Московский Государственный Университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчет по метрическим алгоритмам классификации: k ближайших соседей

Шаталов Н.А
кафедра ММП, группа 317
2017

Содержание

1 Введение.

Перед Вами лежит работа, которая полностью описывает этапы проведения экспериментов над методом k ближайших соседей для решения задачи многоклассовой классификации. Описанные ниже эксперименты проводились на датасете MNIST.

Ниже также использовалась моя реализация алгоритма k ближайших соседей, которая не имеет особых структур данных «под капотом» и производит простой поиск ближайших соседей по заданной метрике.

2 Эксперименты

2.1 Стратегии алгоритма

В первом эксперименте мы исследуем время работы алгоритма поиска ближайших соседей в зависимости от стратегии алгоритма (моя собственная реализация my own, brute, kd tree и ball tree) и размерности пространства признаков. Для второго мы смотрим подмножество признаков размера 10, 20 и 100, которое выбирается случайно и один раз для всех объектов.

Dimension	my own	brute	kd tree	ball tree
10	32.46	31.87	1.27	5.91
20	36.27	27.86	4.38	29.09
100	83.74	31.09	103.83	103.28

Таблица 1: Время поиска ближайших соседей (с)

Как видно из таблицы 1, kd tree и ball tree становятся неэффективными с ростом размерности пространства признаков, что подтверждает проклятье размерности. Самой эффективной на большой размерности является стратегия brute, поэтому в дальнейшем будем использовать ее.

2.2 Параметры алгоритма (k, метрика, веса)

Осталось подобрать наилучшие параметры для модели: число k ближайших соседей, метрику (евклидово или косинусное расстояние), и использовать ли веса объектов или нет.

В качестве метрики качества модели используем accuracy (acc):

$$acc = \frac{\sum_{i=1}^{s} [y_i^{pred} = y_i^{test}]}{s},$$

где s - размер тестовой выборки, y_i^{pred} - отклик модели на i-ый объект тестовой выборки, y_i^{test} - реальный класс i-ого объекта тестовой выборки.

В качестве весов, если мы используем невзвешенную модель, то берем веса $w = 1$, в невзвешенной модели возьмем:

$$w = \frac{1}{\varepsilon + \rho(x', x_{(i)})},$$

где $\rho(x', x_{(i)})$ - расстояние от тестового объекта x' до его i-го ближайшего соседа $x_{(i)}$, $\varepsilon = 10^{-5}$.

Nonweighted		Weighted	
Euclidean	Cosine	Euclidean	Cosine
464.46	721.71	463.53	761.76

Таблица 2: Время кросс-валидации (с)

	Nonweighted		Weighted	
k	Euclidean	Cosine	Euclidean	Cosine
1	**0.9695**	0.9729	0.9687	0.9724
2	0.9627	0.9685	0.9687	0.9724
3	0.9693	**0.9731**	0.9706	0.9737
4	0.9674	0.9729	**0.9715**	**0.9749**
5	0.9678	0.9723	0.9696	0.9737
6	0.9665	0.9718	0.9703	0.9740
7	0.9665	0.9712	0.9685	0.9728
8	0.9655	0.9710	0.9686	0.9728
9	0.9656	0.9702	0.9668	0.9716
10	0.9645	0.9700	0.9670	0.9716

Таблица 3: Точность (accuracy)

Посмотрим на время кросс-валидации по $k \in [1, 10]$ на различных метриках в таблице 2. Как можно заметить, косинусное расстояние считается дольше, чем евклидово, при этом веса не сильно влияют на время работы. Но, нам больше важно качество модели а не время ее работы.

Как видно из таблицы 3, наилучшее качество модель принимает при $k = 4$, с использованием косинусной метрики, учитывая расстояния до соседей с помощью весов.

Далее будем использовать эти параметры.

