

Building Anticipations in an Accuracy-based Learning Classifier System by use of an Artificial Neural Network

Toby O'Hara

Technical Report – UWELCSG05-004
University of the West of England
Bristol BS16 1QY, U.K.
toby.o'hara@uwe.ac.uk

Larry Bull

Abstract- Learning Classifier Systems which build anticipations of the expected states following their actions are a focus of current research. This paper presents a mechanism by which to create learning classifier systems of this type, here using accuracy-based fitness. In particular, we highlight the supervised learning nature of the anticipatory task and amend each rule of the system with a traditional artificial neural network. The system is described and shown able to perform well in a number of well-known maze tasks.

1 Introduction

Holland's Learning Classifier System (LCS) (1976) represents a form of machine learning which exploits evolutionary computing to produce inductive structures within an artificial entity. Typically, such systems use stimulus-response rules to form chains of reasoning. However, Holland's architecture has been extended to include mechanisms by which higher levels of cognitive capabilities, along the lines of those envisaged in (Holland et al., 1986), can emerge; the use of predictive modeling within LCS has been considered through alteration to the rule structure (e.g., Riolo, 1991). Using maze tasks loosely based on those of early animal behaviour experiments, it has been found that LCS can learn effectively when reward is dependent upon the ability to accurately predict the next environment state/sensory input. LCS with such 'lookahead' typically work under latent learning, i.e., they build a full predictive map of the environment without external reinforcement. LCS of this general type have gained renewed interest after Stolzmann presented the heuristics-based ACS (Stolzmann, 1998) (see also (Gerard et al., 2002) for a related system). ACS was found to produce over-specific solutions through the workings of its heuristics and was later extended to include a Genetic Algorithm (GA) (Holland, 1975) - ACS2 (Butz & Stolzmann, 2002). Bull (2002) presented an extension to Wilson's simple payoff-based LCS - ZCS (Wilson, 1994) - that is also able to form anticipations under latent learning. Significantly, this was the first anticipatory system to build such models through the GA alone. Most current work in LCS has shifted to using accuracy as rule fitness, after Wilson presented XCS (Wilson, 1995). Bull

(2004) has recently applied the same mechanism as used in ZCS to a simple accuracy-based LCS. In this paper, a version of XCS in which each rule is represented by an artificial neural network is extended to create such anticipations is presented. Here each rule carries a second neural network by which to produce a description of the anticipated next state, trained using a traditional supervised learning scheme. That is, we present an approach to the construction of anticipations, which highlights the supervised nature of the learning task (after (Tani, 1996)).

Previously, we have presented an extension to accuracy-based LCS, which enables rules to calculate the predicted payoff expected from their use of a given action in a given state (O'Hara & Bull, 2004) (after a suggestion in (Wilson, 1995)). This was achieved by each rule carrying a traditional artificial neural network, trained under a supervised learning scheme. For a given state-action pair input, the output of the network was the expected Q-value. In this paper we successfully apply the same mechanism to the production of anticipations.

2 XCS

As noted above, Wilson (1995) introduced XCS in which rule fitness for the GA is not based on the strength of rule predictions but on the accuracy of the predictions. Each classifier maintains a prediction of expected payoff, and the classifier's fitness is given by a measure of the prediction's accuracy. The intention is to form efficient generalizations and a complete and accurate mapping of the search space, i.e. that are maximally general (Kovacs, 1997), rather than simply focusing on the higher payoff niches in the environment as in other LCS like ZCS.

In XCS on each time step, on receipt of an input message, the rule-base is scanned and any rule whose condition matches the message at each position is tagged as a member of the current match set [M]. A system prediction is then formed for each action proposed by the rules in [M] according to a fitness-weighted average of the predictions of the rules. The system action is then selected, typically either deterministically (exploit) or randomly (explore). An action set [A] is then formed, the appropriate system output given and a reward may or not be received. If [M] is empty covering is used.

Reinforcement in XCS consists of updating three parameters, Error (E), Prediction (p), and fitness (F) for each appropriate rule. Each is updated every time it belongs to $[A_{-1}]$ or $[A]$ if it is a single step problem.

