AITS

Letter recognisation using Machine Learning

Hanish chidpothu

www.aitechsystems.com

AI –Tech Systems

Ai-techsystems.com

Chennai,India

Chhanish.16@gmail.com

Abstract— **This report gives a information on classifier system to a complex letter recognition task with 20 different fonts using Machine learning. A set of 20,000 unique letter images was generated by randomly by distorting pixel images of the 26 uppercase letters from 20 different commercial fonts. The parent fonts represented a full range of character types including script, italic, serif, and Gothic. The features of each of the 20,000 characters were summarized in terms of 16 primitive numerical attributes.**

Keywords- Category learning, Font recognisations, and classifier systems.

I. INTRODUCTION

My report is based on machine induction techniques.

The first component consists primarily of categorization. The second component involves associating one or more action sequences with each of the categories. The solution has a large repertoire of well-practiced action routines that can be selected and applied in a way that is appropriate for the initial categorization decision. My research focuses on the first component of the above paradigm. The computer system that induces general categorization rules within a supervised learning paradigm. A large number of unique examples are presented to the system along with an outcome measure that indicates the appropriate category for each example. Our test implementation involves 26 categories and 20,000 unique test patterns to find out the correct letter from all the given combinations.

II. CHARACTERISTICS AND ATTRIBUTES

To test the methodology,to identify each of a large number of black-and-white rectangular pixel displayed from one of the 26 letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 numerical attributes that were in turn submitted to our classifier system.

A. Generation:

Generation of each item in the character file was generated in the following manner. Twenty thousand calls were made to a character-image generating program with random uniformly distributed parameter values for font type, letter of the alphabet, linear magnification, aspect ratio and horizontal and vertical "warp." Each character image was first produced in the form of vector coordinates of the end-points of its constituent line segments. The specified scale changes and "warping" were applied to these coordinates. The line segments were then "rasterized" to form a rectangular array of pixels, each of which was "on" or "off. "on" pixels represented the image of the desired character. These arrays averaged about 45 pixels high by 45 pixels wide is generated.

B.Recognisation:

In the process of recognisation the linear magnification ranged from 1.0 to 1.6. The additional horizontal magnification, which changed the aspect ratio, ranged from 1.0 to 1.5. The horizontal warp parameter controlled a quadratic transformation of the horizontal coordinates that distorted the horizontal scale by stretching either the left or right region of the image (and correspondingly shrinking the other region). The vertical warp parameter operated similarly in the vertical direction. The range of the warp parameters was chosen so that even when their values were at the limits of their range, the resulting character images, although rather misshapen, were fairly recognizable to humans eye.

III. PROCEDURE:

1. In the horizontal position,the counting pixels from the left edge of the image, of the center of the smallest rectangular box that can be drawn with all "on" pixels inside the box.

2. In the vertical position, counting pixels from the bottom of the above box.

3. The width, in pixels, of the box.

4. The height, in pixels, of the box.

5. The total number of "on" pixels in the character image.

6. The mean horizontal position of all "on" pixels relative to the center of the box and divided by the width of the box. This feature has a negative value if the image is "left- heavy" as would be the case for the letter L.

7. The mean vertical position of all "on" pixels relative to the center of the box and divided by the height of the box.

8. The mean squared value of the horizontal pixel distances as measured in 6 above. This attribute will have a higher value for images whose pixels are more widely separated in the horizontal direction as would be the case for the letters W or M.

9. The mean squared value of the vertical pixel distances as measured in 7 above.

10. The mean product of the horizontal and vertical distances for each "on" pixel as measured in 6 and 7 above. This attribute has a positive value for diagonal lines that run from bottom left to top right and a negative value for diagonal lines from top left to bottom right.

11. The mean value of the squared horizontal distance times the vertical distance for each "on" pixel. This measures the correlation of the horizontal variance with the vertical position.

12. The mean value of the squared vertical distance times the horizontal distance for each "on" pixel. This measures the correlation of the vertical variance with the horizontal position.

