Использование мультимножеств в распознавании символов

О. А. Славин[1]

Г-н Журден. Честное слово, я и не подозревал, что вот уже более сорока лет говорю прозой. Большое вам спасибо, что сказали.

Жан-Батист Мольер.
Мещанин во дворянстве

В статье приводятся примеры использования понятия «мультимножества» в задачах распознавания образов символов. Рассмотрены мультимножества оценок, представлений и классификаций.

Введение

Задача распознавания образов символов состоит в определении (построении) функции классификации, решающей, к каким классам $\{C_i\}_{i=1}^{M}$ принадлежит образ x**.** Функция классификации формирует ответ в виде набора *альтернатив* $A = (a_1, \ldots, a_m)$, $m < M$. Каждая альтернатива a_k представляет собой пару $<c_k, p_k>$, где c_k — *код класса*, p_k — *оценка* принадлежности к k-му классу. Альтернативы в векторе A отсортированы по убыванию оценок принадлежности. Необходимо отметить, что образ x может не соответствовать ни одному из M классов $\{C_i\}_{i=1}^{M}$, поэтому к числу классов добавляется ещё один класс Θ, называемый *отказом*.

Мы считаем, что серые (полутоновые) или черно-белые (бинарные) образы символов представлены растрами, т. е. матрицами $R(M, N)$ размера $M \times N$, элементы R_{ij} которых являются действительными или целыми числами. Функция классификации может оценивать близость к множеству классов

[1] 117312, Москва, В-312, проспект 60-летия Октября, 9, ИСА РАН, OSlavin@cs.isa.ru.

как по растрам непосредственно, так и с помощью *представления* образов символов. В качестве представлений могут выступать, например, наборы признаков, вычисленных для растра, или нормализованные растры.

В статье будут рассмотрены некоторые случаи применения понятий теории мультимножеств в задачах распознавания образов символов, позволяющие органично описывать структуры и алгоритмы распознавания образов символов.

1. Алгоритм 3 × 5

Распространенной процедурой, предшествующей распознаванию образа символа, является *нормализация образа* по различным параметрам, например, углу наклона, толщине линий [1] или форме образа [2]. Наиболее часто производят нормализацию по размерам или масштабирование. При этом решается ряд задач, таких как унификация эталонов распознавания и уменьшение размеров объекта распознавания. Последнее обстоятельство может снижать вариативность образов символов, за счет чего повышается качество распознавания на одних и тех же эталонных наборах. Часто перед распознаванием образы кириллицы или латиницы масштабируют до размеров 16 на 16 [3], используются и другие сетки, например, 20×20 и 28×28 [4]. Существуют работоспособные алгоритмы, базирующихся на масштабных сетках 3×5 и 5×3.

Дадим определение *сжатия* бинарного изображения произвольного размера к серому (полутоновому) образу меньшего размера. Пусть $B(m, n)$ — матрица исходного бинарного растра с размерами m на n. Сначала выполним преобразование матрицы B в матрицу $B^1(mM, nN)$ по следующему правилу

$$B^1_{pq} = B_{lk}, \quad \text{где} \quad m(l-1) < p \le ml \quad \text{и} \quad n(k-1) < q \le nk.$$

Затем определим матрицу $H(M, N)$ со следующими элементами

$$H_{ij} = \sum_{p=m(i-1)+1}^{mi} \sum_{q=n(j-1)+1}^{nj} B^1_{pq} .$$

Полученный сжатый растр мы будем использовать в качестве вектора, получаемого из матрицы $H(M, N)$ следующим образом

$$h = (H_{1,1}; H_{1,2}; \ldots H_{1,M}; H_{2,1}; H_{2,2}; \ldots H_{2,M}; \bullet\bullet\bullet H_{N,1}; H_{N,2}; \ldots H_{N,M}).$$

Нормализованный на единичную длину вектор h будем называть *сжатым образом* или *сверткой*.

