Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра математических методов прогнозирования

Андрейцев Антон Игоревич

Комбинаторный подход к проблеме офлайн-распознавания рукописных цифр

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Научный руководитель:

к.ф.-м.н., доцент

С. И. Гуров

Москва, 2018

Содержание

Аннотация

В работе рассмотрен комбинаторный подход в распознавании геометрических образов на примере рукописных цифр, включающий в себя рассмотрение цепного кодирования и функционального описания геометрических объектов. В диссертации предлагается алгоритм классификации, на основе этих подходов, а так же, сравнение результатов, которые предлагает комбинаторный подход, с существующими подходами решения задачи распознавания геометрических образов.

1 Введение

Распознавание рукописных объектов и в частности цифр является, пожалуй, одной из первых задач компьютерного зрения. Для неё уже разработаны много различных эффективных методов решения, различие в качетве классификации с помощью которых составляет лишь 3-4 знаки после запятой. Данная работа не ставит своей целью повышение точности классификации рукописных цифр, учитывая что современные методы дают почти 100% точность. Краткий обзор методов классификации конкретно рукописных цифр можно найти в Википедии[1]. У всех этих методов очевидно есть свои достоинства и недостатки, в частности, недостатком текущего «state of the art» подхода – нейросетевого, является требование наличия большого количества данных и неинтерпретируемость работы алгоритма. Целью данной работы является разработка алгоритма классификации геометрических структур (в частности рукописных цифр) на основе цепного кодирования Фримена, и отрабатываться этот алгоритм будет на примере задачи распознавания рукописных цифр, как одной из классических задач комьютерного зрения. Такой подход будет обладать интерпретируемостью и не потребует большого количество данных в обучающей выборке для успешной классификации.

В начале работы будет проведён анализ литературы, связанной с теорией цепного кодирования. Затем будет описана адаптация цепного кодирования к задаче классификации с помощью расстояния Левенштейна, о котором так же речь пойдёт в середине работы. Так же будут разобраны два примера реальных задач, к которым был применён описанный метод. В заключении работы будут представлены результаты вычислительных экспериментов с их анализом и сравнением с результатами других методов, применяемых в данных задачах.

1.1 Определения и обозначения

Имеется выборка из N рукописных цифр, представленных оцифрованными матрицами $M_n, \quad n = 1, 2, \dots N$ размера p x p, где на (i, j) – ой позиции стоит число в диапазоне от 0 до 255, символизирующее яркость (оттенок серого: 0 – чёрный цвет,

[1] https://ru.wikipedia.org/wiki/MNIST_(база_данных)

255 – белый). Каждой такой матрице сопоставлена метка $k_n \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, в соответствии с тем, какая цифра закодирована в матрице. Выборка разделена на N_{train} тренировочных объектов – для которых метка известна и N_{test} тестовых, для которых эту метку нужно предсказать.

1.2 Обзор литературы

В данной работе используется метод цепного кодирования Фримена (H. Freeman), описанный им в своей работе 1961г. [1]. Фримен предложил способ кодировки произвольной геометрической фигуры, в частности и пространственной кривой, с помощью последовательности символов, с целью упрощения дальнейшего анализа поведения этого объекта с помощью компьютера.

В оригинальной статье кодировка выглядит следующим образом: на заданном промежутке рассматривается некоторое количество точек, принадлежащих кривой. Кривая кодируется последовательностью углов, которые образует хорда, соединяющая последовательно соседние пары точек, с осью абсцисс (Рис. 1).

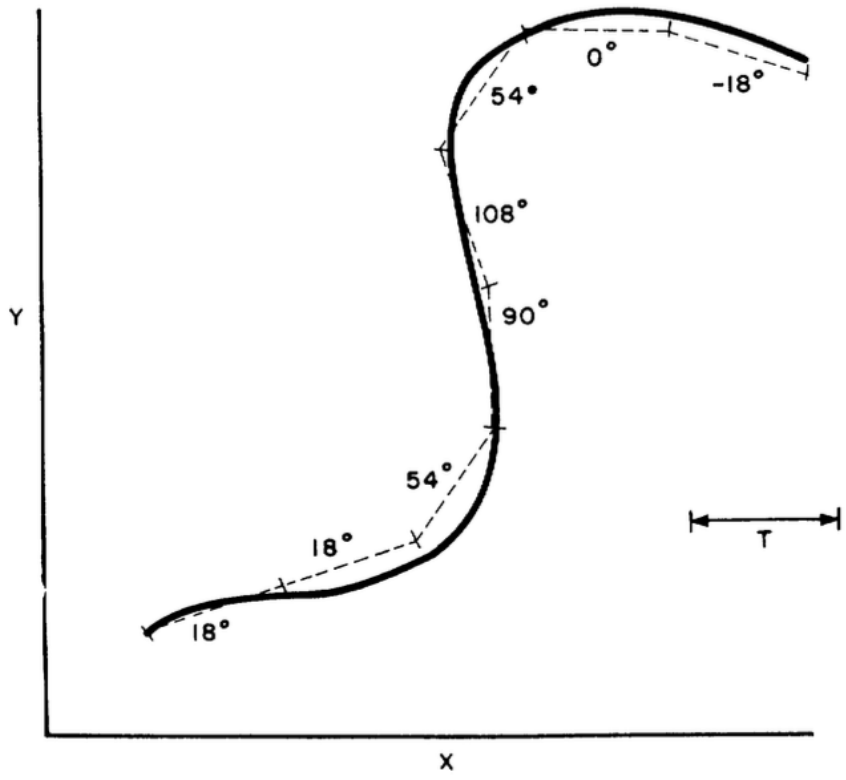


Рис. 1: оригинальная кодировка

Для ограничения мощности множества символов для описания можно разделить окружность на n секторов и вместо текущего значения угла в градусах записывать просто номер сектора, в который попадает данный угол[1]. Тогда для кривой, изображённой на рисунке 1, кодировка примет следующий вид: $(18°, 18°, 54°, 90°, 108°, 54°, 0°, -18°) \to (0, 0, 2, 4, 5, 2, 0, 19)$. В своей работе Фримен в основном описывает способ реализации цепного кодирования и немного говорит о практическом приложении такого подхода: вычисление площади замкнутой кривой. Вопросы классификации, на основе такого кодирования в данной статье не рассматриваются.

В работах [3, 4, 7] рассматриваются более развитые методы создания цепной кодировки и различные манипуляции с полученными последовательностями, в том числе рассматриваются проблемы самопересечения кривых и наличия возможных ответвлений, создание по цепному описанию увеличенной фигуры (Рис. 3), *ротация* и отражение.

