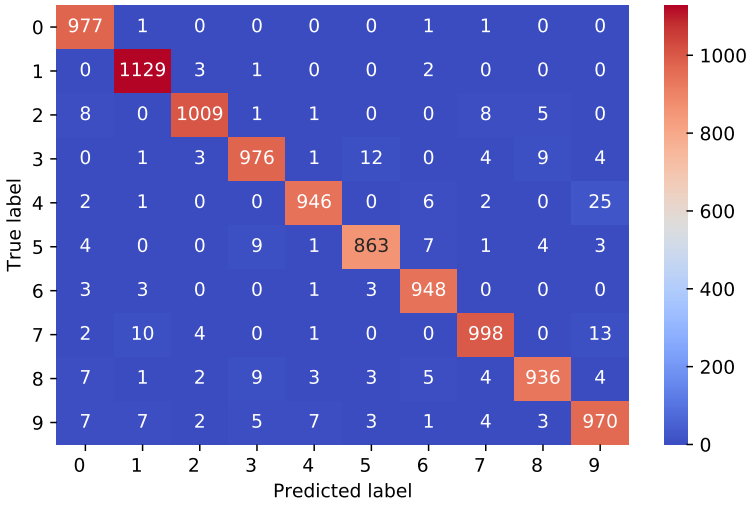
На основе второго и третьего эксперимента в дальнейшем будем рассматривать модель с четырьмя соседями, весами и косинусной метрикой.

Четвертый эксперимент

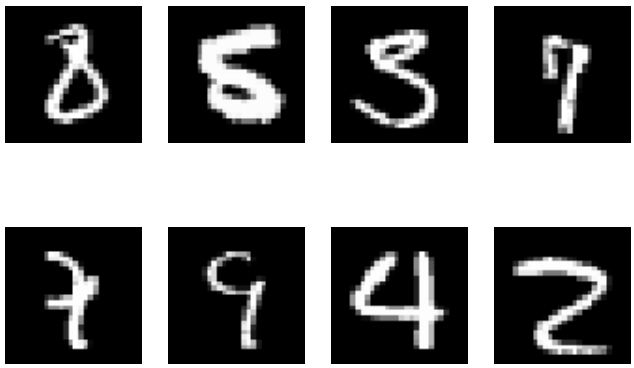
Итак, рассмотрим оценки, полученные по кросс-валидации с тремя фолдами для алгоритма со следующими параметрами: взвешенный brute с косинусной метрикой.

Первый фолд: **0.975**
Второй фолд: **0.974**
Третий фолд: **0.976**

Применяя алгоритм к исходной обучающей и тестовой выборкам, получим точность 0.9752. Этот результат примерно равен результату, полученному на кросс-валидации с тремя фолдами. С помощью больших сверточных сетей удается достигнуть точности 0.9918. С учетом сложности нашей модели, результат получается достойным.



Проанализируем матрицу ошибок. Она имеет явный диагональный вид, что позволяет высоко оценить точность модели. Можно заметить, что чаще всего алгоритм путает четверку с девяткой. Так же часто неверные предсказания получаются для троек (алгоритм выдает пятерку или восьмерку) и для единиц (семерки). Проверим наши выводы, выведя восемь произвольных изображений, на которых модель допустила ошибку.



Действительно, эмпирический анализ оказался верным. Модель ошибается на предсказанном нами наборе цифр. Т.к. датасет изображений состоит из рукописных цифр, написание разных объектов можетоказаться похожим. Так, семерка и единица имеют похожую форму. Тройку, пятерку и восьмерку объединяет округлое написание нижней части числа. Иногда в изображении написание выглядит неаккуратно: не соединены концы числа или число написано под наклоном, из-за этого также возникают ошибки.