В процессе обучения каждый образ из *обучающей последовательности* будем сжимать до определенного размера, а сжатые образы, рассмат-

риваемые как элементы одного эвклидова пространства, подвергнем алгоритму кластеризации.

Будем использовать понятие *алфавита обучения* [5], т. е. перечня классов, на которые разбита обучающая последовательность. Разнообразные с точки зрения начертания типы символов могут содержать несколько *графем*, т. е. типов начертания, соответствующих одному символу. Например, одному символу русского языка «Д» соответствует несколько графем, таких как ***Дд*ϑg**. Алфавит обучения также учитывает неспособность алгоритма различать графемы, соответствующие различным символам. Например, описанная нормализация образов символов по размеру делает невозможным различение прописных и строчных символов с одинаковым начертанием, таких как **Вв, Гг, Дд, Жж, Зз3**. То есть алфавит обучения как множество классов является носителем мультимножества всех допустимых графем. Степень элемента этого мультимножества есть количество графем, неразличимых с точки зрения алфавита обучения.

Каждое из подмножеств обучающей последовательности $R(G)$, соответствующих различным графемам G, кластеризуется отдельно. Образуется набор кластеров, каждый из которых формируется суммированием элементов однотипных растров. Центром кластера является среднее арифметическое растров, вошедших в кластер. Первый элемент подмножества $g_1 \in R(G)$ составляет начальный кластер Cl_1. Очередная свертка g_i сравнивается со всеми образованными кластерами, т. е. вычисляется наименьшее из расстояний $\rho(g_i , E(Cl_k))$ от свертки до центра каждого кластера Cl_k. Если расстояние от свертки g_i до множества центров всех кластеров Cl_1, Cl_2, ..., Cl_m больше заданного заранее порога, то образуется новый кластер Cl_{m+1}. Если же расстояние до какого-либо кластера меньше заданного порога, то свертка g_i суммируется с ближайшим кластером.

Тем самым к концу обучения мы вычислить центр $E(Cl_k)$ для каждого кластера, и отнормировав его на единичную длину, создать массив *эталонов* $B = \{E_1, E_2, \ldots, Em\}$. Некоторые из эталонов соответствуют одной и той же графеме, т. е. эталоны B представляют собой мультимножество, элементами которого являются идеальные свертки графем с кратностями, равными количеству получившихся для одной графемы кластеров.

Распознавание сжатого образа X, одной размерности с эталонами и нормированного на единичную длину состоит в следующем. Определим подмножество эталонов, расстояние до которых от X меньше заранее заданного ε. Соответствующие этим ближайшим эталонам пары, состоящие из кода графемы g_i и расстояния ρ_i до них, образуют мультимножество

$$A(\varepsilon) = \{(g_1, \rho_1), \ldots, (g_k, \rho_k)\}$$

альтернатив распознавания.

2. Событийный алгоритм

Рассмотрим представление образа, содержащего одну компоненту связности, эквивалентное растровому представлению, но отличное него. Бинарный растр, как набор строк, состоящих из черных и белых точек, представим в виде набора интервалов, каждый из которых содержит одну или несколько смежных черных точек. Процедура преобразования бинарного растра в набор интервалов состоит в его просмотре строк сверху вниз, каждая строка пробегается слева направо для обнаружения черных интервалов. Два интервала в смежных строках, имеющие пересекающиеся горизонтальные проекции, называются *сцепленными*. Набор сцепленных интервалов называется *линией*. Таким образом, линия в каждом горизонтальном сечении содержит ровно один интервал. В процессе просмотра возможны следующие ситуации

- Черный интервал не имеет соседей в предыдущей строке растра (соседними считаются точки, граничащие по одному из восьми направлений). Соответствует началу линии со свободным началом (интервалы 1, 2 на рис. 1).
- Черный интервал не граничит ни с одним интервалом из следующей строки. Линия закончилась, и этот конец — свободный (интервалы 3, 4 на рис. 1).