2.3 Промежуточный итог

Применим лучший алгоритм (напомним его параметры: стратегия brute, $k = 4$, используется косинусная метрика, веса соседей обратно пропорциональны расстоянию до тестового объекта) к тестовой выборке и посмотрим на его точность.

$$accuracy = 0.9752$$

Wikipedia говорит, что на ансамбле из пяти свёрточных нейронных сетей смогли добиться точности 0.9979. Нам есть к чему стремиться.

Рассмотрим матрицу ошибок на рис. 3а (для удобства элементы на диагонали обнулены).

Как видим, хуже всего модель распознает цифру 9, путая ее пости со всеми остальными цифрами. В большинстве случаев модель путает 4 с 9, 7 с 9, 3 с 5, 7 с 1.

Ниже приведена выборка ошибочных объектов на рис. 2.

Из приведенных ошибочных объектов видно, что модель путает цифры, написанные «мелко размашистым» почерком с «закорючками» .

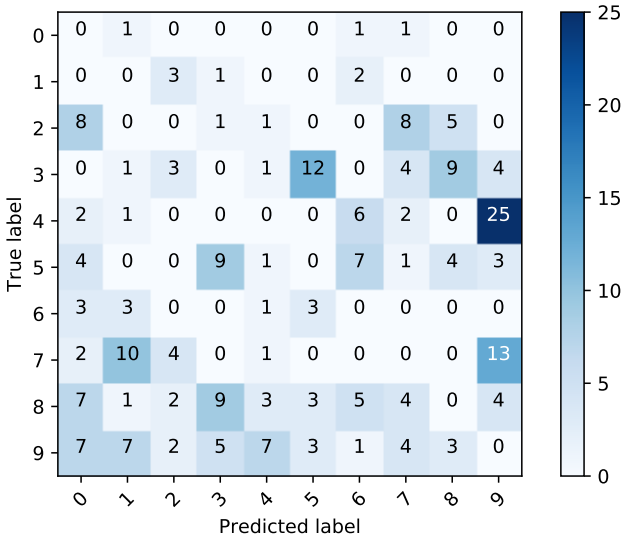


Рис. 1: Матрица ошибок (модель без преобразований)

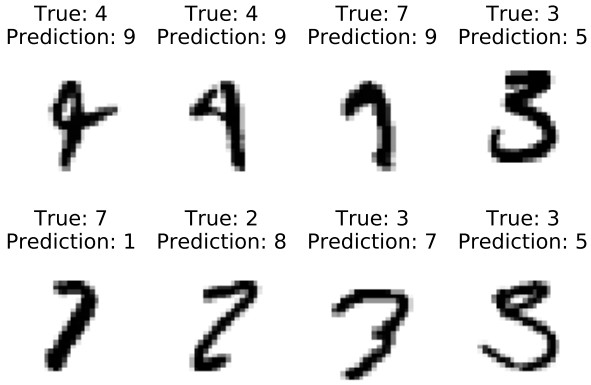


Рис. 2: Ошибочные объекты (модель без преобразований)

2.4 Преобразование обучающей выборки

Попробуем повысить качество модели, добавляя к обучающей выборке объекты, прошедшие преобразования. Рассмотрим три различных преобразования: поворот изображения, смещение изображения и фильтр гаусса с заданной дисперсией.

Подберем параметры преобразований, которые наилучшим образом повышают точность модели на кросс-валидации.

Из-за ограничения вычислительной мощности, далее кросс-валидация проводится на обучающей выборке, обрезанной до 12000 объектов случайным образом.

2.4.1 Повороты

Повороты обучающей выборки производятся в обе стороны: по часовой и против часовой стрелке. По таблице 4 видим, что наилучшим образом точность модели повышается при повороте обучающей выборке на 10°. Преобразование немного улучшило точность предсказания почти для всех цифр (см. рис. 3b).

Поворот	Нет	5°	10°	15°
Точность	0.9552	0.9609	**0.9617**	0.9599

Таблица 4: Точность модели с поворотами обучающей выборки

Напомним, что элементы на диагонали матрицы ошибок обнулены для удобства.