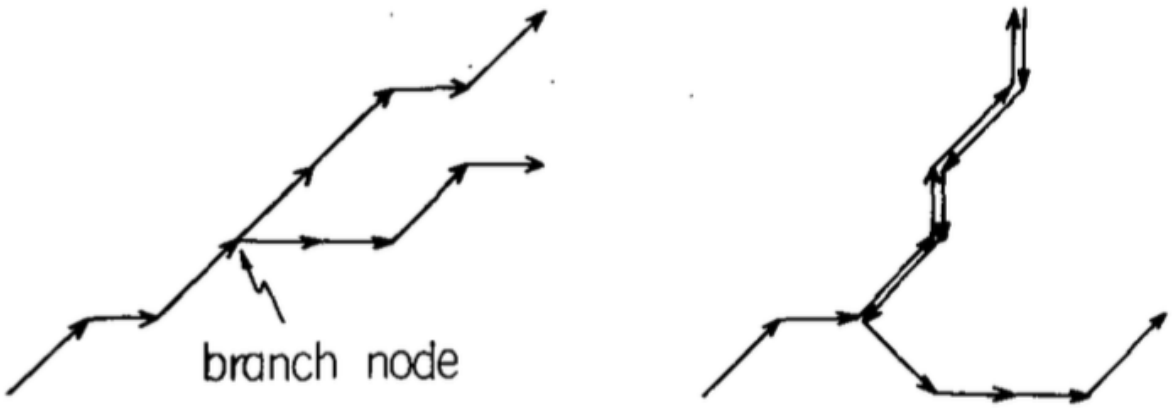


Рис. 2: наличие ответвлений у кривой

[1] В оригинальной статье происходит деление на 20 секторов по 18°

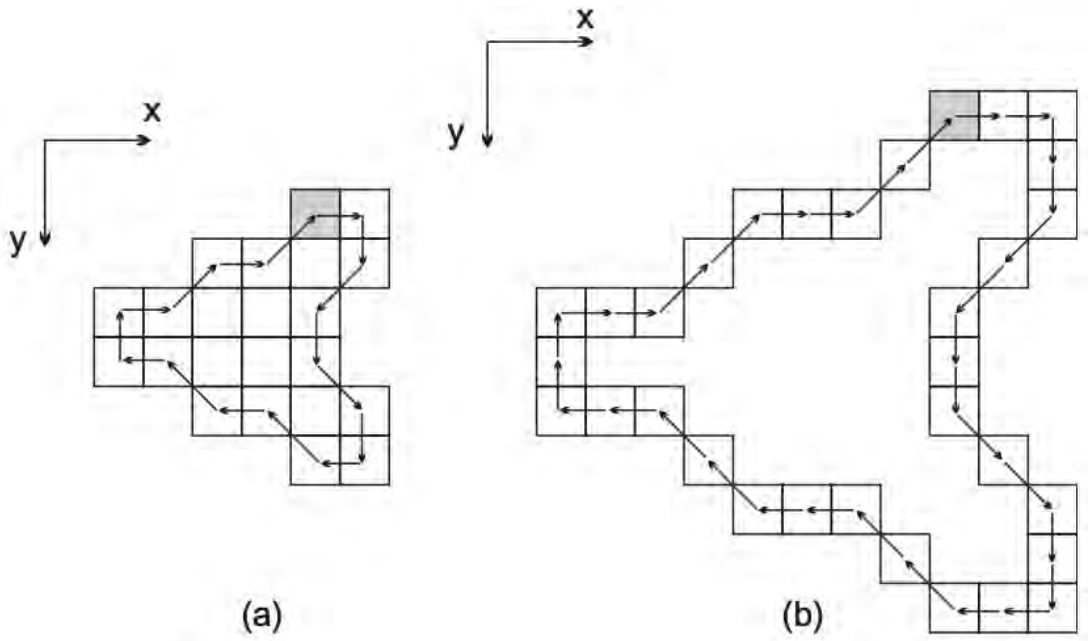


Рис. 3: повторение каждого символа увеличивает фигуру в 2 раза

Несмотря на большое число статей, посвящённых цепному кодированию и его анализу, работ, касающихся непосредственному приминению его для классификации объектов не много.

В дополнении к цепному кодированию в работе также рассматривался подход, основанный на описании цифр неявными полиномами. Такой подход так же распространён в распознавании геометрических структур. Например, в [9, 12] предлагается находить полином, описывающий множество точек на основании трёх линий уровня (Рис. 4)

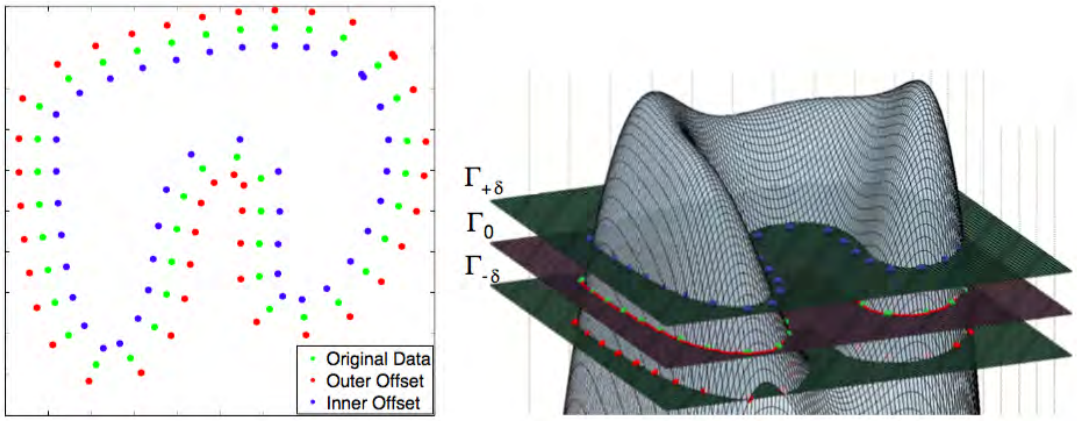


Рис. 4: 3L визуализация

Смысл данного подхода в том, что из множества точек выбирается три уровня точек – верхний, средний и нижний. Верхний и нижний уровни по сути являются объектами разных классов. Средний же уровень служит разделителем этих классов. Тогда задача ставится найти такой полином, который старается отделить эти два класса. В итоге задача сводится к поиску вектора, дающего наименьший квадрат отклонений (МНК).

Кроме того, есть подход, основанный на вписывании полинома, без дополнительных линий уровня[1], основанный на непосредственной минимизации функционала:

$$(1) \sum_{(x,y)\in\Gamma} f^2(x,y), \quad \Gamma = \{(x,y)|f(x,y)=0\}$$

Про геометрический смысл данного функционала можно прочитать в [14].

Подход на основе цепного кодирования и на основе неявных полиномов и будут составлять основу дальнейшего исследования.

2 Комбинаторный подход

В данном разделе будет изложена основная теория и методология распознавания рукописных цифр, связанная с комбинаторным подходом. Будут рассмотрены проблемы применения этого подхода к реальным данным и методы их решения.

2.1 Цепное кодирование Фримена

Несмотря на обширное описание методологии цепного кодирования, в частности в работах [1–3], основной недостаток этих описаний заключается в том, что предложенный алгоритм работает для последовательности точек, которые представляют из себя цепь единичной ширины, то есть точки расположены таким образом, что ближайшими к конкретной точке являются всего две точки. Такое редко встречается в реальных данных, более распространённая форма оцифрованных матриц содержит множество точек, в окретсности которых находятся больше двух соседей (Рис. 5)

[1] по сравнению с подходом в [9, 12]

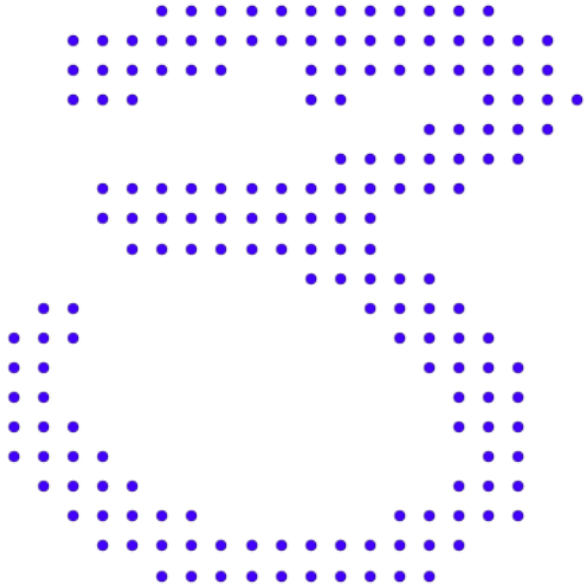


Рис. 5: Пример объекта из базы данных MNIST

Наивный алгоритм Фримена в данной ситуации не сработает. Для его применения требуется тщательная предобработка входных данных, что само по себе является трудоёмкой задачей, которой уделяется отдельное внимание в такой области анализа данных, как компьютерное зрение. Специально отметим, что предобработка данных не является задачей данной работы, поэтому алгоритм, предложенный в дальнейшем, будет своего рода компромиссом между простотой предобработки входного объекта с одной стороны, и пригодностью данного вида предобработки для дальнейшего анализа с другой.

Проводя эксперименты с различными фильтрами для предобработки изображений и перебирая всевозможные эвристики, компромисс был достигнут на алгоритме «Marching squares» [15]. Этот алгоритм используют для выделения линий уровня функций от двух переменных, его часто применяют в метеорологии для визуализации изобар. Алгоритм обрамления с помощью «Marching squares» прост: на вход подаётся бинарная матрица и обрамление строится по правилу, заданному рисунком ниже (Рис. 6). Более подробную иллюстрацию процесса создания обрамления можно увидеть в приложении данной работы.

