УДК 004.932.75'1

ВЫБОР ПРИЗНАКОВ ДЛЯ РАСПОЗНАВАНИЯ ПЕЧАТНЫХ КИРИЛЛИЧЕСКИХ СИМВОЛОВ

Багрова И.А., Грицай А.А., Пономарёв С.А., Сорокин С.В., Сытник Д.А.
ООО «Комплексные системы», Тверь

Поступила в редакцию 02.06.2010, после переработки 04.06.2010.

В данной работе проанализированы четыре типа признаков, которые могут быть применены при распознавании печатных кириллических символов. Проверены возможности построенных с использованием разных типов признаков классификаторов к обобщению и устойчивости к шумам на изображении. Предложен эвристический алгоритм выбора набора признаков для классификатора.

Four types of features for recognition of Cyrillic letters are analyzed. Algorithm for feature selection based on ideas of genetic optimization and clustering is proposed.

Ключевые слова: распознавание текста, кластеризация, выбор признаков, генетическая оптимизация.
Keywords: optical character recognition, clustering, genetic optimization, feature selection.

Введение

Большие объемы информации, накопленные до широкого распространения компьютеров, до сих пор остаются доступными лишь на бумажных носителях. Перевод этой информации в электронный вид может существенно повысить её доступность для широкого круга пользователей. Поскольку в рамках национального проекта «образование» российские школы получили доступ в сеть Интернет, сейчас любой школьник России может получить доступ к учебникам, художественной, научной и исторической литературе, которая представлена в электронных библиотеках. Учителя могут получить оперативный доступ к учебно-методической литературе. Историки – к архивным данным.

Задача распознавания текстов является актуальной и для документооборота коммерческих и государственных организаций. Несмотря на то, что в настоящее время большинство документов составляется на компьютерах, задача создания полностью электронного документооборота ещё далека до полной реализации. Как правило, существующие системы охватывают деятельность отдельных организаций, а обмен данными между организациями осуществляется с помощью традиционных бумажных документов.

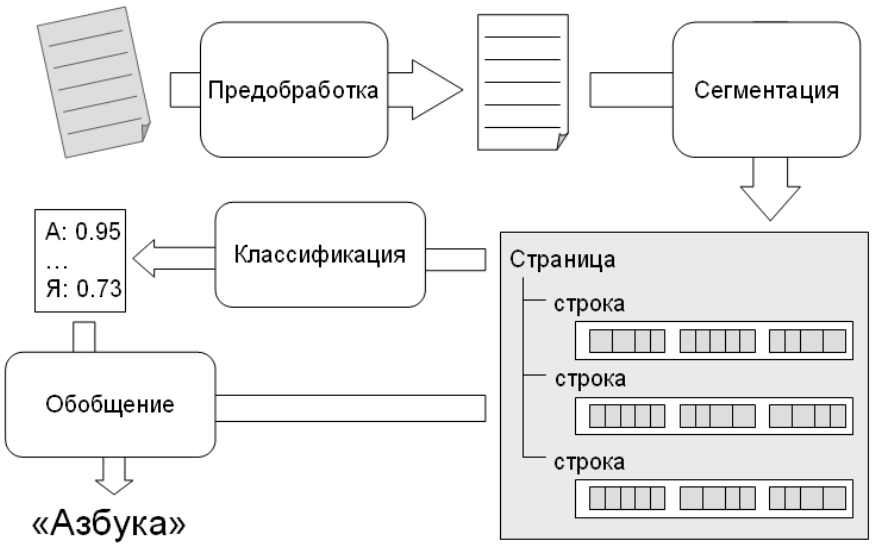

Рис.1: Система распознавания текста

Многие работы по распознаванию текстов велись за рубежом, и соответственно ориентировались на распознавание текстов на английском или других языках. К сожалению, решение многих возникающих в процессе распознавания текста подзадач существенно зависит от языка документа. При адаптации систем для поддержки нового языка во многих случаях возможно использование лишь идей, лежащих в основе алгоритмов распознавания, но и они могут потребовать корректировки и дополнительных исследований.

Большинство систем распознавания текста, поддерживающих кириллические алфавиты, представляет собой коммерческие продукты с закрытым исходным кодом, что делает практически невозможным их всестороннее изучение научным сообществом и дальнейшую модификацию. Кроме того, большинство имеющихся продуктов имеет жесткие ограничения по совместимости с различными операционными системами.

