10

Abstraction for Genetics-Based Reinforcement Learning

Dr Will Browne, Dan Scott and Charalambos Ioannides
University of Reading
UK

1. Introduction

Abstraction is a higher order cognitive ability that facilitates the production of rules that are independent of their associations.

In standard reinforcement learning it is often expedient to directly associate situations (states) with actions in order to maximise the environmental reward signal. This may lead to problems including a lack of generalisation and not utilising higher order patterns in complex domains. Thus standard Q-learning has been developed to include models or genetics-based search (Learning Classifier Systems), which improve learning speeds and generality. In order to extend reinforcement learning techniques to higher-order rules, abstraction is considered here.

The process of abstraction can be likened to Information Processing Theory (a branch of Learning Theory) (Miller, 1956), which suggests that humans have the ability to recognize patterns in data and chunk these patterns into meaningful units. The individual patterns do not necessarily remain in a memory store due to the holistic nature of the individual patterns. However, the chunks of meaningful information remain, and become a basic element of all subsequent analyses.

The need for abstraction arose from the data-mining of rules in the steel industry through application of the genetics-based machine learning technique of Learning Classifier Systems (Holland, 1975), which utilise a Q-learning type update for reinforcement learning. It was noted that many rules had similar patterns. For example, there were many rules of the type 'if side guide setting < width, then poor quality product' due to different product widths. This resulted in a rule-base that was unnecessarily hard to interpret and slow to learn. The initial development of the abstraction method was based on the known problem of Connect4 due to its vast search space, temporal nature and available patterns.

The contribution of this chapter is that the novel method of abstraction is described and shown to be effective on a large search space test problem. Abstraction enabled higher order rules to be learned from base knowledge, which mimic important aspects of human cognition. Tests showed that the abstracted rules were more compact, had greater utility and assisted in developmental learning. The emergence of abstracted rules corresponded with escaping from local minima that would have otherwise trapped basic reinforcement learning techniques, such as standard Q-learning.

Open Access Database www.i-techonline.com

Source: Reinforcement Learning: Theory and Applications, Book edited by Cornelius Weber, Mark Elshaw and Norbert Michael Mayer
ISBN 978-3-902613-14-1, pp.424, January 2008, I-Tech Education and Publishing, Vienna, Austria

2. Background

During the application of the Genetics-Based Machine Learning technique of Learning Classifier Systems (LCS) to data-mine rules in the steel industry, Browne noted that many rules had similar patterns (Browne 2004). For example, there were many rules of the type 'if side guide setting < width, then poor quality product' due to different product widths. This resulted in a rule-base that was unnecessarily hard to interpret and slow to learn. A method is sought to generate higher order (abstracted) rules from the learnt base rules.
A novel Abstraction algorithm has been proposed (see figure 1) to improve the performance of a reinforcement learning genetics-based machine learning technique in a complex multi-step problem (Browne & Scott, 2005). It is hoped that this algorithm will help reinforcement learning techniques identify higher-order patterns inherent in an environment.

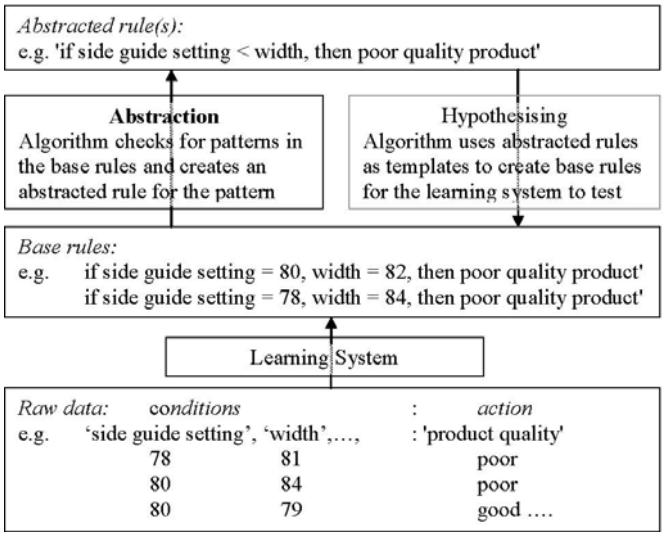

Fig. 1. Abstraction from data to higher order rules.

