Московский Государственный Университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Фоминская Галина
кафедра ММП, группа 317
2017

Метод k ближайших соседей

1 Введение

В данной работе был проведен анализ метода ближайших соседей. Рассмотренная реализация метода, включающая кросс-валидацию для оценки качества работы алгоритма, была написана на языке $Python$. Эксперименты были проведены на датасете изображений цифр $MNIST$.

2 Описание метода

Рассматривается задача многоклассовой классификации.

Имеется обучающая выборка $\{(x_i, y_i)\}_{i=1}^{l}$, состоящая из l векторов $x_i \in \mathbb{R}^n$, называемых векторами признаков, и их классов $y_i \in Y$. Пусть на множестве векторов признаков задана метрика $\rho(\cdot, \cdot)$.

Требуется построить алгоритм $a(\cdot)$, который для произвольного нового вектора признаков автоматически определит его класс. В модели k ближайших соседей определение класса для нового объекта u выполняется по формуле

$$a(u) = \arg\max_{y \in Y} \sum_{i=1}^{k} \mathbb{I}[y_i = y],$$

где суммирование ведется по k объектам обучающей выборки, ближайшим к u по метрике ρ. Заметим, что такой алгоритм не учитывает степень близости u к его соседям. Это можно исправить, учитывая голос каждого соседа с весом, зависящем от расстояния до u:

$$w_i = \frac{1}{\rho(u, x_i) + 10^{-5}}$$

Тогда класс объекта u определяется по формуле

$$a(u) = \arg\max_{y \in Y} \sum_{i=1}^{k} w_i * \mathbb{I}[y_i = y].$$

3 Эксперименты

У приведенной реализации метода ближайших соседей есть 4 параметра:

- k — число ближайших соседей
- стратегия поиска ближайших соседей
- метрика, по которой определяется расстояние между объектами
- единичные веса vs. веса, обратно пропорциональные расстояниям между объектами

Эксперименты проводятся для различных значений этих параметров с целью выбрать те значения, при которых точность предсказания будет лучшей.

3.1 Выбор стратегии поиска соседей

В приведенной реализации метода есть 4 различные стратегии поиска ближайших соседей: стратегии brute, kd-tree и ball-tree, соответствующие реализациям NearestNeighbors из модуля sklearn.neighbors, и стратегия my-own, основанная на подсчете расстояний между двумя множествами векторов. Сравним эти методы по скорости работы в зависимости от размерности пространства признаков. В таблице 1 приведены значения времени работы алгоритма для 5 соседей для рассмотренных стратегий и размерностей пространства 10, 20 и 100.

Размерность:	**10**	**20**	**100**
brute	26	20.4	21.2
ball-tree	2.88	19.2	**122**
kd-tree	1.38	7.08	**109**
my-own	43.6	43.1	45.1

Таблица 1: Время поиска ближайших соседей (с)

Из таблицы видно, что время работы алгоритмов brute и my-own почти не меняется с ростом размерности пространства признаков. Алгоритмы kd-tree и ball-tree при маленьком числе признаков работают значительно быстрее переборных алгоритмов, но с ростом размерности время их работы сильно увеличивается. Это объясняется тем, что в этих алгоритмах число действий, необходимых для поиска ближайшего соседа, растет с ростом размерности пространства признаков, в частности для построения kd-дерева требуется 2^d точек признакового пространства, где d — размерность этого пространства. При этом время поиска ближайшего соседа пропорционально $\ln(2^d)$, то есть растет пропорционально d. В нашем исследовании размерность пространства признаков равна 28^2, что намного больше 100, значит оптимальнее всего будет использовать стратегию brute для вычисления расстояний между объектами.

3.2 Выбор параметров модели

Мы выбрали самую оптимальную по времени стратегию поиска расстояний. Теперь подберем остальные параметры модели так, чтобы получить наилучшее качество. У приведенной реализации метода ближайших соседей помимо вариантов выбора стратегии поиска расстояний есть три параметра:

1. k — число соседей
2. метрика, в которой считаются расстояния
3. веса, с которыми учитываются голоса соседей

Рассмотрим значения k от 1 до 10 и эвклидову и косинусную метрики, и с помощью кросс-валидации на 3 фолдах на обучающей выборке вычислим точность (accuracy) алгоритма при каждом из рассмотренных параметров. Из графика 1 видно, что при выборе косинусной метрики accuracy выше чем при выборе эвклидовой при всех значениях k.

