Московский государственный университет
имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчёт по заданию № 1

Выполнил:
студент 317 группы
Находнов М. С.

Москва
2018

Содержание

Введение

В данном отчёте описаны эксперименты, направленные на изучение качества и скорости работы алгоритма K-ближайших соседей (K-NN).

Все эксперименты проведены на изображениях из датасета MNIST (60000 - размер обучающей выборки, 10000 - размер тестовой выборки).

Эксперименты

Подбор параметра K

N features	my_own	brute	kd_tree	ball_tree
10	**0.0**/48.65/48.65	**0.0**/15.83/15.83	6.81/**4.24**/**11.05**	6.53/5.74/12.27
20	**0.0**/70.78/70.78	**0.0**/16.58/16.59	6.43/**3.54**/**9.97**	6.18/14.36/20.54
100	**0.0**/73.03/73.03	0.01/**17.05**/**17.06**	6.72/177.03/183.75	6.77/236.52/243.28

Таблица 1: Время работы различных алгоритмов в зависимости от размерности пространства признаков (fit time/predict time/fit + predict time). Измерения проводились усреднением по 3 независимым запускам

Из таблицы 1 видно, что при небольшой размерности пространства признаков kd_tree и ball_tree являются самыми быстрыми, однако, при увеличении размерности время работы алгоритмов, использующих деревья, значительно увеличивается и, brute становится самым быстрым алгоритмом. В свою очередь, my_own уступает brute в скорости работы. Это, вероятно, связано с тем, что brute использует реализацию K-NN из библиотеки sklearn, написанной на C, за счёт чего время исполнения программ, содержащих циклы значительно уменьшается.

Так как в дальнейшем тестирование будет происходить в пространстве признаков высокой размерности (728), то логично выбрать алгоритм brute, как самый быстрый и по времени обучения, и по времени поиска ближайших соседей.

Подбор метрики и функции взвешивания

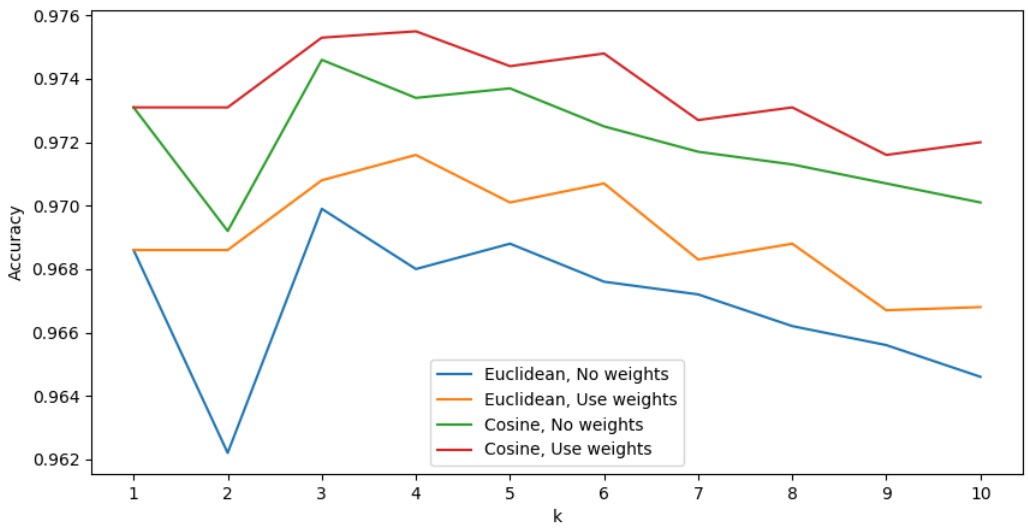

Рис. 1: Точность на тестовой выборке в зависимости от числа соседей, метрики и функции взвешивания

Euclidean; No weights	Euclidean; Use weights	Cosine; No weights	Cosine; Use weights
199.60	193.57	413.96	**414.58**

Таблица 2: Время работы алгоритмов в зависимости от используемой метрики и функции взвешивания. Замер производился на кросс валидации с 3 фолдами.

Алгоритм K-NN зависит от выбора метрики в пространстве признаков. Было рассмотрено две метрики — евклидово расстояние и косинусное расстояние. Также, для вычисления предсказания алгоритм может использовать различные способы взвешивания ближайших соседей. Первый способ — все соседи равнозначны и дают одинаковый вклад. Второй — вес от соседа обратно пропорционален расстоянию между этим соседом и объектом тестовой выборки: $w_i = \frac{1}{||x-x_i||+10^{-5}}$.