XCS uses a niche-GA (Booker, 1985); the GA acts in action sets $[A]$, instead of globally. Rule replacement is based on the estimated size of each action set a rule participates in with the aim of balancing resources across niches. The GA is triggered (see also (Booker, 1989)) within a given match set if the number of time steps since its last invocation in that set passes a fixed threshold θ , based on the average time-stamp of the rules. The reader is referred to (Butz & Wilson, 2001) for full details of XCS.

3 X-NCS: A Neural LCS

The neural learning classifier system X-NCS (Bull & O'Hara, 2004) is based on XCS and the majority of its internal mechanisms are unchanged. Each traditional condition-action rule is replaced by a single, fully connected multi-layer perceptron. All rules have the same number of nodes in their hidden layers and one more output node than there are possible actions to signify not matching, the "action network" in figure 1. All weights are randomly initialized in the range $[-1.0, 1.0]$ concatenated together in an arbitrary order and thereafter determined by the GA. The system starts with an initial random population, containing the maximum number of classifiers specified in the particular experiment rather than varying N as in (Wilson, 1995).

Rule discovery operates in the same way as usual for XCS with real numbers (Wilson 2000). Hence the mutation operator is altered to adjust gene values using a normal distribution.

Previously (O'Hara & Bull, 2003), we have investigated ways in which to include backpropagation (BP) (Rumelhart & McClelland, 1986) within the "action network" of X-NCS in order to increase performance. Belew, et al. (1991) were perhaps the first to highlight the potential of combining the two search techniques in the evolution of artificial neural networks, suggesting that "local search performed by backpropagation and other gradient descent procedures is well complemented by the global sampling performed by the GA." We investigated in this context the effects of varying the amount of BP undertaken and which rules to update in various tasks. It was found that the best performance was achieved where the target for the BP was taken as the fittest classifier in the given $[A]$, where only relatively unfit classifiers in $[A]$ were updated. That is, on every formation of an $[A]$, the least fit % of rules use the output of the fittest rule within the niche as their target for the BP process.

The neural learning classifier system with lookahead X-NCS (LNN) used here to predict the next environment state is based on X-NCS and its internal mechanisms are unchanged except for the estimation of rule error. The rule error (XCS "absolute error") is the sum of prediction error

and lookahead error. Lookahead error is simply a measure whether the rule's lookahead is correct or not. It is the instantaneous value at the presentation of one example. So the lookahead error does not change over time by means of a time-averaged process, as does the standard X-NCS prediction error for example. A scheme was investigated where this lookahead error was adjusted via a Widrow-Hoff action but the performance was disrupted by the production of over-general rules, i.e., where rules cover multiple niches and are accurate in some but not others. Therefore there is not the ACS2 concept of "reliable" rules where the reliable rule is reliable if it is accurate over 90% of the time, or unreliable ones where the rule has less than 10% accuracy. Instead the anticipation accuracy of rules is based on the simple percentage of accurate anticipations per presentation. Another key difference with ternary systems is that, of course, with neural rules the wildcards are not explicit and hence there is no explicit "pass through" of input to anticipation.

3.1 Rule Network Architecture

A classifier's anticipation network, like its action network counterpart, is a fully connected MLP. Unlike the action network it has one output node for each input node, which provides the estimated input vector were the action to be undertaken (see figure 1). The output of the lookahead network translates into the next vector binary value as follows. If the output node has a value of 0.5 or greater it is given a value of 1 and less than 0.5 a value of 0. From this it can be seen that there is quite a wide range within which a node can travel, this helps robustness, and makes it easier for different vectors to be contained within the same neural network. This also prevents the nodes becoming saturated as would happen if the values were at the extremes of 1 and 0 (e.g., see (Belew et al., 1991)).