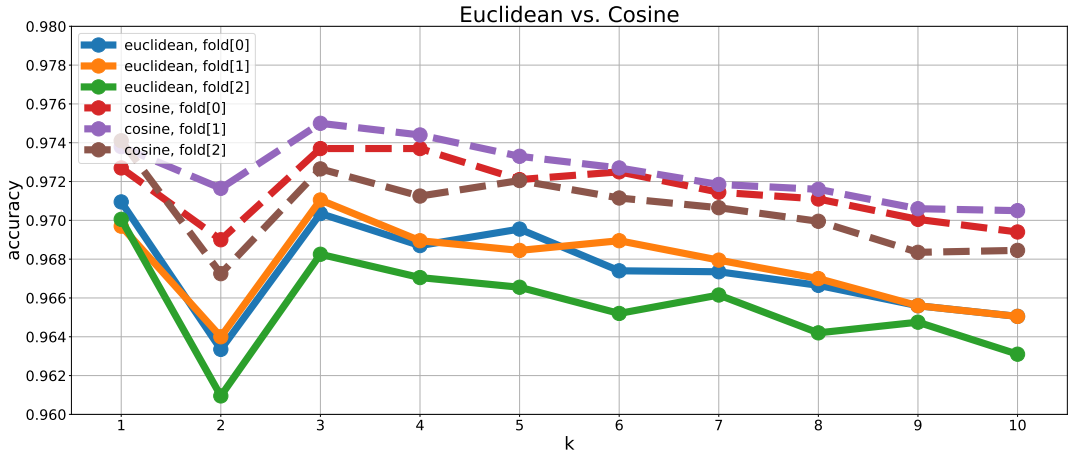

Рис. 1: Точность при использовании эвклидовой и косинусной мер

На графике заметен провал при значении $k = 2$. Это объясняется тем, что в этом случае появляется большая вероятность того, что решение будет принято по соседу, который относится к другому классу.

Повторим этот же эксперимент, но теперь учитывая веса голосов соседей. Здесь они были выбраны равными $\frac{1}{dist+10^{-5}}$. Результаты показаны на графике 2. Из него видно, что и при учете весов косинусная метрика дает выигрыш в качестве по сравнению с эвклидовой.

Осталось сравнить качество для модели с косинусной мерой в зависимости от того, учитываем мы веса или нет. Из графика 3 видно, что метод с учетом весов дает лучшее качество по сравнению с методом без их учета, и лучший результат получается при $k = 4$. Таким образом, лучшее качество получается при $k = 4$, использовании косинусной метрики и с учетом весов голосов соседей.

Это разумный результат, так как добавление весов позволяет понизить влияние большого числа объектов, которые входят в k ближайших соседей нашего объекта, но находятся далеко от него и с большой вероятностью относятся к другому классу.

3.3 Анализ алгоритма и ошибочных объектов

Теперь построим предсказание для отложенной тестовой выборки с помощью модели с параметрами, подобранными в предыдущем эксперименте. Полученная точность приближенно равна 0.9752. Рисунок 4 показывает матрицу ошибок для нашего эксперимента, с обнуленными диагональными элементами, которые иначе показывали бы сколько объектов были классифицированы правильно. Видно, что очень много ошибок, например, при классификации "четверок".