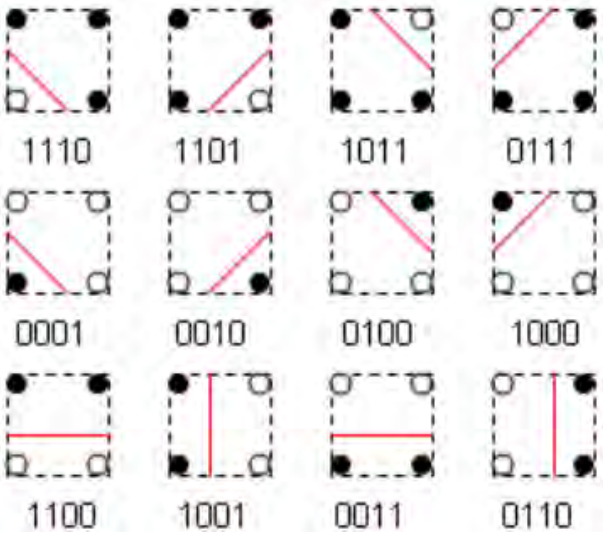


Рис. 6: Marching squares алгоритм

Метод также может быть обобщён на построение нескольких линий уровня тем, что матрица будет подаваться не бинарная, а заполненная вещественными числами, однако для решения задачи, поставленной в данной работе, будет достаточно одного обрамления. После реализации алгоритма, для каждого числа получаем обрамление, как показано ниже (Рис. 7). В дальнейшем для построения кривых Фримена будет использована эта «оболочка» числа.

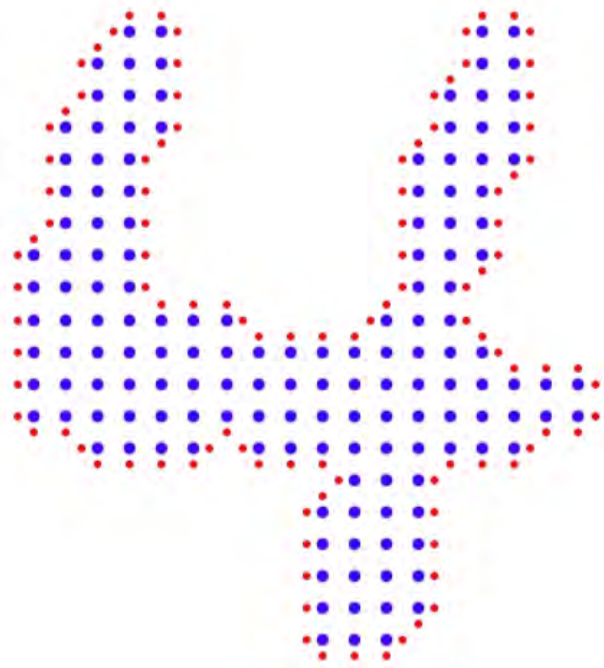


Рис. 7: Результат обрамления числа алгоритмом Marching squares

На текущем этапе нам удалось привести объекты набора данных MNIST в вид, пригодный для применения цепного кодирования, так как теперь в окрестности каждой точки находится по 2 соседа, тем самым получается цепь[1].

Итак, выбирается начальная точка, фиксируется направление обхода[2] и, вычисляя положения текущей и предыдущей точек относительно друг друга, производится кодировка по схеме (Рис. 8). Более подробную визуализацию процесса кодировки можно увидеть в приложении.

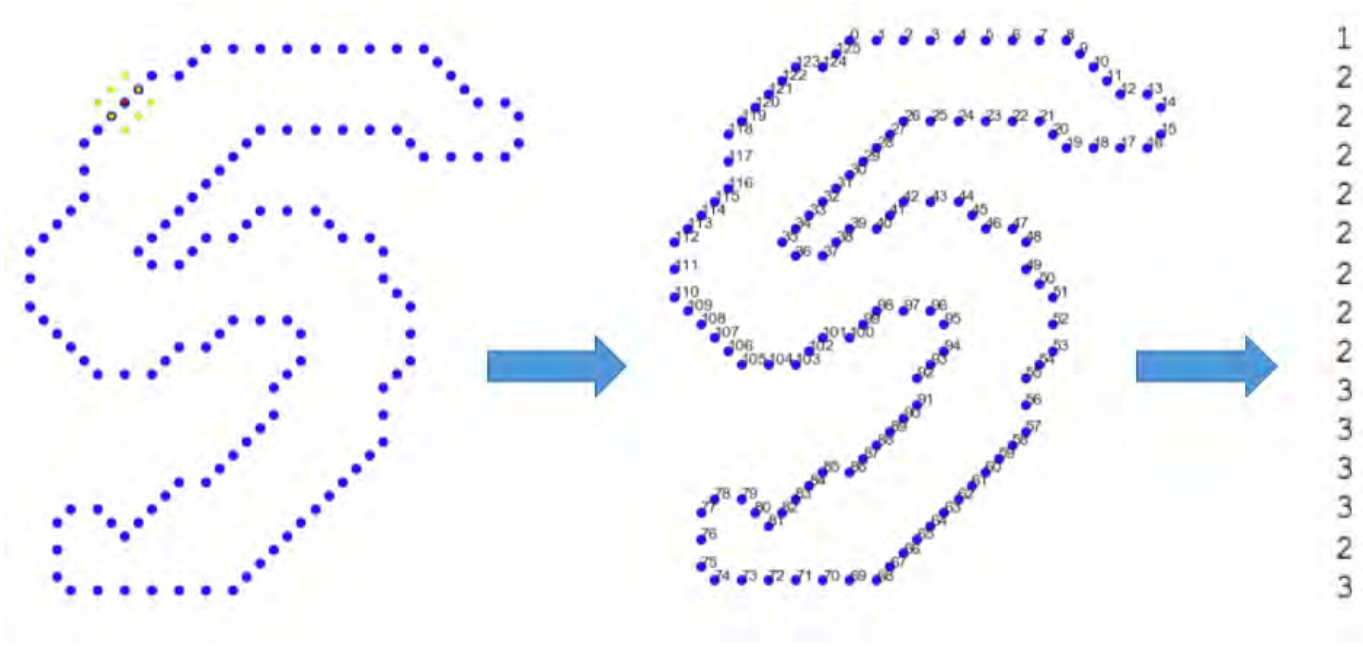


Рис. 8: Схема кодировки

Заметим, что такой способ построения цепной кодировки обходит две потенциальные проблемы: во-первых, для каждого такого набора точек всегда в качестве начальной точки можно выбрать точку, которая ближе всего к верхнему левому углу, тем самым это разрешит проблему выбора начальной точки, а во-вторых, благодаря описанию числа с помощью алгоритма Marching squares пропадает «проблема перекрёстков». Проблема, выражающаяся в том, стоит или нет учитывать в цепной кодировке среднюю черту у цифры «7» например, и в каком направлении продолжать обход цифры «8», если цепь дошла до точки самопересечения. Подобных про-

[1] Случаи, когда в окрестности точки находится более двух соседей, будут рассмотрены ниже.

[2] Обход осуществлялся по часовой стрелке.

блем в данном подходе не возникает, так как описание получается односвязным без точек сочленения.

Несмотря на всю стройность картины, изложенной до текущего момента, у такого подхода всё-же есть проблемное место, которое возникает непосредственно при реализации на практике. Связано оно с той ситуацией, когда у текущей точки больше двух ближайших соседей (Рис. 9). Поскольку решаемая задача предлагает работу с конкретными объектами, предсказать места таких проблемных точек не сложно. Данная проблема решалась рассмотрением различных эвристик, в частности непосредственным указанием правил построения цепного кода в отдельных таких случаях, которых на реальном наборе данных оказалось не много.

Рис. 9: Проблемная точка

2.2 Расстояние Левенштейна

После получения кодировок возникает задача их сравнения для классификации объекта. Полученные последовательности можно сравнивать различными способами, в частности вводить некоторую меру расстояния для строк. Одним из популярных методов вычисления расстояния является расстояние Левенштейна [16]. Оно показывает сколько операций вставки, удаления и замены минимально нужно совер-

шить, чтобы превратить одну последовательность символов во вторую. В классическом определении расстояния Левенштейна полагают стоимость каждой операции (замена, удаление, вставка) равной единице. Данное расстояние можно вычислить с помощью методов динамического программирования. Рекуррентная формула для вычисления расстояния Левенштейна между двумя последовательностями приведена ниже.

