Abstract

In reinforcement learning problems, the sparse nature of the feedback makes credit assignment difficult. Consequently, feature extraction becomes much more difficult. Others have addressed this difficulty by use of bottom-up methods which aim to produce a feature set corresponding to the probability density function of the inputs. In contrast, we present a new approach which aims to bias feature extraction toward features which are important, rather than those which are merely frequent. To do this, we propose a model for on-line feature extraction in which the feature detectors are tuned by a bottom-up, competitive interaction, but this tuning is guided by top-down information as to the importance of the features. We define important features to be features which are highly correlated with the reinforcement feedback rather than those which are seen often. We have implemented these ideas in a simple neural network model, and we present results for a simple game application. A unique feature of our model is its use of a monitor circuit to avoid saturation of the output nodes, and thereby constrain feature extraction to yet unrepresented features.

1 Introduction

In reinforcement learning problems the feedback is simply a scalar value which may be delayed in time. This reinforcement signal reflects the success or failure of the entire system after it has performed some sequence of actions. Hence the reinforcement signal does not assign credit or blame to any one action (the *temporal credit assignment* problem), or to any particular node or system element (the *structural credit assignment* problem).

In contrast, in supervised learning the feedback is available after each system action, removing the temporal credit assignment problem; in addition, it indicates the error of individual nodes instead of simply telling how good the outcome was. Supervised learning methods, for instance back-propagation, off-line clustering, mathematical optimization, and ID3, rely on having error signals for the system's output nodes, and typically train on a fixed set of examples which is known in advance. But not all learning problems fit this paradigm. Reinforcement learning methods are appropriate when the system is required to learn on-line, or a teacher is not available to furnish error signals or target outputs. Examples include:

Game playing If there is no teacher, the player must be able to determine which actions were critical to the outcome and then alter its heuristics accordingly.

Learning in a micro-world The agent must develop the ability to categorize its perceptions, and to correlate this awareness of its environment with the satisfaction of primitive drives such as pleasure and pain.

On-line control Controllers of automated processes such as gas pipelines or manufacturing systems must adapt to a dynamically changing environment, where the optimal heuristics are usually not known.

Autonomous robot exploration Autonomous robots may make feasible exploration of hazardous environments such as the ocean and outer space, using on-line learning to adapt to changing and unforeseen conditions.

Our work focuses on feature extraction, the development of the system's input representation. Since the reinforcement feedback is not an error signal for individual system elements, it gives little guidance for feature extraction. One reason is that if the system fails by choosing the wrong action, the feedback does not specify which of the output nodes was wrong. In a system which chooses its action by selecting the most active output node, an error can be caused by either having a node be too active for a given input, or by other nodes not being active enough.

If the system has a hidden layer of feature detectors, another reason for poor feature extraction is that acting properly depends on both identifying the current context as well as selecting an action appropriate to that context. A scalar feedback signal does not indicate which of these processes is at fault. The feedback does not distinguish between the case where the system rightly identified its context but selected the wrong response, and the case where the system's learned responses are correct, but its feature detectors misidentified the context. In terms of a typical neural network implementation, the system needs to know whether it should tune its feature detectors, or the weights placed on the outputs of those feature detectors, or both. The sparse reinforcement signal does not furnish this information. Thus learning methods for reinforcement learning problems may need bottom-up information or some type of internal feedback to supplement the top-down feedback.

The next section of the paper presents the reinforcement learning model used in the experiments. Section 3 presents a test application and the results obtained, and section 4 discusses the model in terms of importance-based feature extraction. Section 5 reviews related work, and section 6 discusses some extensions of the model for future work.

First we describe the architecture and algorithm of our learning model.

2 A Simple Reinforcement Learning Model

2.1 The Network Architecture

We illustrate our feature extraction ideas with a simple neural net system for reinforcement learning. The system is shown in Figure 1. The inputs come from a layer of receptor nodes, and feed into a winner-take-all layer of effector nodes. The winning effector node controls the system action; if the effector activations are very close, the winner is chosen randomly. This simple model contains no hidden nodes for feature detection; consequently, each effector node acts as a feature detector which is sensitive to the input pattern specified by its input weights. The weights are tuned by a combination of internal and external feedback. The internal feedback comes from a simple monitor circuit which regulates the level of activation across the effector layer. When too many effectors are active, the monitor circuit sends negative feedback, and when no effectors are active, the monitor circuit sends positive feedback. The external feedback is a reinforcement signal of -1 if the system failed in this trial, or 0 otherwise; this feedback is given to the winning effector node at the end of the trial. Each effector tunes toward or away from the current input pattern, according to whether it has received positive or negative feedback.