2.1 Test domain

Connect 4 is a turn-based game between two players, each trying to be the first to achieve four counters in a row (horizontally, vertically or diagonally). The game takes place on a 7 * 6 board; players take it in turns to drop one of their counters into one of the seven columns. The counters will drop to the lowest free space in the column. Play continues until the board is full or one player gets four in a row, see figure 2. Optimum strategies exist (Allis, 1988; Watkins, 1989), so the problem is both known and bounded.
A client-server program of Connect 4 was written in Java, as Java Applets can easily be viewed on the internet, allowing a website to be constructed for this project [please visit: http://sip189a.rdg.ac.uk].
A Q-Learning (Sutton & Barto, 1998) approach to the problem is implemented in order to provide benchmark learning performance. Two different approaches were taken to training the Q-Learning system. The first progressively trained the algorithm against increasingly hard opponents, whilst the second trained for the same number of games, but against the hardest opponent from the outset.

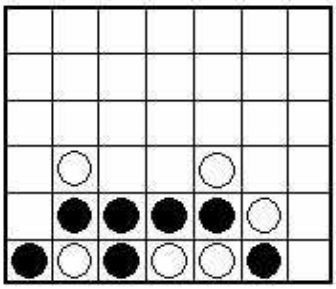

Fig. 2. Connect 4 board, black horizontal win

The Abstraction algorithm requires rules in order to perform abstraction. A well-known LCS, XCS (Butz, 2004) was implemented to create rules and provide a second benchmark learning performance.

3. Biological inspiration for abstraction

The human brain has inspired artificial intelligence researchers, such as the development of Artificial Neural Networks that model aspects of low-level neuronal activity. Higher-level functional modelling has also been undertaken, see ACT-R and SOAR architectures (Anderson et al, 2004; Laird et al, 1987). Behavioural studies suggest that pattern recognition, which includes abstraction, is important to human cognition. Thus this section considers how the brain abstracts. This includes using the common neuroscience technique of studying subjects with liaisons to specific brain areas.

It has been observed in cases of autism that there is a lack of abstraction. A well studied case is that of Kim Peek -due to his Savant abilities and popularity as the inspiration for the main character in the film Rain Man. He was born with macrocephaly (an enlarged head), an encephalocele (part of one or more of the skull plates did not seal) and agenesis of the corpus callosum (the bundle of nerves that connects the two hemispheres of the brain is missing). Brain studies, such as MRI, show that the there is also no anterior commissure and damage to the cerebellum.

Kim has the ability to analyse certain types of information in great detail, e.g. Kim's father indicates that by the age of 16-20 months Kim was able to memorize every book that was read to him. It is speculated that neurons have made other connections in the absence of a corpus callosum, resulting in the increased memory capacity (Treffert & Christensen, 2005). However, Kim has difficulty with motor skills, such as buttoning a shirt, which is likely to be caused by the damaged cerebellum as it normally coordinates motor activities. His general IQ is well below normal, but he scores very highly in some subtests.

An absent corpus callosum (ACC) does not regenerate as no new callosal fibers emerge during an infant's development. Although people with ACC lead productive and meaningful lives there are common developmental problems that may occur with disorders of the corpus callosum (DCC). The National Organization for Disorders of the Corpus Callosum states:

> Behaviorally individuals with DCC may fall behind their peers in social and problem solving skills in elementary school or as they approach adolescence. In typical development, the fibers of the corpus callosum become more efficient as

children approach adolescence. At that point children with an intact corpus callosum show rapid gains in abstract reasoning, problem solving, and social comprehension. Although a child with DCC may have kept up with his or her peers until this age, as the peer-group begins to make use of an increasingly efficient corpus callosum, the child with DCC falls behind in mental and social functioning. In this way, the behavioral challenges for individuals with DCC may become more evident as they grow into adolescence and young adulthood.