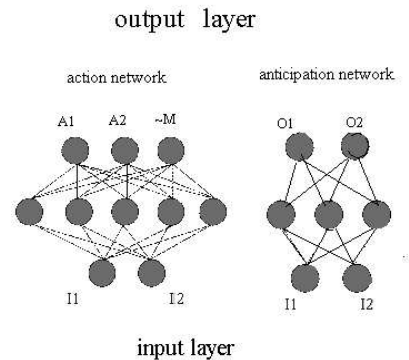


Fig. 1: Action Network with Anticipation Network.

In X-NCS (LNN) the action is chosen in the standard XCS way by the prediction array, in other words the anticipation accuracy plays no part in selection of the action unlike ACS2, which chooses actions by a product of these prediction and anticipation variables. To make lookahead error (1 or 0) of equal importance to the prediction error, a new parameter called lookahead error multiple (LEM) has been introduced, which multiplies the

lookahead error by a value between zero and the payment range. If this parameter is given the same value as the payment range, then apart from at the beginning of the trial, the rule error is weighted towards the lookahead error, if the LEM is put at or near XCS ϵ_0 then the bias will be towards prediction error.

Rule discovery operates such that crossover and mutation only act upon the action network; the second network for the child is either passed directly from either parent. The cover operator is altered such that when the action-set is empty, random neural networks are created until one gives its highest activation on an action node for the given input and a random anticipation network is generated for each new rule. Subsumption is not included.

Information is kept by the rule of each separate input vector that matches, as well as other related information such as the action produced per input. This information provides metrics for external comprehension as well as diagnostics. From this can be ascertained “input range specificity” (ir-specificity) which is similar to ternary specificity and its counterpart generality, in that it describes how many input vectors a particular rule matches.

In order to measure lookahead accuracy a metric is kept of the percentage accuracy of the rule with the highest numerosity in the action set [A]. Keeping the metric of the rule with the highest numerosity is to confirm that if the rules are extracted for modelling, this selection method can be used (after (Bull, 2004)).

3.2 Processing Cycle

The processing cycle is as follows:-

1. Matching is standard X-NCS as described above.
2. Choosing the action and getting reward from the environment is standard XCS.
3. The update cycle is slightly different as follows:
 - a) As with standard XCS the update/reinforcement is carried out on the previous action set [A_i] unless at food when it is also carried out on the current action set [A].
 - b) The lookahead value for the rule is compared to the next vector, if correct, zero is added to the prediction error and standard processing XCS continues. That is the rule error is merely the prediction error.
 - c) If the lookahead value does not equal the next vector, this next vector is fed to the BP local learning on the lookahead network. There it is used as the target value for the BP. The BP update cycle continues until either the preset number of cycles is finished (LMC) or until the rule vector matches the target next vector.
 - d) The new error value (0 or 1) is then multiplied by the LEM and added to the prediction error to form the absolute error for the rule.

4 Woods 1

Usually XCS investigations into multi-step Markov tasks start with Woods 2 (see (Wilson, 1995)) which is a toroidal grid environment containing two types of food

(encoded 110 and 111), two types of rock (encoded 010 and 011). However in this case it is problematic because in any position in the maze for the same action there may be two or more valid next vectors. This means it is difficult to solve for X-NCS (LNN).

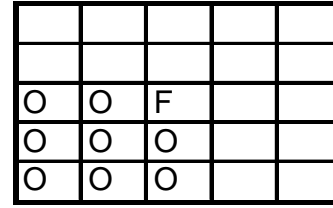


Fig. 2: Woods 1.

Therefore to test the scheme on a simple multi-step task, the Woods 1 (Wilson, 1994) maze problem has been used, which essentially is the same as Woods 2 except with one type of object and food. Woods 1 is also a toroidal grid environment containing one type of food (encoded 11), and one type of rock (encoded 01) in 3 by 3 cells, and free space (00) - figure 2. Like in Woods 2, the LCS receives the contents of its eight surrounding cells per cycle as input and can move in one of eight directions (if free space or food). The aim is to find the shortest path to a food location from a random start position. A record is kept of the moving average (over the previous 50 exploit trials) of how many steps it takes the system to move into a food cell on each trial. All results presented are the average of ten runs.