2.4.2 Смещение

Смещение	Нет	1 px	2 px	3 px
Точность	0.9552	**0.9594**	0.9573	0.9557

Таблица 5: Точность модели с смещением обучающей выборки

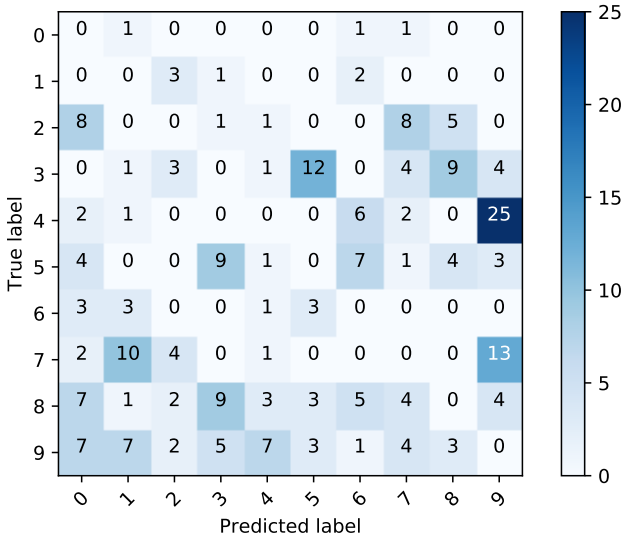
Смещение обучающей выборки производится по всем четырем сторонам. Из таблицы 5 видно, что лучше всего модель работает с смещением обучающей выборки на 1 пиксель. Преобразование совсем немного улучшило качество предсказывания (см. рис. 3c).

2.4.3 Размытие (фильтр Гаусса)

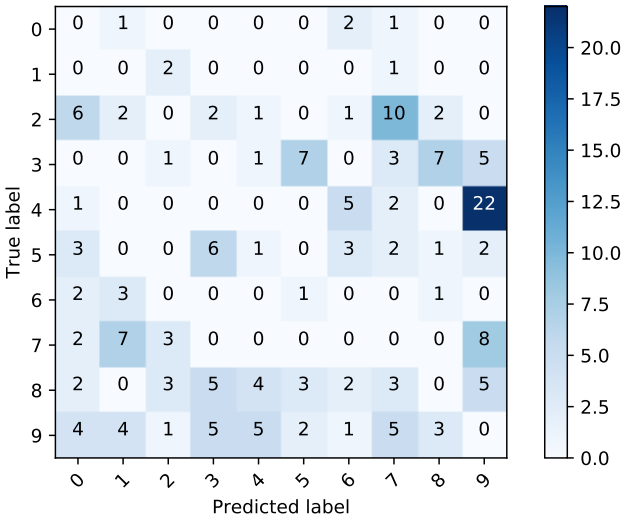
Дисперсия	Нет	0.5	1	1.5
Точность	0.9552	0.9605	**0.9613**	0.9611

Таблица 6: Точность модели с размытием обучающей выборки

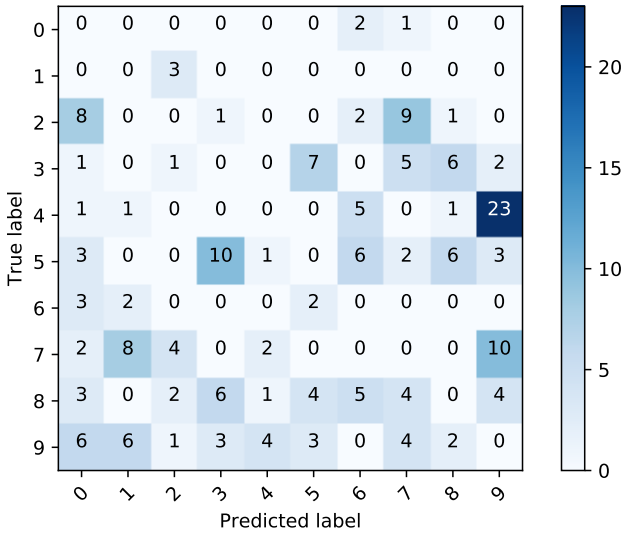
Из таблицы 6 видно, что лучше всего модель работает с смещением размытием выборки фильтром Гаусса с дисперсией $\sigma^2 = 1$. Преобразование заметно улучшило качества предсказываний отдельных ошибок, модель больше не путает так сильно цифру 4 с 9, 3 с 5 (см. рис. 3d).