Behavioural characteristics related to DCC difficulties on multidimensional tasks, such as using language in social situations (for example, jokes, metaphors), appropriate motor responses to visual information (for example, stepping on others' toes, handwriting runs off the page), and the use of complex reasoning, creativity and problem solving (for example, coping with math and science requirements in middle school and high school, budgeting) (NODCC, 2007).

The connection between the left and right half of the brain is important as each hemisphere tends to be specialised on certain tasks. The HERA model asserts that the left pre-frontal cortex is associated with semantic (meaning) memory, whilst the right is associated with episodic (temporal) memory (Tulving et al., 1994). Memories themselves are associated with the hippocampus, which assists in transforming short to long term memory. This is intact in many savants, such as Kim Peek. Thus, it is postulated here that a link is needed between the separated episodic and semantic memory areas in order for abstract, higher order, knowledge to form -it is not sufficient just to create long-term generalised memories.

A caveat of the above analysis is that even with modern behavioural studies, functional MRI, PET scans and other neurological analysis, the brain/mind is highly complex, plastic and still not fully understood.

4. Learning classifier systems

This section outlines the architecture of XCS, including the required adjustments for the Connect 4 domain, so that it may train against a pre-coded expert system. A standard XCS (Butz, 2004, available from www-illigal.ge.uiuc.edu/) was implemented with the Abstraction algorithm (see section 5). Following these results tests were also conducted with a modified version of XCS (mXCS) that had its reinforcement learning component adjusted to complement the Abstraction algorithm.

4.1 Setup and board representation

The board representation formed an important part of the LCS. Each space on the board could be one of three possible states, red, yellow or empty, however it was considered useful to further split down the empty squares into two categories, playable and unplayable (unplayable squares are above the playable squares and become playable in the future as the game progresses).

A two character representation for each space was chosen, leading to an 84 character long string representing the board (running from top row to bottom row). The encoding for a red was chosen as “11” and a yellow was “10”, a playable space was “00” whilst an unplayable was “01”. Mutation may only generalize by replacing specific characters with a “#”; where hashes can stand for either a “1” or a “0”.

4.2 Gameplay and reward

LCS must decide upon the best move to play at its turn without knowing where its opponent will play in the subsequent turn. An untrained LCS will often play randomly as it attempts to learn the best moves to play. After each move has been played by the opponent, the LCS attempts to match the state of the board to its rules. Attached to each of these classifiers are three pieces of information: the move that should be played, the win score (the higher this is the more likely a win will occur) and the accuracy score (accuracy of the win score). Win scores of less than 50 indicate a predicted loss, greater than 50 is a projected win. After matching, an action must be selected through explore, exploit or coverage. Exploring (which is most likely to happen) uses a weighted roulette wheel based on accuracy to choose a move. Exploiting chooses the move that has the greatest win score and is used for performance evaluation. Coverage generates a new rule by simply selecting a random move to play for the current board position.

θ_{GA} the GA threshold was set to 1000 games, the GA would run after a set of 1000 games had been played and the maximum population size was set to 5000. χ, the crossover possibility was set to generate 500 random crossovers every time the GA is run. Of the 500 crossovers generated, approximately 100 in every GA run passed validity checks and were inputted into the new population. μ, the mutation rate was set at a 1% chance to receive a mutation and then a 2% that each character in that rule would receive a mutation. Deletion θ probabilities (θ_{del}) were based upon tournament selection of rule fitness and the number of rules deleted was chosen to keep the population size at 5000.

The standard reinforcement update for LCS is the Widrow-Hoff update (Butz & Wilson, 2002), which is a recency weighted average. A Q-learning type update is used within the LCS technique for multistep decision problems (Lanzi, P-L., 2002).

5. Abstraction algorithm

The Abstraction algorithm was designed to work upon generated rules, e.g. by the LCS. Abstraction is independent of the data itself. Other methods, such as the standard coverage operator, depend directly on the data. Crossover and mutation depend indirectly on the data as they require the fitness of the hypothesized rules, which is dependent on the data. Abstraction is a higher order method, as once good rules have been discovered; it could function without the raw data being available.