Описанные в статье исследования проводились в рамках научно-исследовательской работы по теме «Создание открытого алгоритма распознавания кириллических печатных символов на графических носителях и создание на его основе прототипа системы обработки информации» выполняемого в рамках федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009-2013 гг.

1. Архитектура системы распознавания

Для понимания места рассматриваемой в статье задачи и требований, которым должны отвечать наборы признаков рассмотрим архитектуру разрабатываемой нами системы распознавания символов (рис.1).

На вход системы распознавания поступает растровое изображение страницы документа. Как правило, перед началом распознавания исходное изображение

Рис. 2: Объединение нескольких букв в одну компоненту связанности

Рис. 3: Распадение изображений букв на несвязанные компоненты вследствие низкого качества сканирования

необходимо подготовить. Это производится на этапе предобработки. Он включает фильтрацию изображения от шумов, повышение резкости и контрастности изображения, выравнивание и преобразование в используемый системой формат (в нашем случае 8-битное изображение в градациях серого).

Подготовленное изображение попадает на вход модуля сегментации. Задачей этого модуля является выявление структурных единиц текста – строк, слов и символов. В результате получается дерево, отражающее структуру текста на странице, с каждым элементом которого сопоставлен фрагмент изображения. Выделение фрагментов высоких уровней, таких как строки и слова, может быть осуществлено на основе анализа промежутков между тёмными областями.

К сожалению, такой подход не может быть применён для выделения отдельных букв, поскольку в силу особенностей начертания или искажения изображения соседних букв могут объединяться в одну компоненту связанности (рис. 2) или наоборот, изображение одной буквы может распадаться на отдельные компоненты связанность (рис. 3). Во многих случаях для решения задачи сегментации на уровне букв используются сложные эвристические алгоритмы [1].

Мы полагаем, что для принятия окончательного решения о прохождении границы букв на таком раннем этапе обработки системе распознавания недостаточно информации. Поэтому задачей модуля сегментации на уровне букв в нашем случае является нахождение возможных границ символов внутри слова, а окончательное решение о разбиении слова будет приниматься на последнем этапе обработки, с учётом идентификации отдельных фрагментов изображения как букв. Дополнительным преимуществом такого подхода является возможность работы с начертаниями букв, состоящих из нескольких компонент связанности без специальной обработки таких случаев.

Выявленные фрагменты изображения подаются на вход классификатора, выходом которого является вектор возможности принадлежности изображения к классу той или иной буквы. Работа классификатора осуществляется в два шага (рис. 4). Сначала по исходному изображению вычисляются признаки. Значение каждого признака является функцией от яркостей некоторого подмножества пикселей изображения. В результате получается вектор значений признаков, который поступает на вход нейронной сети. Каждый выход сети соответствует одной из

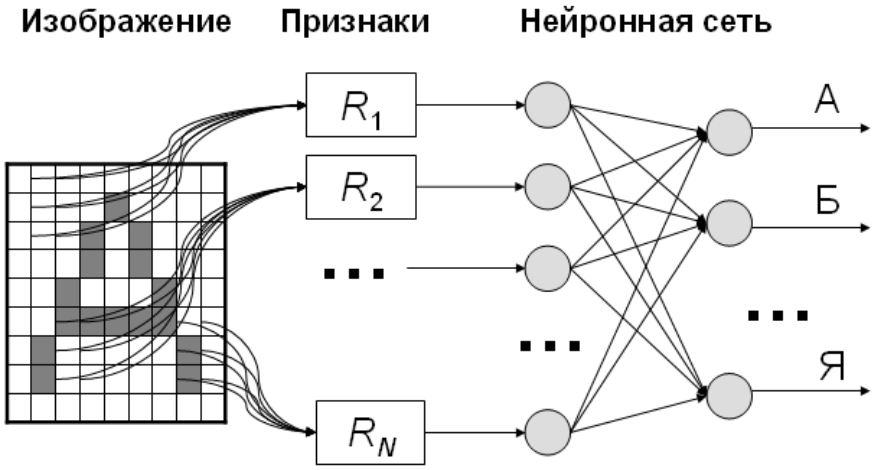

Рис. 4: Классификатор

букв алфавита, а получаемое на выходе значение можно рассматривать как уровень возможности того, что входное изображение содержит ту или иную букву.