Из рисунка 2 видно, что использование косинусного расстояния в качестве метрики значительно превосходит евклидову метрику вне зависимости от числа соседей. Добавление взвешивания с учётом расстояния между объектами тестовой и обучающей выборок позволяет улучшить точность как в случае евклидовой метрики, так и в случае косинусного расстояния.

Как следствие, для дальнейших экспериментов будет использоваться косинусная метрика, взвешивание объектов с четырьмя ближайшими соседями.

Однако, использование косинусной метрики замедляет время работы K-NN примерно в два раза 2. При этом взвешивание объектов с учётом расстояния не влияет на скорость работы.

Анализ работы K-NN

-	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 3: Confution matrix, 4-NN, Cosine with weights

Test	SOTA	Cross validation
0.9752	0.9979	0.9755

Таблица 4: Точность на тестовой выборке с помощью 4-NN; лучший достигнутый результат [1]; точность на кроссвалидации с помощью 4-NN

Таблица 4, показывает, что оценка полученная на кросс валидации близка к оценке на тестовой выборке. Это говорит о том, что подбор параметров с помощью обучающей выборки на кросс валидации не приводит к переобучению. При этом лучший результат, полученный с помощью K-NN, значительно (на 4.5%) уступает state-of-the-art результатам.

Из confution matrix 3 видно, что наибольшее число ошибок допускается на визуально похожих объектах, например, 4 и 9, 3 и 8, 7 и 1. Рисунок 2 показывает, что ошибки классификации происходят на тех объектах, где цифра изображена, или неразборчиво, или похожей по начертанию на другую цифру.

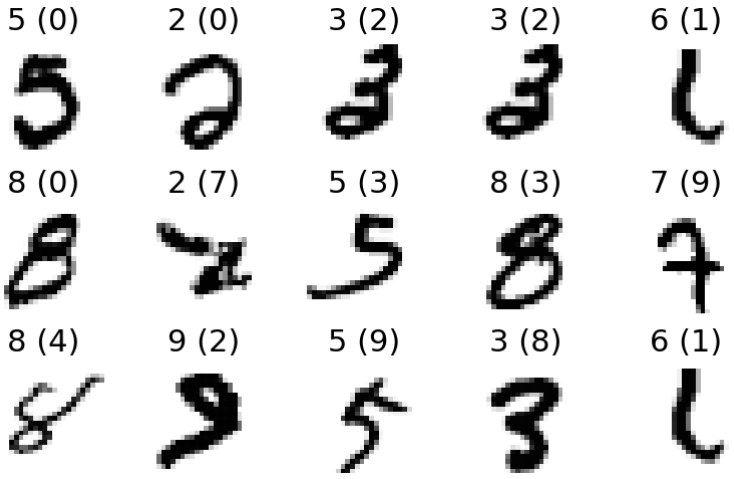

Рис. 2: Неверно классифицированные объекты. Корректная метка (Предсказанная метка)

Аугментация

-	Train transformation	Test tranformation
Rotation, $angle = 5$	0.9765	0.9764
Rotation, $angle = 10$	0.9802	**0.9796**
Rotation, $angle = 15$	0.9785	0.9781
Shift, $x\_shift = y\_shift = 1$	0.9760	0.9760
Shift, $x\_shift = y\_shift = 2$	0.9756	0.9756
Shift, $x\_shift = y\_shift = 3$	0.9752	0.9752
Blur, $sigma = 0.5$	0.9758	0.9729
Blur, $sigma = 1.0$	**0.9814**	0.9632
Blur, $sigma = 1.5$	0.9782	0.9278

Таблица 5: Точность в результате трансформации обучающей или тестовой выборки

«Размножение» выборки, как обучающей, так тестовой (за исключением размытия) или улучшает качество (повороты), или оставляет его без изменения (сдвиги).

Ожидаемо, что линейные трансформации дают одинаковый эффект, что при применении к обучающей, что при применении к тестовой выборке. Это связано с тем, что, например, для поворотов в выборку добавляются как повороты на лево, так и направо с одним и тем же углом поворота. Небольшое отличие в точности, связано с тем, что при поиске k ближайших соседей для аугментированной тестовой выборки, находился список из k соседей для каждого варианта изменения тестового объекта. Затем эти списки объединялись и выбирались k ближайших объектов. Такой подход допускал повторения соседей в итоговом списке k ближайших соседей.