The abstraction attempts to find patterns in the rules that performed best within the LCS. Having found a pattern common to two or more of the LCS rules, the Abstraction algorithm is to generate a new rule in the abstracted population based solely on this pattern. This allows the pattern to be matched when it occurs in any state, not just the specific rules that exist within the LCS.

Not all of the rules generated by the LCS are worthwhile and therefore the Abstraction algorithm should not be run upon all of the rules within the LCS. The domain is noiseless, so the parameters chosen to govern the testing of rules for abstraction were the conditions that a rule must have a 100% win score and a 100% accuracy. Therefore the rules abstracted by the Abstraction algorithm should only be rules that lead to winning situations.

The main mechanism that allowed the abstraction to perform was a windowing function that was used in rule generation as well as rule selection (when it came to choosing an abstracted rule to play). The windowing function acted as a filter that was passed over the

'good' rules generated by the LCS. This filter would compare two rules at a time for similarities that could lead to abstracted rules.
The windowing function worked in all directions on the board, horizontally, vertically and in both diagonal directions. The window size was set to 4 space/counters (8 characters in terms of the board representation). However code allowed for a window size of between 4 and 6 spaces/counter (8 – 12 characters in terms of the board representation), any greater than a window size of 6 and the vertical and diagonal windows no longer fit on the board.
Any match that is found is turned into an abstracted rule, each rule had 8 characters (assuming a window size of 4) to represent the pattern occurring on the board. Each rule also had to be assigned a move to play whenever that rule was used. The move assigned was always chosen from one of the playable spaces within the pattern. An example rule is '10,10,10,00:11', which translate to 'if three red counters in a row and payable space in the next position, then play in the next position'. All rules entered the abstracted population with a win and accuracy of 50.
Several limitations were placed upon what was considered a valid match for the Abstraction algorithm, including ignoring all unplayable areas. A valid pattern had to contain at least one playable space and no more than 2 playable spaces. Patterns without a playable space are useless because rules as they offer nowhere for a move to be played. The second limitation placed upon the abstraction process was that a valid rule could have a maximum of one unplayable space. This helps limit the generation of "empty" rules. Figure 3 shows an example of two windowing functions finding a match and generating an abstracted rule.

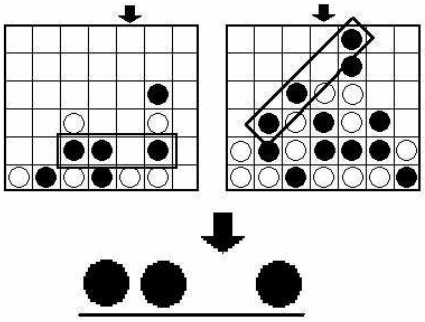

Fig. 3. Example of Abstraction Algorithms generating a new rule.

5.1 Base rule discovery

As with the LCS, the Abstraction algorithm also had a GA that was run upon the population to generate new rules. It had a single point crossover and mutation; however it had no deletion algorithm as all the abstraction rules were kept. Duplication was prevented through a duplication check that was made each time a rule was to be inserted into the rule-base, including those generated by crossover and mutation.
A LCS can function alone, but the Abstraction algorithm cannot function without a rule-base to work on; hence it needs an LCS to function alongside it. How the two are combined and work together is detailed in this section.
When the LCS with abstraction needs to play a move, the system searches the board for any matches within its abstracted rule set. The board is searched by passing the windowing function over the board (horizontally vertically and diagonally). A rule is then chosen out of

all matched rules. When exploiting the rule with the best win score is chosen, whilst when exploring a roulette wheel based upon accuracy is used.
The chosen abstracted rule also has a move associated with it, however unlike the LCS rules the move does not relate directly to the board. With a window size of 4 counters the rule could occur anywhere on the board, horizontally, vertically or diagonally. Therefore an extra calculation is required to translate the abstracted rules' move into the corresponding move on the actual board.
If no abstracted rule is found after the initial search of the board state, then control of playing the move is handed to the base LCS.