The effector nodes must compete to represent the current input pattern because the monitor circuit limits their aggregate response. Nodes which trigger inappropriate actions for the state represented by the input pattern are penalized in this competition by the negative external feedback. As a result, these nodes are tuned away toward other input patterns, and the remaining effectors become more likely to win out on the current input pattern. In this way, the nodes learn to recognize states of the environment in which their actions are helpful to the performance of the system.

In addition, our system shows that importance-based feature extraction need not be incompatible with the spirit of local, "neurally-plausible" computation. The winner-take-all network layer could be implemented in terms of lateral inhibition links between effectors (Feldman and Ballard, 1982). The monitor circuit could also be implemented in terms of local computation; for example, the positive feedback half might be implemented by a node which sums the effector's activation, where this summation node inhibits another node which provides a wave of positive feedback to the effectors.

2.2 The Weight Update Algorithm

In our notation, x_i is the activation of receptor node i, y_j is the output of effector j, and $w_{i,j}$ is the weight on the link from receptor i to effector j. $x_i \in [0.0, 1.0]$, $y_j \in [0.0, 1.0]$, and $w_{i,j} \in [-1.0, 1.0]$. The output value of effector node j is $y_j(t) = f(\sum_i x_i(t) w_{i,j}(t-1))$, where f is a sigmoid function with output in the range [0.0, 1.0]. The winning effector node is chosen as the effector with greatest output value, $y_{max} = \max_j\{y_j(t)\}$.

The weight update algorithm is as follows:

$$w_{i,j}(t) = \begin{cases} w_{i,j}(t-1) + \alpha x_i(t)(FB_j(t) - w_{i,j}(t-1)) & \text{if } FB_j(t) \neq 0 \\ w_{i,j}(t-1) & \text{otherwise} \end{cases}$$

Here

$$FB_j(t) = r_j(t) + m(t),$$

the sum of the external and internal feedback seen by effector j. If effector j is not chosen as the winning effector at time t, $r_j(t) = 0$, but if j is the winner, then $r_j(t) = r(t)$, the reinforcement feedback given by the environment:

$$r(t) = \begin{cases} 0 & \text{if the move was not a terminal move, or result was success} \\ -1 & \text{if the move resulted in a failure} \end{cases}$$

$m(t)$ is the internal feedback given by the monitor circuit, where

$$m(t) = \begin{cases} \Delta & \text{if } \max\{y_j(t)\} < \epsilon \\ -\Delta & \text{if } \sum y_j(t) > E \\ 0 & \text{otherwise} \end{cases}$$

α is the learning rate and Δ is the magnitude of the internal feedback. Δ has a magnitude slightly smaller than that of the external feedback, -1. Our intent was to give internal feedback an influence comparable to that of the external feedback without overriding it. ϵ and E determine when there is too little or too much competition among effectors, respectively. The monitor circuit parameter tunings apparently reflect the number of detectors competing, the size of their input space, and the frequency of external feedback for the task. For our experiments, the system performed well for $\Delta \in [0.5, 0.8]$, $\epsilon \in [0.01, 0.5]$, and $E \in [4.5, 6.5]$.

2.3 Analysis

We note first that the weight values $w_{i,j}$ are bounded, as a consequence of the boundedness of the receptor values and the feedback. The combined feedback signal, $FB_j(t) = r_j(t) + m(t)$, lies in [-1.65, 0.65]. We can rewrite the update rule as

$$w_{i,j}(t) = (1 - \beta(t)) w_{i,j}(t-1) + \beta(t) FB_j(t),$$

where $\beta(t) = \alpha x_i(t) \in [0.0, 0.10]$. Since $\beta(t) \in [0, 1]$, the updated value $w_{i,j}(t)$ is a linear combination of $w_{i,j}(t-1)$ and $FB_j(t)$. Therefore, $w_{i,j}(t)$ is bounded by those values. In particular, since $w_{i,j}(0)$ is initially bounded by [-1.0, 1.0], and $FB_j(t)$ is bounded by [-1.65, 0.65], it is easy to show that $w_{i,j}(t)$ is bounded by [-1.65, 1.0] for all t.

