Machine Learning, 6, 161-182 (1991)

Letter Recognition Using Holland-Style Adaptive Classifiers

PETER W. FREY
Department of Psychology, Northwestern University, Evanston, IL 60208

DAVID J. SLATE
Pattern Recognition Group, Odesta Corporation, 1890 Maple Avenue, Evanston, IL 60201

Editor: Tom Dietterich

Abstract. Machine rule induction was examined on a difficult categorization problem by applying a Holland-style classifier system to a complex letter recognition task. A set of 20,000 unique letter images was generated by randomly 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. Our research focused on machine induction techniques for generating IF-THEN classifiers in which the IF part was a list of values for each of the 16 attributes and the THEN part was the correct category, i.e., one of the 26 letters of the alphabet. We examined the effects of different procedures for encoding attributes, deriving new rules, and apportioning credit among the rules. Binary and Gray-code attribute encodings that required exact matches for rule activation were compared with integer representations that employed fuzzy matching for rule activation. Random and genetic methods for rule creation were compared with instance-based generalization. The strength/specificity method for credit apportionment was compared with a procedure we call "accuracy/utility."

Keywords. Category learning, parallel rule-based systems, exemplar-based induction, apportionment of credit, fuzzy-match rule activation.

Human experts often solve difficult problems quickly and effortlessly by categorizing complex situations as special cases of familiar paradigms and applying solution strategies that are known to be effective for these paradigms (de Groot, 1965; Chase & Simon, 1973). Problem solving in this context involves partitioning a complex task into two components that can be solved independently and executed in a serial fashion. The first component consists primarily of categorization. Humans acquire this ability after many years of observing a wide-range of related examples. The expert's skill seems to be based primarily on memory for past experiences rather than on logical deduction or symbolic reasoning (Charness, 1981). The second component involves associating one or more action sequences with each of the categories. The problem solver 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.

Our research focuses on the first component of the above paradigm. We examine a 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.

In this article we describe an adaptive classifier system similar in many respects to the approach pioneered by Holland (1975, 1976, 1980, 1986). These systems create a list of fixed-length condition-action rules (i.e., classifers) that are applied in parallel to "messages" representing the presence or absence of specific features in the current environment. There are typically three major parts: a performance algorithm that compares rules with messages to determine which rule(s) should be activated, a reinforcement algorithm that modifies the strength of each rule on the basis of its "fitness" in the current environment, and a rule-creation algorithm that generalizes exemplars or combines current rules to produce new ones. In a supervised learning categorization task, the effectiveness of each rule can be determined directly, because there is immediate feedback on each training trial. Therefore, it is not necessary to use Holland's bucket brigade, which is normally employed to address the complex credit allocation problem that arises when there is intermittent feedback. The simplification of the "standard" Holland approach used for this application is very similar to the modifications made by others for studies on category learning (Wilson, 1985, 1987, 1988; Davis & Young, 1988).

1. Characters and attributes

To test our methodology, we selected a difficult classification task. The objective was to identify each of a large number of black-and-white rectangular pixel displays as 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. The identification task was especially challenging because of the wide diversity among the different fonts and because of the primitive nature of the attributes.

The 20 different fonts were designed by Dr. Allen V. Hershey, a mathematical physicist at the U.S. Naval Weapons Laboratory. Our local computer center obtained the font set from the National Bureau of Standards. In our research, we employed the following Roman alphabet fonts: HASTR, HCART, HCITA, HCROM, HCSCR, HDROM, HGENG, HGGER, HGITA, HIITA, HIROM, HISYM, HIUMT, HMETE, HMUSI, HSROM, HSSCR, HTITA, HTROM, and HUMAT. The fonts represent five different stroke styles (simplex, duplex, triplex, complex, and Gothic) and six different letter styles (block, script, italic, English, Italian, and German).

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." The

"on" pixels represented the image of the desired character. These arrays averaged about 45 pixels high by 45 pixels wide.

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.

Examples of the character images generated by these procedures are presented in Figure 1. Each character image was then scanned, pixel by pixel, to extract 16 numerical attributes. These attributes represent primitive statistical features of the pixel distribution. To achieve compactness, each attribute was then scaled linearly to a range of integer values from 0 to 15. This final set of values was adequate to provide a perfect separation of the 26 classes. That is, no feature vector mapped to more than one class. The attributes (before scaling to 0–15 range) are:

1. The horizontal position, 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. 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