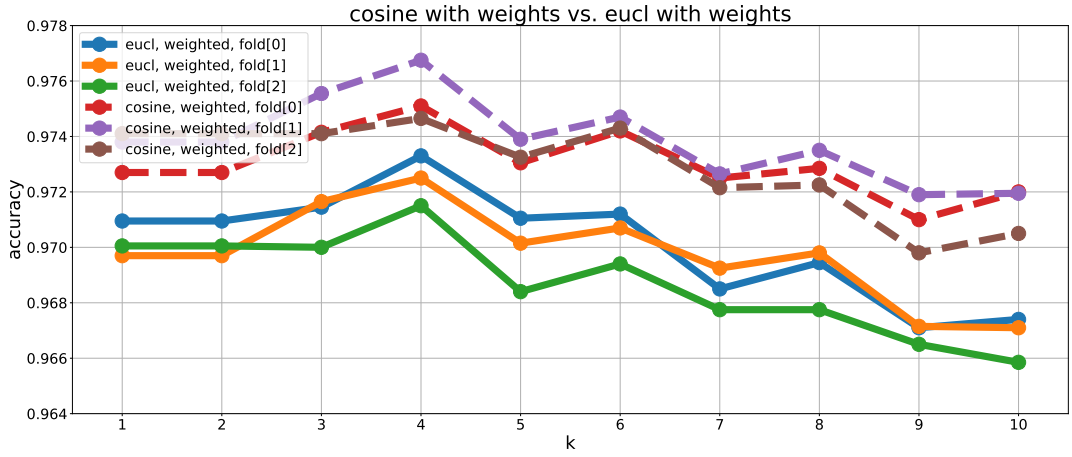

Рис. 2: Точность при использовании эвклидовой и косинусной мер с весами

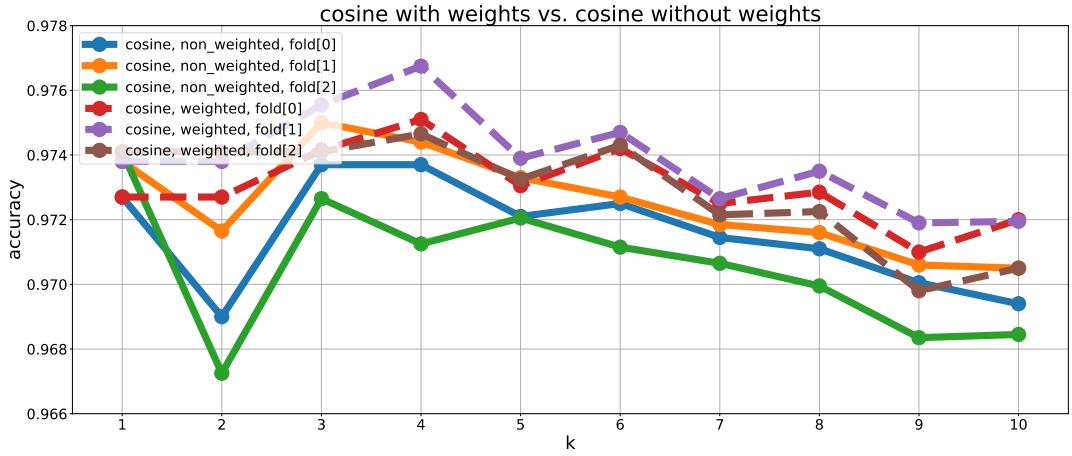

Рис. 3: Точность при использовании косинусной меры с весами и без

25 из них были ошибочно классифицированы как "девятки". На рисунке 5 показаны некоторые примеры таких объектов.

3.4 Расширение обучающей выборки

Рассмотрим следующие преобразования объектов обучающей выборки:

1. поворот на углы от -15 до 15 градусов
2. сдвиг на число пикселей от -3 до 3 по горизонтали или по вертикали
3. применение гауссовского фильтра с дисперсией от 0 до 1.5

Заметим, что такие преобразования не меняют класс, к которому относится объект. Теперь наша цель — выбрать некоторые параметры преобразований и расширить обучающую выборку, добавив в неё преобразованные объекты. Для того, чтобы подобрать оптимальные значения