4.1 Results for Woods 1 of varying LEM

The parameters used for all the following are: $N=1800$, $\beta=0.2$, $\Phi=0.5$, $\mu=0.08$, $\alpha=0.1$, $\chi=0.8$, $\theta_{GA}=25$, $\theta_{del}=25$, $\delta=0.1$, $\epsilon_q=0.01$. Rules contain three hidden layer nodes in both action and lookahead networks. The transfer functions in both the networks are sigmoid. The learning rate for BP for both networks was the same as the XCS learning rate, β , and the momentum term was 0.5. Other investigations (not shown) had confirmed the efficacy of these values.

The effects of keeping constant maximum the number of cycles of BP each lookahead network undertakes per update (LMC) and of varying the lookahead error multiple (LEM) were investigated.

In the first experiment the LMC was kept at 20 and the LEM varied. The rationale for the LMC limit was that, on first somewhat cursory inspection of the speed of learning, most networks seem to learn well within that time.

From figure 3 it can be seen that using the neural representation alone (no LNN) requires around 1,500 problems to solve the task. It can also be seen that with the lookahead network the speed of solution varies slightly depending upon the LEM. With LEM of 10 after a slower start the lookahead solves the problem in the same time as without. For LEM of 200 and 1000 from a slower start, the problem is solved faster in approximately 800 problems. It can be seen that there is not much difference between LEM values of 200 and 1000. This indicates that

the lookahead network is improving the speed of solving the task, and that this improvement is related to the accuracy of the lookahead prediction. The results can also be seen where the LEM is zero, i.e., where the lookahead does **not** contribute to the rule error calculation. In this case the performance is similar to that without the second network.

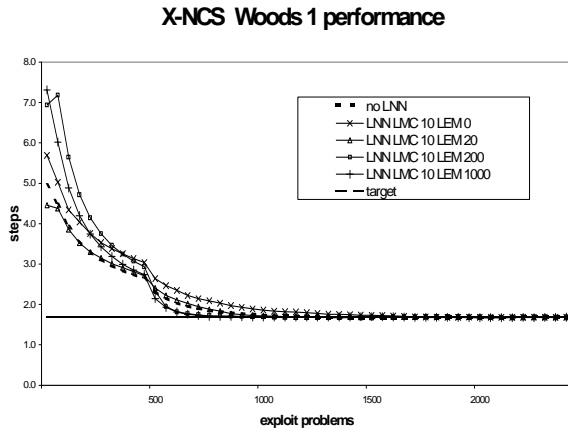


Fig. 3: Woods 1 performance varying lookahead error multiple.

Figure 4 shows the speed of lookahead learning using the metric, as discussed before, of the percentage accuracy of the most numerous rule in the action set. Not unsurprisingly the higher the LEM, and so with bias towards the lookahead learning, the faster the lookahead learning. Also that with a LEM of 20, the system sometimes struggles to reach 100% but is always well over 95%. The lookahead learning achieves optimal figures faster than it solves the problem as, not surprisingly, local learning is faster than the GA. Interestingly when the lookahead error is not part of the rule error the lookahead performance struggles to reach 40% showing that system performance is dependant on this combination of reward prediction and lookahead accuracy.

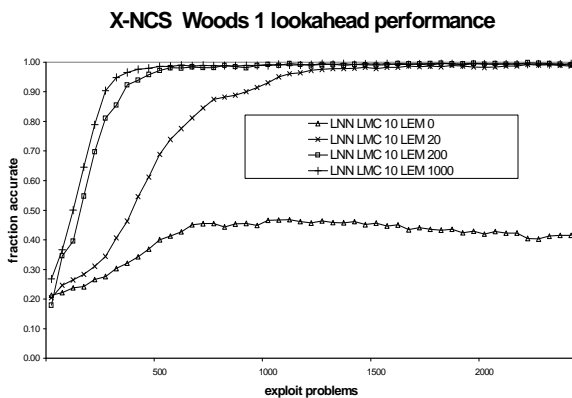


Fig. 4: Woods 1 system error varying lookahead error multiple.