6. Results

The following section details the results found during the trials of the LCS and Abstraction algorithm. Initial trials investigated the difficulty of the problem domain with standard Q-learning and XCS techniques. Preliminary tests of the Abstraction algorithm with XCS were followed by tests of the Abstraction algorithm with a modified XCS (mXCS) where the reinforcement learning complemented the abstraction. The use of abstraction as the training progressed was investigated. During these tests, each system was trained for 20,000 games against an opponent that played randomly. Finally, the robustness of the Abstraction algorithm to changes in the domain was tested by increasing the difficulty of the opponent.

6.1 Q-Learning and standard XCS

The Q-Learning Algorithm performed well in the initial 20,000 games (see figure 4), achieving an average win percentage of 69%. However, there was no progress in the wins as the 20,000 games progressed, with the win percentage always remaining at around 69%. This exhaustive search nature of the algorithm meant it took several weeks of computation on a 3GHz PC. Ideally, each test would have been repeated 10 times and the average results taken, but this was impractical due to time constraints.
The XCS performance trend was similar, with an average win percentage of 62% reached quickly, but no further improvements. Analysis of the rules showed that they had become trapped in local optima. A few specific strategies had been learnt, such as initially trying to build a column of counters in a given column. However, if this column happened to be blocked, then the overall strategy failed.

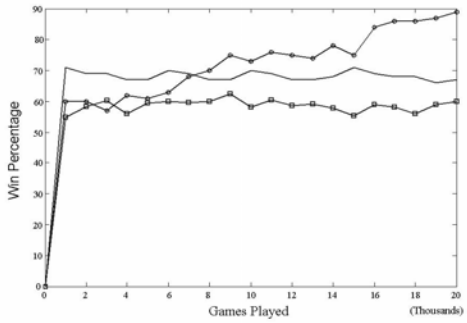

Fig. 4. Graphs of win percentages for the 3 algorithms Solid Line -Q-Learning Algorithm, square -XCS Algorithm, circle -mXCS with Abstraction.

When the Abstraction algorithm is added it produces a similar trend until 6000 trials. A significant improvement is noted after 8000 trials as the performance increases to 90%. This compares favorably with both Q-learning (69%) and standard XCS (62%).
During testing the rules that the Abstraction algorithm produced were observed and an interesting pattern arose in the order in which the abstractions were discovered. In early generations no abstracted rules are found, whilst mXCS attempts to establish a set of good rules that have a win and accuracy of 100. The first abstracted rules found are not rules for a direct win (i.e. 3 in a row and play in the fourth). The first rules that emerge are those rules that cause a 3 in a row situation with an empty playable fourth space.
Learning to form 3 in a row followed by learning to form 4 in a row is a novel example of incremental learning. Intuitively, it could be expected that learning to form 4 in a row, which is closer to obtaining the reward, would be achieved first. Incremental learning is hypothesized to be an important cognitive ability (Butz, 2004).
Whilst there is no direct feedback from the abstraction rule-base to the mXCS rule-base, it is possible to see them evolve together and there is a definite dependency between the two. With the introduction of abstracted rules to make 3 in a row, this is likely to occur far more often (as abstracted rules take preference over mXCS rules). With 3 in a row occurring more often, mXCS has more opportunities to conceive of rules that directly give a win. Therefore, with more winning rules the Abstraction algorithm is more likely to come up with abstracted rules that lead to a direct win, greatly bolstering the winning ability of the algorithm.

6.2 Effect of abstraction

The use of abstracted rules as training progresses can be monitored, see figure 5. As outlined in section 5, the combined system always plays a matching abstracted rule in preference to a matching base rule. After 8000 trials the base rules were accurate enough to allow abstraction to start. Once abstraction had started, the performance of the system continued to improve beyond that of standard XCS and Q-learning (see figure 4). A further 8000 trials occur where the system uses a combination of both base and abstracted rules. After this period the system just uses abstracted rules in its decision-making. Small improvements in performance occurred due to the action of the genetic algorithm in the abstracted population.