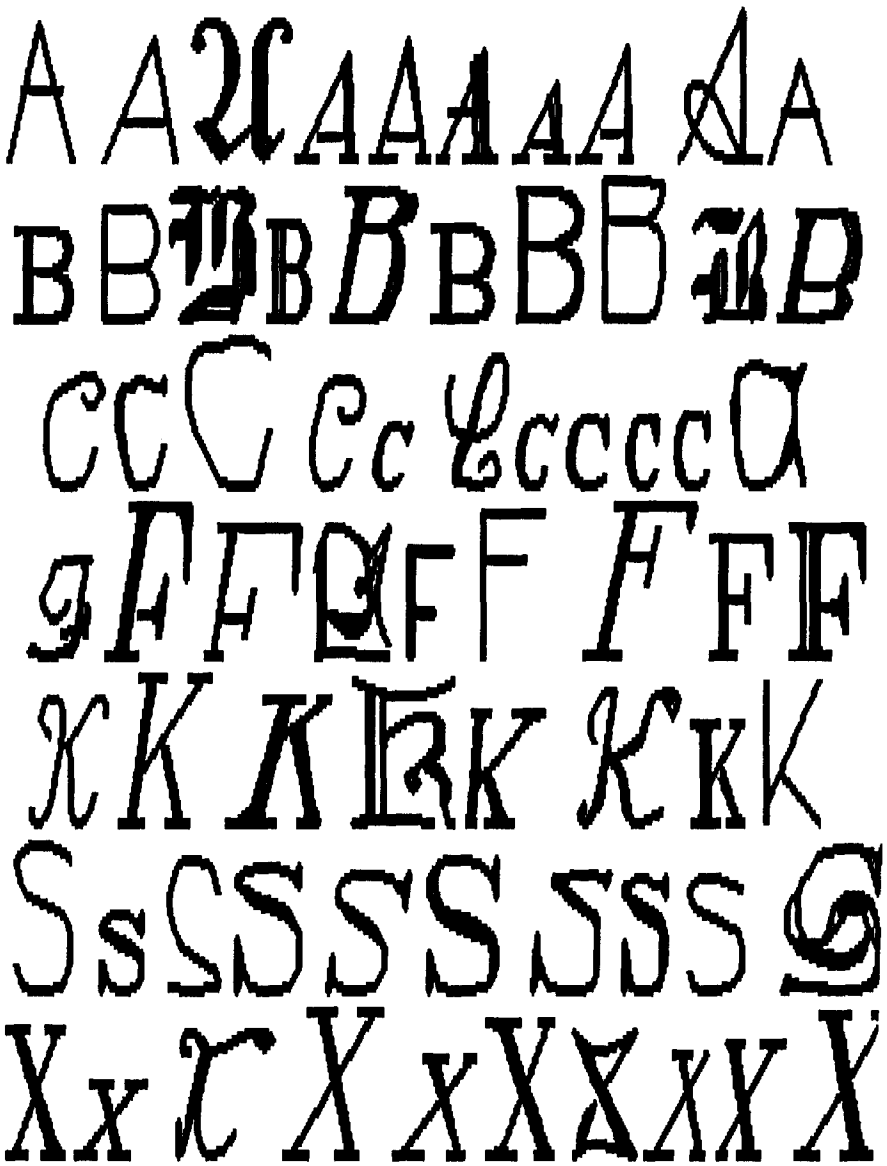

Figure 1. Examples of the character images generated by "warping" parameters.

to right at all vertical positions within the box. This measure distinguishes between letters like "W" or "M" and letters like "I" 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 data file of the 16 attribute values and outcome category for each of the 20,000 stimulus items is on file with David Aha (aha@ics.uci.edu).

2. 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.

When this phase was complete, a sixth pass was performed in which no new rules were created. Rules that did not maintain pre-specified performance levels were discarded. This "verification" process greatly reduced the number of rules. Each classifier in the final group was an important contributor to the overall performance of the system.

The last step in the process consisted of applying the final rule set to the 4000 test items to determine the overall accuracy of the system. The rule set was "frozen" during this test phase in that no rule creations, rule deletions, or strength changes took place. The primary performance measure for evaluating the different algorithms was the percentage correct on the 4000 test items. In addition, measures are reported for the total number of rules created during training, the final number of rules after verification, and the mean specificity of the rules after verification.

3. Description of the classifier systems

Our procedure is based on a typical Holland classifer system that has been modified for one-step categorization problems (i.e., no bucket brigade). The processing sequence can be summarized as follows:

1. Compare the attribute vector of a test item with the attribute specifications of each of the classifers in the current rule buffer.
2. Select a match set [M] consisting of all classifers whose conditions are satisfied by the test item's attribute vector.
3. Compute a bid for each classifer 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.

The primary focus of our research was to examine three different attribute coding procedures (BIN, GRA, INT), three different rule creation procedures (RAN, GEN, EXM), and different bidding systems. In addition, the criterion for rule retention was systematically varied for each algorithm to determine how well each method performed under different resource constraints. This manipulation influences the size of the rule buffer and indirectly determines what level of rule specificity is most suitable.

4. Attribute coding

Binary coding (BIN) represents the value of each of the 16 attributes as a 4-bit binary number. Each data item (message) is thus a string of 64 (4×16) bits. Similarly, the left-hand side of each classifier is a 64-element string, where each position is either a wild card, a 0, or a 1. A classifier matches a test item when all of its non-wild card positions are identical to the test item.

Gray coding (GRA) is similar to binary coding, but the mapping of attribute values to 4-bit codes is different. We employed the procedure described by Goldberg (1989, p. 100). Number representation with Gray-codes insures that adjacent values are identical except for a change in a single bit.

Integer coding (INT) represents each of the 16 attributes as an integer. Each data item (message) is a 16-position vector in which each position is an integer in the range 0 to 15. The left-hand side of each classifier consists of a 16-position vector in which each position is either a wild card or a target value for the attribute. With integer coding, we also specify a window size for fuzzy matches. A uniform window size is employed for all attributes. If the window size is set at zero, an exact match is required on all non-wild card attributes. If the window size is set at 1, a classifier matches a test item if the target values for all non-wild card attributes are either identical to the value for the test item or are within 1 unit of that value. If the window size is set at 2, a classifier matches if the target values for all non-wild card attributes are within 2 units of the test item values. Therefore, window size determines the degree of fuzziness tolerated in defining a match. Booker (1988, p. 182) describes another method for implementing partial matches. In his system, rules that match on most, but not all, of the non-wild card attributes can be eligible for the auction.

5. Rule creation

Holland et al. (1986) describe a variety of methods for creating new rules, including random generation, two forms of genetic mutation (generalization and specialization), genetic crossover, and instance-based generalization. In our research, we examined three of these

methods: random generation, exemplar-based generation, and a hybrid procedure that started with random rules and evolved offspring from the fittest parents using single-point genetic crossovers and specializing mutations.

In the random procedure (RAN), 3900 rules, 150 for each of the 26 categories, were produced prior to the training phase. Each classifier was created by randomly assigning wild card status or a uniformly distributed random value to each position in the attribute vector and randomly assigning an outcome value (one of the 26 categories). Whenever a rule failed to meet a pre-specified performance level, it was immediately discarded and replaced by a newly created rule. Successful rules were retained indefinitely by the system.