X-NCS Woods 1 population

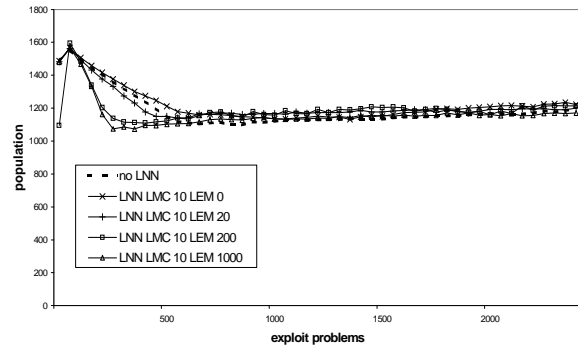


Fig. 5: Woods 1 population varying lookahead error multiple.

The lookahead network easily switches between different anticipated vectors. This is probably because the output node has only to switch between two values, 0 and 1, and secondly the ir-specificity of the rules with lookahead learning is lower than with prediction processing.

From figure 5 it can be seen that compared to non-LNN in all of the lookahead learning systems the macro population dropped more sharply and then reverted to a figure fairly close to that of the non-LNN. As the inspection of the rules shows lower generality on the LNN systems this indicates that rules become fitter and more numerous earlier.

X-NCS Woods 1 performance

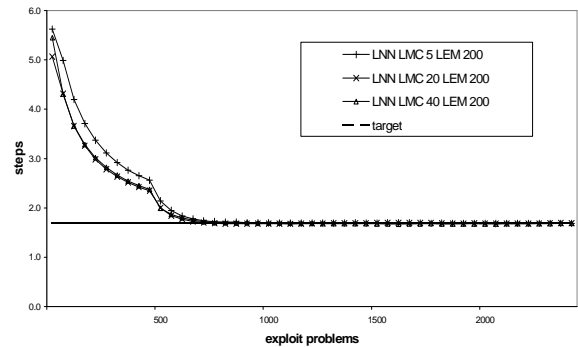


Fig. 6: Woods1 performance varying BP cycles

The effect of raising the maximum number of LNN BP cycles, the (LMC) parameter was also investigated. The value of LEM was kept at 200. Figure 6 shows that varying the number of cycles doesn't give improvements.

4.2 Woods 1 with ActNN local learning

Local learning on the ActNN was then investigated. Parameters were as before except for an LMC value of 10 cycles and LEM values of 200. Figure 7 shows system performance. Although BP always starts much better than without, and lookahead starts much slower, both combined eventually learn faster than without either. That is, BP improves the performance initially but X-NCS(LNN) solves the problem in about the same time as both separately.

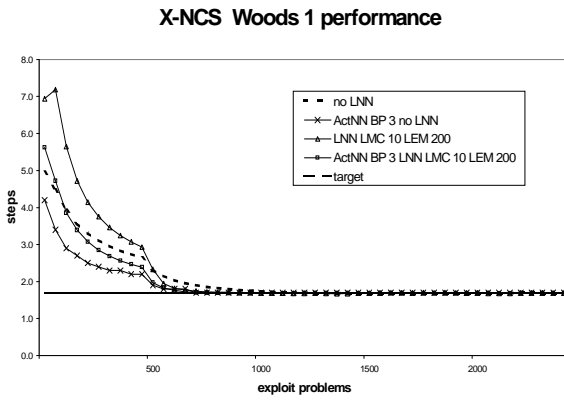


Fig. 7: Woods 1 performance with local learning.

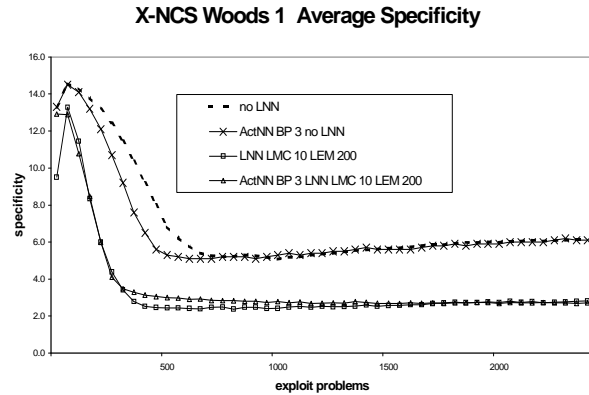


Fig. 8: Woods 1 ir-specificity.