$dist(S_1, S_2) = D(M, N)$ где M, N – длины S_1, S_2

$$(2) \quad D(i,j) = \begin{cases} 0, i = 0, j = 0 \\ i, j = 0, i > 0 \\ j, i = 0, j > 0 \\ min\left(D(i, j-1) + 1, D(i-1, j) + 1, D(i-1, j-1) + m(i,j)\right) \end{cases}$$

$$m(i,j) = \begin{cases} 1, S_1[i] \neq S_2[j] \\ 0, S_1[i] = S_2[j] \end{cases}$$

Приведём пример: вычислим расстояние Левенштейна между последовательностями (1,3,2,4) и (0,1,3,1,2,4) (Рис. 10).

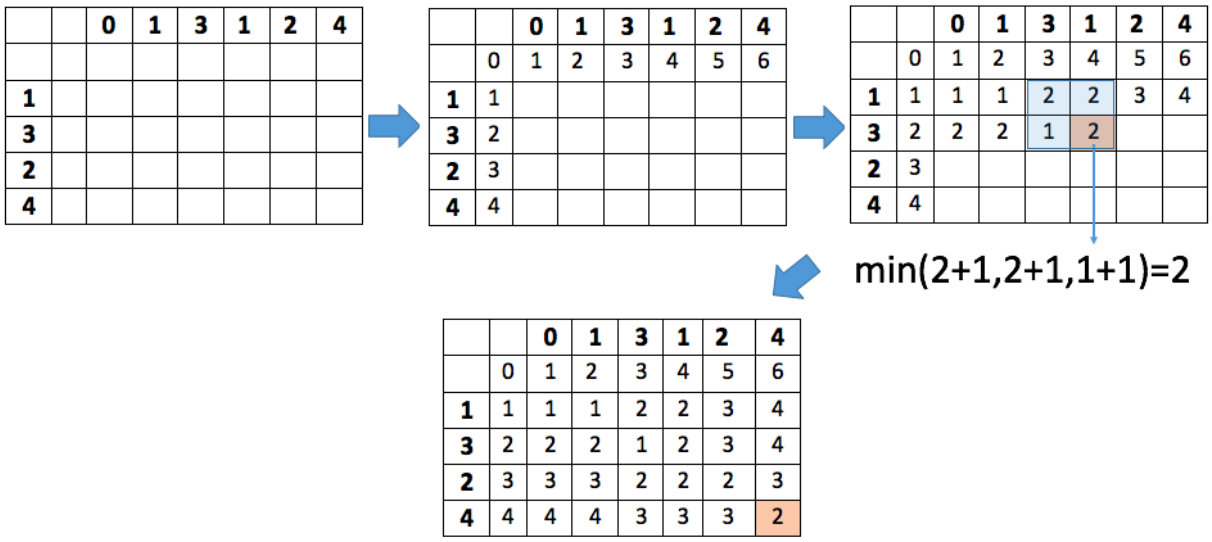


Рис. 10: Вычисление расстояния Левенштейна.

В данном примере расстояние Левенштейна между последовательностями равно двум (нижняя правая ячейка таблицы). Действительно, чтобы превратить строку (1,3,2,4) в строку (0,1,3,1,2,4) нужно сделать две вставки (Рис. 11).

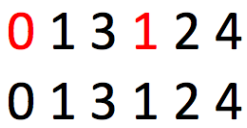


Рис. 11:

В расстоянии Левенштейна стоимость каждой операции определяется в одну единицу. Можно определить стоимость операций по-другому, чтобы добиться лучшей классификации. Например, логично предложить, чтобы стоимость замены близких в некотором смысле элементов друг на друга была меньше, чем замена друг на друга элементов не близких. Так, например, понятно, что стоимость замены в кодировке символа 0 на символ 1 или 7 должна быть меньше, чем замены символа 0 на символ 4 или 5 например, так как 0,1,7 в целом указывают на направление вверх, в то время как 4 указывает на направление вниз. Тем самым, можно в формуле (1) определить член $m(i, j)$, как расстояние между направлениями (Рис. 12).

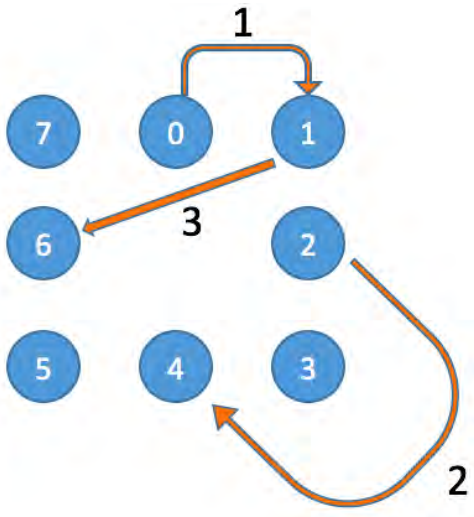


Рис. 12: Расстояние между направлениями.

Таким образом, мы учтём некоторую специфику, которая скрывается в наших данных, что в теории должно улучшить итоговую классификацию. Такая модификация стоимости операции замены, разумеется, не единственная, можно вводить различные требования. Например, можно считать стоимостью операции замены вели-

чину, обратно пропорциональную вероятности встретить символ, на который предлагается сделать замену, вслед за последовательностью из n символов. Эвристики придумывать можно различные, в этой работе им не уделено особое внимание, ввиду того, что, как будет изложено далее, уже какие-то базовые идеи дают хороший итоговый результат.

Разумеется, помимо расстояния Левенштейна существует ещё много способов измерения расстояния между последовательностями (метрика Жаккара, расстояние Хэмминга и др.), но как и в случае с экспериментами вокруг стоимости операции замены символа, эти расстояния в работе не будут описаны. Само по себе расстояние Левенштейна, вместе с изменённой функцией стоимости замены, как будет видно дальше, даёт неплохое качество классификации рукописных цифр набора данных MNIST.

2.3 Функциональное описание

Одним из современных подходов в распознавании геометрических образов является подход, основанный на вписывании неявных полиномов в облако точек [8–10]. Плюсом данного подхода является его универсальность, и кроме того, для его применения не нужно проводить тщательную предобработку данных. Непосредственное развитие в итоговом алгоритме классификации данный подход не получил, и в этой работе он рассматривается скорее как перспективное направление дальнейшего исследования. Ниже будут приведены текущие результаты, достигнутые с помощью такого подхода. Для некоторых цифр, метод, описанный в [8–10] даёт достаточно стабильное описание[1] (Рис. 13)

Для большинства цифр без предобработки такой метод даёт неосмысленный результат.

К счастью, для преодоления недостатков метода, описанного выше, можно воспользоваться его модиифкацией – 3L алгоритмом [12]. Смысл этого алгоритма в том, что мы пытаемся разделить наш объект на два класса и ищем такую кривую, которая наилучшим образом[2] отделяла бы эти два класса. Этому методу фактически

[1] с точки зрения визуального совпадения

[2] в среднеквадратическом смысле

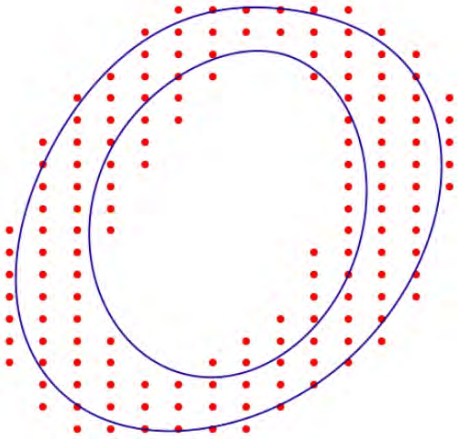


Рис. 13: Описание цифры 0 полиномом 4 степени

необходимо указать «корридор», внутри которого мы хотим чтобы прошёл полином. Результаты применения этого метода можно увидеть на (Рис. 14).