In the hybrid procedure (HYB), 3900 rules were initially created as described above. When a rule dropped below the minimum performance level, it was immediately replaced. On each trial, the replacement process could invoke one of three separate rule creation methods. One method was the RAN procedure described above. A second method for creating a new rule (MUT), involved selecting a strong rule, randomly selecting one of the attributes, and randomly assigning a numerical value between 0 and 15 to that attribute. This procedure had the effect of either maintaining the current rule specificity (in cases where the selected attribute was already relevant) or of increasing specificity (in cases where the selected attribute had previously been a wild card). This property was intentional. The rules that were generated by the random procedure needed low specificities in order to have even a weak chance of matching test items. The mutation procedure was designed to produce an increase in rule specificity in order to enhance the performance of the system.

The third method for rule creation (CROSS) selected two strong rules, randomly selected a single splitting location, and created a new rule by combining components from the left and right side of the split. All crossovers were one-point crossovers. To act as a parent in methods MUT and CROSS, a rule had to have the same outcome as the rule that was being replaced, and its strength had to be higher than the strength it was initially assigned. Rules that qualified by these criteria were selected for parenthood randomly in proportion to their fitness scores.

Each time a rule was created by the hybrid procedure (HYB), one of the three methods was chosen randomly in the proportion 20% RAN, 60% MUT, and 20% CROSS. When MUT or CROSS was selected, but no rules qualified for parenthood, method RAN was employed. When CROSS was selected, but only one rule qualified for parenthood, method MUT was employed. The proportion 20-60-20 was chosen, because it produced better results than 20-20-60, 20-30-50, 20-40-40, or 20-50-30. The number of rules for each category remained at 150 throughout training.

In the exemplar-based generalization procedure (EXM), the system started with an empty rule buffer and created two new rules each time a misidentification was made by the system or when there were no rules which qualified for the auction. Each of the two new rules was created by randomly assigning wild card ($p = .5$) or non-wild card status to each of the attribute positions of the training item. The non-wild card positions retained the values of the misidentified training item. The category assigned for the new rule was the category of the training item. This procedure is similar to Wilson's (1985, p. 19) "create operator." It differs in that it is triggered not only when no rule matches but also when the winning rule identifies the wrong category.

An option for each of these procedures is a feature called "attribute wild card tuning" (AWCT), which dynamically adjusts the wild card probability separately for each attribute. The new probability is computed as the weighted mean frequency of occurrence of a wild card in that position over all existing classifiers, but with each classifier weighted according to its fitness. This increases the chance that less informative attributes will be assigned wild-card status. We employed attribute wild-card tuning in all of the tests that follow, including the BIN and GRA runs in which there were 64 positions for tuning.

6. Strength-based bidding systems

We examined several variations of the more common methods of bidding and apportioning credit. For all of these methods, an initial strength value (NEWST) was assigned to each new rule, all rules were taxed each time an item was presented, and each rule that qualified for the auction paid a fixed proportion of its current strength for the privilege of attending. Two methods of apportioning credit to bidders were examined: winner-take-all and sharing in proportion to one's bid. Two methods of bidding were also examined: proportional to strength and proportional to the product of strength and specificity. The algorithms examined in this research were based on the approach described by Wilson (1988). The basic algorithm for the SS approach was:

1. A match set [M] is selected consisting of all classifiers whose conditions are satisfied by the input string.
2. Each classifier in [M] makes a bid equal to its strength (condition STR) or equal to the product of its strength and specificity (condition SS). Specificity is defined as the number of non-wild card attributes.
3. The system makes its decision by selecting the classifier from [M] that has the highest bid and outputs the selected classifier's category as the system's decision.
4. A fixed reward value (200% of NEWST) is given to the highest bidder if it advocated the correct category (condition WTA) or is shared among all classifiers in [M] that advocated the correct category (condition SHR). Each reward share is proportional to the size of the classifier's bid. If none of the classifiers in [M] advocate the correct category, no reward is given.
5. Every classifier in [M] has its strength reduced by a fixed proportion of its bid. This proportion was 10% in condition STR and 1.5% in condition SS. These values were selected to maximize performance under each of the two conditions.
6. Each classifier is taxed 1 unit of strength for each item that is presented. If the classifier's strength drops below 1, the classifier is immediately removed from the rule buffer. In the random and hybrid conditions, the rule is replaced by a newly created classifier that represents the same category.

For each algorithm using these bidding systems, performance was examined under 4 different values of NEWST. These values were 1000, 2000, 4000, and 8000. These values determine the length of time a rule can stay in the system without receiving all or part

of the reward. If NEWST is initially set at 1000, a rule that receives no reward is discarded after 1000 training items because of taxation. If NEWST is initially set at 8000, the rule will stay in the system much longer before it is discarded for erroneous identifications or inactivity. With the higher settings for NEWST, the system can retain rules of higher specificity that contribute to the system's performance less frequently.

A simple factorial design was performed to examine the relative effectiveness of reward sharing versus winner-take-all, bidding on the basis of strength versus the product of strength and specificity, and rule creation by random, hybrid, or exemplar methods. In all of the conditions reported in the initial results (Tables 1, 2, 3, and 4), attribute values were represented as integers (INT) and the window size for defining a match was set at 1. The primary dependent measure for these analyses was the number of items from the 4000-item test set that were correctly identified after 5 learning passes and 1 verification pass through the 16,000-item training set.

The results presented in Table 1 indicate the accuracy of identifying items in the test set under these various conditions. Sharing reward among all the bidders which advocated the correct outcome was clearly superior to the winner-take-all procedure. This is consistent with recent reports in the literature (e.g., Goldberg, 1989, p. 225; Wilson, 1988, p. 707), which advocated sharing reward. In the current application, reward sharing consistently produced more than a 50% improvement over the performance observed with the winner-take-all procedure.

The comparison between the two bidding procedures produced less clear results. Very similar performance levels were achieved with bidding based on strength alone or based on the product of strength and specificity. Holland (1986) advocates factoring specificity into the bid. Wilson (1988) presents evidence suggesting that performance may be superior when specificity does not influence bidding or reward distribution. Our results indicated that the two approaches produced equally accurate identifications of the test items.