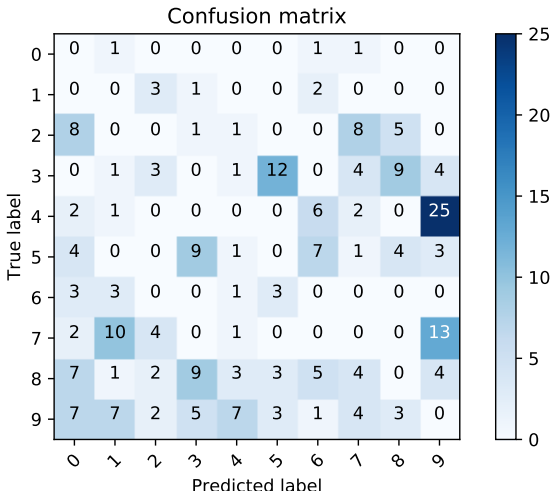

Рис. 4: Матрица ошибок для исходной выборки

параметров преобразований, проведем кросс-валидацию на обучающей выборке. В таблице 2 приведены значения accuracy для различных значений углов поворота, жирным шрифтом выделены максимальные по каждому из фолдов значения. Для сдвигов и поворотов эксперимент был проведен на трети обучающей выборки. Для них результаты приведены в таблицах 3 и 4. Из таблиц видно, что лучшее значение accuracy получено при углах 10 градусов, сдвигах на 1 пиксель и фильтрации с дисперсией 1.

угол (градусы)	fold-1	fold-2	fold-3
0	0.9747	0.9734	0.9738
5	0.9808	0.9807	0.9816
10	**0.9809**	**0.9809**	**0.9817**
15	0.9797	0.9789	0.9786

Таблица 2: accuracy для разных значений углов поворота

сдвиг (пиксели)	fold-1	fold-2	fold-3
0	0.9618	0.9574	0.9634
1	**0.9709**	**0.9652**	**0.9689**
2	0.9670	0.9624	0.9646
3	0.9657	0.9629	0.9636

Таблица 3: accuracy для разных величин сдвига(на трети выборки)

Теперь расширим обучающую выборку, добавив в нее преобразованные объекты, и посчитаем accuracy для предсказания классов для тестовой выборки. Она получилась равна 0.985. Видно, что проведенное нами расширение обучающей выборки привело к приросту качества классификации. Построим по проведенному эксперименту матрицу ошибок (Рис. 6). Видно, что по сравнению с предыдущим экспериментом расширение выборки исправляет часть ошибок на изображениях "четверок но при этом, например, "двойки"начинают классифицироваться хуже.

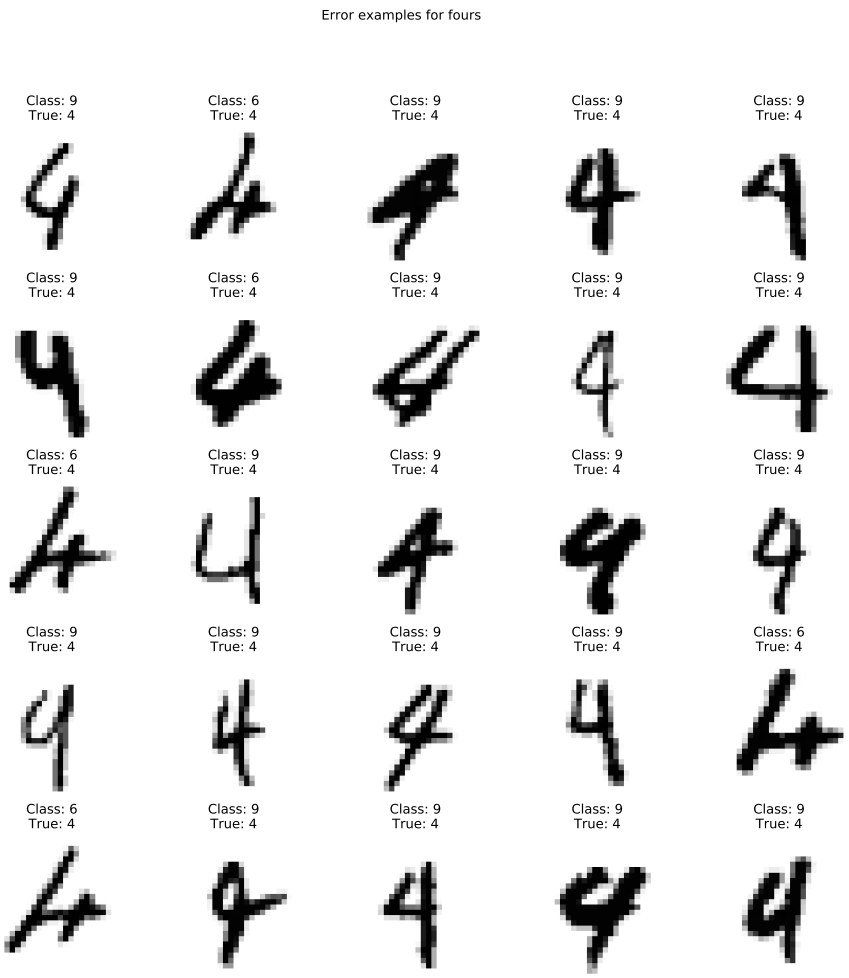

Рис. 5: Примеры ошибочно классифицированных "четверок"