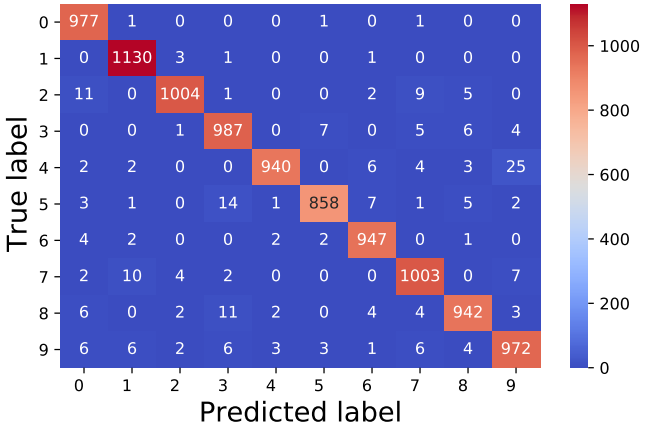
Предсказанное значение	Правильное значение
3	8
8	5
5	3
9	7
1	7
4	9
9	4
7	2

Пятый эксперимент

Попробуем повысить качество предсказания за счет размножения обучающей выборки. Для начала повернем каждый объект из выборки на 5, 10 и 15 градусов соответственно и посмтрим, как это отразится на качестве предсказания. Рассмотрим кросс-валидацию с тремя фолдами.

Поворот	Первый фолд	Второй фолд	Третий фолд
на 5 градусов влево	0.990	0.990	0.989
на 10 градусов влево	0.983	0.983	0.982
на 15 градусов влево	0.978	0.978	0.977
на 5 градусов вправо	0.990	0.990	0.990
на 10 градусов вправо	0.984	0.982	0.983
на 15 градусов вправо	0.977	0.977	0.979

Можно заметить, что больше всего качество предсказания повышается при повороте на пять градусов влево. Проанализируем, как в этом случае изменится матрица ошибок.



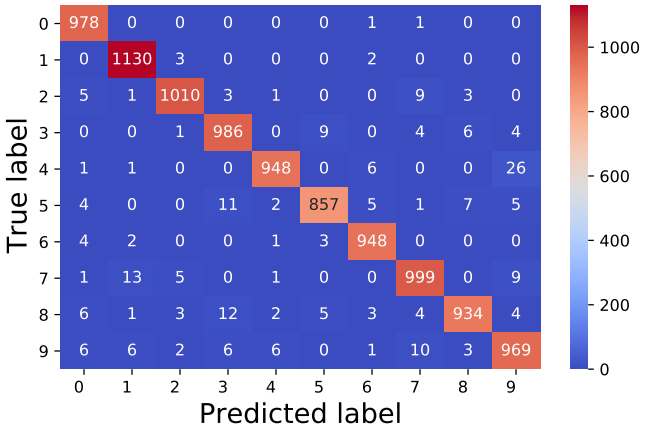
Проведенное преобразование позволило частично понизить уникальные ошиб-

ки. При этом критические места остались практически неизменными: значительно понизились только ложное предсказание троек в класс пятерок и семерок в класс девяток. При обычном запуске на расширенной обучающей выборке и неизменной тестовой качество предсказания повысилось до 0.976.

Теперь произведем сдвиг изображений на 1, 2 и 3 пикселя влево и вверх. Также рассматриваем кросс-валидацию с тремя фолдами:

Сдвиг	Первый фолд	Второй фолд	Третий фолд
на 1 пиксель влево	0.983	0.982	0.983
на 2 пикселя влево	0.977	0.977	0.975
на 3 пикселя влево	0.976	0.975	0.976
на 1 пиксель вверх	0.983	0.982	0.982
на 2 пикселя вверх	0.977	0.976	0.976
на 3 пикселя вверх	0.976	0.974	0.975

Смещение на 1 пиксель влево улучшает предсказание на кросс-валидации. Однако при предсказании на изначальных тестовых данных точность падает до 0.9748. Тогда попробуем расширить выборку за счет смещения на один пиксель вверх. Точность повышается до 0.9759. Проанализируем матрицу ошибок.

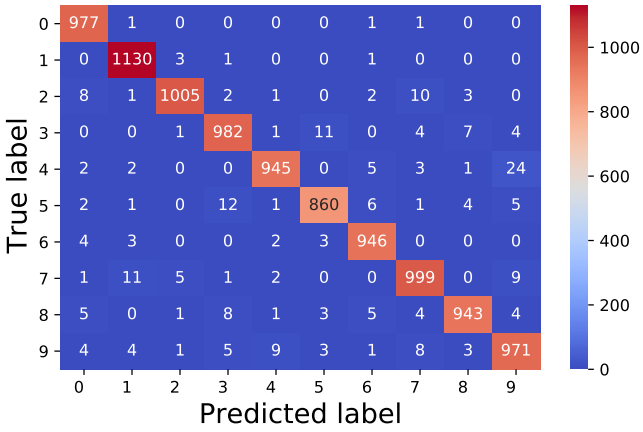


Удалось значительно сократить количество неправильных предсказаний для троек. При этом точность предсказания для пятерок понизилась, а для остальных цифр осталась примерно на том же уровне.