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

		Reward Sharing		Winner-Take-All	
Method of Rule Creation	NEWST	Strength	Str*Spec	Strength	Str*Spec
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
Exemplar	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

Table 2. Mean rule specificity after verification with integer attribute representation and fuzzy matching.

		Reward Sharing		Winner-Take-All	
Method of Rule Creation	NEWST	Strength	Str*Spec	Strength	Str*Spec
Random	1000	2.83	2.49	2.95	3.30
	2000	2.92	2.69	3.03	3.23
	4000	3.00	2.74	3.20	3.18
	8000	3.03	2.83	3.13	3.11
Hybrid	1000	3.99	3.58	3.13	3.45
	2000	3.98	3.50	3.48	3.73
	4000	4.61	3.67	3.91	3.67
	8000	4.53	3.73	3.77	3.40
Exemplar	1000	6.65	6.48	7.44	7.63
	2000	6.97	6.90	7.90	8.36
	4000	7.50	7.79	8.07	8.86
	8000	8.02	8.23	8.41	9.01

Table 3. Number of rules after verification with integer attribute representation and fuzzy matching.

		Reward Sharing		Winner-Take-All	
Method of Rule Creation	NEWST	Strength	Str*Spec	Strength	Str*Spec
Random	1000	107	136	38	64
	2000	183	216	64	70
	4000	266	333	88	96
	8000	335	437	108	119
Hybrid	1000	93	122	61	62
	2000	177	224	93	111
	4000	371	360	128	116
	8000	651	610	159	126
Exemplar	1000	242	236	187	197
	2000	434	435	345	379
	4000	747	772	550	667
	8000	1302	1313	878	990

The different methods for rule generation did have noticeable effects on performance. The exemplar-based procedure (EXM) produced higher accuracy scores than the other two methods. The hybrid method (HYB) was also clearly superior to the random method (RAN). EXM was also notable in that it took much greater advantage of the additional resources that were available when NEWST was set at the higher values (e.g., 4000, 8000). Its superior performance over HYB and RAN was accentuated at the higher NEWST settings. When valid rules are highly complex, RAN and HYB may not generate good candidates at a high enough frequency to take advantage of the larger rule buffer. The superior performance observed for EXM in this application may be a general characteristic of supervised learning

Table 4. Number of rules created during training with integer attribute representation and fuzzy matching.

		Reward Sharing		Winner-Take-All	
Method of Rule Creation	NEWST	Strength	Str*Spec	Strength	Str*Spec
Random	1000	324,432	309,202	337,126	325,151
	2000	164,805	156,098	173,764	167,151
	4000	85,266	79,870	90,411	86,700
	8000	44,925	41,622	47,711	45,666
Hybrid	1000	311,315	301,819	351,352	341,920
	2000	156,004	150,308	187,503	175,722
	4000	77,426	76,003	98,761	95,452
	8000	40,618	39,462	52,601	50,591
Exemplar	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

situations in which highly informative examples are available. The fact that each of our 26 outcomes came in different "flavors" may also help explain the advantage of the exemplar-based procedure.

We also examined several other dependent measures. Table 2 summarizes the effect of the several manipulations on rule specificity. In this research, specificity is defined as the number of non-wild card attributes in the rule.

Under most conditions, the average specificity of the rules increased as the value of NEWST increased. This is consistent with the expectation that, in general, more specific rules will participate in the auction less often than more general rules. Since the tax was fixed at 1 unit under all conditions, an increase in the value of NEWST has the effect of permitting a rule to maintain a viable level of strength for longer periods of time between successful participations in the auction. This favors a rule base of higher average specificity, which is consistent with the general trend observed in Table 2.

Our results also indicated that the average specificity of the rules was greater with rule creation by HYB than by RAN. This difference can probably be attributed to the fact that both the mutation procedure and the crossover procedure create rules with a wider range of specificities than their parent rules. This provides an opportunity for the reward allocation process to discover rules that are more accurate because they are more specific. The exemplar-based rule creation procedure produced rules that had a much higher average specificity than either RAN or HYB. This probably results from the lower wild card probability ($p = .5$ for exemplar as compared to $p = .75$ for random) combined with the greater tendency for these more specific exemplar-based rules to match test items and provide correct answers. When we examined the random procedure with the wild card probability set at .5, the rules were more specific, but hardly ever matched a test item, and when one did, it hardly ever had the correct outcome. Therefore, almost all of the high-specificity rules generated by the random process were discarded by the system.

The relationships between average rule specificity and our manipulations of the bidding system seem to be complex. With rule creation by the random method or exemplar method, winner-take-all reward allocation produces a rule set of higher specificity than the reward sharing procedure. This relationship appears to be reversed with the hybrid rule creation procedure. When the reward was shared, bids proportional to strength tended to produce higher specificity rules than bids proportional to the product of strength and specificity. When the reward was allocated on a winner-take-all basis, however, the opposite relationship tended to occur: strength bidding produces less specific rules than strength-times-specificity bidding. We do not have a ready explanation for these complex relationships.

The size of the final rule set (after verification during the sixth pass) seems to correlate closely with the accuracy scores for the various procedures. The relationship is positive: the greater the number of rules, the higher the accuracy score. The procedure that created and retained the largest number of successful rules involved rule creation by exemplars and reward allocation by sharing. The procedure that produces the fewest successful rules employed random rule creation and allocated reward by the winner-take-all procedure. Another relationship that was observed is that bidding by strength times specificity tends to produce a larger set of successful rules than bidding by strength alone. In this case, identification accuracy is not improved by the greater number of rules; SS produces more successful rules than STR but the identification accuracy of the two procedures tends to be equivalent.

The final dependent measure we examined was the total number of rules created during the training phase. These results are summarized in Table 4. The number of rules created by RAN and HYB depends on how often rules are purged, and thus upon the quality of the existing rule base. With both of these procedures, a fixed-size rule buffer (n = 3900) was maintained throughout training. The number of rule creations was much greater for RAN and HYB than for EXM. With the RAN and HYB procedure, there were more rule creations with winner-take-all and with bidding by strength alone.