Figure 8 shows how lookahead affects the average ir-specificity of the population. Lookahead causes rules to be more ir-specific, not surprisingly as there are two prediction hurdles for the rule to overcome. Local learning via BP seems to have no great influence on ir-specificity.

4.3 Discussion of Results for Woods 1

The results are encouraging with performance improvements, although Woods 1 is a simple task. Comparing the population and the lookahead graphs it can be seen that the population drops whilst the lookahead is learning, reflected in a population lower in ir-specificity but higher in numerosity. For subsequent experiments with the more complex Maze5 (Lanzi, 1997) there will be further investigation of LEM, LMC and re-initialisation.

5 Maze 5

Maze 5 was previously used in investigations of X-NCS with respect to action network local learning via BP (O'Hara & Bull, 2003). This investigation builds on that by combining the effect of two local learning schemes. Like Woods 1, the LCS receives the contents of its eight surrounding cells per cycle as input and can move in one of eight directions (if free space or food).

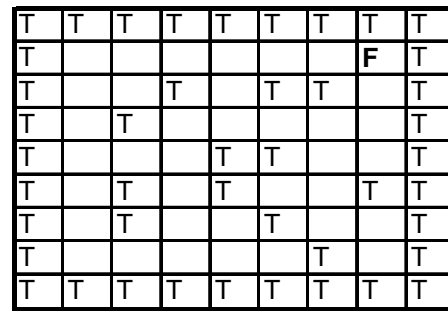


Fig. 9: Maze 5.

The aim is to find the shortest path to a food location from a random start position. Without any local learning it was found necessary to have a population of 6500 to enable the problem to be solved in a reasonable time. The parameters used for all the following as standard X-NCS for Maze 5: $N=6500$, $\beta=0.2$, $\Phi=0.5$, $\mu=0.08$, $\alpha=0.1$, $\chi=0.8$, $\theta_{GA}=25$, $\theta_{del}=25$, $\delta=0.1$, $\epsilon_0=0.01$. As in Woods 1, rules contained three hidden layer nodes in both the action network and the lookahead network and the transfer functions in both the networks are sigmoid. This is in spite of the fact that this is more difficult task than Woods 1 therefore a higher number of nodes might seem advantageous. Investigations (results not shown) showed no significant improvement in performance resulted from a higher number of hidden nodes. This is almost certainly a result of the fact that the networks have only to describe individual niches, and don't have to describe the whole problem space, and can therefore be simpler. In these experiments, the learning rate for BP for both networks was not the same as the XCS learning rate, β , a value of 0.5 was found to have better performance (results not shown) and similarly the momentum term was better at 0.2. All results presented are the average of ten runs.

5.1 Results for Maze 5

Taking into account the results from Woods 1 in the previous experiment, the maximum number of LNN cycles, (LMC) were kept at 10 as this seemed to work well and we made the LEM value 100. The rationale for this was that 1000 would result in faster lookahead learning, but 100 might produce a more reliable result. Increasing the value of the ActNN BP parameter did not increase the speed of learning (not shown).

X-NCS Maze 5 performance

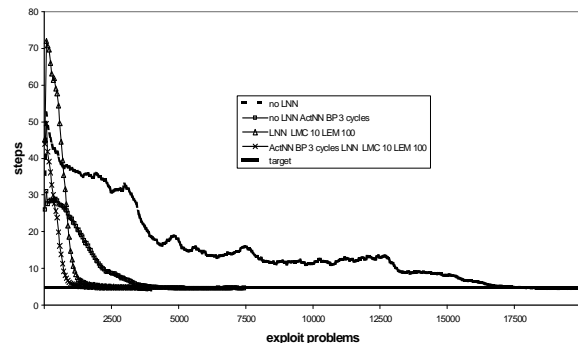


Fig. 10: Maze 5 performance.