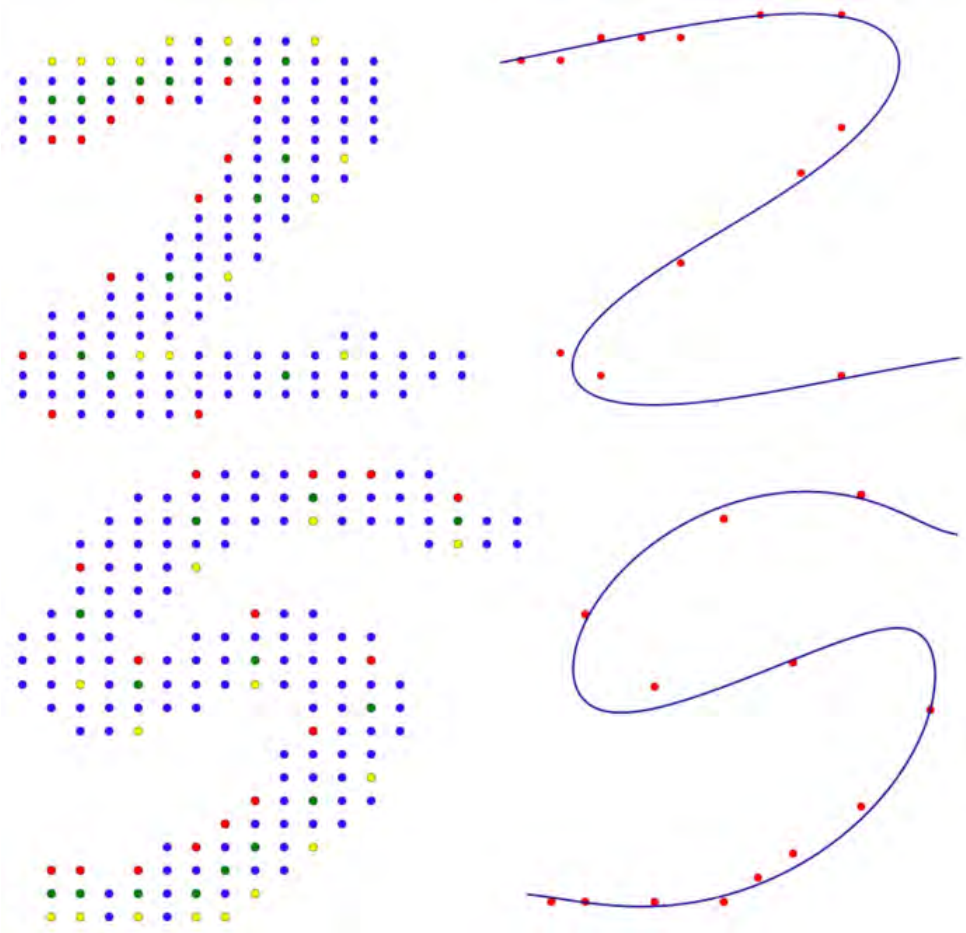


Рис. 14: Результат применения 3L алгоритма к цифрам 2 и 5

Критичным для данного метода является необходимость указывать алгоритму набор внешних и внутренних точек[1]. Вопрос о том, как автоматически подбирать эти точки в данной работе не рассматривается – можно предлагать какие-то эвристики, но этот метод не был разработан дальше, ввиду того, что алгоритм усложняется непропорционально улучшению качества классификации, который этот метод потенциально может обеспечить.

Кроме того, некоторые цифры, такие например как «3», плохо опишутся полиномом таким способом, так как полином функция гладкая, а у «3» серединную часть представляется сложным описать гладко.

Все эти вопросы поддалкивают к дальнейшему исследованию приложения данного функционального метода к задаче классификации рукописных цифр. Непосредственная классификация с помощью данного метода могла бы проводиться, благодаря скрещиванию его с цепным кодированием: после нахождения полинома, можно применить цепную кодировку уже не для исходного объекта, а для полинома, его описывающего. Интуитивно кажется, что такая кодировка будет более «гладкой», а следовательно все объекты одного класса будут больше похожи на свой класс и сильнее отличаться от объектов не своего класса.

3 Вычислительные эксперименты

В данном разделе будет продемонстрирован результат применения описанного выше цепного кодирования для классификации объектов из набора данных MNIST[2]. В качестве возможности применения цепного кодирования не только к цифрам будет рассмотрен ещё набор данных COIL-20[3]. Первый из этих наборов состоит из рукописных цифр, представленных векторами, содержамими значения от 0 до 255, символизирующими уровень серого в конкретной позиции. Второй набор содержит фотографии различных бытовых предметов с различных ракурсов.

[1] на графике внешние точки – красные и жёлтые, внутренние – зелёные

[2] http://yann.lecun.com/exdb/mnist/

[3] http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

3.1 Исходные данные и условия эксперимента. Набор данных MNIST

Исходный набор данных имеет вид (Рис. 15)

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0

Рис. 15: MNIST

Каждая строка в этом датасете содержит метку объекта (какая цифра) и вектор длины 784. Этот вектор получается вытягиванием матрицы размера 28 × 28 в одну строку. Матрица содержит в себе числа от 0 до 255, символизирующие уровень серого цвета (0 – чёрный цвет, 255 – белый)

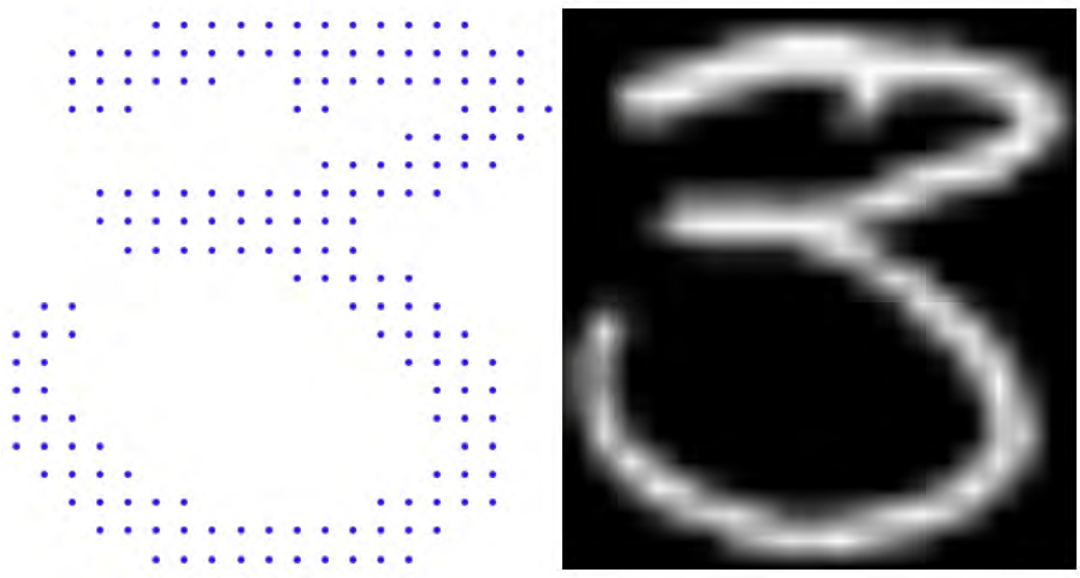


Рис. 16: визуализация одной строки

Набор данных разбивается на обучение и контроль. В ходе эксперимента оказалось, что 700 объектов для обучения достаточно, для получения конкурентноспособ-

ного качества классификации (по 70 объектов каждого из 10 классов). Ради экономии времени в тестовую выборку были включены 1000 объектов.

Псевдокод классификации представлен ниже. Для его работы необходимо предварительно создать цепное описание для тренировочных объектов. Назовём множество этих цепных описаний $\mathcal{A}_{train}$, а множеством классов этих объектов $\mathcal{K}_{train}$.

$\mathcal{A}_{train} = \{z \mid z = freeman\_chain(marching\_squares(x_i)), \quad \forall x_i \in train\}$

$\mathcal{K}_{train} = \{k_{z_i}\}_{i=1}^{|\mathcal{A}_{train}|}$

$train$ – множество тренировочных объектов, $marching\_squares$ – функция, возвращающая обрамление объекта, $freeman\_chain$ – функция, создающая по обрамлению цепную кодировку, k_{z_i} – класс объекта z_i, $levenstein\_distance(a, b)$ – функция, которая вычисляет расстояние Левенштейна между последовательностями a и b.

Algorithm 1 Алгоритм классификации

Require: x – матрица текущего объекта

Ensure: $k_x \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ – класс объекта

$chain_x = freeman\_chain(marching\_squares(x))$

for $i = 1, \ldots, |\mathcal{A}_{train}|$ **do**

$dist_i = levenstein\_distance(chain_x, z_i)$

end for

$distances = \{dist_i\}_{i=1}^{|\mathcal{A}_{train}|}$

$position = \underset{i \in \{1,\ldots,|\mathcal{A}_{train}|\}}{\operatorname{argmin}} (distances)$

$k_x = (\mathcal{K}_{train})_{position}$

Для каждого из объектов обучения составлялось обрамление алгоритмом Marching squares, затем, по этому обрамлению строилась цепная кодировка. Таким образом, получено 700 последовательностей. После этого, для каждого объекта обучения составлялось обрамление и так же считалась цепная кодировка, а классификаци происходила методом одного ближайшего соседа, то есть, считалось расстояние Левенштейна от текущего тестового объекта до всех 700 объектов обучения и тестовому объекту присваивался класс объекта обучения, до которого расстояние Левентшейна от этого объекта минимально (Рис. 17, Рис. 18).