The lower rule creation rate for the exemplar-based procedure results from the higher quality of the rules which are being produced. It is apparently unnecessary to create and purge a large number of rules in order to find a few effective ones. Rule turn-over is greatly reduced when the rules have a decent probability of being effective in their initial form. The larger number of rules in the winner-take-all condition is most likely the result of the poorer overall performance of this procedure which leads to more frequent rule purgings. The reason for the greater number of rule creations in the strength-only bidding condition is not clear. The two bidding procedures generally produced similar levels of performance and thus some additional factor, other than final performance level, must be responsible for the additional number of rules created by strength-only bidding.

The rate of rule creation, even in the most unfavorable conditions, did not appear to destabilize the rule set. In the RAN and HYB procedures, the size of the rule set was fixed at 3900. During training, 80,000 training items were examined. In the most active condition, 350,000 rules were created. This averages 4.5 new rules per trial or a turn-over rate in relation to the entire set of rules of 0.12% per trial. This is about 1% of the rules being changed every 9 trials. When NEWST was set at 8000, the rate of change for these same conditions was considerably slower, approximating 1% turn-over every 60 trials.

7. 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 Table 5 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: $p = .7$ for window size 0, $p = .5$ for window size 1, and $p = .3$ for window size 2. These values were selected because they optimized performance for each procedure.

The results in Table 5 indicate that the classifier system performs best with a window size of 1. Under conditions of optimal resource allocation (NEWST = 8000), requiring exact matches (window size 0) led to fewer correct identifications, a larger rule set, and rules of lower average specificity. Setting a wider window (window size 2) produced fewer correct identifications, a smaller rule set, and rules of higher average specificity. There appears to be a direct trade-off between window size and specificity. With a wider window, it is feasible to increase the number of relevant attributes and still maintain a reasonable level of matching. When exact matches are required, only a few attributes can be specified as non-wild cards if a reasonable level of matching is to be maintained.

The results in Table 5 also indicate a clear relationship between window size and the number of rules that the system employs. With exact matching, many rules are required. With a wider window, fewer rules are required in order to maintain a comparable level of performance.

Table 5. Effect of fuzzy match window size on exemplar-based rule creation with strength bidding and reward sharing.

Dependent Measure	NEWST	Window Size		
		0	1	2
% Correct Identifications on 4000-Item 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

8. Binary and gray code attribute representation

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.

The results in Table 6 indicate that the integer representation seems to be more effective on the supervised learning task than either the binary or gray code representations. The gray code representation was slightly superior to the binary representation. In the binary and gray code conditions, rule specificity refers to the number of relevant positions out of a 64 item vector. With the integer representation, only 16 positions are available. With this in mind, the integer representation produced rules with higher average specificity (in a relative sense) than either of the other two methods. For example, when NEWST was set at 8000, 50% of the positions were relevant in the integer condition, while 33% were relevant in the binary and gray code conditions. This higher level of specificity is feasible with the integer representation, because exact matches are not required. When exact matches are required in the integer condition (see window size = 0 in Table 5), the proportion of relevant attributes is 24%, which is even less than with the binary and gray code conditions. Note however that the accuracy level when exact matches are required with the integer representation is still superior to both the binary and gray code representations. Thus, even

Table 6. Effect of binary, gray code, and integer representations on exemplar-based rule creation with strength bidding and reward sharing.

Dependent Measure	NEWST	Type of Attribute Representation		
		Binary	Gray Code	Integer
% Correct Identifications 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 Rule Specificity	1000	22.00	21.34	6.65
	2000	21.54	21.22	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 task. One should note, however, that we did not examine partial matching under the binary and gray code conditions. These methods may have been more effective with partial matching.

We did not examine binary and gray-code representations with single-point crossovers. It is possible that binary and gray-code representations may compete more successfully with the integer representation under conditions in which crossovers are employed for rule creation. One of the advantages of these representations is that they provide a rich set of schemata which make crossovers more effective.

The results in Table 6 also indicate that the number of rules present after verification is influenced by the type of representation. When NEWST is set at 8000, the integer representation with fuzzy matching produces a higher accuracy level with a smaller set of rules than the binary and gray code representations. When NEWST is set at lower values, the integer method employs a larger number of rules than the other methods. The gray code representation uses fewer rules to outperform the binary representation with NEWST set at 8000. At lower settings, the gray code procedure still has higher accuracy than the binary procedure but employs a similar number of rules.

9. Accuracy-utility bidding system

The concepts of rule strength and rule specificity have played an important role in the development of Holland's classifier system. In the current research, alternative measures of rule fitness were examined in an effort to address problems we 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. Our experience with these difficulties provided motivation to explore an alternative reward allocation system.

The "accuracy-utility" procedure (AU) separates the concept of rule strength into two distinct measures: accuracy and utility. These two measures bear a functional resemblance to Holland's (1976) measures of average payoff rate and age. Holland defined the rate measure (p. 282) as the payoff received over an epoch divided by the length of the epoch. Our accuracy measure follows this precedent except that it tracks the relative frequency of payoff rather than the average amount. Holland's age measure (p. 186) was defined as the number of time steps since the rule received a positive payoff. Our utility measure assesses a similar aspect of performance except that it tracks overall performance rather than focusing exclusively on the rule's most recent exploit. Booker (1988, p. 180) also employs multiple performance measures to determine reward distribution.

In our AU procedure, accuracy tracks the ratio, for the life of the rule, of the number of correct bids to the total number of bids. This frequency ratio estimates the individual likelihood that each rule in [M] will forecast the correct category. From a baseball perspective, accuracy is the rule's batting average for the occasions on which it qualifies for the auction (i.e., is a member of [M]). The rule in [M] with the highest accuracy score wins the auction and determines the system's output.