By recursively applying the update rule, we see that $\mathbf{w}_j$, the vector of weights for effector j, is a time-averaged trace of terms $\vec{x}(t)FB_j(t) = \vec{x}(t)(m(t) + r_j(t))$ (see Appendix). Here $\mathbf{w}_j = (w_{1j}, w_{2j}, \ldots, w_{mj})$ and represents the pattern which excites effector j. $\vec{x}(t) = (x_1(t), x_2(t), ..., x_m(t))$ is the input pattern at time t. Although all the effectors see the same internal feedback m, they receive different external feedback, r_j. If the winning effector's action proves unsuccessful, that effector is penalized in the competition for the current input pattern, $\vec{x}$; consequently, $\mathbf{w}_j$ is tuned away by an additional term $\alpha\vec{x}r$ which the other effectors do not see. This makes the other detectors more likely to win the competition for this pattern in the future. The result is that, over time, the system tries other effectors (actions) for this pattern (system state) until it finds an action which does not result in failure from this state. The effectors which tune away are pushed to other areas of feature space where they may represent more appropriate behaviors for the system. Therefore, the system uses external feedback to tune effectors toward important features—those which indicate a situation where the effector's action is useful.

Instead of merely avoiding failure, the system may benefit from maximizing some indication of success. If the task includes positive reinforcement for successful trials, r_j may be positive; in this case, the winning effector would be tuned closer to $\vec{x}$. Therefore it would gain a competitive advantage over the other effectors for this pattern. So far, our studies have been limited to reinforcement learning where reinforcement is only given upon failure.

2.4 The Role of the Monitor Circuit

The internal feedback, $m(t)$, produced by the monitor circuit plays an important role in our algorithm. Without the monitor circuit, the system did not converge, even when it was made more powerful by addition of a layer of hidden nodes or inclusion of a temporal-difference learning component. With the monitor circuit, our system outperformed that of Robinson and Fallside, which used a combination of back-propagation and temporal-difference learning.

The monitor circuit provides the system with internal feedback in addition to the reinforcement feedback provided by the environment. In this way it is similar to Sutton's Adaptive Heuristic Critic (AHC) algorithm (Sutton, 1984). The difference is that the AHC aims to provide top-down feedback regarding the system's performance, while the monitor circuit provides bottom-up feedback regarding the distribution of the effectors.

The monitor circuit affects the system's performance in several ways. First, the monitor circuit helps control the dynamic range of the effector activations, as discussed above. This insures that there will be an active effector which triggers an action, but that choosing an effector will not become a meaningless random decision due to a saturation of the effector nodes.

Second, the monitor circuit causes the effectors to explore the input space. The monitor circuit drives the effectors to recognize "unclaimed" system states $\vec{x}$, by adding an update term $\vec{x}(t)m(t) > 0$ to the link weights for all the effectors. But the monitor circuit also prevents these effectors from all trying to recognize the same states, by sending a negative update term, $\vec{x}(t)m(t) < 0$, to the link weights when too many effectors are active.

Finally, the monitor circuit is a key element in importance-based feature extraction. When unsuccessful effectors are tuned away from responding to a particular situation, the system must search for alternative actions which will result in better feedback. The monitor circuit is the mechanism which draws other effectors in, by sending positive internal feedback. Since the unsuccessful effectors are also receiving negative external feedback, the other effectors see better feedback in comparison, and are drawn in to replace the unsuccessful effectors. Without the monitor circuit this competition of effectors would not be present. The monitor circuit leads the system to choose the best effectors for each system state, and causes the exploration of the input space to be done by unsuccessful effectors in order to find system states for which they will be more effective.

3 An Application

We have used tic-tac-toe as the initial application for our study of importance-based feature extraction. Although tic-tac-toe is a trivial game for humans or computers with search algorithms, it is a difficult task to learn purely by reinforcement learning. The learner is given no guidance as to the target action to be learned or the direction of weight updates, and the sparse feedback is delayed until the end of the game. These properties make tic-tac-toe a useful initial study for importance-based feature extraction in reinforcement learning.

3.1 Methodology

We trained the neural network system against an optimal opponent: a minimax-based search program which always chooses the best move and which chooses randomly between moves with equal-valued outcomes. Since this opponent never errs, the best result our network could achieve is a draw game. Training proceeded as follows. Before each run, the network's weights, $\{w_{i,j}\}$, were initialized to random values in [-1.0, 1.0]. The network moved first in each game, and was given a reinforcement signal at the end of each game: -1 for a loss, or 0 for a draw game. Training continued until the network learned to play at least 50 draw games in a row, or until game number 4000, whichever came first.