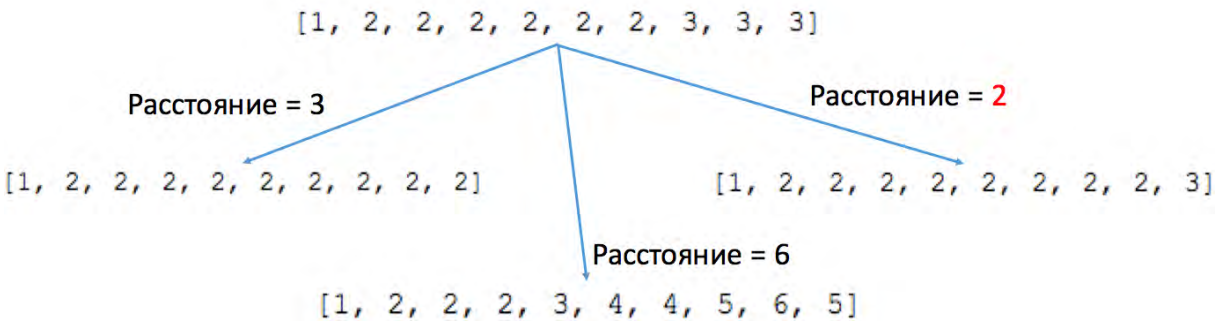


Рис. 17: Визуализация процесса классификации

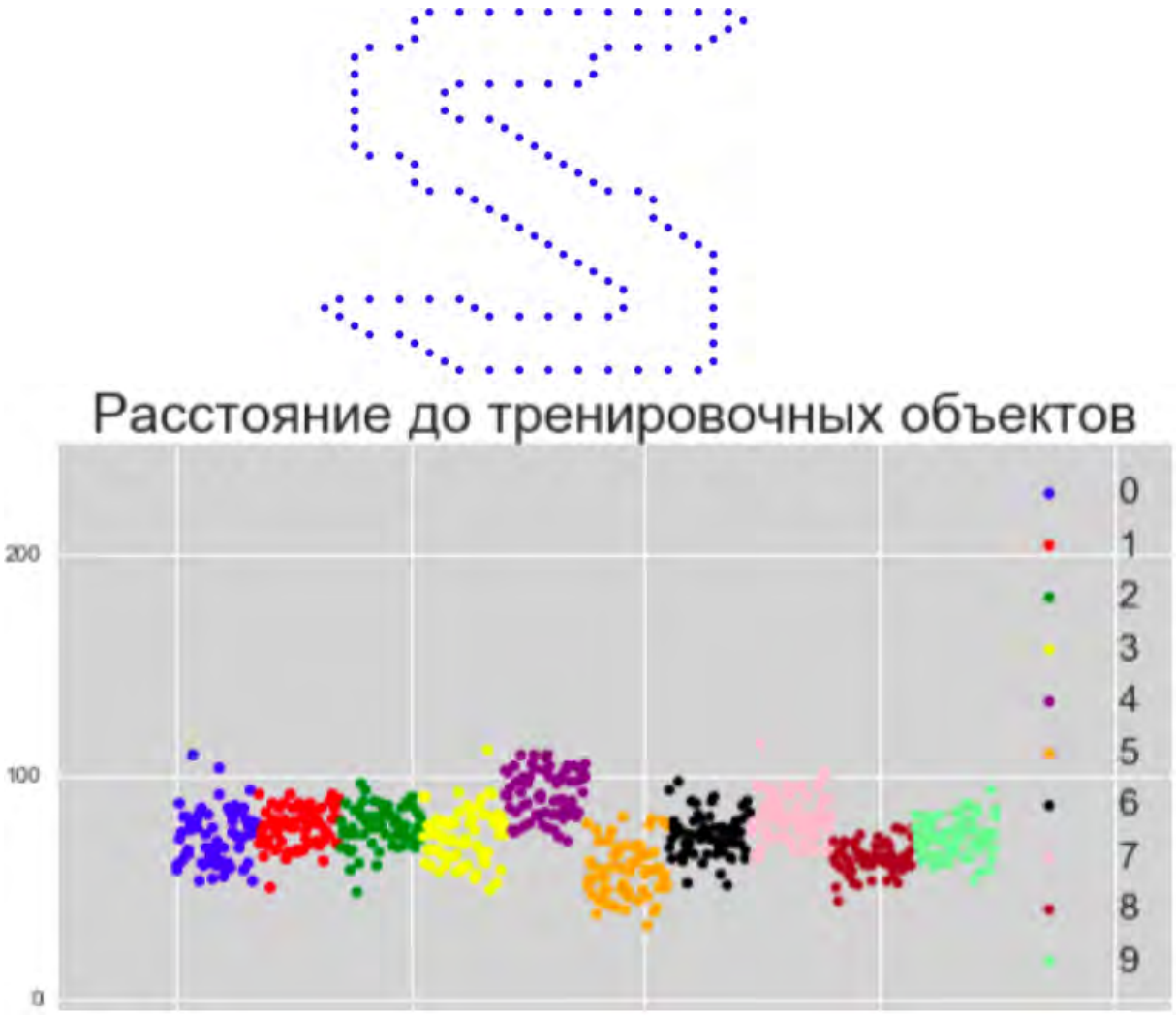
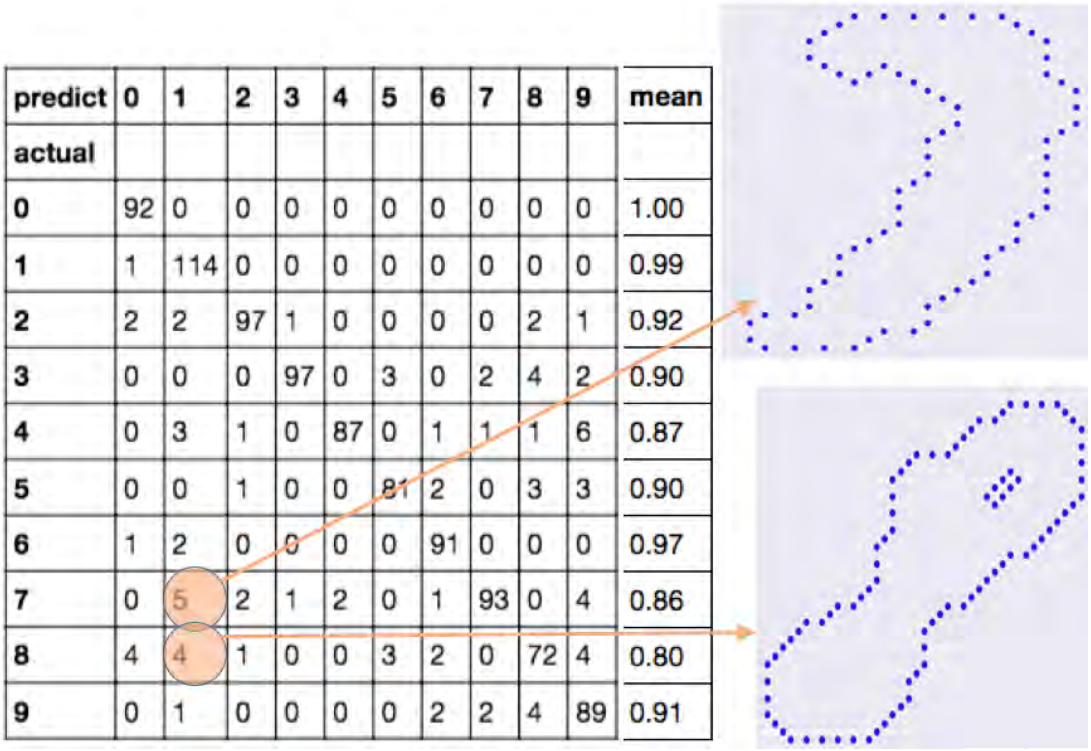