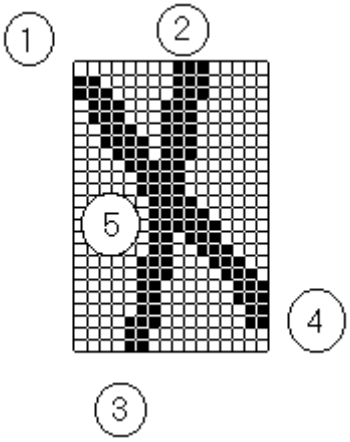

Рис. 1. Пример свободных начал (1, 2), свободных концов (3, 4) и линий с несвободными началами и концами (5)

- Черный интервал граничит с черным интервалом в предыдущей строке, и этот интервал первый из всех интервалов, граничащих с интервалом

предыдущей строки. Интервал является продолжением линии (интервалы, продолжающие линии, заданные свободными концами 1, 2 на рис. 1).

- Существует несколько интервалов, сцепленных с одним интервалом из предыдущей строки. Первый (самый левый) из них относится к продолжению линии, а остальные являются началами новых линий с несвободными началами (пересечение линий 5 на рис. 1).
- У двух или более линий имеется общий соседний интервал в следующей строке. Первая (самая левая) линия продолжается, а остальные имеют несвободные концы (пересечение линий 5 на рис. 1).

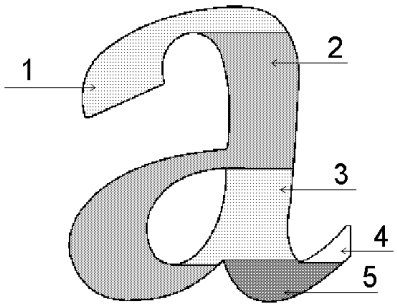

Рис. 2. Пример разбиения образа символа «а» на линии

Данная процедура выделения интервалов и линий позволяет получить *линейное представление* односвязного образа символа (пример разбиения растра символа на линии см. на рис. 2), состоящее из набора линий:

$$L_1 = \{B_1, E_1, M_1, I^1_1, \ldots, I^1_{N1}\},$$
$$L_2 = \{B_2, E_2, M_2, I^2_1, \ldots, I^2_{N2}\},$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$L_k = \{B_k, E_k, M_k, I^k_1, \ldots, I^k_{Nk}\},$$

где k — число линий в растре,

B_i, E_i — признак наличия свободного начала и свободного конца в i-й линии,

M_i — номер строки в растре первого интервала i-й линии,

I^i_j — j-й интервал i-й линии, состоящий из координат начала и конца.

Огрубим линейное представление следующим образом. Для каждой линии L_i определим ее начало S_i и конец F_i как координаты середин первого и последнего из интервалов линии, после чего отмасштабируем коор-

динаты начала и конца на сетке $M \times N$. Отмасштабированную таким образом линию назовем *прямым событием*.

Последовательность

$$S = \{N_L, N_B, N_E, M_1, F_1, \ldots, M_N, F_N\},$$

содержащую N_L линий и информацию о структуре символа, т. е. количество свободных начал и концов N_B и N_E, назовем *линейным представлением символа*. Линейное представление сохраняет описание структуры символа, и, в то же время, разнообразие линий ограничивается размером использованной сетки. Повернув образ на 90° по часовой стрелке и проделав процедуры выделения линий и определения событий с повернутым растром, получим набор событий

$$S_r = \{N_{Lr}, N_{Br}, N_{Er}, M_1, F_1, \ldots, M_{Nr}, F_{Nr}\},$$

называемый *поворотным линейным представлением символа*. Из построенных представлений удалим линии с размерами менее заданной величины. Эта процедура еще более огрубляет представление, пренебрегая малыми выбросами в образе. Для игнорирования аналогичных вариаций образа из-за малых отверстий необходима процедура удаления отверстий.