The receptors represent the state of the system in the environment. We tested the system using two different receptor configurations. The original receptor configuration incorporates the following encoding scheme: Two receptor nodes are always set to the value 1.0. The rest are sensitive to a particular row, column, or diagonal, and are stimulated by the marks of one player and inhibited by marks from the other player. Eight of these nodes are stimulated by the network's marks, while the remaining eight are stimulated by the minimax player's marks. There are 18 receptor nodes in all. This representation scheme allows the learning system to do credit assignment in terms of simple pre-wired features: the row-by-row count of marks.

We also wanted to test whether this simple system could discover good feature detectors from scratch; we ran a few simulations using a minimalist representation which lacked any pre-wired features other than the presence of marks on the board. In this receptor scheme, each square on the board has two associated receptors, one of which senses the presence of the network's mark on that square, and the other senses the presence of the opponent's mark. In addition, two receptors are permanently set to the value 1.0, bringing the total to 20 receptor nodes in the minimalist representation. The receptor vector is then normalized to unit length. The challenge to the system is to develop feature detectors on its own for the concepts of “row”, “column”, and “diagonal”.

Finally, our simulation of the winner-take-all effector layer incorporates a legal move filter, which inhibits effectors corresponding to illegal moves. Therefore, the most active effector always corresponds to a legal move. We feel that this is a reasonable addition to the model, since we could easily add an additional sensor for each board position, which simply inhibits the corresponding effector when that position is occupied. Attempting to combine the learning of features for legal play with those of good strategy apparently makes the feature set much more complex, especially since features for the rules of play tend to be very abrupt in the input space.

3.2 Results

Using the original receptor configuration we ran 15 trials with different random initializations. All the trials converged to correct play, requiring an average of 563 games before converging. The standard deviation was 497 games. The feedback seen by the network for a typical trial is shown in Figure 2. The feedback value plotted for each game is actually the sum of the external feedback (gotten at the game's conclusion) added together with the monitor circuit's internal feedback updates (which are possible after any move). A running average of the feedback is shown in Figure 3, which is a plot of the function $y(n) = 0.9y(n-1) +$ (the cumulative feedback for game n), where $y(0) = 0$. These graphs show the stability of the system once correct performance has been achieved and the external feedback becomes zero; the monitor circuit's internal feedback also goes to zero as the system learns to distribute its effectors according to their relative importance in different situations.

We also ran several simulations with the minimalist receptor configuration and a slightly different tuning for the monitor circuit parameters ($\epsilon = 0.10$ and $E = 5.5$). This system seemed more sensitive to the initial tuning of the effectors. In a set of 5 trials, the network converged to correct play in 4 of the trials, averaging 1168 games until learning was complete, with a standard deviation of 1272 games.

For comparison, we note the results of (Robinson and Fallside, 1989) in their application of reinforcement learning to the tic-tac-toe task. Their system played against a much weaker opponent than ours did and showed poorer results. Their system played against an opponent which either completed a line of two to win a game, or else moved randomly if no such win moves were available. At the beginning of training their network moved randomly but still won approximately 30% of the time, indicating the weakness of its opponent. After training on over 300000 games, Robinson and Fallside's system beat the same opponent 59% of the time. In contrast, our network learned correct play in less than 3000 games and played to a draw against an optimal opponent 100% of the time, although one of our five runs failed to converge. Robinson and Fallside's system may have fared better if it had included a legal move filter as ours did; however, we feel that the main reason for our system's better performance is that it incorporates importance-based feature extraction.

4 Importance-Based Feature Extraction

The reinforcement signal does not indicate how the input representation ought to be tuned, since it simply indicates the overall success of the entire system over a period of time. Since the top-down feedback is not sufficient to guide feature extraction, we may wish to supplement it with a bottom-up self-organization of the feature detectors (the effectors, in our system). A bottom-up competitive interaction among feature detectors can produce a feature set which spans the input space; however, without any sort of top-down influence, the feature set will be independent of the goals of the learning task. (While a particular system may not have explicit goals, we may consider its goals to be whatever state/action combinations result in maximizing the value of its reinforcement signals). Therefore, the system may benefit from biasing this self-organization toward features which are "important."

We define "important" features to be those which enable the system to recognize situations where one action is preferable over another. One action is preferable over another if it is likely to lead to better reinforcement than the other. If the system is designed to *model* the input space, all areas of the space may be equally important. In contrast, if the learning task is a *control* task, then the only important features are those which distinguish between contexts requiring different behaviors. Therefore, important areas of feature space are those for which the values of the system's actions differ greatly. Regions of feature space for which the system's actions have the same values are *un*important, no matter how frequently these regions are encountered. The concept of importance is illustrated by the human tendency to make distinctions for classes of features which are useful to us. For example, both Eskimos and skiers have reason to make more distinctions regarding types of snow than a typical non-skier living in Florida, to whom all snow may be just "snow."