На последнем этапе принимается решение о наиболее правдоподобном варианте прочтения слова. Для этого используются уровни возможности прочтения отдельных букв, межбуквенной сегментации и частоты сочетаний букв в русском языке. Далее мы рассмотрим некоторые проблемы, которые возникают в процессе создания подсистемы распознавания символов, предназначенной для использования в качестве компонента описанной системы распознавания текста.

2. Используемые типы признаков

Как было указано выше, первым этапом работы классификатора является вычисление по изображению символа определённых числовых характеристик, называемых признаками. Использование такого подхода позволяет сократить размерность пространства, в котором решается задача классификации, и тем самым существенно упростить задачу создания классификатора, что, как ожидается, должно привести к улучшению качества его работы.

Для использования в задачах распознавания символов используются различные способы вычисления признаков. Одним из наиболее общих подходов является использование вычислений, использующих в качестве входа количество чернил (чёрных точек), попадающих в ту или иную ограниченную область изображения. Для определения признака подобного рода необходимо задать некоторое подмножество точек входного изображения P и функцию $R : P \to [0, 1]$. Нами были исследованы четыре различных типа признаков:

1. Линейные бинарные.

   В этом случае признак представляет собой отрезок прямой на изображении,

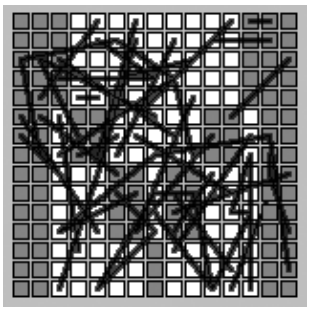

Рис. 5: Линейные признаки

заданный координатами начальной и конечной точки:

$$P^{linear}((x_1,y_1),(x_2,y_2)) = \{(x,y)|\exists d_x, d_y \in [-0.5, 0, 5], \exists \alpha \in [0,1] \\ (x+d_x, y+d_y) = (x_1,y_1) + \alpha(x_2,y_2)\}, \quad (1)$$

где P^{linear} - множество точек изображения, входящих в признак; (x_1, y_1) и (x_2, y_2) - координаты точек, определяющих отрезок.

Входящие в признак точки могут быть сгенерированы, например, с помощью алгоритма Брезенхэма [2].

Выходное значение признака равно единице, в случае если среди входящих в него точек имеется точка, уровень черноты которой превышает заданный порог:

$$R(P) = \begin{cases} 1, \text{ если } \min\limits_{p \in P}\{brightness(p)\} < \theta, \\ 0, \text{ иначе,} \end{cases} \quad (2)$$

где: R – выходное значение признака; P – множество точек изображения, входящих в признак; $brightness(p)$ – яркость точки p из диапазона от 0 до 255, 0 – точка чёрная, 255 – точка белая; θ – порог.

2. Линейные непрерывные.

   Эти признаки также представляют собой отрезок, однако в данном случае выходное значение признака представляет собой отношение числа чёрных точек к общему числу точек признака и может непрерывно изменяться в диапазоне от 0 до 1:

$$R(P) = \frac{\sum_{p \in P}(255 - brightness(p))}{255 * |P|}. \quad (3)$$

3. Прямоугольные бинарные.

   Признак имеет форму прямоугольника, со сторонами параллельными границам изображения, заданного координатами двух противоположных углов:

$$P^{rect}((x_1,y_1),(x_2,y_2)) = \{(x,y)|\, \min\{x_1,x_2\} \le x \le \max\{x_1,x_2\}, \\ \min\{y_1,y_2\} \le y \le \max\{y_1,y_2\}\}. \quad (4)$$

Принимает значение 0 или 1 в зависимости от наличия чёрных точек заданного порога, согласно (2).

4. Прямоугольные непрерывные. Признак прямоугольной формы, определяющий отношение числа чёрных точек к общему их числу, согласно (3).

Подобные признаки часто используются для сокращения размеров входного пространства в задачах распознавания, такая методика известна в области ИИ как грубое кодирование [3]. Например, в работе [4] для распознавания рукопечатных символов использовались признаки, которые мы называем прямоугольными непрерывными.

Для создания классификатора необходимо задать набор признаков $R = \{R_1, \ldots, R_n\}$, включающий заданное при определении архитектуры классификатора число признаков. Нами рассматривались наборы, включающие по 50 признаков одинакового типа.

