Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Learning from Data using XCS

Elias B. Ayele * **Abdollah Homaifar** ** **Albert Esterline** ***
Robert Dean **** **Dan Rodgers** †

* *Electrical Engineering Department, North Carolina A&T State University, Greensboro, NC 27411 USA (e-mail: ebayele@ncat.edu)*
** *Electrical Engineering Department, North Carolina A&T State University, Greensboro, NC 27411 USA(e-mail: homaifar@ncat.edu)*
*** *Computer Science Department, North Carolina A&T State University, Greensboro, NC 27411 USA (e-mail: esterlin@ncat.edu)*
**** *General Dynamics Robotics System, Westminster, MD 21157 USA (e-mail: rdean@gdrs.com)*
† *General Dynamics Robotics System, Westminster, MD 21157 USA (e-mail: drodgers@gdrs.com)*

Abstract: In this paper, we present first of all the working principles of an accuracy based learning classifier system. We also discuss the use of learning classifier systems for learning from data by considering a sample application. The sample application, the Terrain Reasoner Weight Adapter (TRWA), is a system that learns near optimal weights to be used by a path planner while generating routes. Manually generated weights are used to generate a sample data set for training the TRWA. We detail the TRWA and the significant improvements made to the usual XCS strategies in order to achieve our goal of using a supervised learning technique for the TRWA. A reward assignment scheme is developed. The use of tournament selection instead of roulette wheel selection for selecting two parents in the GA is also analyzed. The results obtained show the efficiency of the method.

1. INTRODUCTION

Among machine learning approaches, learning from training examples is one track of significant interest. In statistical machine learning, two different scenarios are clearly distinguished : supervised and unsupervised learning (Seeger (2001)).

In supervised learning, learning is facilitated by a teacher. Various input–output mappings are provided as part of the training examples, and the learner is given some sort of reinforcement for selecting the best output for a given input. In the end, input-ouptut mapping rules, also known as classifiers, are evolved. For some classes of problems, identified as classification or pattern recognition problems, the output space (set of labels) is finite whereas for others, categorized as regression estimation problems, the output space is infinite. Minimizing errors in generalization and expected loss are of great importance for classification and regression estimation problems, respectively. In unsupervised learning, there is no *a priori* output, that is, no labeled examples are available. Instead, a model is built for the given set of inputs.

This work considers the use of a supervised learning technique for learning from training data. The next section details a learning classifier system, specifically XCS, that is currently widely used. Section 3 discusses a specification of the system developed, the TRWA. Section 4 then presents the use of XCS as a supervised learning technique playing a central role in the TRWA. It emphasizes the modifications introduced to the usual XCS strategies. Results found are shown in section 5, and section 6 concludes.

2. XCS

An XCS is a learning classifier system (LCS) where classifier fitness is based on the accuracy of the payoff prediction rather than the prediction itself. An XCS strives to achieve generalization through the evolution of a population of the condition–action–prediction rules (classifiers). An XCS thus has the tendency to evolve accurate, maximally general classifiers (Butz and Wilson (2001)). [1]

The system gets its input $\sigma(t)$ from the environment and then executes actions $\alpha(t)$ upon the environment. These actions are those favored by classifiers that have matching condition with the sensory inputs and yet are more accurate than others. For the input–output pair, a reinforcement program (RP) determines the reward $\rho(t)$. In multi–step problems, the RP also determines the point of termination, which could be the receipt of reward or exceeding the maximum number of trials.

Figure 1 illustrates the interaction of the XCS with the environment and the RP and could address either a single step or a multistep problem. In the former case, successive situations are not related to each other, and reward is received at every step. In the latter case, a reward is received only in the final step. This reward is then propagated back and distributed to all classifiers that led to this final step.

[1] Butz and Wilson (2001) details the XCS algorithm.