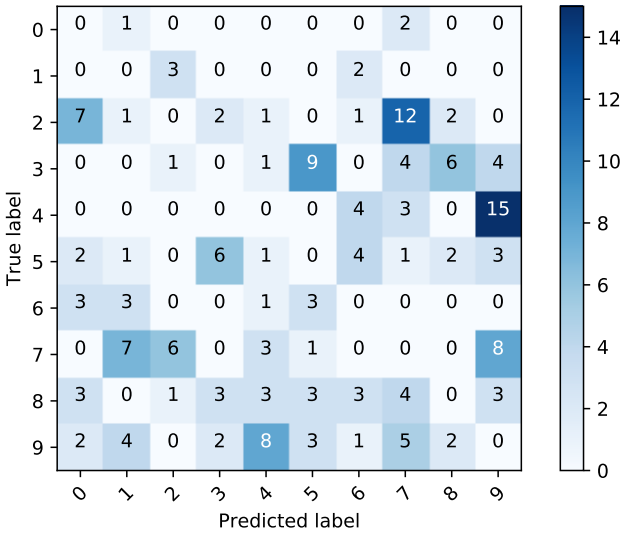
(a) без преобразований



(b) поворот на 10°



(c) смещение на 1 px



(d) фильтр Гаусса, $\sigma^2 = 1$

Рис. 3: Матрицы ошибок с добавлением различных преобразований обучающей выборки

2.5 Финальная модель

Добавим к обучающей выборке (ее длина — 60000) объекты, преобразованные поворотами на 10°, смещениями на 1 пиксель и размытием фильтром Гаусса с дисперсией 1.

Получим точность на тестовой выборке.

$$accuracy = 0.9851$$

После преобразований мы смогли улучшить точность модели на 1%, что является неплохим результатом.

2.6 Преобразование тестовой выборки

Вместо добавления преобразованных объектов к обучающей выборке попробуем размножить тестовую выборку, поставив вместо каждого объекта тестовой выборки множество его преобразований $\Phi(x) = \{\phi_j(x)|j \in \{0, \ldots, m\}\}$, где $\phi_0(x) = x$, $\phi_j(x), j \in \{1, \ldots, m\}$ - преобразования объекта тестовой выборки. Расстояние от объекта x' до множества $\Phi(x)$ считается как

$$\rho(x', \Phi(x)) = \min_j \{\rho(x', \phi_j(x)|j \in \{1, \ldots, m\}\}.$$

Аналогично найдем лучшие параметры для преобразования тестовой выборки. Напомним, что мы рассматриваем повороты, смещения и размытие фильтром Гаусса выборки.

Из-за ограничения вычислительной мощности, аналогично проводим кросс-валидацию на обрезанной обучающей выборке из 12000 объектов.