3. Обобщение начертаний символов

Обучение классификатора в задачах распознавания символов используется в рамках концепции обучения с учителем, при этом для подбора параметров нейронной сети используется алгоритм обратного распространения ошибки [5]. Классификатору предъявляются образцы изображений известных букв и добиваются минимизации среднеквадратического отклонения выхода нейронной сети от правильного ответа по всей предъявляемой выборке. Однако в процессе эксплуатации классификатору придется распознавать не только те изображения, которые предъявлялись в процессе обучения. Распознаваемые изображения могут быть зашумлены или созданы с использованием отличающихся шрифтов. Успех работы системы распознавания в целом во многом определяется тем, насколько хорошо классификатор сможет решить задачу распознавания заранее неизвестных ему символов. Общепризнано, что нейронные сети обладают свойством обобщения [6], что позволяет использовать их для решения этой задачи. Однако результат работы классификатора в целом зависит и от используемого набора признаков.

Для оценки способности классификатора распознавать текст неизвестного ему заранее начертания нами были использованы два теста, в которых после обучения классификатора на одном наборе изображений букв, ему предлагалось распознать набор изображений, включающий начертания, не входившие в обучающую выборку.

В первом случае для обучения использовались шрифты семейства Times New Roman: обычный, жирный и курсив (всего 99 обучающих примеров: 3 шрифта по 33 буквы). Для проверки обобщения сети предъявлялись символы как из этих шрифтов, так и из Times New Roman жирный курсив. Этот тест, включающий 132 изображения букв, в дальнейшем будем называть простым.

Как для обучения, так и для проверки изображения буквы генерировались с помощью функции печати текста операционной системы и масштабировались на входное поле классификатора, которое имело размер 15x15 пикселей.

Для оценки способности к генерализации учитывалось число букв, которые не были правильно распознаны классификатором (максимальная возможность соответствовала другой букве) и число случаев, когда правильная буква не попала в тройку с наибольшими уровнями возможностей.

Для тестирования каждого типа признаков генерировалось 10 случайных наборов признаков данного типа, по 50 признаков в каждом. Для генерации каждого признака случайным образом определялись координаты двух точек, задающих расположение признака. Признаки, у которых длина отрезка между точками была меньше $1/10$ или больше $1/2$ от размера стороны входного изображения, отбрасывались.

Вторая часть классификатора представлялась нейронной сетью с 50 нейронами входного слоя и 33 выходного. Нами были проверены варианты сети без скрытых слоёв, с одним и двумя скрытыми слоями. Добавление скрытых слоёв не оказало существенного влияния на качество работы классификатора, однако существенно замедляло как его обучение, так и работу. Поэтому мы остановили свой выбор на наиболее быстром варианте без скрытых слоёв.

Обучение нейронной сети велось методом обратного распространения ошибки.

Поскольку алгоритм обратного распространения ошибки является, по сути, алгоритмом градиентного спуска, он может находить лишь локальный, а не глобальный экстремум функции ошибки. Для уменьшения влияния начального состояния нейронной сети на качество распознавания процесс обучения проводился 10 раз для каждого набора признаков из разных начальных состояний сети. Таким образом, для каждого типа признаков было проведено 100 испытаний.

Результаты тестов приведены в таблице ниже. Для практического использования может быть отобран классификатор с наилучшими показателями, отраженными в столбце «Лучшее».

Таблица 1: Результаты тестирования обобщения на простом тесте

Вид признаков	Неправильно классифицировано букв		Правильная буква не вошла в тройку	
	Лучшее	Среднее	Лучшее	Среднее
Линейные бинарные	19	24.18	11	16.23
Линейные непрерывные	5	8.57	1	3.23
Прямоугольные бинарные	18	23.87	8	14.48
Прямоугольные непрерывные	3	6.50	0	1.80

В более сложном случае классификатор обучался на всех четырёх шрифтах семейства Times New Roman, а для проверки использовался полный набор из 138 шрифтов содержащих кириллические символы из комплекта Windows XP. Таким образом, тестовый набор включал 4554 изображений букв.

Необходимо отметить, что этот набор включает в себя шрифты с начертаниями символов, серьезно отличающимися от Times New Roman, включая вычурные декоративные варианты. Вряд ли можно ожидать, что такие шрифты могут быть хорошо распознаны подобным классификатором. Поэтому достаточно большое количество ошибок на этом тесте является вполне ожидаемым.