To prevent young rules from prematurely dominating the bidding after a few lucky guesses, the accuracy ratio is defined with a bias of 2 added to the denominator. This bias forces each new rule to serve an "apprenticeship" during which it develops a stable accuracy record without disturbing the performance of the system. This bias also has the advantage of selecting the more experienced rule when two or more rules have 100% accuracy. Note that the accuracy measure weighs initial performance as heavily as recent performance and thus assumes a time-invariant sampling process.

For the results presented in this article, a further adjustment was made to the accuracy measure. Empirical analysis indicated that the behavior of some rules clearly invalidated the implicit assumption that the accuracy of a rule was independent of whether or not it won the auction. For unexplained reasons, some rules had significantly worse performance records when they won the bidding than when they lost to another rule. To compensate for this phenomenon, a second measure was tabulated. This second measure was computed as the number of correct identifications on the occasions when the rule won the bidding divided by the number of times the rule won the bidding. This measure was given an initial optimistic bias (2 out of 2) to prevent it from prematurely crippling a young rule. Each rule's net accuracy was defined as the minimum of the two accuracy measures. The effect of this "win-accuracy" adjustment was a modest but noticeable improvement in performance. This modification did not seem to compromise the stability of the system.

Our utility measure attempts to measure the usefulness of a rule to the system and thereby the desirability of its retention. The utility measure is defined as the number of correct winning bids divided by the number of stimulus items presented during the lifetime of the rule. Note that an accurate, frequent bidder that is consistently outbid by a more mature or more accurate rule will have a low utility.

Rules stay in the system as long as their utility values exceed a pre-specified criterion value. In the current research, we examined utility criteria of 1/1000, 1/2000, 1/4000, and 1/8000. Each new classifier was given a probationary period during which it was retained by the system without regard to its utility value. The probationary period extended until the number of stimulus items "seen" by the rule exceeded the reciprocal of the utility criterion, i.e., 1000, 2000, 4000, or 8000 items. The utility criteria provides a simple and direct parameter to control the number of rules in the system. An investigator can set this parameter to produce a rule set of a desired size and the system should then select a set of rules which optimizes performance for a rule set of that size.

In the accuracy-utility system, five performance characteristics are retained for each classifier. These measures are the number of bidding opportunities, the number of matches, the number of wins when the classifier matches, the number of correct identifications when the classifier matches, and the number of correct identifications when the classifier wins the bidding.

Although the strength concept has been broadened into two separate measures, the revised system appears to be conceptually simpler. There are no auxiliary constructs like taxes. Considerations like strength alone versus strength times specificity or reward sharing versus winner-take-all are no longer problematic. In addition, the AU bidding system has only a single parameter, the utility criterion, which needs to be adjusted for the specific application. Traditional strength systems require parameter optimization for amount of reward, amount of tax, and bid cost.

Our algorithm for the AU bidding system is:

1. A match set [M] is selected consisting of all classifiers whose conditions are satisfied by the input string.
2. Each classifier in [M] makes a bid equal to its accuracy.
3. The system makes its decision by selecting the classifier from [M] that has the highest bid and outputs the selected classifier's category as the system's decision.
4. Each classifier's accuracy and utility measure are recalculated.
5. If a classifier's utility drops below the pre-specified criterion, the classifier is immediately removed in condition EXM or replaced by a newly created classifier that represents the same category in condition RAN or GEN.

As was the case with the strength bidding system, accuracy-utility was most successful in creating and retaining effective classifiers when it employed integers for attribute coding, an exemplar-based rule creation procedure, a wild card probability set at 0.5, and a window size for fuzzy matches set at 1. During the 80,000 training trials (16,000 items × 5 passes), two new rules were created each time the winning rule misidentified the test item. During the sixth and final pass through the 16,000 items, no new rules were created. Rules that fell below the predetermined utility criterion were discarded throughout all six passes. The rules remaining after the sixth pass were tested against the 4000-item test set.

System	Utility Criterion	% Correct	Specificity	Final Rules	Rules Created
INT.EXM.AU	1/1000	68.5	6.84	252	52,612
window = 1	1/2000	73.3	7.32	426	44,278
wild card = 0.5	1/4000	79.2	7.71	674	37,082
AWCT = yes	1/8000	81.6	8.16	1040	33,446

These results are comparable to the results observed with the strength-based bidding system with exemplar rule creation and reward sharing. As the utility criterion is reduced, the system retains a larger number of rules, the average rule specificity increases, and the number of items in the test set that are correctly identified increases. This system produces its highest performance level (81.6%) with the utility criterion set at 1/8000. The strength-based bidding system, employing exemplar rule creation, produced a comparable performance level (e.g., 80.8%).

In comparison to the best strength systems (reward sharing, exemplar-based rule creation, NEWST = 8000), the accuracy-utility system seems to produce rules of similar specificity, creates slightly fewer rules during training, and has a more compact rule set after verification. The accuracy-utility system also greatly simplified our task in finding the parameter settings which led to optimal performance.

10. Exemplar/hybrid rule creation

The prior results indicated 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 relationship 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.

Rules were created initially by generating two new exemplar-based rules each time the system failed to identify the test item correctly. When a rule's strength decreased to zero or less, it was replaced by a new rule using either a mutation or a single-point crossover. Preliminary analysis indicated that a 50-50 mix of mutations and crossovers produced the best performance. This 50% mixture was employed for all subsequent analyses. Mutations and crossovers were operationalized exactly in the manner described previously. Hybrid rule creation was not activated if the rule to be replaced was above position 8000 in the rule buffer. This specification kept the rule buffer at a manageable size (approximately 9000 to 11,000 rules) during training.

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	6.10	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

As compared to the system using exemplar-based rule creation with all other settings identical, the exemplar/hybrid system demonstrated several interesting properties. The rule base that was retained after verification was consistently more compact and consistently had rules of lower specificity. These differences were quite dramatic. Apparently the hybrid procedure had the effect of producing more general rules. At the lower NEWST settings (1000 and 2000), the exemplar/hybrid system had lower accuracy in identifying test items. At the higher settings (4000 and 8000), this system had superior performance in regard to the test items. It appears that with a large rule buffer, the exemplar/hybrid combination is the best procedure for producing a high quality set of rules (i.e., maximizing accuracy and compactness).