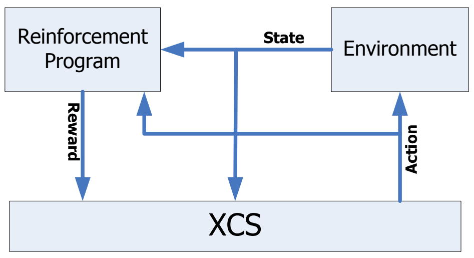

Fig. 1. XCS interacts with the environment and RP [Butz and Wilson, 2001]

2.1 Classifiers in XCS

A classifier in an XCS has three slots: condition (matched against the current environmental state), action and parameter slots. The condition is a string bits from the set $\{0, 1, \#\}$, the hash, #, being a *don't care*, matches 0 or 1. The action is a member of the set $\{a_1, a_2, ...a_n\}$, where a_i represents an action the agent can perform on the environment. The parameter slot mainly consists of four parameters: the prediction p, the prediction error ϵ, the fitness F, and the numerosity num. In order to speed processing, a classifier is represented as a *macroclassifer*, which represents all num traditional classifiers (or *microclassifiers*) in the population $[P]$ with the same condition and action. Each classifier keeps additional parameters: the experience exp, the time stamp ts, and the action set size as. It is convenient to partition the description of a single operating cycle into the traditional performance, reinforcement, and discovery components. A schematic illustration of XCS is given in Figure 2.

Performance Component Classifiers in $[P]$ with conditions that match the current sensory inputs form the match set $[M]$. If no classifier matches the sensory input or if $[M]$ contains fewer than the threshold θ_{mna} number of actions, then a classifier is added to $[M]$, that is, the input is "covered". This *covering* procedure assures that at least a certain number of actions is present in each match set and causes the program to converge more quickly (Butz and Wilson (2001)). Covering creates a new classifier that matches the current input and has a random action. But, when $[M]$ does not have sufficient number of actions, then the new classifier's action is selected from $\{a_1, a_2, ...a_n\}$ excluding those actions already present in $[M]$. If $[P]$ has reached its allowable maximum size N, then a classifier is deleted from $[P]$.

For each possible action a_k present in $[M]$, XCS computes the system's prediction. The prediction array is the set of all predictions PA_k for all possible actions a_k. PA_k is the fitness–weighted average of the predictions p_i of the classifiers in $[M]$ advocating that action and is given by (where F_j is the fitness of the i^{th} classifier in $[M]$)

$$PA_k = \frac{\sum_i F_i p_i}{\sum_i F_i} \tag{1}$$

The XCS chooses an action from those present in $[M]$. This action selection could be random, roulette wheel (prediction–based probability) or deterministic (maximum

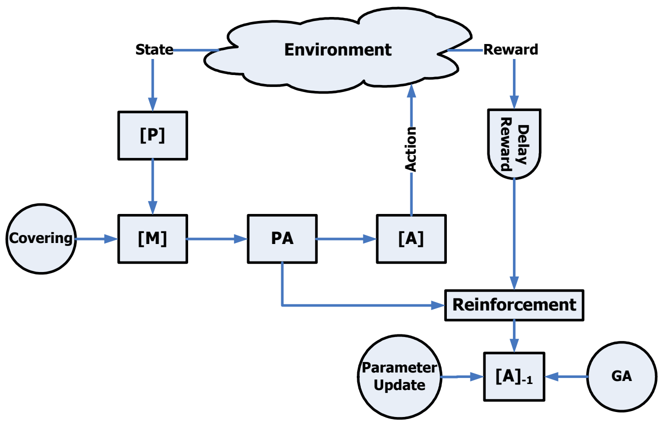

Fig. 2. Schematic illustration of XCS

prediction). An action set $[A]$, consisting of those classifiers in $[M]$ that advocate the selected action, is then formed.

Reinforcement component When a scalar reward ρ is received from the environment, the payoff P is computed using a Q–learning–like reinforcement learning technique given by

$$P = \rho_{-1} + \gamma * \max(PA) \tag{2}$$

where ρ_{-1} is the reward from the previous time-step and γ is the discount factor. For single-step problems, the payoff reduces to ρ.

For multi–step problems, all parameter updates are performed on the previous action set $[A]_{-1}$. For one–step problems, updates are done on $[A]$, and, in fact, there is no $[A]_{-1}$ for such problems. Accordingly, the term "action set" in the preceding discussion may refer to either $[A]$ or $[A]_{-1}$.