Figure 10 shows that as before, BP acting on the system without the LNN network improves system performance and that adding the LNN improves it again here.

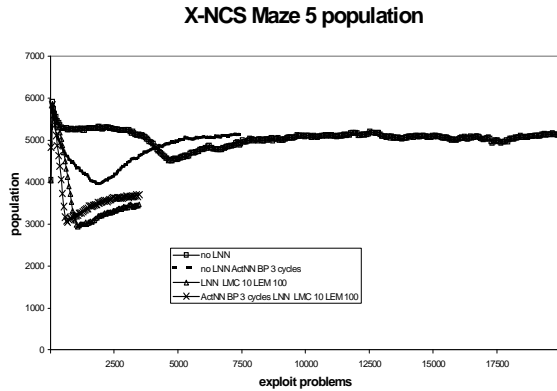


Fig. 11: Maze 5 population.

The populations in figure 11 confirm that the LNN results in an increase in generalization. In the case of lookahead and local learning the trials were stopped when the task was learned, as run times for these experiments are relatively long, hence these lines are truncated.

In terms of lookahead learning performance, it can be seen from figure 12, that results from Woods 1 are confirmed; BP in the action network increases the speed of learning.

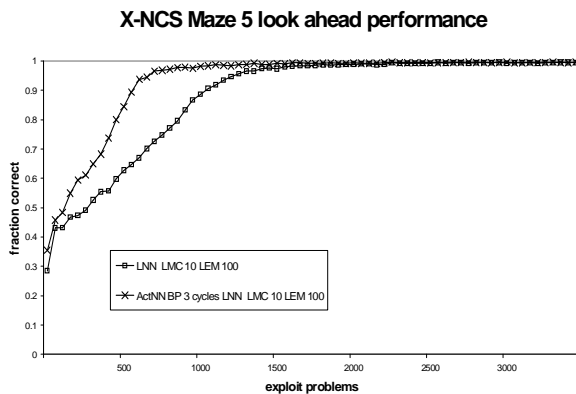


Fig. 12: Maze 5 lookahead performance.

5.2 Discussion of Results for Maze 5

This task is a relatively difficult multi-step problem and it appears that speed and reliability must be balanced to give both lookahead performance and reliability in solving the task. As far as lookahead learning is concerned even if the task is not solved, efficient lookahead learning still takes place. As the lookahead learning uses local learning this is not surprising, and confirms the value of having local learning running in addition to global learning, and the idea (Belew et al., 1991) that local learning can add extra functionality to evolving a neural system.

The results show good performance and the fact that the LNN works well with local learning on the ActNN is very promising.

6 Conclusions

This paper has presented results from using a separate lookahead neural network within an accuracy-based learning classifier system. By using the lookahead network the generality of the rules decreases compared to rules without a second network. As expected the increase in irspecificity significantly improves performance, i.e., it decreases the number of problems it takes to solve the task. This is especially so with harder problems, such as Maze 5. The effect of the second task seems to have an additional specialization pressure. It appears to act in parallel with the action network and complement and reinforce it. The investigation of the use of this second lookahead network in combination with using BP on the Action network showed surprisingly that considering two networks are being trained at the same time they work well together, and the performance improves. That is even though both networks are being altered, and in the case of the Action network possibly to change the action of the rule, and hence probably a different next vector. Lookahead learning via the supervised scheme seems to work well, working in parallel with the normal XCS mechanisms.

Bibliography

- Belew, R.K., McInerney, J. & Schraudolph, N.N. (1991) Evolving Networks: Using the Genetic Algorithm with Connectionist Learning. In C.G. Langton, C. Taylor, J.D. Farmer & S. Rasmussen (eds) *Artificial Life II*, Addison-Wesley, pp511-548.
- Booker, L. (1985) Improving the Performance of Genetic Algorithms in Classifier Systems. In J.J. Grefenstette (ed.) *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, Lawrence Erlbaum Assoc., pp80-92.
- Booker, L.B. (1989) Triggered Rule Discovery in Classifier Systems. In J.D. Schaffer (ed.) *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, pp265-274.
- Bull, L. (2002) Lookahead and Latent Learning in ZCS. In W.B. Langdon, E.Cantu-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A.C. Schultz, J. F. Miller, E. Burke & N. Jonoska (eds) *GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, pp 897-904.