11. Discussion

Holland (1986) proposed an adaptive general-purpose rule-based system that is designed to cover a broad range of applications. The procedures discussed in this report are relevant to a more limited set of applications in which problems can be solved by categorizing test

cases from a time-invariant population. If one chooses an appropriate representation, a wide range of problems can be approached within this framework. Our analysis of the literature suggests that more than half of the common expert system applications can be represented as categorization problems.

Our efforts to adapt Holland's approach to this more specialized environment have produced several useful findings. There are many real-world applications in which attribute values come naturally as ordered numerical variables. Our results indicate that one can successfully map these values onto a limited range of integers and employ a "fuzzy match" strategy for rule activation. On our character recognition problem, integer representations were more effective than either binary or gray-code (Caruana & Schaffer, 1988; Goldberg, 1989) representations.

Our results also demonstrated that exemplar-based rule creation methods can be quite effective. On the character recognition task, random rule generation and a hybrid procedure (random creation, mutations, and single-point crossovers) were noticeably less successful. In the character recognition task, rules created by randomly selecting attribute values had a very low probability of being useful. Even when 350,000 rules were created in this fashion, only a small number were consistently successful. This small group seemed to be inadequate for the needs of the genetic procedures. If we had presented many more training examples to the system, it might have eventually produced enough information for a genetic process to become effective in evolving better rules. Other types of crossovers, such as two-point crossovers or within-feature crossovers, might produce more effective results. It is clear, however, that exemplar-based rule creation produced good candidates at a much faster rate than the random or hybrid procedures we examined.

The hybrid procedure created rules initially by a random process and then evolved better rules with mutations and crossovers. An unusual finding was that this system performed best when genetic mutations were more common than genetic crossovers. This outcome may result from the composition of the outcome categories in the character recognition task. In particular, the different outcomes appear in specific flavors. For example, the Gothic fonts are quite different from the others. When crossovers are the predominant rule creation method, the system tends to produce many rules that are variations of the same main theme. This focus tends to favor the most common "flavor" at the expense of the others. The mutation operator tends to produce a more diverse set of offspring and this characteristic may be helpful in coping with the multiple flavors of the same category.

When the exemplar-based procedure was combined with the genetic procedures, the hybrid system produced rules that achieved the highest performance level. This rule set was also more general and more compact than any of the other rule sets that performed at a similar level. Of the various methods we examined for rule creation, the exemplar-genetic combination was the most effective.

The concepts of rule strength and rule specificity have played an important role in the development of Holland's classifier system. Wilson (1988) has published results that question whether it is useful to consider rule specificity in determining reward allocation and bid cost. Our results were consistent with Wilson's in that rule specificity did not provide any particular advantage. When bid cost and reward allocation were computed without regard to specificity, system accuracy was comparable and other performance measures were at least as good.

Our results with the character recognition task indicated that reward sharing is much more effective than a winner-take-all reward allocation procedure. The superiority of reward sharing was observed in every condition we explored. From a conceptual perspective, there are several advantages of sharing reward among the top bidders who advocate the correct outcome. One advantage of reward sharing is that more reward enters the system. When the top bidder in a winner-take-all procedure advocates the incorrect category, no reward is given to any rule. When the reward is shared, it is more probable that at least one of the bidders will advocate the correct category. This has the effect of distributing reward more completely with the reward sharing procedure than with winner-take-all. A second advantage of reward sharing is that individual rules receive feedback more frequently when reward is distributed among all the bidders. More frequent feedback seems to enhance the rate at which rule strengths adapt to the environmental contingencies.

In the current research, we explored a variation of Holland's (1976) multiattribute procedure for evaluating rule fitness. With strength-based classifier systems that have a large number of rules, we have observed instabilities in the performance of the system. Although there are several possible explanations for this undesirable phenomenon, we believe that the problem relates to the many parameters that need to be adjusted for strength-based bidding systems. Investigators often have difficulty finding a proper balance among these parameters. If the balance between bid cost and reward allocation is not optimal or if the tax rate is not optimal, significant performance decrements can result.

We explored the accuracy-utility bidding system in an attempt to clarify the conceptual basis for establishing a rule's fitness and to reduce the number of parameters associated with evaluating fitness. In a strength-based system, a rule's fitness is expressed as a single strength measure based on initial strength, payoffs, bid costs, and taxes. In our research, we observed that fitness had two separate aspects that can be measured independently: skill at performing classifications and contribution to overall system performance. A rule can be skillful in making correct classifications and yet be so specific that it contributes very little to overall system performance. Or a rule may be very general and yet be judged redundant because it is usually outbid by one or more rules that are more skillful.

The definitions we chose for these two fitness measures are consistent with the conceptual distinctions described by Holland (1976). The skill measure, **accuracy**, is defined as the cumulative ratio of the number of correct bids to the total number of bids, with an initial pessimistic bias to help counter "beginner's luck." This bias permits new rules to develop reliable accuracy values without influencing the performance of the system. The accuracy measure is a fraction bounded by 0 and 1 and provides a historical record of the rule's success in providing a correct response each time it matches a test case. When several rules match a test item, the one with the highest accuracy is heeded. Accuracy scores are modified for all rules that matched the test item. This consistent feedback enhances the reliability of the accuracy measure.

The second measure, **utility**, reflects the relative frequency with which each rule contributes positively to the system's overall performance. This definition is based on the idea that overall system performance is simply the rate of making correct classifications and that an individual's utility is the proportion of that rate attributable to the rule's correct winning bids. At any given moment, system performance is equal to the sum of the utilities of all of the rules currently in the system. With infinite resources, all rules of high accuracy

can be maintained in the system. If resources are limited and the system must operate with a small set of rules, the rules that contribute least often to system performance are discarded, regardless of their accuracy scores. This heavy reliance on utility is predicated on the principle that a rule with a low accuracy score should be retained by the system as long as there is no other rule which can out-perform it in its own arena. If there were a rule with more accurate predictions, that rule would be winning the auctions and have accumulated a higher utility score.