At generation j, the prediction p_j, a statistic estimating the payoff P, and the error ϵ_j of a classifier are updated when that classifier enters $[A]$ using the moyenne adaptive modifée (MAM) technique (Wilson (1995)). In this technique, the Widrow–Hoff procedure is used only after a classifier has been adjusted at least $1/\beta$ times; otherwise, the new value is simply the average of the previous and the current. The Widrow–Hoff procedure is given by

$$p_j \leftarrow p_j + \beta(P - p_j) \tag{3}$$
$$\epsilon_j \leftarrow \epsilon_j + \beta(|P - p_j| - \epsilon_j) \tag{4}$$

where β $(0 < \beta \leq 1)$ is a learning rate constant. The average of the current and previous values is computed as

$$p_j \leftarrow (P + (\exp_j - 1)p_j)/\exp_j \tag{5}$$
$$\epsilon_j \leftarrow (|P - p_j| + (\exp_j - 1)\epsilon_j)/\exp_j \tag{6}$$

or equivalently

$$p_j \leftarrow p_j + (P - p_j)/\exp_j \tag{7}$$
$$\epsilon_j \leftarrow \epsilon_j + (|P - p_j| - \epsilon_j)/\exp_j \tag{8}$$

The fitness calculation is a bit involved and is tied to the classifier's accuracy in prediction. First, the classifier's accuracy k_j is computed using the formula

$$k_j = \begin{cases} \alpha(\epsilon_j/\epsilon_0)^{-v}, & if\ \epsilon_j > \epsilon_0 \\ 1 & , otherwise \end{cases} \tag{9}$$

A classifier is considered "accurate" if its error is less than a threshold, ϵ_0, and is given accuracy 1. If the error is greater than ϵ_0, the resulting lower accuracy is given by a decaying polynomial of the error as in (9) where v ($v > 0$) and α ($0 < \alpha < 1$) are constants that control the rate of decline in the accuracy. Then a relative accuracy k'_j is calculated for each classifier in the action set by dividing its k_j by the sum of the k_js of the set, that is,

$$k'_j = \frac{k_j}{\sum_j k_j} \tag{10}$$

Finally, the fitness of each classifier is computed using the Widrow–Hoff procedure,

$$F_j \leftarrow F_j + \beta(k'_j - F_j) \tag{11}$$

These cyclic and interlinked updates of p_j, ϵ_j, and F_j make the fitness of a classifier represent the accuracy of its payoff prediction relative to the prediction accuracies of other classifiers in its action sets. This is the basis for the selective pressure in XCSs toward more accurate classifiers.

Discovery Component In several LCSs, the genetic algorithm (GA) has no constraint on mating, that is, it panmictically uses the entire population. But in XCS the GA operates on members of $[A]_{-1}$ or $[A]$. This shift in GA locus increases the proportion of accurate, maximally general classifiers in $[P]$. Besides, the GA is not executed at every time step. Instead, the time stamp ts values of classifiers in $[A]$ is averaged and subtracted from the current time step, and, if the difference exceeds a certain threshold, θ_{GA}, the GA is invoked.

Crossover and mutation maintain genetic diversity from one generation to the next. In crossover, condition bits (beyond a certain cross–site for one point crossover) are swapped between two parent classifiers. In mutation, the value of a classifier bit is changed to another valid value (chosen from $\{0, 1, \#\}$ for condition bits or from $\{0, 1\}$ for action bits). The parents are selected from $[A]$ with fitness–based probability or using tournament selection (Butz et al. (2002)) and copied. A cross–site is selected randomly and the copies are crossed with probability χ and mutated with probability μ per allele. The resulting offspring are inserted into $[P]$, possibly replacing two others if the limit for the population size, N, is exceeded. Less fit classifiers that participated in a threshold number of action sets are susceptible to deletion. Using such a threshold guarantees that individuals have had sufficient time for their parameters to be estimated (Kovacs (1999)).

To deal with sparseness and to evolve classifiers that are as formally general as possible without sacrificing accuracy, an optional procedure called subsumption deletion is applied (Wilson (1998)). A classifier subsumes another if its condition is more general than the condition of the subsumed one.

Some applications of XCS to a real-world problem include data mining (()), and optimization (Ross et al. (2002)) among several others. Investigation of XCS's working principles is available in (Butz et al. (2001)).