- Bull, L. (2004) Lookahead and Latent Learning in a Simple Accuracy-based Learning Classifier System. In X. Yao et al. (eds) *Parallel Problem Solving from Nature - PPSN VIII*. Springer Verlag, pp1042-1050.
- Bull, L. & O'Hara, T. (2002) Accuracy-based Neuro and Neuro-Fuzzy Classifier Systems. In W.B.Langdon, E.Cantu-Paz, K.Mathias, R. Roy, D.Davis, R. Poli, K.Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A.C. Schultz, J. F. Miller, E. Burke & N.Jonoska (eds) *GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, pp905-911.
- Butz, M. & Wilson, S.W. (2001) An Algorithmic Description of XCS. In P-L. Lanzi, W. Stolzmann & S.W. Wilson (eds) *Advances in Learning Classifier Systems: IWLCS 2000*. Springer, pp253-272.
- Butz, M.V. & Stolzmann, W. (2002) An Algorithmic Description of ACS2. In P-L. Lanzi, W. Stolzmann & S.W. Wilson (eds) *Advances in Learning Classifier Systems: Proceedings of the Fourth International Workshop*. Springer, pp211-230.
- Gerard, G. Stolzmann, W. & Sigaud, O. (2002) YACS a new Learning Classifier System using Anticipation *Soft Computing* 6(3-4), 216-228.
- Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press.
- Holland, J.H. (1976) Adaptation. In R. Rosen & F.M. Snell (Eds) *Progress in Theor. Biology 4*. Plenum.
- Holland, J.H., Holyoak, K.J., Nisbett, R.E. & Thagard, P.R. (1986) *Induction: Processes of Inference, Learning and Discovery*. MIT Press.
- Kovacs, T. (1997) XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions. In *Soft Computing in Engineering Design and Manufacturing*. Springer, pp59-68.
- Lanzi, P-L (1997) A Model of the Environment to Avoid Local Learning. In *Technical Report N. 97*. Politecnico di Milano.
- O'Hara, T. & Bull, L. (2003) Backpropagation in Accuracy-based Neural Learning Classifier Systems. *Technical Report UWELCSG03-007*. Available from <http://www.cems.uwe.ac.uk/lcsg>.
- O'Hara, T. & Bull, L. (2004) Prediction Calculation in Accuracy-based Neural Learning Classifier Systems. *Technical Report UWELCSG04-004*. Available from <http://www.cems.uwe.ac.uk/lcsg>.
- Riolo, R.L. (1991). Lookahead planning and latent learning in a classifier system. In Meyer, J.-A., & Wilson, S. W. (Eds.), *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior* pp. 316-326. Cambridge, MA: MIT Press.
- Rumelhart, D.E. & McClelland, J.L. (1986) *Explorations in Parallel Distributed Processing*. MIT Press.
- Stolzmann, W. (1998). Anticipatory Classifier Systems In *Genetic Programming 1998* pp. 658-664. University of Wisconsin, Madison, Wisconsin: Morgan Kaufmann.
- Tani, J. (1996). Model-based learning for mobile robot navigation from the dynamical system perspective. *IEEE Transactions on System, Man and Cybernetics*, 26(3) :421-436.
- Wilson, S.W. (1994). ZCS: A zeroth level classifier system. *Evolutionary Computation* 2(1): 1-18, 1994.
- Wilson, S.W. (1995). Classifier Fitness Based on Accuracy. *Evolutionary Computing* 3: 149-175
- Wilson, S.W. (2000) Get Real! XCS with Continuous-Valued Inputs. In P-L. Lanzi, W. Stolzmann & S.W. Wilson (eds) *Learning Classifier Systems: From Foundations to Applications*. Springer, pp209-222.