4. Устойчивость к шумам

Для оценки устойчивости классификации к шумам на изображении классификатор обучался на всех четырех шрифтах семейства Times New Roman, а затем

Таблица 2: Результаты тестирования обобщения на сложном тесте

Вид признаков	Неправильно классифицировано букв		Правильная буква не вошла в тройку	
	Лучшее	Среднее	Лучшее	Среднее
Линейные бинарные	2540	2813.01	1537	1820.78
Линейные непрерывные	1667	1922.42	919	1074.41
Прямоугольные бинарные	2521	2762.27	1558	1790.28
Прямоугольные непрерывные	1745	1994.65	969	1137.47

тестировался на изображениях букв этих же шрифтов, на которых у заданного числа случайно выбранных пикселей цвет изменялся на выбранный случайно из диапазона 0. . . 255. Так же как и в предыдущем случае создавалось 10 случайных наборов признаков и для каждого 10 раз проводилось обучение нейронной сети из случайного начального состояния.

Результаты тестирования приведены в таблицах 3 и 4.

Таблица 3: Результаты тестирования обобщения на тесте с отношением сигнал/шум 12 дБ

Вид признаков	Неправильно классифицировано букв		Правильная буква не вошла в тройку	
	Лучшее	Среднее	Лучшее	Среднее
Линейные бинарные	63	87.29	29	54.92
Линейные непрерывные	12	15.51	2	4.36
Прямоугольные бинарные	73	85.38	41	57.56
Прямоугольные непрерывные	8	12.7	0	3.4

Таблица 4: Результаты тестирования обобщения на тесте с отношением сигнал/шум 6 дБ

Вид признаков	Неправильно классифицировано букв		Правильная буква не вошла в тройку	
	Лучшее	Среднее	Лучшее	Среднее
Линейные бинарные	87	107.25	51	78.84
Линейные непрерывные	10	22.72	2	7.22
Прямоугольные бинарные	92	106.6	60	79.79
Прямоугольные непрерывные	11	19.97	1	6.42

5. Автоматический выбор признаков

Рассмотренные ранее результаты получены для случайных наборов признаков. При этом одни из наборов оказывались более удачными, чем другие. Практический интерес представляет целенаправленный поиск таких наборов, которые обеспечивали бы хорошие показатели работы классификатора.

В работах [7], [8] показано, что при разработке систем нечёткого вывода хорошим критерием для автоматического выбора конфигурации нечётких терм-множеств является минимизация вероятности (или энтропии) одновременной активации терм-множеств одной переменной, при максимизации одновременной активации терм-множеств разных переменных. Используемые нами наборы признаков могут быть рассмотрены как нечёткие терм-множества одной переменной, поэтому представляется естественным проверить эффективность данного подхода в нашем случае.

Энтропия совместной активации двух признаков есть

$$H^{i,k} = -\frac{1}{T}\sum_{t=1}^{T}[R_i(t) * R_k(t)] * \log(R_i(t) * R_k(t)), \tag{5}$$

где T – число компонентов обучающей выборки, а $R_j(t)$ – реакция j-го признака на t-ом элементе обучающей выборки.

Описанная в [7], [8] технология проектирования нечётких систем управления подразумевает минимизацию суммы совместных энтропий вида (5) по всем парам терм-множеств одной переменной, что для нашего случая даёт следующий критерий качества набора признаков:

$$\sum_{i \neq k} H^{i,k} \to \min. \tag{6}$$

Для генерации наборов признаков, удовлетворяющих критерию (6), мы использовали генетическую оптимизацию. Проведённые вычислительные эксперименты показали, что характеристики полученных таким образом наборов признаков не превосходят таковые для случайно сгенерированных наборов.

Однако использование интегрального критерия (6) не является единственным возможным способом построения наборов признаков, наименее похожих друг на друга по реакции на изображения обучающей выборки. Для решения этой задачи был предложен следующий алгоритм $ClusterSelection$:

$ClusterSelection(InitialCount, ClusterCount, SelectCount)$

1. Сгенерировать большое количество случайных признаков:

$$R^{initial} = \{R_1, R_2, \ldots, R_{InitialCount}\}.$$

2. Провести кластеризацию этого набора, используя (5) как функцию расстояния между элементами:

$$C = Cluster(R^{initial}, H, ClusterCount),$$

где: $Cluster(A, D, N)$ – процедура кластеризации множества A на N кластеров используя расстояние D, $C = \{C_1, \ldots, C_{ClusterCount}\}$, $C_i \subset R^{initial}$, $C_i \cap C_j = \varnothing$ для $i \neq j$, H – используемая в качестве расстояния энтропия совместной активации (5).