Рис. 18: Визуализация процесса классификации

3.2 Результаты эксперимента

В качестве результата приведём матрицу ошибок для классификации 1000 тестовых объектов (Рис. 19).



predict	0	1	2	3	4	5	6	7	8	9	mean
actual											
0	92	0	0	0	0	0	0	0	0	0	1.00
1	1	114	0	0	0	0	0	0	0	0	0.99
2	2	2	97	1	0	0	0	0	2	1	0.92
3	0	0	0	97	0	3	0	2	4	2	0.90
4	0	3	1	0	87	0	1	1	1	6	0.87
5	0	0	1	0	0	81	2	0	3	3	0.90
6	1	2	0	0	0	0	91	0	0	0	0.97
7	0	5	2	1	2	0	1	93	0	4	0.86
8	4	4	1	0	0	3	2	0	72	4	0.80
9	0	1	0	0	0	0	2	2	4	89	0.91

Рис. 19: Матрица ошибок

Общая точность, полученная на тестовых данных достигла 91%. Модифицированное расстояние Левенштейна (стоимость замены зависит от близости направлений) даёт более хорошие результаты (Рис. 20): общая точность 94,6%. На обеих таблицах (рисунки 19 и 20) по вертикали представлен реальный класс объектов (actual), по горизонтали прогнозный (predict).

Стоит отметить, что тестовые объекты выбирались случайно и равномерно, это обеспечивает репрезентативность искусственной тестовой выборки[1]. Точность корретно вычислять, ввиду сбалансированности классов в исходном датасете и в тестовой выборке.

[1] Она искусственная в том понимании, что для теста было отобрано 1000 объектов из тестового датасета, размер которого в оригинальном составлении на порядок больше.

predict	0	1	2	3	4	5	6	7	8	9	mean
actual											
0	92	0	0	0	0	0	0	0	0	0	1.00
1	1	114	0	0	0	0	0	0	0	0	0.99
2	0	2	100	1	0	0	0	0	1	1	0.95
3	0	0	0	102	0	1	0	0	3	2	0.94
4	0	2	0	0	93	0	0	0	1	4	0.93
5	0	0	0	0	0	85	1	1	2	1	0.94
6	0	0	0	0	0	0	94	0	0	0	1.00
7	0	5	2	1	2	0	1	93	0	4	0.86
8	2	1	0	0	0	2	3	0	80	2	0.89
9	0	0	0	0	0	0	1	0	4	93	0.95

Рис. 20: Матрица ошибок для модифицированного расстояния Левенштейна

3.3 Исходные данные и условия эксперимента. Набор данных COIL-20

Пример объектов, содержащихся в этом наборе данных (Рис. 21).

Рис. 21:

Исходный датасет содержит 20 классов изображений, по 72 фотографии на каждый класс. Каждая из 72 фотографий была сделана под разными углами. Ввиду того, что задача стоит не просто классифицировать этот набор данных, а классифицировать с помощью предложенного метода, из этого датасета было выбрано 9 классов (Рис. 22), различающийся своими контурами

Рис. 22: Пример объектов из отобранных для классификации классов

Набор из $72 \times 9 = 648$ объектов был поделен на тренировочный и тестовый. В тренировочный вошли по 20 объектов из каждого класса, отобранные для каждого класса случайным образом. В тестовый набор соответстенно вошли все остальные 468 объектов.

Для классификации бинаризуем матрицы, отвечающие каждому изображению, затем алгоритмом Marching squares составим обрамление, а по нему уже найдём цепную кодировку. Классификация, как и в случае датасета MNIST, производится

методом одного ближайшего соседа в пространстве последовательностей с введённым на нём расстоянием в виде расстояния Левенштейна.

3.4 Результаты эксперимента

На 468 объектах алгоритм показал точность классификации 96.8% (Рис. 23).

predict	0	1	2	3	4	5	6	7	8
actual									
0	52	0	0	0	0	0	0	0	0
1	0	49	0	0	0	0	3	0	0
2	0	0	46	0	5	0	1	0	0
3	0	0	0	52	0	0	0	0	0
4	0	0	1	0	51	0	0	0	0
5	0	0	2	0	0	49	0	1	0
6	0	0	1	0	0	0	51	0	0
7	0	0	0	0	0	0	0	52	0
8	1	0	0	0	0	0	0	0	51

Рис. 23: Матрица ошибок для набора данных COIL-20

Классы в этой таблице соответствуют классам из (Рис. 22) в следующем порядке: картинки перебирались слева направо, сверху вниз, то есть 0 – утка, 1 – брус, 2 – машина, 3 – кабриолет и тд.

3.5 Обсуждение и выводы

Метод классификации, на основе цепного кодирования и расстояния Левенштейна, продемонстрировал достойное качество классификации для наборов данных MNIST и COIL-20. Текущим бейзлайном в распознавании геометрических структур является нейросетевой подход. На наборе данных MNIST пятислойная конволюционная нейронная сеть с входным слоем длины 784 после одной эпохи обучения даёт качество 98%. Если довести количество эпох до 30 и более, то тестовый набор цифр распознаётся почти на 100%.

Итак, к плюсам текущего подхода можно отнести:

- Метод ближайшего соседа является «ленивым» алгоритмом классификации, то есть фактический его смысл это просто запоминание эталонных объектов. Это делает его гибким инструментом, который легко подстроить под конкретную задачу (вместе в цепным кодированием) без особого углубления в её отличительные черты (Это было продемонстрировано на датасетах MNIST и COIL-20 – хотя объекты в этих задачах сильно различаются, изменения в алгоритме были минимальные).

- Метод показал усточнивою работу при использовании даже малой части обучающей выборки. В обоих экспериментах для обучения было использовано на порядок меньше объектов, чем представлено в оригинальных тренировочных датасетах (тренировочный набор данных MNIST содержить 42000 объектов, для обучения было использовано всего 700 – меньше 2% всей выборки).

- Метод обладает интерпретируемостью, что на сегодняшний день в индустриальной data science является одним из важнейших качеств алгоритма.

- Даже при простой реализации метод даёт качество сравнимое с бейлайном (нейросетевой подход). Учитывая, что на сегодняшний день качество распознавания конкретно рукописных цифр доходит до 100%, не имеет смысла ставить своей целью превосходство предложенного алгоритма над бейзлайном.

Границы применимости этого метода так же довольно ясны. Данный алгоритм, в том виде, в котором он описан в этой работе применим для классификации геометрических объектов, чьи контуры принципиально различны. В связи с этим было ограничено множество классов в датасете COIL-20, так как два прямоугольных бруса с разными надписями на них (Рис. 24) данный алгоритм распознать не сможет.

Рис. 24: Объекты из COIL-20, которые нельзя распознать в помощью предложенного алгоритма

Конечно, можно усложнять алгоритм дальше – выделять на объекте не только контур, но и ещё какие-то геометрические струткуры (в примере с рисунком выше это могут быть надписи) и так же для них строить цепное описание. В таком случае алгоритм потеряет ту простоту, которой он обладает на текущем этапе, ведь придётся углубляться в предобработку данных, что само по себе является большой задачей в области компьютерного зрения. Однако данную идею можно считать толчкном и направлением к дальнейшему развитию метода, основанного на цепном кодировании.

4 Заключение

В данной работе был рассмотрен подход к распознаванию рукописных цифр, основанный на цепном кодировании. В результате исследования были получены следующие результаты:

- Ранее известный подход, использовавшийся для сжатия информации, адаптирован к задаче классификации
- В результате экспериментов был разработан универсальный алгоритм, позволяющий проводить классификацию геометрических объектов различной природы

(а именно: неунифицированные фотографии объектов удалось привести к единообразному виду, позволяющему в дальнейшем классифицировать объект)

- Проведены вычислительные эксперименты на различных наборах данных, показывающие компетентность предложенного метода для задач классификации

Список литературы

1. Freeman, Herbert. “On the Encoding of Arbitrary Geometric Configurations.” IEEE Transactions on Electronic Computers, EC-10, no. 2, 1961, pp. 260–268., doi:10.1109/tec.1961.5219197.

2. Arica, N., and F.t. Yarman-Vural. “An Overview of Character Recognition Focused on off-Line Handwriting.” IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews), vol. 31, no. 2, 2001, pp. 216–233., doi:10.1109/5326.941845.