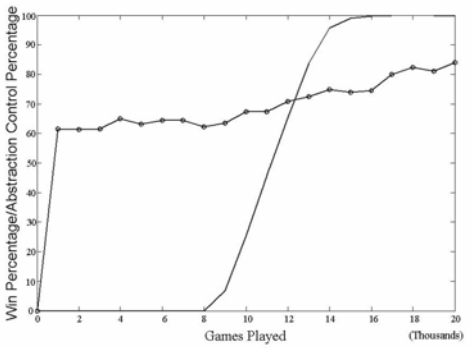

Fig. 5. Graph of percentage base rules versus abstracted rules (solid line) as training progresses (circle line).

The random opponent still defeats the system in 10% of the games when it chances upon a good strategy. As there are multiple positions for good strategies to occur in, the system is rarely presented with them, which makes them difficult to learn. In order to determine the robustness and scalability of the techniques the difficulty of the opponent was increased.

6.3 Robustness of the systems

The opponent could now block a potentially winning three in a row state. The system has to learn to create multiple win situations. This is a significantly harder problem, especially as the opponent could win either randomly or in the act of blocking, which halts the game. All algorithms perform poorly as all win percentages are under 20%. If no good base rules are found, then the Abstraction algorithm will not start.

Instead of training from the start with the harder opponent, it was decided to train first with the simple opponent and then switch to the harder opponent, see figure 6. After the switch, standard XCS performed better than the Q-Learning Algorithm, achieving a win percentage of 15%, it should be noted that the performance was less than the Q-Learning algorithm during the first 20000 games. Analysis of the Q-Learning algorithm testing showed that progressive training, from the easiest to the hardest opponent, caused it to get stuck in a local optimum with a win percentage of only 11%. The generality and adaptability of the standard XCS algorithm enables it to switch opponent without as great a penalty.

The performance of the Abstraction algorithm was significant. Not only did it outperform standard XCS and Q-learning (53%, compared with 15% and 11% respectively), but it performed significantly better then when it had been trained only on the harder opponent (53% compared with 19%). This is a good example of incremental learning, where it is necessary to build up the complexity of the problem domain.

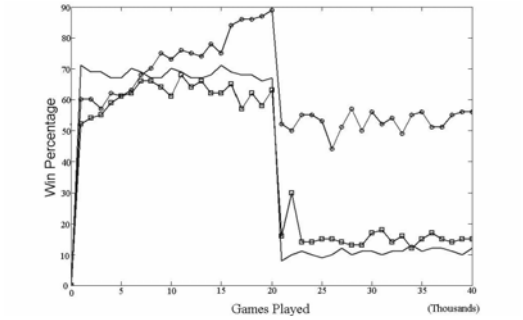

Fig. 6. Change in opponent at 20x103 games played (Solid Line -Q-Learning Algorithm, square -XCS, circle -mXCS with Abstraction).

The concept of abstraction has been applied to the alternative domain of the Multiplexer problem (Browne & Ioannides, 2007). This was to test if a different representation (alphabet) could be used between the initial population (e.g. binary alphabet) and the abstracted population (e.g. s-expression alphabet). Result showed on hypothesised base data that abstraction is capable of scaling well on the formed rules (max length 1034 bits compared with 84 bits for the Connect4 domain). A significant advantage was the compacting of the rule-based, see figure 7 and see table 1, compared with a bit string of 1034. Abstraction also selected the most appropriate functions within the s-expressions for the domain (two from a possible 10).

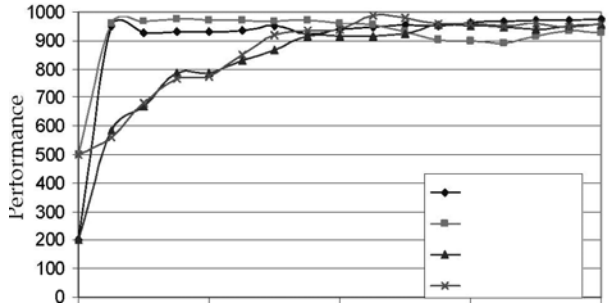

Fig. 7. Abstracted XCS with hypothesised base rules on 3-MUX ♦, 6-MUX ■, 135-MUX ▲, 1034-MUX **x** problem