3. Для получения итогового набора выбрать заданное число представителей каждого кластера:

$$R = \cup_{i=1}^{ClusterCount} R^i, \text{ где } R^i \subseteq C_i, |R^i| = SelectCount.$$

В этом алгоритме параметрами являются: *InitialCount* – число изначально создаваемых случайных признаков, *ClusterCount* - число выделяемых кластеров, *SelectCount* – число признаков, выбираемых из каждого кластера.

Результатом работы алгоритма является набор признаков R.

Особенностью задачи кластеризации, возникающей на 2 шаге алгоритма, является то, что при её решении не должны использоваться вычисляемые алгоритмом искусственные элементы (например, центры кластеров), поскольку их построение в данной ситуации не представляется возможным. Это сужает круг алгоритмов, которые могут быть применены в данном случае, исключая например такие варианты, как например алгоритм K-средних. Нами был использован алгоритм кластеризации, основанный на алгоритме построения минимального остовного дерева. Сначала рассматривается полный граф всех возможных связей между элементами, для него строится минимальное остовное дерево и затем для получения N кластеров достаточно удалить $N-1$ наиболее длинных рёбер.

Проверка эффективности данного подхода осуществлялась на простом тесте, описанном в 3 разделе. Сравнивались качества наборов признаков, полученных случайным выбором и выбранных алгоритмом *ClusterSelection* (выбор производился из одних и тех же наборов). Использовались прямоугольные непрерывные признаки, ранее показавшие себя наиболее эффективными в этом тесте. Результаты тестирования представлены в таблице 6.

Алгоритм	Неправильно классифицировано букв		Правильная буква не вошла в тройку	
	Лучшее	Среднее	Лучшее	Среднее
Случайный выбор	3	6.64	0	2.19
Кластеризация с выбором 2 элементов из кластера	3	6.46	0	2.04

Как можно видеть, наборы признаков, отбираемые алгоритмом *ClusterSelection*, в среднем оказываются более эффективными, чем случайный выбор.

Поиск оптимального классификатора с использованием алгоритма *ClusterSelection* подразумевает генерирование большого числа случайных наборов, из которых с помощью кластеризации выделяются наборы-кандидаты. Для каждого кандидата необходимо провести несколько раундов обучения нейронной сети, провести тестирование и выбрать лучший классификатор.

Построение кластеризации с помощью минимального остовного дерева осуществляется достаточно медленно: сложность применённого нами алгоритма Борувки $O(N^2 \log(N))$ [9]. Поэтому при использовании больших исходных наборов алгоритм *ClusterSelection* может уступить случайному поиску, так как последний будет за то же время проверять гораздо большее число вариантов. А в небольших исходных наборах может оказаться недостаточно далеких друг от друга элементов. Для решения этой дилеммы мы совместили идею генетической оптимизации с алгоритмом выбора признаков *ClusterSelection*.

В отличие от классического генетического алгоритма (ГА), который в каждый момент времени оперирует с набором закодированных в хромосомах решений задачи оптимизации, в нашем алгоритме каждая хромосома кодирует только один

признак, а решение (набор признаков) кодируется всем текущим поколением. Для получения нового поколения используются традиционные для ГА генетические операции, но в качестве процедуры отбора индивидуумов для следующего поколения используется идея алгоритма $ClusterSelection$. Таким образом, генетические операции обеспечивают создание новых признаков, а алгоритм кластеризации, работая в каждый момент на достаточно небольшом наборе элементов, поддерживает их «непохожими» с точки зрения (5) друг на друга. Этот процесс генерации наборов признаков продолжается некоторое время, после чего перезапускается заново на случайном начальном наборе.