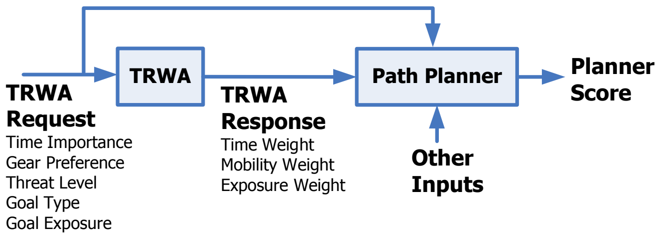

Fig. 3. Exploitation phase system block diagram

3. TERRAIN REASONER WEIGHT ADAPTER (TRWA)

In this work, an example area of operation, qualitatively characterized and with recognizable topographical features, is considered. A mission is set for an agent with source, target, and tasks. In areas of operation where there are constraints other than goal achievement, routes should be carefully planned. These constraints include time spent by an agent for traversal, exposure to the enemy, mobility, coverage for reconnaissance, and obstacle avoidance. It is desired to generate routes that meet most of these constraints while achieving the basic goal of reaching the destination.

The Terrain Reasoner Weight Adapter (TRWA) is a supervised learning program that learns from a given training dataset. (The adaptation component of the TRWA to topographical features will be covered in future work.) Once the learning is over, the TRWA generates near optimal values of weights for stealthy movement of a single agent. Consequently, there are two phases for the TRWA, learning and exploitation. In the learning phase, manually generated weights are used to produce a training dataset. In the exploitation phase, the already–developed planner uses the TRWA outputs, in collaboration with other inputs, to plan routes that best satisfy the constraints. Figure 3 is a block diagram of the overall system in the exploitation phase. We next explain the inputs and outputs shown in this diagram.

3.1 System Inputs and Outputs

TRWA Request
The TRWA request (input) comprises the following attributes.

(1) *Time Importance (TI)* indicates the level of importance given to the time minimization while planning a path. The levels of time importance are HARD (maximum TI), SOFT and DON'T CARE (minimum TI).
(2) *Gear Preference (GP)* is the preference for the gear to use for navigation. Gear levels 0 (slowest) to 3 (fastest) are used.
(3) *Threat Level (TL)* measures the level of exposure (NONE, POSSIBLE, or EXISTS) in the environment.
(4) *Goal Type (GT)* specifies the spatial representation of the goal as POINT, LINE, or AREA.
(5) *Goal Exposure (GE)* is a percentage measure (0 to 1) specifying the exposure level of the goal to threats.

TI and *GP* dictate the preference level given for time minimization and gear level, respectively. The remaining specify the current environmental condition.

Condition	Route Information	Weights	Scores
Threat Level	Time Estimate	Time	Time
Goal Type	Distance Estimate	Mobility	Exposure
Goal Exposure	Gear Percentage	Exposure	

Table 1. Training data set format

TRWA Response The TRWA response (output) is a combination of three weight factors, *time weight* (T_w), *mobility weight* (M_w), and *exposure weight* (E_w) from the interval [0,1]. For training data generation, the weights are manually generated in an incremental fashion.

Planner Score Planner scores evaluate the routes generated by the planner based on time and exposure minimization. The range for both time and exposure scores is [0, 1]. The TRWA is unaware of the score generation process as the planner uses its own independent algorithm to evaluate the performance of a route. Once generated, the scores guide the learner during the learning phase.

3.2 Training Data Presentation

A dataset, produced by the path planner and covering different scenarios for various manually generated weight values, is used for training. A training data point is composed of 13 attributes and has the format shown in Table 1. In the training dataset, scenarios with threat level, goal type and goal exposure triples of (0,0,0), (0,1,0), (0,2,0), (2,0,0), (2,0,1), (2,1,0), (2,1,1), and (2,2,1) are present. Some scenarios are represented more than once for varying source-to-goal distance. The gear percentage indicates the percentage of the route covered at each gear level.

One challenge with the training set is that, for a given scenario, there are multiple weight combinations that generated maximum-scoring routes.

4. TRWA: LEARNING PHASE

The TRWA is a single step XCS system. It gets its inputs from the environment (the training dataset in this context) via the detectors (Figure 4). The environmental condition and route information are matched with the condition of a classifier. The TRWA outputs are compared with those present in the training dataset. Then the environment generates the appropriate score for the winning action. Finally, the RP, which is effectively the reward generator, computes a reward and sends it to the TRWA. This cycle continues until specified convergence criteria are met.

4.1 TRWA Classifier Format

The goal exposure input and all TRWA output parameters are real numbers within the interval [0,1] while the remaining input parameters are integers with only limited sets of values. Accordingly, a way to represent the real-valued attributes is developed. To simplify the representation, three regions are considered for the goal exposure (GE), as shown in Table 2. This clustering is also used to identify the regions for the weights T_w, M_w, and E_w that result in near optimal scores for time and exposure. It also serves as a validation means in the exploitation.