дисперсия	fold-1	fold-2	fold-3
0	0.96370181	0.95935203	0.95950202
0.5	0.97120144	0.96895155	0.96925154
1	**0.97225139**	0.97135143	**0.97105145**
1.5	0.97105145	**0.97180141**	0.97075146

Таблица 4: accuracy для разных дисперсий гауссовского фильтра(для трети выборки)

3.5 Анализ качества предсказаний на расширенной тестовой выборке

Аналогичным образом расширим тестовую выборку. Теперь для каждого объекта обучающей выборки есть множество объектов, полученных из него путем рассмотренных преобразований. Все объекты из этого множества относятся к тому же классу, что и исходный объект. Будем вычислять расстояния до объектов обучающей выборки не для исходных объектов тестовой, а для полученных множеств. Качество предсказаний (accuracy), полученных таким образом,

получилось равным 0.9699. Видно, что оно почти не изменилось по сравнению с моделью без применения преобразований. Это соответствует интуитивному представлению о том, что объекты, близкие с точностью до рассмотренных преобразований, должны относиться к одному классу. При этом качество упало по сравнению с моделью с применением преобразований к обучающей выборке. Это может быть связано с большей обобщающей способностью алгоритма, обученного на большем числе объектов.На рисунках 6 и 7 приведены матрицы ошибок для проведенных предсказаний на расширенных обучающей и тестовой выборок. Видно, что случаи, в которых происходят ошибки, почти совпадают.

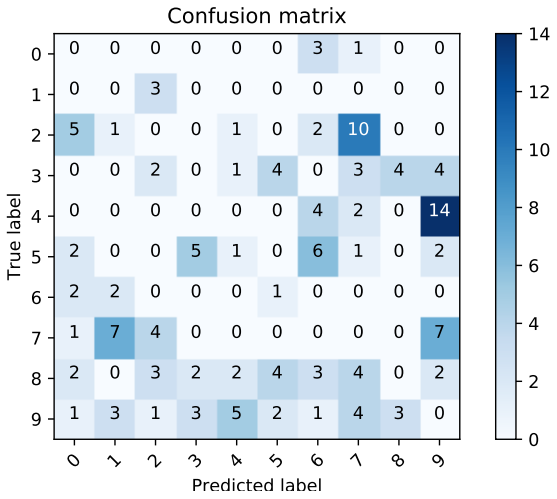

Рис. 6: Матрица ошибок для расширенной обучающей выборки

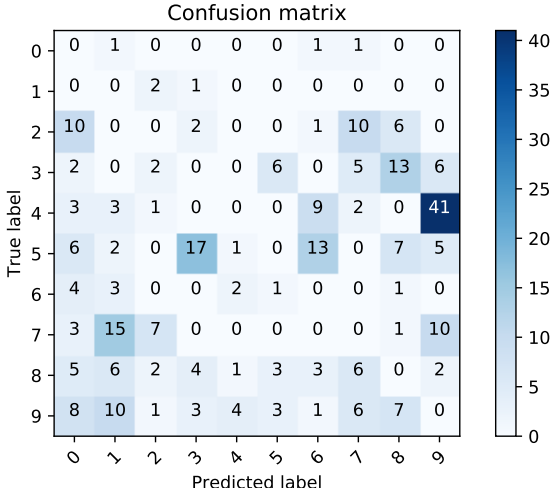

Рис. 7: Матрица ошибок для расширенной тестовой выборки

4 Выводы

В работе был рассмотрен метод ближайших соседей для решения задачи многоклассовой классификации. На коллекции изображений цифр MNIST были проведены эксперименты по нахождению с помощью кросс-валидации оптимальных параметров алгоритма и были посчитаны значения accuracy предсказаний на отложенной тестовой выборке.