Таким образом, получаем алгоритм $ClassificatorSearch$ построения классификатора:

$$ClassificatorSearch(ClusterCount, SelectCount, RepeatCount, p^{\Gamma}, p^{\Delta}, NetCount, BPSteps)$$

1. Создать случайным образом первое поколение
   $S^0 = \{s_1, \ldots, s_{ClusterCount*SelectCount}\}$,
   где s_i – битовая строка.
2. Повторять для i от 0 до $RepeatCount$
   i. используя генетические операции (мутация, скрещивание) на основе исходной популяции создать новых индивидуумов:
   $$S^{new} = S^i \cup \Gamma(S^i, p^{\Gamma}), \cup\Delta(S^i, p^{\Delta}),$$
   где: Γ - оператор скрещивания, p^{Γ} - вероятность скрещивания, Δ - оператор мутации, p^{Δ} - вероятность мутации.
   ii. С помощью кластеризации отобрать из расширенного набора индивидуумов следующее поколение:
   $$C = Cluster(F(S^{new}), H, ClusterCount),$$
   $$S^{i+1} = \cup_{j=1}^{ClusterCount} F^{-1}(R^j), \text{ где } R^j \subseteq C_j, |R^j| = SelectCount.$$
   где: F – функция декодирования битовой строки в параметры признака, F^{-1} – обратное преобразование.
   iii. Провести тестирование текущего набора, запомнить лучший вариант:
   $$BestClassifier = Test(BestClassifier, F(S^{i+1}), NetCount, BPSteps).$$
3. Перейти к шагу 1.

Параметрами алгоритма $ClassificatorSearch$ являются: $ClusterCount$ - число выделяемых кластеров, $SelectCount$ – число признаков, выбираемых из каждого кластера, $RepeatCount$ – число поколений генетического поиска, p^{Γ} - вероятность скрещивания для генетического алгоритма, p^{Δ} - вероятность мутации, $NetCount$

– число различных нейронных сетей со случайными начальными весами, создаваемых для каждого набора признаков. $BPSteps$ – число шагов алгоритма обратного распространения для обучения нейронных сети.

Результатом работы алгоритма является лучший найденный классификатор $BestClassifier$, включающий набор признаков и обученную нейронную сеть.

Тестирование наборов признаков осуществлялось следующим образом. Для текущего набора сначала создавалось несколько нейронных сетей. Каждая сеть обучалась с помощью алгоритма обратного распространения ошибки на небольшое количество шагов, и затем из них выбиралась одна сеть, имеющая наименьшую ошибку распознавания. Для этой сети проводился второй раунд обучения, включавший большее количество шагов алгоритма обратного распространения ошибки. Полученный в результате классификатор тестировался и сравнивался с найденными ранее.

$Test(CurrentBestClassifier, ReceptorSet, NetCount, BPSteps)$

1. Создать набор нейронных сетей $\Psi_1, \ldots, \Psi_{NetCount}$ со случайными весами.

2. Выполнить процедуру обучения сетей

$$\Psi_i = BackPropogation(\Psi_i, BPSteps/NetCount), i \in 1, \ldots, NetCount,$$

где $BackPropogation(\Psi, Steps)$ выполняет $Steps$ шагов обучения сети Ψ методом обратного распространения ошибки.

3. Найти сеть с наименьшим значением ошибки на обучающей выборке:

$$\Psi^* = \arg\min_i E(\Psi_i),$$

где E – функция ошибки вида

$$E = \frac{1}{2}\sum_{t=1}^{T} |\Psi(R(t)) - l(t)|^2,$$

где $R(t) = (R_1(t), \ldots, R_{ClusterCount*SelectCount(t)})$ – вектор выходных значений признаков набора ReceptorSet на t-ом компоненте обучающей выборки, $l(t)$ – соответствующее желательное значение выхода классификатора.

4. Завершить обучение выбранной сети:

$$\Psi^* = BackPropogation(\Psi^*, BPSteps/(1 - NetCount)).$$

5. Сравнить параметры полученного классификатора $\Psi^*(ReceptorSet)$ с наилучшим имеющимся классификатором $CurrentBestClassifier$. Заменить, если новый классификатор показал лучшие результаты.

В этом алгоритме параметрами являются: $CurrentBestClassifier$ - наилучший найденный на данный момент классификатор, не определён при первом вызове, $ReceptorSet$ – число признаков, выбираемых из каждого кластера, $NetCount$ –

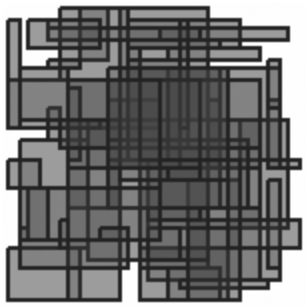

Рис. 6: Признаки, полученные алгоритмом $ClassificatorSearch$

число различных нейронных сетей со случайными начальными весами, создаваемых для каждого набора признаков. $BPSteps$ – число шагов алгоритма обратного распространения для обучения нейронных сети. Алгоритм запоминает лучший из протестированных классификаторов в переменной $CurrentBestClassifier$. В случае использования нескольких критериев, например минимальной ошибки обучения, минимального числа ошибок обобщения в разных условиях, целесообразно запоминать не один классификатор, а Парето-оптимальное множество.