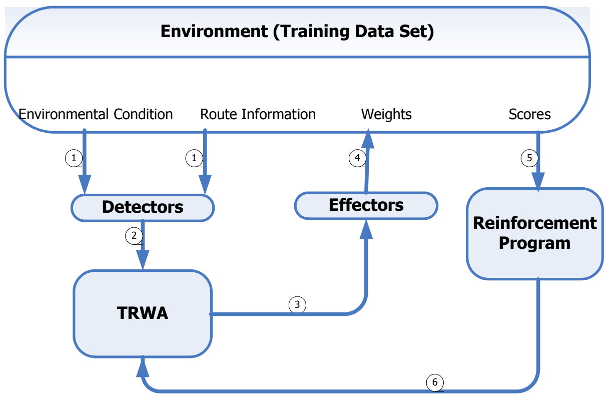

Fig. 4. Learning phase system block diagram. The encircled numbers indicate the sequence of operations.

Range	Region	Value
$0 \leq GE \leq 1/3$	Low	0
$1/3 < GE \leq 2/3$	Medium	1
$2/3 < GE \leq 1$	High	2

Table 2. Regions for goal exposure

Condition					Action (Weight)		
TI	**GP**	**TL**	**GT**	**GE**	$\mathbf{T}_W$	$\mathbf{M}_W$	$\mathbf{E}_W$

Table 3. TRWA classifier format

Every condition slot is represented by two bits, and every action slot is represented by n bits. (The slots are shown in Table 3.) The parameter n is dependent on the weight increment used for generating the training dataset. For a weight increment of 0.1, where 11 different weights are represented, the value of n is four, and there are 1330 data points per scenario. (No data point exists with all zero weights.) Not all bit combinations are valid. In all condition slots other than the gear preference slot and in all action slots, there are invalid bit combinations. For instance, there are only three possible values, 0, 1, and 2, for time importance (i.e., bit strings 00, 01, and 10, respectively). Hence bit string 11 is invalid.

With ten condition bits and $3n$ action bits, the TRWA is much more complex than the usual XCS benchmark problems (e.g., the Woods and Maze problems), where the numbers of actions are far smaller. Parameter n can be changed in accordance with the level of precision required.

4.2 Strategies

Initialization Usually, in an XCS, the initial population is either generated randomly or left empty. For the TRWA, a number of data sets (representing a small percentage of the population size) with maximum score are deliberately injected into the initial population. This has a positive impact on the speed of convergence.

Covering Unlike the usual case of inserting one classifier when covering is necessary, a small percentage or a fixed number of classifiers is inserted. Since there are several possible actions, the injection of classifiers advocating different actions during covering hastens the action coverage process and intensifies early diversification.

Exploration and Exploitation Exploration alone is done until a point where individuals get enough experience. Thereafter, exploration and exploitation act alternatively.

Action Selection For exploration, either random or roulette-wheel action selection schemes are used, and exploitation uses only deterministic action selection.

Partially Matched Actions As a result of the extraordinarily large number of actions, it is likely that only one classifier advocating a given action exists in the action set. Therefore, crossover might not operate at all. To alleviate this problem, a classifier is inserted into the action set if at least one of its action slots matches the corresponding slot of the selected action. This is done for a small percent of generations and only during exploration.

Genetic Algorithm (GA) Parent classifiers for the GA are selected using either roulette wheel or tournament selection mechanisms. Crossover is applied only to the condition part of a classifier. A cross-site is selected randomly. Due, however, to the presence of invalid strings, not all cross-sites result in valid offspring, hence a check for validity is done. If one or more attributes in the offspring are invalid, the process is repeated until all attributes become valid or the number of trials exceeds a threshold. In the latter case, the cross-site selection will be limited to one of the junction points of the condition slots. In the original XCSs and later derivations, the mutation rate is usually kept fixed and very low. Diversity in early generations, however, is beneficial in cases where there is a relatively bigger domain for the actions than with standard XCS problems. Thus, in this work, the mutation rate is made to decrease gradually to its final, low value. If mutation generates an invalid attribute value, the change is simply rejected. Mutation is also dependent on the probability of generating hashes in new classifiers. A bit is mutated to a don't care with probability proportional to the probability of hash ($P_{\#}$), likelihood that a newly created classifier will have a '#' at a position.