MUX	Condition	Length
3	VALUEAT OR 2 2	4
3	VALUEAT AND 2 2	4
3	VALUEAT ADDROF 2 2	4
3	VALUEAT AND 2 POWEROF 1	5
3	VALUEAT OR POWEROF 2 2	5
3	VALUEAT OR POWEROF 1 2	5
6	VALUEAT ADDROF 4 5	4
6	VALUEAT ADDROF 5 4	4
6	VALUEAT ADDROF 4 POWEROF 5	5
6	VALUEAT ADDROF POWEROF 3 4	5
6	VALUEAT ADDROF POWEROF 5 4	5
135	VALUEAT ADDROF 128 134	4
135	VALUEAT ADDROF 134 128	4
135	VALUEAT ADDROF POWEROF 22 128	5
135	VALUEAT ADDROF 128 PLUS 133 134	6
1034	VALUEAT ADDROF 1033 1024	4
1034	VALUEAT ADDROF PLUS 1029 1029 1024	6
1034	VALUEAT ADDROF MULTIPLY 1025 324 1024	6
1034	VALUEAT ADDROF PLUS 1029 1024 1024	6
1034	VALUEAT ADDROF MULTIPLY 1029 324 1024	6
1034	VALUEAT ADDROF PLUS 1033 1033 1024	6

Table 1. Abstracted rules on 3-MUX, 6-MUX, 135-MUX & 1034-MUX problem

7. Discussion

Abstraction may appear a trivial task for humans and the positive results from this work intuitive, but abstraction has not been routinely used in genetics-based reinforcement learning.
One reason is that the time each iteration requires is an important consideration and abstraction increases the time for each iteration. Typically XCS takes 20 minutes to play 1000 games (and remains constant), mXCS with abstraction takes 20 minutes for 100 games (although this can vary greatly depending on the choice of parameters) and the Q-Learning algorithm ranges from 5 minutes for 1000 games initially to 90 minutes for 1000 games after 100,000 games training. However, given a fixed amount of time to train all three algorithms mXCS with abstraction would perform the best, once the initial base rules were found.
The Q-Learning algorithm has to visit every single state at least once in order to form a successful playing strategy. Whilst the Q-Learning system would ultimately play a very good game, weeks of computation failed to achieve the level of success the Abstraction algorithm had in a very short space of time (hours rather than weeks). Although better Q-learning algorithms (including generalization capabilities) exist (Sutton & Barto, 1998) this choice of benchmark algorithm showed the scale of the problem, which is difficult to calculate.
The improvement in abstraction performance from standard XCS to the modified XCS was due to using simpler reinforcement learning. The Widrow-Hoff delta rule converges much faster, which for simpler domains that can be solved easily is beneficial. However, slower and more graceful learning may be required in complex domains when interacting with higher level features.
The abstracted rules allow the system to play on states as a whole, including those that have not been encountered, where these states contain a known pattern. This is useful in data-mining, but with the inherent dangers of interpolation and extrapolation. The abstracted rule-base is also compact as an abstracted rule covers more states than either a generalized LCS rule or a Q-learning state. Unique states may still be covered by the base rules.
Abstraction has been shown to give an improvement in a complex, but structured domain. It is anticipated that the Abstraction algorithm would be suited to other domains containing repeated patterns.

8. Future work

Instead of the current linear filters in the Abstraction algorithm, it is possible to vary the size and shape in order to represent and hopefully discover advantageous multi-win situations. The abstraction method is static and determined a priori, which is successful for this structured domain. The next stage is to evolve the abstracted rules and/or filters thus reducing the searching time.
A process termed 'hypothesizing' is proposed (see figure 1) where the abstracted rules form a template in order to produce new rules for the base population, with the worth of the abstracted rule being determined by the success of their hypothesized rules.

9. Conclusion

A novel Abstraction algorithm has been developed to successfully improve the performance of a genetics-based machine learning technique in a complex multi-step problem. It is hoped

that this algorithm will help to fulfill the intended use of the LCS technique as a test bed for artificial cognitive processes.

