Computer Science 448
Evolutionary Computation

Christopher Mark Gore
chris-gore@earthlink.net
http://www.cgore.com

Wednesday, October 4, AD 2006

1 Reinforcement Learning [2]

Reinforcement learning is the process of learning how to map situations to actions to maximize a numerical reward. The learning system is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by exploration. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. There are the two primary distinguishing characteristics of reinforcement learning:

1. trial-and-error search and
2. delayed reward.

Reinforcement learning is defined not by characterizing learning methods, but by characterizing a learning problem. Any method that is well suited to solving that problem, we consider to be a reinforcement learning method. The idea is to capture the most important aspects of the problem facing the learning agent interacting with its environment to achieve its goal. Such an agent must be able to

1. perceive the state of the environment,
2. act on the environment, and
3. have a goal or goals relating to the state of the environment.

More simply put: *sensation*, *action*, and *goal*.

1.1 Exploration versus Exploitation

A primary challenge is the trade-off between exploration and exploitation. A reinforcement learning agent will prefer actions that it has tried in the past and found to be effective in producing reward in order, to reliably obtain more reward. But to discover such actions, it has to try actions that it has not selected before. The agent has to *exploit* existing knowledge to obtain reward, but it also must *explore* to make better action selections in the future. Neither exploration nor exploitation can be pursued exclusively without failure. The agent must try a variety of actions and progressively favor those that appear to be best. On a stochastic task, each action must be tried many times to gain a reliable estimate its expected reward. The exploration-exploitation dilemma has been intensively studied by mathematicians for many decades.

1.2 The Whole Problem

Reinforcement learning explicitly considers the whole problem of a goal-directed agent interacting with an uncertain environment, starting with a complete, interactive, goal-seeking agent, instead of considering sub-problems without addressing how they might fit into a larger picture. All reinforcement learning agents have explicit goals, can sense aspects of their environments, and can choose actions to influence their environments. From the beginning the agent operates with significant uncertainty about its environment. For learning research to make progress, important subproblems have to be isolated and studied, but they should be subproblems that play clear roles in complete, interactive, goal-seeking agents, even if all the details of the complete agent cannot yet be filled in.

2 Learning Classifier Systems

A learning classifier system is a type of evolutionary algorithm in which a description of a current situation is used in an attempt to map that description to some classification or action. This is achieved through simulated evolutionary processes, where the population being evolved consists of various rules; our entire population forms a rule set, and we apply concepts from Darwinism to our individual rules. This is known in learning classifiers as the *Michigan approach*. The other primary method employed, where each individual is an entire solution and therefore a whole rule set is known as the *Pittsburgh approach*. We use a modification of XCSR here, which uses the Michigan approach, and our method also uses the Michigan approach.

2.1 ZCS

ZCS is a *zeroth level classifier system* originally proposed in [3]. ZCS preserves most of the functionality of traditional learning classifier systems, but it is a very simplified version, which aids in understanding of the classifier and its actions. This was a very useful contribution, because much of the problems with learning classifiers before then was their very baroque nature. A good short summary of ZCS can be found in [1], and this summary is based primarily on that.

In ZCS there is no message list, a much-welcomed simplification of the traditional LCS. This comes with a cost: there is no explicit method for transmitting information between cycles without the message list. This makes the interface entirely dependent on the interface of the system with its environment, and thus assumes a Markov process. This is most definitely an invalid assumption for real-world traded markets and for other time-series data.

Each rule r is of the form $r = (c, a, s)$ where:

- c is the condition matched by r, comprised of elements from the set $\{0, 1, \#\}$, where # is the matching symbol, matching both 0 and 1;
- a is the action that the rule r recommends;
- and s is a strength measurement of the rule r, $s \in \mathbb{R}$.

In each cycle t the match set M_t is found, $M_t \subseteq P$, with P being the entire population of rules (the rule set.) M_t can be divided into disjoint subsets based on the action they recommend. With a finite set of possible actions

$$\mathcal{A} = \{a_0, a_1, \ldots, a_{|\mathcal{A}|}\} \tag{1}$$

and $\mathcal{A}' \subseteq \mathcal{A}$ where

$$\mathcal{A}' = \{a'_0, a'_1, \ldots, a'_{|\mathcal{A}'|}\} \tag{2}$$

comprises all of the actions represented in M_t, $M_{t,a'_i} \subseteq M_t$ with

$$M_{t,a'_i} = \{r \colon r \in M_t \wedge a_r = a'_i\}\,, \tag{3}$$

the fitness of an action $a'_i \in \mathcal{A}'$ is

$$\text{fitness}(a'_i) = \sum_{r \in M_{t,a'_i}} s_r. \tag{4}$$

The action to take is selected in a fitness-proportionate method, choosing action a' with the greatest fitness. If $M_t = \emptyset$ then covering must take place; a random rule that matches the current situation is created by initially setting c to exactly the current situation and then replacing a few elements of c at random with the # symbol, and that suggests a randomly-selected action.

The credit assignment scheme used by ZCS is somewhat involved, and is referred to as an *implicit bucket brigade*. It attempts to reward sequences which lead to reward from the environment and which are short.