Reward To ensure that individuals in early generations have sufficient time for their parameters to be estimated, they are given a chance to participate in subsequent generations rather than replaced early. The RP subsidizes such classifiers, that is, it gives them a fixed bonus.

Reward Computation The reward (R) is computed as a weighted sum of two factors,

(1) the time and exposure scores for a given weight combination in a scenario (Individual Score) and
(2) the average performance of the weight combination across different scenarios (Cumulative Score).

In order to favor the weight combination that performed best for the given scenario, more weight is given to the former. The reward is computed as

$$R = MR * \left\{ \begin{array}{c} W_{IS} * \left(\dfrac{(TI * TS) + (2 - TI) * ES}{2} \right) + \\ W_{CS} * CM(TI) \end{array} \right\} \tag{12}$$

where MR is the maximum reward (1000), W_{IS} is the weight for the individual score, W_{CS} is the weight for cumulative score, and the cumulative measure, $CM(TI)$, is computed as the average score performance of the weight combination in different scenarios, as in (13)

$$CM(TI) = \frac{\sum_m (TI * TS) + (2 - TI) * ES}{2m} \tag{13}$$

where m is the total number of scenarios. CM is a function of the time importance (TI). The parameters W_{IS} and W_{CS} are to be tuned while subject to the constraint

$$W_{IS} + W_{CS} = 1 \tag{14}$$

Thus, the reward scheme discussed above is continuous, that is, the reward given could assume any value between 0 and MR. In contrast, for a step-wise reward, either zero or the maximum reward is given, as determined by (15). This aids in favoring best classifiers to the next generation when there are many competing classifiers.

$$R = \left\{ \begin{array}{l} MR,\ if\ \left(\dfrac{(TI * TS) + (2 - TI) * ES}{2} \geq R_T \right) \\ \quad 0\ \ ,otherwise \end{array} \right. \tag{15}$$

where R_T is a reward threshold ($0 \leq R_T \leq 1$).

Another variation uses the continuous reward scheme only for some portion of the valid reward interval, that is, the reward is computed as in (12) if $\frac{(TI*TS)+(2-TI)*ES}{2} \geq R_T$; otherwise, it is reset to zero.

The reward schemes discussed thus far favor more general classifiers. However, it is not desirable to have all maximally general classifiers in the final population as it is least likely for one set of TRWA–generated weights to perform equally good across a number of scenarios. Maximally general classifiers can be screened out using a reward assignment strategy based on the level of generalization. Accordingly, (12) is modified as shown in (16) below.

$$R = MR * \left\{ \begin{array}{c} W_{IS} * \left(\dfrac{(TI * TS) + (2 - TI) * ES}{2} \right) + \\ W_{CS} * CM(TI) + W_{GEN} * (1 - LG) \end{array} \right\} \tag{16}$$

where LG is the level of generalization, which is computed as the ratio of the number of non-hashes (bits that are not don't cares) in the condition slot of a classifier to the total number of bits in a condition slot, and W_{GEN} is the weight given for the level of generalization in computing the reward. The parameters W_{IS}, W_{CS} and W_{GEN} are subject to the constraint

$$W_{IS} + W_{CS} + W_{GEN} = 1 \tag{17}$$

5. RESULTS

At first, the learner is tested for several single scenario cases. Once satisfactory results are obtained, then the scenarios are grouped together and tests conducted.

5.1 Settings

For all tests conducted, the gear preference is fixed at level 3. The maximum number of steps per problem is set to 10000, and the maximum population size is 500. Other parameters are given in Table 4.

Parameter	Meaning
$\beta = 0.2$	learning rate
$\gamma = 0.71$	discount rate
$\theta_{GA} = 25$	GA threshold
$\nu = 0.1, \alpha = 0.1,$ $\varepsilon_0 = 0.01$	used in fitness calculation
$\chi = 0.2$	probability of crossover
$\mu = 0.01$	probability of mutation per character
$\theta_{del} = 20, \delta = 0.1$	used in classifier deletion from [P]
$P_{\#} = 0.1$	likelihood of '#'
$p_I = 10, \varepsilon_I = 0,$ $F_I = 10$	initialization values for new classifier
$\theta_{sub} = 20$	if $exp > \theta_{sub}$, do subsumption
$\theta_{mna} = 8$	do covering if # of actions in $\|M\| < \theta_{mna}$
$R_T = 0.95$	reward threshold
TS	percentage tournament set size

Table 4. Regions for goal exposure

5.2 Test Results

XCS with its modifications has shown excellent performance in generating optimal weights. Without the modifications, XCS was unable to converge. Except for one case where the time and exposure scores are exceptionally low ($TL = 2$, $GT = 1$, $GE = 0.4$), satisfactory results are found. Step-wise reward computation resulted in better convergence than continuous reward computation, which did not converge at all. The use of tournament selection in place of roulette wheel selection for selecting parents for the GA, however, resulted in no better performance.