Both measures, accuracy and utility, are defined in the context of a time-invariant population of data items. This is appropriate for the character recognition data set. The use of lifetime "batting averages" makes sense if each bidding opportunity provides equally valid feedback for determining rule fitness. If the application involves an evolving data environment, these definitions might be more effective if recent activity were weighted more heavily. Holland's (1976) age measure is similar to our utility measure but is computed on the sole basis of the rule's most recent contribution to system performance. The relative importance of these differences is probably worth exploring. Our time-invariant method has the advantage of increasing reliability by statistical averaging. It has the disadvantage of being less sensitive to changes in the data environment over time.

One of the important characteristics of Holland's parallel rule-based system is the ability to create a default hierarchy of rules (Holland, Holyoak, Nisbett, & Thagard, 1986, p. 18) that allows a group of rules to act in concert in a way that promotes both efficiency and correct performance. The accuracy-utility approach seems to foster default hierarchies in a natural way. The rules with the highest accuracy scores tend to be very specific ones that attend auctions infrequently. They only bid in highly specialized situations and are almost always correct. More general rules tend to have lower accuracy scores and therefore only win the bidding when no specialist is present. By directly measuring each rule's accuracy, our system seems to foster stable default hierarchies without having to factor rule specificity into the bidding process.

A major disadvantage of the current formulation of the accuracy-utility bidding system is the requirement for feedback after each test item. Many applications for classifier systems involve intermittent feedback. Holland's bucket brigade approach was developed specifically to address the credit apportionment problem when rules contribute positively to the system's performance by preparing (setting up) circumstances that eventually lead to positive outcomes. The manner in which accuracy and utility are currently defined makes them insensitive to the consequences of delayed reward. The accuracy-utility approach might be adapted to intermittent feedback applications by employing Sutton's (1988) method of temporal differences.

There are several other interesting contemporary approaches for classifying complex stimuli into a fixed set of categories. In this report, we have focused on learning algorithms that produce parallel rule-based systems. This methodology is a natural outgrowth of the ideas pioneered by Holland (1975, 1976, 1980, 1986). There are also other promising approaches that are not explored in this report but might profitably be examined in future research. These include methods that build a decision tree in a recursive manner (Hunt, Marin, & Stone, 1966; Quinlan, 1979, 1986), methods that categorize individual items based on their similarity to previously classified cases (Stanfill & Waltz, 1986), and methods

based on connectionist networks (Ackley, Hinton, & Sejnowski, 1985; Anderson, 1983; Rumelhart & Zipser, 1985). In particular, it would be of interest to determine the effectiveness of these other methods on our letter recognition problem.

Acknowledgments

We thank Henry Cejtin for help in preparing the character images and both Henry and Rico Tudor for comments and suggestions. We have also greatly benefited from the insights and helpful suggestions of four anonymous reviewers. Computational resources for this project were provided by the Odesta Corporation, Northbrook, Illnois.

References

Ackley, D.H., Hinton, G.E., & Sejnowski, T.J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science, 9,* 147–169.

Anderson, J.A. (1983). Cognitive and psychological computation with neural models. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-13,* 799–815.

Booker, L.B. (1988). Classifier systems that learn internal world models. *Machine Learning, 3,* 161–192.

Caruana, R.A., & Schaffer, D. (1988). Representation and hidden bias: gray vs. binary coding for genetic algorithms. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 153–161). Ann Arbor, MI: Morgan Kaufmann Publishers.

Charness, N. (1981). Aging and skilled problem-solving. *Journal of Experimental Psychology: General, 110,* 21–38.

Chase, W.G., & Simon, H.A. (1973). Perception in chess. *Cognitive Psychology, 4,* 55–81.

Davis, L., & Young, D.K. (1988). Classifier systems with Hamming weights. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 162–173). Ann Arbor, MI: Morgan Kaufmann Publishers.

de Groot, A.D. (1965). *Thought and choice in chess.* The Hague: Mouton, 2nd edition, 1978.

Goldberg, D.E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley Publishing.

Holland, J.H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.

Holland, J.H. (1980). Adaptive algorithms for discovering and using general patterns in growing knowledge bases. *International Journal of Policy Analysis and Information Systems, 4,* 217–240.

Holland, J.H. (1986). Escaping brittleness: The possibilities of general purpose machine learning algorithms applied to parallel rule-based systems. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. II). San Mateo, CA: Morgan Kaufmann Publishers.

Holland, J.H., Holyoak, K.J., Nisbett, R.E., & Thagard, P.R. (1986). *Induction: Processes of inference, learning, and discovery.* Cambridge, MA: The MIT Press.

Hunt, E.B., Marin, J., & Stone, P.J. (1966). *Experiments in induction.* New York: Academic Press.

Quinlan, J.R. (1979). Discovering rules by induction from large collections of examples. In D. Michie (Ed.), *Expert systems in the micro electronic age.* Edinburgh: Edinburgh University Press.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning, 1,* 81–106.

Robertson, G.G. (1988). Population size in classifier systems. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 142–152). Ann Arbor, MI: Morgan Kaufmann Publishers.

Rumelhart, D.E., & Zipser, D. (1985). Feature discovery by competitive learning. *Cognitive Science, 9,* 75–112.

Stanfill, C., & Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM, 29,* 1213–1228.

Sutton, R. (1988). Learning to predict by the method of temporal differences. *Machine Learning, 3,* 9–44.

Wilson, S.W. (1985). Knowledge growth in an artificial animal. *Proceedings of an International Conference on Genetic Algorithms and Their Applications* (pp. 16–23). Pittsburgh, PA: Lawrence Erlbaum Associates.

Wilson, S.W. (1987). Classifier systems and the animat problem. *Machine Learning, 2,* 199–228.

Wilson, S.W. (1988). Bid competition and specificity reconsidered. *Complex Systems, 2,* 705–723.