Использованная в алгоритме Test двухступенчатая схема обучения позволяет сократить время работы алгоритма, сосредоточившись на обучении и тестировании наиболее перспективной сети для каждого классификатора. Как показала накопленная статистика, та сеть, которая имеет минимальную ошибку после первой тысячи шагов обратного распространения ошибки, чаще всего сохраняет своё преимущество и с продолжением обучения, а потом показывает наилучшие способности к обобщению.

В результате работы алгоритма $ClassificatorSearch$ в течение 40 минут был построен классификатор, неправильно классифицирующий на простом тесте лишь две буквы (правильные решения входят в тройку наиболее возможных с точки зрения классификатора), что превосходит все ранее построенные в многочисленных экспериментах классификаторы.

Заключение

Проведенные эксперименты показывают сильное превосходство непрерывных признаков перед признаками бинарными. Разница между признаками, представленными прямоугольными областями и отрезками гораздо менее значительна. Интересно заметить, что в случае двоичных признаков прямоугольный вариант практически всегда оказывается чуть хуже, чем у отрезков, тогда как для непрерывных признаков ситуация меняется на противоположную. В частности, только прямоугольные непрерывные признаки смогли обеспечить безошибочное распознавание зашумленных на $1/8$ площади изображения символов.

Худшие свойства бинарных признаков проявляются и в процессе обучения. При их использовании практически всегда наблюдаются осцилляция функции ошибки в процессе обучения, что говорит о неблагоприятной для алгоритма обратного распространения ошибки форме поверхности функции ошибки [5]. Так при выполнении первого теста такое явление наблюдалось в 94 случаев из 100 для линейных

признаков, и во всех 100 для площадных. В то же время при использовании непрерывных признаков осцилляции наблюдались лишь в исключительных случаях.

На основании вышеизложенного представляется естественным рекомендовать использование прямоугольных непрерывных признаков для решения задачи классификации изображений символов.

Установлено, что для задачи распознавания символов, использование минимизации суммы совместной энтропии не является подходящим критерием для автоматического выбора набора признаков. Однако показатели случайного выбора набора признаков могут быть превзойдены при использовании алгоритма кластеризации с функцией совместной энтропии в качестве расстояния между объектами.

Построенный на основе объединения идей генетической оптимизации и кластеризации алгоритм $ClassificatorSearch$ позволяет быстро находить удачные наборы признаков и строить классификаторы.

Список литературы

[1] Арлазаров В.Л., Куратов П.А., Славин О.А. Распознавание строк печатных текстов // Сб. трудов ИСА РАН «Методы и средства работы с документами». – М.: Эдиториал УРСС, 2000. – С. 31-51.

[2] Роджерс Д. Алгоритмические основы машинной графики. М.: Мир, 1989.

[3] Hinton G. E. Distributed representations. Technical Report CMU-CS-84-157, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1984.

[4] Дробков А.В., Семёнов А.Б. Исследование одного метода распознавания рукопечатных символов // Вестник Тверского государственного университета, серия «Прикладная математика». Тверь, 2009. №15. С. 15-26.

[5] Rojas R. Neural Networks. A Systematic Introduction. Springer-Verlag, Berlin, 1996.

[6] Хайкин С. Нейронные сети: полный курс, 2е издание. : Пер. с англ. М. Издательский дом «Вильямс», 2006.

[7] Панфилов С. А. Методы и программный комплекс моделирования алгоритмов управления нелинейными динамическими системами на основе мягких вычислений. Диссертация на соискание ученой степени кандидата технических наук. Тверь, 2005.

[8] Сорокин С.В., Литвинцева Л.В., Ульянов С.В. Технология интеллектуальных мягких вычислений в проектировании робастных нечётких систем управления: оптимизатор баз знаний // Нечёткие системы и мягкие вычисления. Том 3, №1, 2008. Тверь: ТвГУ.

[9] Седжвик Р. Фундаментальные алгоритмы на C++, часть 5. Алгоритмы на графах. М.: ДиаСофт ЮП, 2003.