Фильтр	Первый фолд	Второй фолд	Третий фолд
дисперсия = 0.5	0.992	0.993	0.993
дисперсия = 1	0.988	0.989	0.988
дисперсия = 1.5	0.981	0.981	0.981

Теперь применим фильтр Гаусса с дисперсиями 0.5, 1 и 1.5 на кросс-валидации с тремя фолдами.

Применение фильтра Гаусса с дисперсией 0.5 дает наибольший результат. Применим это преобразование к изначальной выборке: точность удалось повысить до значения 0.9758. Проанализируем матрицу ошибок.



Нам удалось повысить точность предсказания для восьмерок, семерок и троек.

Все три преобразования оказывают эффект на предсказания примерно одного и того же типа объектов. Попробуем объединить их и посмотреть, какая точность получится в этом случае. При применении всех трех преобразований точность предсказания падает до 0.9751. При применении сдвига фильтра Гаусса до 0.9749. При применении поворота с фильтром Гаусса и поворота со сдвигом точность вырастает до 0.9753. Получается, наибольшая точность достигалась при размножении выборки за счет поворота: 0.976.

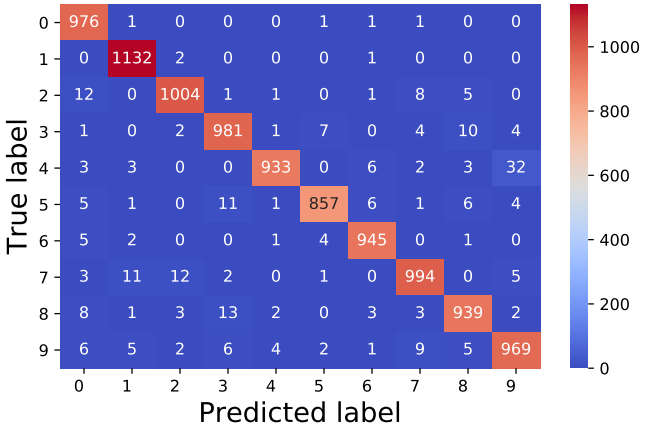
Шестой эксперимент

Теперь рассмотрим похожий алгоритм, но будем применять указанные в пятом эксперименте преобразования в тестовой выборке и проанализируем, как это повлияет на точность предсказания.

Поворот на 5 градусов влево:	**0.974**
Поворот на 10 градусов влево:	**0.965**
Поворот на 15 градусов влево:	**0.946**
Поворот на 5 градусов вправо:	**0.974**
Поворот на 10 градусов вправо:	**0.969**
Поворот на 15 градусов вправо:	**0.949**
Смещение на 1 пиксель влево:	**0.955**
Смещение на 2 пикселя влево:	**0.840**
Смещение на 3 пикселя влево:	**0.577**
Смещение на 1 пиксель вверх:	**0.948**
Смещение на 2 пикселя вверх:	**0.790**
Смещение на 3 пикселя вверх:	**0.609**

Фильтр Гаусса с дисперсией 0.5: **0.974**
Фильтр Гаусса с дисперсией 1: **0.961**
Фильтр Гаусса с дисперсией 1.5: **0.919**

Применение преобразований к тестовой выборке скорее снижает качество прогноза, в некоторых случаях даже довольно значительно. Попробуем применить к выборке систему преобразований, состоящую из поворота на 5 градусов вправо и фильтра Гаусса с дисперсией 0.5. В таком случае точность равна 0.973. Проанализируем, как изменилась матрица ошибок.



Улучшилось предсказание для троек и восьмерок, для остальных цифр результаты в среднем стали незначительно хуже.

Качественно два подхода различаются тем, что в первом случае мы увеличиваем выборку, соответственно, возрастает время обучения алгоритма и затраты по памяти, однако это позвляет делать более аккуратные прогнозы. Во втором случае мы преобразовываем тестовые объекты, что не влияет на время обучения. Но могут возникнуть ситуации, когда примененное преобразование скорее мешает модели, чем улучшает ее, как получилось и в нашем случае. Я думаю, подход с размножением обучающей выборки предпочтительнее.