Поворот	Нет	5°	10°	15°
Точность	0.9594	0.9656	**0.9681**	0.9632

Таблица 7: Точность модели с поворотами тестовой выборки

Смещение	Нет	1 px	2 px	3 px
Точность	0.9594	**0.9681**	0.9646	0.9579

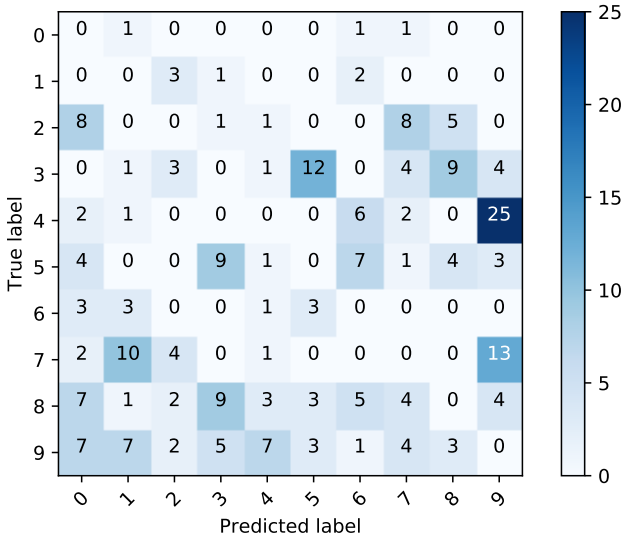
Таблица 8: Точность модели с смещением тестовой выборки

Дисперсия	Нет	0.5	1	1.5
Точность	**0.9594**	0.9563	0.9460	0.9303

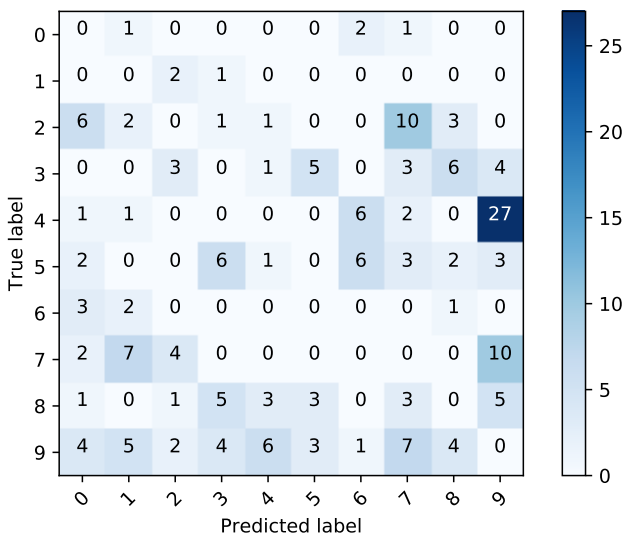
Таблица 9: Точность модели с размытием тестовой выборки

Из таблиц 7, 8, 9 видно, что лучше всего модель улучшает качество при повороте тестовой выборки на 10° и смещении на 1 пиксель.

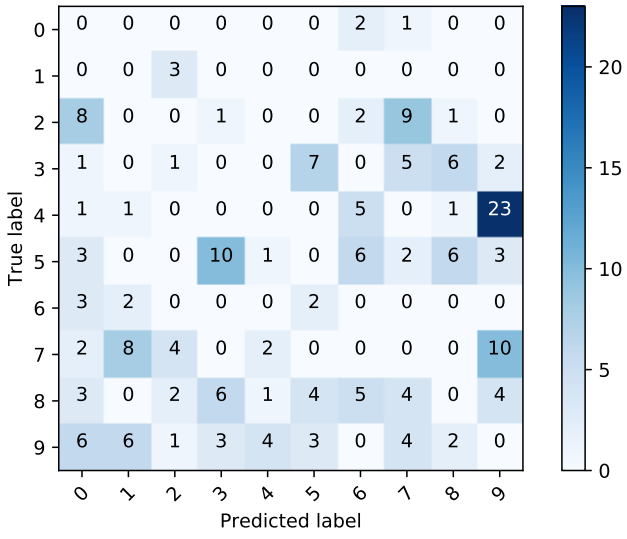
По рис. 4 можно видеть, как преобразования улучшают качество предсказаний некоторых цифр, но ухудшают предсказание других.



(a) без преобразований



(b) поворот на 10°



(c) смещение на 1 px

Рис. 4: Матрицы ошибок с добавлением различных преобразований тестовой выборки

Посмотрим точность на преобразованной тестовой выборке (повороты и смещения) на модели, обученной на непреобразованной обучающей выборке (60000 объектов).

$$accuracy = 0.982$$

Как видно, преобразование тестовой выборки улучшило качество модели в сравнении с моделью без преобразований, но дает почти такую же точность, как модель с преобразованием обучающей выборки. Однако, преобразование тестовой выборки более эффективно по памяти.