13. The mean number of edges (an "on" pixel immediately to the right of either an "off" pixel or the image boundary) encountered when making systematic scans from left to right at all vertical positions within the box. This measure distinguishes between letters like "W" or "M" and letters like 'T' or "L."

14. The sum of the vertical positions of edges encountered as measured in 13 above. This feature will give a higher value if there are more edges at the top of the box, as in the letter "Y."

15. The mean number of edges (an "on" pixel immediately above either an "off" pixel or the image boundary) encountered when making systematic scans of the image from bottom to top over all horizontal positions within the box.

16. The sum of horizontal positions of edges encountered as measured in 15 above.

A. Learning paradigm:

The set of 20,000 unique letter images was organized into two files. Sixteen thousand items were used as a learning set and the remaining 4000 items were used for testing the accuracy of the rules. The program traversed the 16,000 learning items 5 separate times, creating new rules, discarding unsatisfactory ones, and modifying the performance statistics for each rule as appropriate. This process provided 80,000 learning trials. During our preliminary investigations, we explored various numbers of passes through the training file, ranging from 1 to 20. With the better methods, most of the benefits of training (approximately 80%) occurred during the first pass through the 16,000 item data set. We selected 5 passes as our standard procedure, because this seemed to provide a reasonable approximation of the asymptotic improvement level that could be expected from training.

B. Description of classifier systems:

The processing sequence can be characterized as:

1. Compare the attribute vector of a test item with the attribute specifications of each of the classifiers in the current rule buffer.

2. Select a match set [M] consisting of all classifiers whose conditions are satisfied by the test item's attribute vector.

3. Compute a bid for each classifier in set M. Assign the category associated with the highest bidder as the system output.

4. If in learning phase, modify the performance statistics of one or more classifiers as specified by the bidding system algorithm.

5. If in learning phase, discard weak rules and create new rules according to the rule creation algorithm.
6. Select the next test item, and repeat the process starting at step 1.

C. Percent correct identifications:

Table 1: Percent correct identifications on 4000-item test set with integer attribute representation and fuzzy matching.

Method of rule creation	NEWST	Strength	Str*spec	strength	Str*spec
				Reward sharing winner	
Random	1000	49.5	51.0	24.5	30.7
	2000	52.6	49.0	25.6	30.4
	4000	47.9	45.7	24.6	31.5
	8000	40.2	43.9	30.0	28.4
Hybrid	1000	62.6	67.1	30.4	30.6
	2000	68.8	68.5	37.7	32.6
	4000	69.4	70.4	36.6	34.3
	8000	65.2	67.8	32.0	30.4
Exempler	1000	70.4	69.7	53.4	54.5
	2000	74.8	76.7	58.8	60.3
	4000	78.3	77.4	65.0	64.2
	8000	80.8	80.3	66.0	67.8

IV. ADAPTIVE CLASSIFIERS

Method of rule creation	NEWST	Strength	Str*spc	Strength	Str*spc
				Reward sharing	
	1000	324,432	309,202	337,126	325,151
	2000	165,804	156,098	173,764	167,151
	4000	85,266	79,870	90,411	87,000
	8000	44,925	41,622	47,711	45,666
Hybrid	1000	311,315	301,819	351,852	341,952
	2000	156,004	150,308	187,503	175,722
	4000	77,426	76,003	98,761	95,452
	8000	49,618	39,462	52,601	50,591
Exempler	1000	49,742	47,896	64,564	60,300
	2000	42,882	41,584	57,620	53,178
	4000	39,316	36,642	52,198	47,268
	8000	36,428	33,184	50,006	43,990

A. Window size:

In all of the conditions discussed so far, classifiers and test items were based on an integer attribute representation and a fixed window size of 1 was employed for "fuzzy" matching. If the attribute value of a test item was within 1 unit of the rule value, the rule and test item matched on that attribute. In our preliminary analyses, various window sizes were explored to determine which setting would produce the best performance. Results are presented in above table comparing window sizes of 0, 1, and 2. In each case, rule creation is exemplar based, bidding is by strength, reward is shared, the amount of reward is set at 2 times NEWST, and the tax is set at 1. The wild card probability for rule creation was adjusted for each window size: $p = 0.7$ for window size 0, $p = 0.5$ for window size 1, and $p = 0.3$ for window size 2. These values were selected because they optimized performance for each procedure.