Процедура обучения начинается с преобразования каждого растра $\mathbf{r_i}$ с графемой g_i из обучающей последовательности в линейное представление $S(r_i)$. Полученный вектор $S(r_i)$ сравнивается со всеми существующими эталонными линейными представлениями B_E с тем же количеством компонент вектора. Если вектор $S(r_i)$ отсутствует в B_E, то он добавляется как новый эталон с кодом графемы g_i. Если же вектор $S(r_i)$ совпал с одним из элементов B_E, то графема g_i суммируется с мультимножеством графем одного линейного представления

$$\{(g_1, N_1), \ldots, (g_k, N_k)\},$$

где N_i — количество графем g_i в обучающей последовательности, обладающих линейным представлением $S(r_i)$.

Распознавание растра r_i, преобразованного в линейное представление $S(r_i)$, состоит в точном поиске вектора $S(r_i)$ в массиве эталонов. Результатом распознавания является мультимножество графем

$$A = \{(g_1, N_1), \ldots, (g_k, N_k)\},$$

где N_i — количество графем g_i в обучающей последовательности, обладающих линейным представлением $S(r_i)$.

Наличие прямых и поворотных массивов эталонов позволяет получить два мультимножества A и A_r результатов. В качестве окончательного результата могут быть взяты как сумма $A + A_r$, так и разность $A - A_r$.

Обсуждение

Приведенные примеры свидетельствуют о том, что понятия теории мультимножеств органично используются в практике алгоритмов распознавания символов. Может возникнуть вопрос: «а что же дает для разработки алгоритмов распознавания символов теория мультимножеств?»

Во-первых, без теории мультимножеств приходится определять собственные понятия, аналогичные мультимножественным. Например, в описанном событийном алгоритме невозможно обойтись без операций суммы и разности. В свою очередь, использование собственной системы понятий затрудняет взаимодействие разработчиков алгоритмов, а возможная неполнота системы понятий усложняет и саму разработку.

Во-вторых, представление набора классов мультимножеством снимает вопросы, связанные с неоднозначными описаниями символов. В распознавании образов символов кириллицы и латиницы мультимножеством является набор классов $\{C_i\}_{i=1}^{M}$, в которых символы имеют несколько модификаций (признаки курсива, полужирность, тип шрифта), причем некоторые из них могут не иметь содержательного названия (шрифтовые модификации). При этом определяются несколько классов отказа Θk, соответствующих невозможности классификации по признаку модификации.

Наконец, методическая ориентация на мультимножества полезна и в других алгоритмах распознавания, в которых неоднозначность является существенной. К таким алгоритмам относятся задачи распознавания символов с неизвестными границами, распознавание текстовых строк с неизвестным алфавитом, распознавание выражений по описанию, сегментация образа документа.

Благодарности

Автор выражает благодарность большому коллективу разработчиков описанных алгоритмов: В. Л. Арлазарову, А. В. Карзанову, А. Б. Талалаю, Э. А. Комиссарчику, А. Тхиру, А. Д. Астахову, А. А. Подрабиновичу, А. Я. Подрабиновичу.

Литература

1. *Lam L., Suen C. Y.* An Evaluation of Parallel Thinning Algorithms for Character Recognition // IEEE Transactions on Pattern Analysis and Machine Intelligence. 1995. V. 17. № 9. P. 914–919.
2. *Wakahaga T., Odaka K.* Adaptive Normalization of Handwritten Characters Using Global/Local Affine Transformation // IEEE Transactions on Pattern Analysis and Machine Intelligence. 1998. Vol. 20. № 12. P. 28–33.
3. *Мисюрёв А. В.* Использование искусственных нейронных сетей для распознавания рукопечатных символов // Интеллектуальные технологии ввода и обработки информации. 1998. С. 122–127.
4. *Portegys T. E.* A Search Technique for pattern Recognition Using Relative Distances // IEEE Transactions on Pattern Analysis and Machine Intelligence. 1995. Vol. 17. № 9. P. 910–912.
5. *Арлазаров В. Л., Логинов А. С., Славин О. А.* Характеристики программ оптического распознавания текста // Программирование. 2002. № 3. С. 45–63.