1. Rules in the population but excluded from the match set are originally unchanged:
$$s'_r = s_r \forall r \notin M_t. \tag{5}$$
2. Rules in the match set but excluded from the action set A_t (those advocating weaker actions than the one chosen) have their strengths reduced by a factor $\tau \in [0,1)$:
$$s'_r = \tau s_r \forall r \in M_t \setminus A_t. \tag{6}$$
3. The strength of the rules in the current action set have a fraction $\beta \in [0,1)$ of their strengths transferred to the members of the previous action set, reduced by a factor $\gamma \in [0,1)$:
$$s'_r = (1-\beta)s_r \forall r \in A_t, \tag{7}$$
$$s''_r = s'_r + \frac{\gamma \sum_{r \in A_t} \beta s_r}{|A_{t-1}|} \forall r \in A_{t-1}. \tag{8}$$
4. Any feedback P_t from the environment is reduced by β and distributed to the rules in the current action set:
$$s'''_r = s''_r + \frac{\beta P_t}{|A_t|}. \tag{9}$$

Algorithm 1: ZCS credit assignment

A mostly standard GA is run on the population (the rule set) periodically, with parent selection directly related to s and death selection inversely related s. The new rules are usually assigned the mean of their parents' strength initially.

2.2 XCS

ZCS has many positive features, especially its simplicity and the benefits derived from its cooperative fitness sharing, but there are some notable drawbacks, primarily that it usually will not evolve a complete mapping of the environmental states and allowable actions to the possible rewards, often quickly selects local optima, and breeds across niches. These features led Wilson to heavily modify ZCS into what is called XCS [4]. In XCS, several of the deficiencies in ZCS are addressed.

In ZCS, the GA is run on the entire population, a *panmictic* approach. This is ineffective for most problems, so in XCS the GA is run only in the match set originally, and only in the action set in the later versions. This allows for a more accurate rule set to be evolved, since each niche is best viewed as its own sub-problem.

In ZCS, a rule is allowed to continue by the GA on the basis of its payoff. This is problematic, since it biases against rules early in a chain of events that are eventually profitable, and because rules that may be the most appropriate for an event might have a relatively low payoff. This caused ZCS to often fail to create a complete mapping, and to often fail to evolve accurate generalizations. This is remedied in XCS by creating a fitness measure for the rules, separate from the predicted payoff, used by the GA.

Each rule r is now of the form

$$r = (c, a, p, \epsilon, F, exp, ts, as, n), \tag{10}$$

where:

- c is the condition matched by r, comprised of elements from the set $\{0, 1, \#\}$, where # is the matching symbol, matching both 0 and 1.
- a is the action that the rule r recommends.
- p is the predicted payoff.
- ϵ is an estimate of the prediction error.
- F is the fitness used by the GA. It is vital that the fitness used by the GA is a measure of the *accuracy* of the rule, and not a measure of the *magnitude*.
- exp is the experience of the rule, a count of the number of times since this classifier's creation that it has belonged to the action set.
- ts is a time stamp of the last occurrence of a call to the GA in an action set that this classifier was a part of, as the generational number.
- as is an estimate of the average action set size this classifier has belonged to.
- n is the numerosity of this macro-classifier. This is how many traditional micro-classifiers this macro-classifier represents.

2.3 XCSR

Wilson extended his concept of XCS with XCSR in [5]. Classifier systems had typically taken binary strings only as input until then, even though many real-world problems have input from the environment of the form $\mathbb{R}^n$ for some $n \in \mathbb{Z}, n > 0$. Wilson's XCSR allows XCS to operate on just such an input. XCSR is identical to normal XCS except for the input interface, the nature of the predicates, the mutation operator, and the details of covering.

Originally the predicates in XCSR were intervals of the form

$$interval_i = \{center_i, spread_i\}, \tag{11}$$

such that an environmental input x_i was matched by $interval_i$ iff

$$center_i - spread_i \leq x_i \leq center_i + spread_i, \tag{12}$$

but this was discovered to induce a bias, so the representation was eventually changed to be

$$interval_i = \{lower_i, upper_i\}, \tag{13}$$

where now x_i is matched by $interval_i$ iff

$$lower_i \leq x_i \leq upper_i. \tag{14}$$

We use the $\{lower, upper\}$ form in this work.

Crossover is simple two-point crossover, but on the sequence

$$\{center_0, spread_0, \ldots\} \tag{15}$$

or

$$\{lower_0, upper_0, \ldots\} \tag{16}$$

depending on the predicate type, in both cases therefore allowing the crossover points to fall within a single allele.

Mutation is performed by adding a small random quantity from the range $[-0.1, 0.1]$ to each allele. The variation of XCSR used here is capable of scaling outside of $[0.0, 1.0]$, so instead mutation is performed as the addition or subtraction of a small percentage. This change from traditional XCSR was necessary due to the nature of the problem at hand.

References

[1] EIBEN, A. E., AND SMITH, J. E. *Introduction to Evolutionary Computing.* Springer-Verlag, 2003.

[2] SUTTON, R. S., AND BARTO, A. G. *Reinforcement Learning.* The MIT Press, Cambridge, Massachusetts, 1998.

[3] WILSON, S. W. ZCS: A zeroth level classifier system. *Evolutionary Computation 2*, 1 (1994), 1 – 18.

[4] WILSON, S. W. Classifier fitness based on accuracy. *Evolutionary Computation 3*, 2 (1995), 149 – 175.

[5] WILSON, S. W. Get real! XCS with continuous-valued inputs. In *Learning Classifier Systems: From Foundations to Applications* (2000), P. L. Lanzi, W. Stolzmann, and S. W. Wilson, Eds., vol. 1813 of *Lecture Notes in Artificial Intelligence (LNAI)*, Springer-Verlag, pp. 209 – 219.