Dependent Measure	NEWST	0	1	2
		Window size		
Correct identifications on 4000-Test set	1000	53.3	70.4	59.4
	2000	62.3	74.8	61.9
	4000	67.1	78.3	66.2
	8000	69.4	80.8	67.3
Mean rule specificity	1000	3.05	6.65	10.70

	2000	3.24	6.97	11.13
	4000	3.54	7.50	11.50
	8000	3.89	8.02	11.77
Number of rules after Verification	1000	210	242	191
	2000	392	434	314
	4000	860	747	449
	8000	1872	1302	681

B. Binary and grey code attribute distributions

All of the previous results represented attribute values as integers and employed a window of fixed size to determine when matching occurred. Traditional classifier systems more commonly employ binary or gray code attribute representations for numerical attributes. Table 6 compares the performance of the classifier systems while varying the method of attribute representation. Binary and gray code methods are compared directly to the integer method. In each case, rule creation is exemplar-based, bidding is by strength, reward is shared, the amount of reward is set at 2 times NEWST, and the tax is set at 1. The wild card probability in rule creation was adjusted for each method: $p = .65$ for binary and gray code and $p = .5$ for integer representation. These values were selected to optimize performance.

	Type of attribute representation			
Dependent measure	NEWST	Binary	Grey code	Integer
%Correct Identification On 4000-Item test set	1000	43.2	45.9	70.4
	2000	48.0	52.4	74.8
	4000	52.4	56.5	78.3
	8000	54.7	59.3	80.0
Mean specificity	1000	22.0	21.34	6.65
	2000	21.54	1.22	6.97
	4000	21.52	20.80	7.50
	8000	21.37	21.06	8.02
Number of rules after Verification	1000	124	116	242
	2000	233	251	434
	4000	519	588	747
	8000	1600	1441	1302

without fuzzy matching, the integer method out-performs the binary and gray code methods on the character recognition.Though we did not examine partial matching under the binary and gray code conditions. These methods may have been more effective with partial matching.

C. Accuracy-utility content system:

The concepts of rule strength and rule specificity have played an important role in the development of the classifier system. In the current research, alternative measures of rule fitness were examined in an effort to address problems encountered with this application. With some of our parameter selections, stable performance was observed for a period of time and then an abrupt deterioration occurred. Useful default hierarchies, which had been developed gradually, appeared to lose one or two critical rules, seemingly for complex reasons. These losses triggered a major calamity in which other rules, no longer protected from over-generalizing their knowledge, began to make many errors. Within a short time, a large number of previously successful rules had been discarded and the performance of the system dropped significantly. We believe that this problem resulted from selecting the wrong set of parameters for the bidding and reward system. In essence, the strength-specificity bidding system seems to be very sensitive to parameter values and can perform quite poorly if these values are not set within a precise range alternative to reward allocation system.

EXEMPLER HYBRID ROUTE CREATION

The prior results indicate that hybrid procedures for creating rules by altering or combining existing strong rules have beneficial results. A limitation of this procedure appears to be the quality of the rules created initially by a random process. To explore this relation- ship more thoroughly, we examined a rule creation process in which the hybrid methods were used to augment the exemplar-based method. All observations were taken using integer attribute representation with the window size set at 1 and a strength based bidding system with reward shared among the correct bidders. In all cases, the tax value was set

at 1 and the reward value at 2 times NEWST. Bids were based on strength only, and the bid cost was set at 10% of the current strength.

System	Utility criterion	%correct	Specificity	Final rules	Rules created
Int.Eg.Str	1000	60.0	5.94	129	723,219
Window=1	2000	71.4	61.0	250	374,096
Wild card=0.5	4000	79.0	6.72	537	184,721
AWCT=yes	8000	82.7	7.50	1190	101,969

CONCLUSION:

Thus, an letter recognisation program for 20 different font styles is developed using the method of machine learning according to the data set given above.

□