Рассмотрим те классы для которых K-NN стал давать больше правильных ответов. Для этого рассмотрим лучшее из всех преобразований (blur, $sigma = 1.0$, train transformation) и построим разность исходной confution matrix и матрицы после применения аугментации

-	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	-1	1	0	0
1	0	1	0	-1	0	0	0	0	0	0
2	-1	1	-3	1	0	0	1	4	-3	0
3	0	-1	-2	10*	0	-3	0	0	-4	0
4	-2	-1	0	0	14*	0	-2	1	0	**−10**
5	-2	1	0	-3	0	9	-3	0	-2	0
6	0	0	0	0	0	0	0	0	0	0
7	-2	-3	2	0	2	1	0	5	0	-5
8	-4	-1	-1	**−6**	0	0	-2	0	15*	-1
9	**−5**	-3	-2	-3	1	0	0	2	-1	11*

Таблица 6: Разность confution matrix до и после аугментации.

Как видно из таблицы 6 наибольшие улучшения произошли для классов 4 и 9, 3 и 8, 0 и 9 — классы, которые могут быть сложно отличимы друг от друга даже визуально. При этом, применение размытия больше всего повлияло на классы, в которых цифры имеют «округлые», выпуклые части (3, 8, 9).

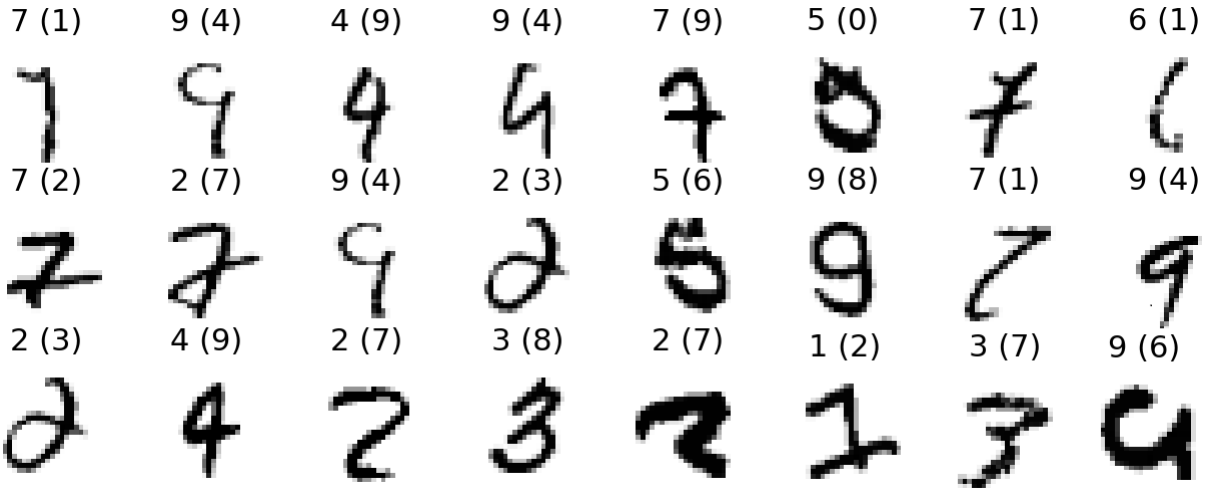

Рис. 3: Неверно классифицированные объекты. Корректная метка (Предсказанная метка)

Изучение неверно классифицированных объектов (рис. 3) показывает, что наиболее часто ошибки происходят на цифрах, где нечёткое написание петелек или стиль написания цифры (например 1 и 7) делает затруднительным определение правильного класса даже человеку.

Несмотря на то, что применение размытия к обучающей выборке приводит к наибольшему увеличению точности классификации, однако, применение аналогичного преобразования к тестовой выборке напротив значительно ухудшает качество работы.

При этом, как было сказано ранее, при применении линейных трансформаций разницы между аугментацией тестовой выборки и аугментации обучающей выборки нет, но при этом, аугментация тестовой выборки работает значительно быстрее, чем аугментация обучающей выборки.

Итого, можно сказать, что аугментация данных может как улучшать качество, так и ухудшать качество и, как следствие, на практике правильный метод изменения данных следует подбирать как гиперпараметр, оценивая качество, например, по кросс валидации.

Выводы

Алгоритм K-NN показал, что он может применяться для задачи классификации изображений и достигать приемлемой точности. Однако, от правильного выбора гиперпараметров (алгоритма поиска ближайших соседей, числа ближайших соседей, метрики, функции взвешивания, вида аугментации) существенно зависит как скорость работы алгоритма, так качество классификации тестовой выборки.

Список литературы

[1] http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html