Figure 5 depicts the result (averaged over 50 iterations) obtained for the test conducted using most of the scenarios in the training dataset. The plot labelled "Correct" indicates whether the suggested weights at that iteration are considered correct. An output weight is considered correct if the corresponding average score is above the threshold value R_T. The other plot, labelled "Prediction Error", indicates an estimate of the error in the payoff prediction.

The payoff is effectively predicted and correct weights are generated by iteration 5000. Though convergence began much earlier, due, possibly, to the termination of some strategies like decaying mutation probability, partially matched actions, and early bonus, it is completed only later in the run.

6. CONCLUSION

This work has shown the use of LCSs for learning from training data. Usual XCS does not converge when using large number of action possibilities. Moreover, the search for the most optimal value for each scenario constrains the flexibility on generalization, and, the presence of several maximally scoring data points adds to this challenge. Thus, for the supervised learning problem at hand, some XCS strategies are adapted and some customized. Strategies like initialization, covering, formation of the action set (partially matched action), crossover and mutation are modified. The changes introduced proved to be fruitful and resulted in better convergence. XCS is best suited for the TRWA than other methods in the field of machine learning, such as neural network, as it provides solutions better understandable by the human-expert. The techniques developed could be adapted for other similar problems where learning from data is required. The overall results found dictate the use of XCS in the area.

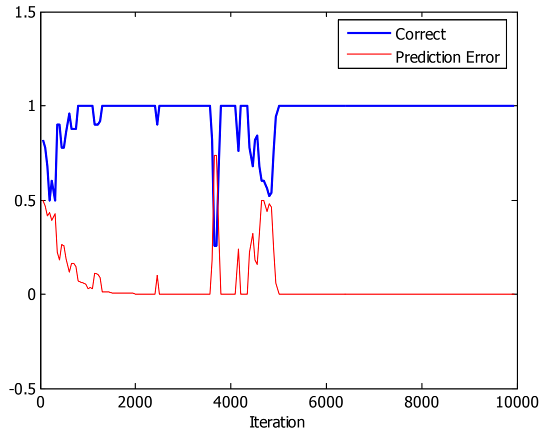

Fig. 5. TRWA Percentage of Correct Prediction and Prediction Error

ACKNOWLEDGEMENTS

This work is sponsored by the US Army Research Laboratory, under contract Robotics Collaborative Technology Alliance (Contract Number DAAD19-01-2-0012). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the US Government.

REFERENCES

A. Barry, J. Holmes, and X. Llora. Data mining using learning classifier systems.

M. Butz, T. Kovacs, P.L. Lanzi, and S.W. Wilson. How xcs evolves accurate classifiers. In *Proceedings of the 2001 Genetic and Evolutionary Computation Conference : Gecco 2001*, pages 927–934. Morgan Kauffman, 2001.

M.V. Butz, K. Sastry, and D.E. Goldberg. Tournament selection in xcs. Technical Report 2002020, IlliGAL, July 2002.

M.V. Butz and S.W. Wilson. An algorithmic description of xcs. In *Advances in Learning Classifier Systems, Proceedings of the Third International Conference - IWLCS2000*, pages 253–272. Springer, 2001.

T. Kovacs. Deletion schemes for classifier systems. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-99*, pages 329–336, 1999.

P. Ross, S. Schulenburg, J. Marin-Blazquet, and E. Hart. Hyper-heuristics: Learning to combine simple heuristics in bin-packing problems. In *Proceedings of the 2002 Genetic and Evolutionary Computation Conference : Gecco 2002*, pages 942–948. Morgan Kauffman, 2002.

M. Seeger. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2001.

S.W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

S.W. Wilson. Generalization in the xcs classifier system. In *Proceedings of the Third Annual Conference*. Morgan Kaufmann, 1998.