10. Acknowledgements

Our thanks to the Nuffield Foundation for their support through grant NUF-URB04.

11. References

Allis, V. (1988). *A Knowledge Based Approach of Connect 4*. Masters Thesis, Vrije Universiteit, Netherlands.

Anderson, JR.; Bothell, D.; Byme, MD.; Douglass, S.; Lebiere C. & Qin Y (2004). An integrated theory of the mind. *Psychological Review* 111 4, pp. 1036-1060.

Browne, WN. & Scott, D. (2005). An abstraction agorithm for genetics-based reinforcement learning. *GECCO 2005*, editors Hans-Georg Beyer et al. Washington D. C., USA, pp. 1875-1882.

Browne, WN. & Ioannides, C. (2007) Investigating Scaling of an Abstracted LCS Utilising Ternary and S-Expression Alphabets. *International Workshop on Learning Classifier Systems*, London

Browne, WN. (2004). The development of an industrial learning classifier system for data-mining in a steel hot strip mill. *Applications of Learning Classifier Systems*. Bull, L. (Ed.), pp. 223-259, Springer, Berlin.

Butz, M. & Wilson, SW. (2002). An algorithmic description of XCS. *Soft Computing: a fusion of foundations, methodologies and applications*, 6 pp. 162-170.

Butz, M. (2004). *Rule-base evolutionary online learning systems: learning bounds, classification and prediction*. PhD thesis University of Illinois, Illinois.

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

Laird, J.; Newell, A. & Rosenbloom, P. (1987) Soar -An architecture for general intelligence. *Artificial Intelligence* 33 pp. 1-64.

Lanzi, P-L. (2002). Learning classifier systems from a reinforcement learning perspective. *Soft Computing: a fusion of foundations, methodologies and applications*, 6 pp. 162-170.

Miller, GA. (1956). The magical number seven, plus or minus two; Some limits on our capacity for processing information. *Psychological Review*, 63, pp. 81-97.

NODCC. (2007) National Organization for Disorders of the Corpus Callosum http://www.nodcc.org

Sutton, RS. & Barto, AG. (1998). *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA.

Treffert, DA. & Christensen, DD. (2005) Inside the Mind of a Savant *Scientific American* pp. 50-55.

Tulving, E.; Kapur, S.; Craik, FIM.; Moscovitch, M. & Houle. S. (1994) Hemispheric encoding/retrieval asymmetry in episodic memory: positron emission tomography findings. Proc. Natl. Acad. Sci. U. S. A. 91, pp. 2016–2020

Watkins, CJCH. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, England.

Reinforcement Learning

Edited by Cornelius Weber, Mark Elshaw and Norbert Michael Mayer

ISBN 978-3-902613-14-1
Hard cover, 424 pages
Publisher I-Tech Education and Publishing
Published online 01, January, 2008
Published in print edition January, 2008

Brains rule the world, and brain-like computation is increasingly used in computers and electronic devices. Brain-like computation is about processing and interpreting data or directly putting forward and performing actions. Learning is a very important aspect. This book is on reinforcement learning which involves performing actions to achieve a goal. The first 11 chapters of this book describe and extend the scope of reinforcement learning. The remaining 11 chapters show that there is already wide usage in numerous fields. Reinforcement learning can tackle control tasks that are too complex for traditional, hand-designed, non-learning controllers. As learning computers can deal with technical complexities, the tasks of human operators remain to specify goals on increasingly higher levels. This book shows that reinforcement learning is a very dynamic area in terms of theory and applications and it shall stimulate and encourage new research in this field.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Will Browne, Dan Scott and Charalambos Ioannides (2008). Abstraction for Genetics-Based Reinforcement Learning, Reinforcement Learning, Cornelius Weber, Mark Elshaw and Norbert Michael Mayer (Ed.), ISBN: 978-3-902613-14-1, InTech, Available from: http://www.intechopen.com/books/reinforcement_learning/abstraction_for_genetics-based_reinforcement_learning

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2008 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.