3. Kaneko, T., and M. Okudaira. “Encoding of Arbitrary Curves Based on the Chain Code Representation.” IEEE Transactions on Communications, vol. 33, no. 7, 1985, pp. 697–707., doi:10.1109/tcom.1985.1096361.

4. Liu, Yong Kui, and Borut Žalik. “An Efficient Chain Code with Huffman Coding.” Pattern Recognition, vol. 38, no. 4, 2005, pp. 553–557., doi:10.1016/j.patcog.2004.08.017.

5. Bribiesca, Ernesto. “A New Chain Code.” Pattern Recognition, vol. 32, no. 2, 1999, pp. 235–251., doi:10.1016/s0031-3203(98)00132-0.

6. Žalik, Borut, et al. “Unsigned Manhattan Chain Code.” Journal of Visual Communication and Image Representation, vol. 38, 2016, pp. 186–194., doi:10.1016/j.jvcir.2016.03.001.

7. Žalik, Borut, and Niko Lukač. “Chain Code Lossless Compression Using Move-to-Front Transform and Adaptive Run-Length Encoding.” Signal Processing: Image Communication, vol. 29, no. 1, 2014, pp. 96–106., doi:10.1016/j.image.2013.09.002.

8. Interian, Ruben, et al. “Curve and Surface Fitting by Implicit Polynomials: Optimum Degree Finding and Heuristic Refinement.” Computers & Graphics, vol. 67, 2017, pp. 14–23., doi:10.1016/j.cag.2017.05.002.

9. Rouhani, Mohammad, and Angel D. Sappa. “A Fast Accurate Implicit Polynomial Fitting Approach.” 2010 IEEE International Conference on Image Processing, 2010, doi:10.1109/icip.2010.5654043.

10. Wolovich, W.a., and M. Unel. “The Determination of Implicit Polynomial Canonical Curves.” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 10, 1998, pp. 1080–1090., doi:10.1109/34.722620.

11. Li, Q., et al. “Implicit Fitting Using Radial Basis Functions with Ellipsoid Constraint.” Computer Graphics Forum, vol. 23, no. 1, 2004, pp. 55–69., doi:10.1111/j.1467-8659.2004.00005.x.

12. Blane, M.m., et al. “The 3L Algorithm for Fitting Implicit Polynomial Curves and Surfaces to Data.” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 3, 2000, pp. 298–313., doi:10.1109/34.841760.

13. Pratt, Vaughan. “Direct Least-Squares Fitting of Algebraic Surfaces.” Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH ’87, 1987, doi:10.1145/37401.37420.

14. Cooper, and Yalabik. “On the Computational Cost of Approximating and Recognizing Noise-Perturbed Straight Lines and Quadratic Arcs in the Plane.” IEEE Transactions on Computers, C-25, no. 10, 1976, pp. 1020–1032., doi:10.1109/tc.1976.1674543.

15. “Marching Squares.” Wikipedia, Wikimedia Foundation, 18 Dec. 2017, ru.wikipedia.org/wiki/Marching_squares

16. “Расстояние Левенштейна.” Wikipedia, Wikimedia Foundation, 9 Feb. 2018, ru.wikipedia.org/wiki/Расстояние_Левенштейна

5 Приложение

Пример вычисления обрамления (Рис. 25)

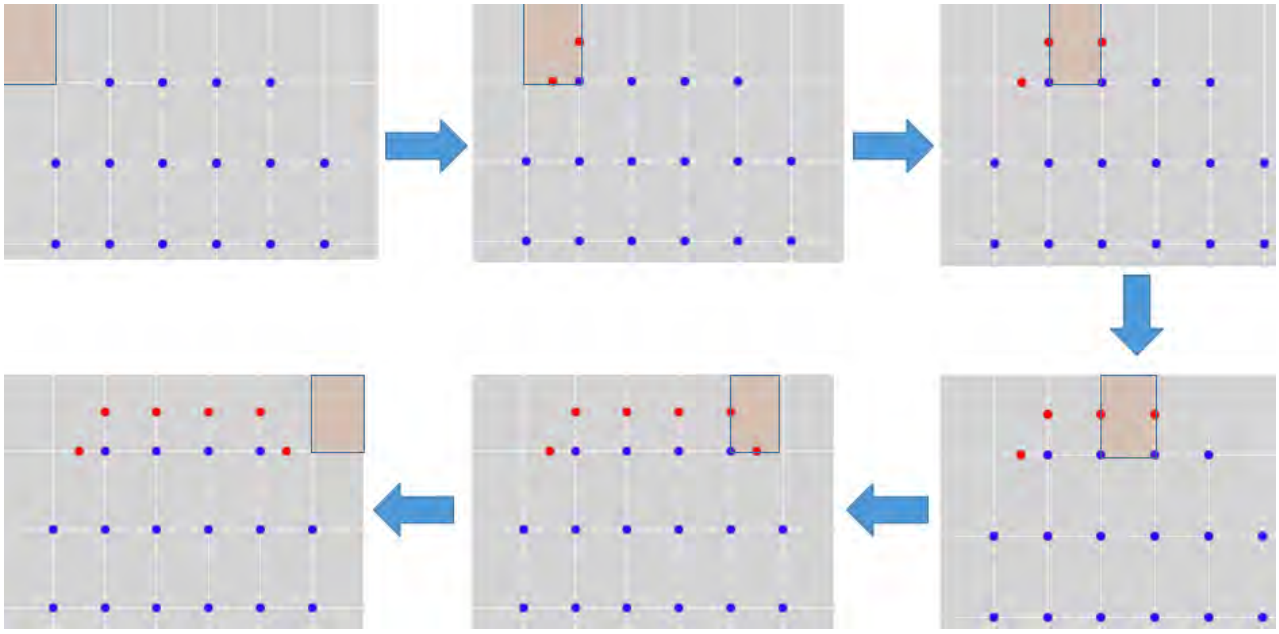


Рис. 25:

Пример вычисления цепной кодировки (Рис. 26)

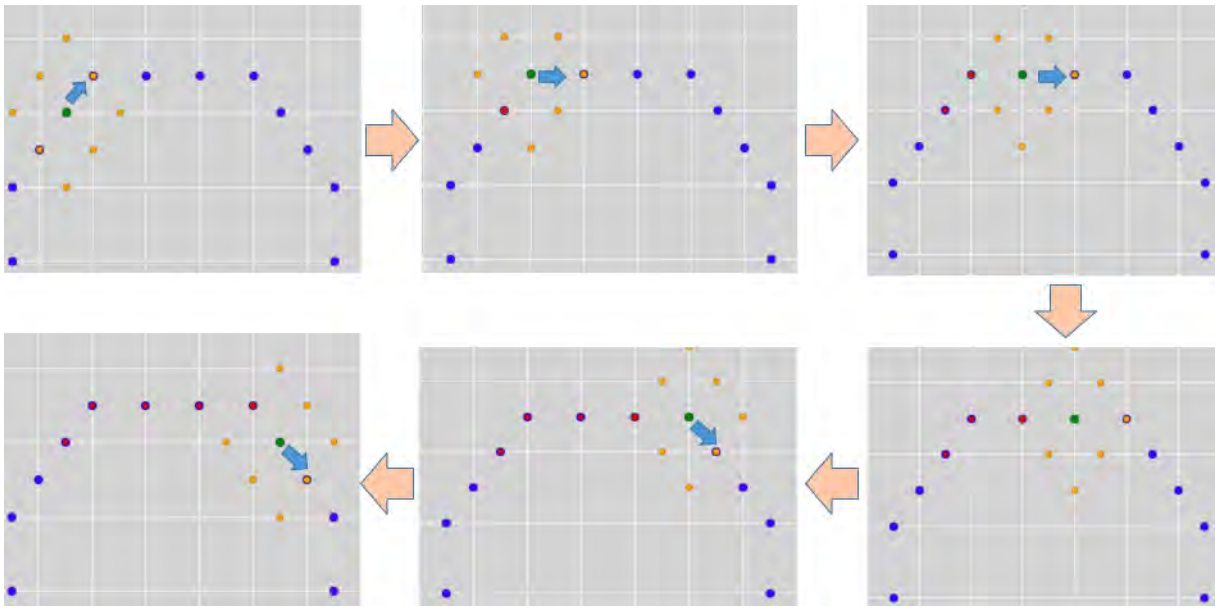


Рис. 26: