Исследование работы алгоритма KNearestNeighbors на примере датасета MNIST

Тыцкий Владислав

Октябрь 2020

Введение

Требуется решить задачу классификации с помощью метрического метода KNearestNeighbors(метод K ближайших соседей) на примере известного датасета MNIST.

MNIST - база данных рукописных цифр. Каждая цифра представляется в виде черно-белого изображения 28×28 пикселей, что эквивалентно вектору $x \in \mathbb{R}^{784}$. Датасет содержит 70000 размеченных цифр. В данном исследовании мы будем использовать обучающую выборку размера 60000, а тестовую соответсвенно 10000. [1]

Задание №1

Требуется сравнить различные алгоритмы нахождения ближайших соседей — brute, kd_tree, ball_tree и my_own.

my_own — самописная реализация, которая вычисляет полную матрицу расстояний $D^{T \times N}$, где T – размер тестовой выборки, N – размер обучающей выборки.

brute, kd_tree, ball_tree — реализации поиска соседей из библиотеки sklearn.

Сравнение скорости работы

Так как описанные выше методы нахождения соседей являются детерминированными (100% точныи), то главным критерием выбора одного из них для дальнеших исследований будет служить скорость работы. В случае MNIST важно знать как хорошо ведут себя алгоритмы в пространстве большой размерности R^{784}.

Для экспериментов были выбраны подпространства размерности 10, 50 , 100. В качестве меры расстояния возьмем евклидову метрику.

[1] В некоторых частях исследования будет использоваться уменьшенная выборка т.к. вычислительная машина Тыцкого В.И. тяжело справляется с такой нагрузкой. Во всех случаях, где не оговаривается иное, будет использоваться полная выборка

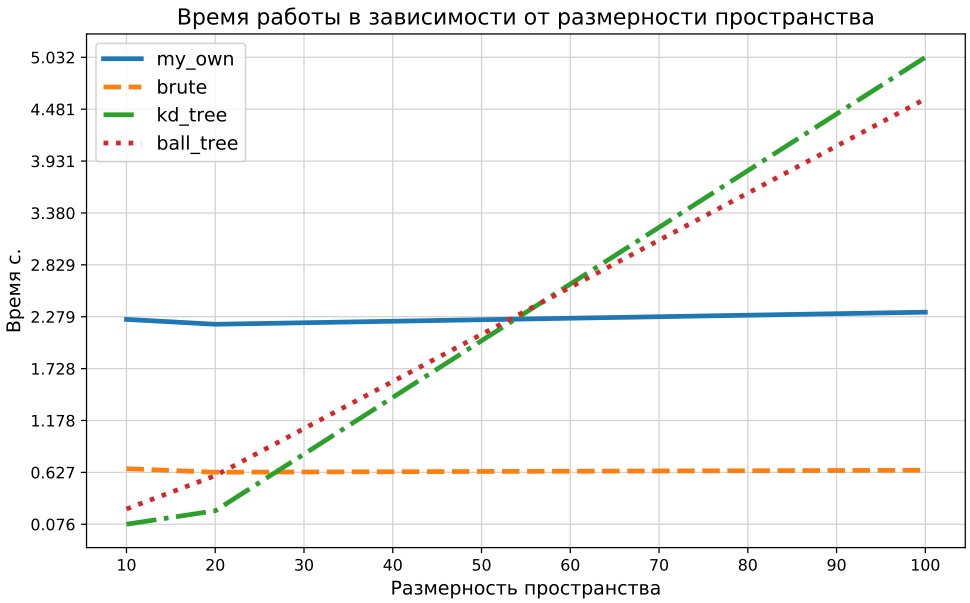

Рис. 1:

На графике (Рис.1) представлены результаты вычисления ближайших соседей для тестовой выборки размера 3000 и 10000 для обучающей выборки.

Из графика время работы kd_tree и ball_tree с ростом размерности пространства увеличивается линейно. Это связано с принципами работы алгоритма и так явлением называемым “Проклятие размерности”. Brute и my_own имеют практически константое время работы алгортима, потому что основаны на простом построении матрицы расстояний, вычисление нормы разности $||x_i - x_j||$ по сравнению с построением матрицы имеет незначительное количество операций.

Далее везде будем использовать либо brute, либо my_own [2]

Задание №2/3

Требуется по кросс-валидации с тремя фолдами оценить точность и время работы в зависимости от следующих факторов:

1. k от 1 до 10 (только точность)
2. евклидова или косинусная метрика
3. используются ли веса или нет (только точность) [3]

[2] Кроме евклидовой метрики нам понадобится косинусное расстояние— $cos(x, y) = 1 - \frac{(x,y)}{||x||_2||y||_2}$. Sklearn метод, реализующих поиск соседей не поддерживает косинусное расстояние, поэтому часто будем использовать метод my_own, у которого есть поддержка этого расстояния.

[3] $w_k = \frac{1}{\rho(X,X_k)+10^{-5}}$, где w_k вес K-ого ближайшего соседа X_k для X

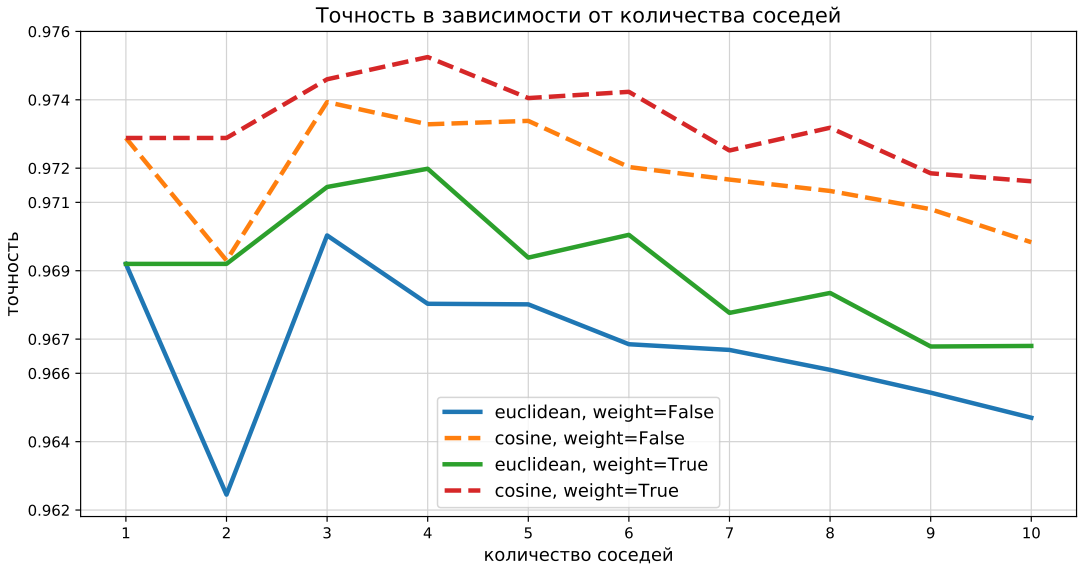

Рис. 2:

test size	**euclidean**		**cosine**	
	mean	std	mean	std
1000	0.917	0.015	0.900	0.001
2000	1.710	0.027	1.796	0.113
3000	2.547	0.021	2.608	0.0257

На графике (Рис.2) представлены результаты вычисления качества в зависимости от вышеперечисленных параметров

- Косинусное расстояние(с весами и без) лучше евклидова для любых k.
- Использование весов улучшает качество для любых k.
- Качество постепенно падает у всех алгоритмов, если $k > 4$
- Лучшим оказался алгоритм с $k = 4$ использующий косинусное расстояние и веса.

Интересно, что без весов лучшая точность достигается при $k = 3$, а с весами при $k = 4$. Это говорит о том,что веса в некотором смысле регуляризуют модель — она использует информацию от большего числа соседей, но не “доверяет” слишком далеким объектам.

Задание №4