One consequence of importance-based feature extraction is that infrequently occurring features may be important if they reliably signal the need for taking a particular action. Frequently occurring features may be unimportant if they are not relevant to the system's goals. Therefore, importance-based feature extraction may produce a feature set which looks very different from the probability density function of the inputs.

In our system, the monitor circuit provides the bottom-up competition which distributes the effectors across the input space. Since there is no separate layer of feature detectors in this model, the effectors themselves are the feature detectors. Therefore, finding important features means tuning the effectors to respond when their associated actions are preferable over the others. This judgement of importance is given by the external feedback, which indicates the value of the winning effector's action in the current context. In unimportant areas of the input space, all the system's actions will have similar values; therefore, external feedback will not distinguish between the effectors here. But in important areas of input space, the external feedback and monitor circuit act together to drive feature extraction toward important features.

If the winning effector's action is harmful in the current context, the negative external feedback tunes it away from responding to similar situations. The monitor circuit's internal feedback draws detectors to tune toward the current input point; however, the negative external feedback seen by the "harmful" effector gives the other effectors an advantage over it, and they are drawn in to replace it. In this way the monitor circuit causes the system to search for effectors which are preferable in the given context. The result is that the effectors are tuned to respond to situations (represented by areas of input space) where their actions are helpful to the system. These effector tunings correspond to important features.

5 Related Work

Most research in reinforcement learning explores feature extraction either purely in terms of the top-down feedback, or as a result of bottom-up self-organization. Importance-based feature extraction combines elements of both approaches. We propose that feature extraction be done by a bottom-up competitive interaction among feature detectors, but that it be guided by top-down information as to the importance of the features.

5.1 Top-Down Approaches

Some researchers have studied reinforcement learning strictly in terms of the top-down feedback, but with a set of simple, pre-computed feature detectors. As (Tesauro, 1989) notes, without including some simple, precoded features, the learning system's order of computation can be quite high, requiring a possibly intractable number of examples. Hence, the inputs to Tesauro's Neurogammon system included a few simple pre-computed features. Tesauro's more recent temporal-difference backgammon program (Tesauro, 1992) contains no pre-wired features, although he comments that adding pre-wired feature detectors to a similar system seems to promise significantly better results. Our hope is that techniques such as importance-based feature extraction may enable learning systems to achieve these benefits by learning good features on-line.

(Barto, Sutton and Anderson, 1983), (Sutton, 1984), (Anderson, 1987), (Robinson and Fallside, 1989) and others have studied temporal difference learning systems composed of a critic module and an action module; the critic module learns to predict the reinforcement feedback, and the action module learns to perform the task based on regular feedback from the critic module. Thus, the critic module provides internal feedback to the system to supplement the external reinforcement feedback, as does our monitor circuit. However, the monitor circuit feedback has to do with bottom-up information regarding the distribution of the effectors, rather than top-down information about the system's performance. We feel that the bottom-up information furnished by the monitor circuit is important for feature extraction in reinforcement learning. Barto, Sutton, and Anderson's pole-balancing system had the input categories built into the system; the system then learned a heuristic estimate of success for each category. Anderson's extension of this work, (Anderson, 1987), used back-propagation to learn the features, as did (Robinson and Fallside, 1989). We feel that top-down feedback is insufficient for adequate feature extraction in reinforcement learning, and that this is the reason why our system outperformed that of Robinson and Fallside.

5.2 Bottom-Up Approaches

In our system the monitor circuit provides a competitive interaction among detectors which is similar to both competitive learning, (Rumelhart and Zipser, 1986), and Kohonen's Self-Organizing Map (SOM), (Kohonen, 1982). Competitive learning tunes the the winning detector to be more responsive to the current pattern, while not tuning the losing detectors. Kohonen's algorithm is similar, but tunes those detectors within a gradually shrinking window about the winner. In the absence of external feedback, our monitor circuit acts to tune the closest detectors to respond to the current input vector, $\vec{x}$, while tuning other detectors away from $\vec{x}$. This is similar to competitive learning and the SOM, except that they do not tune "losing" detectors away from $\vec{x}$. But tuning the losing detectors away appears to encourage exploration in the feature space and prevents "lost" detectors in our system.

The main difference between importance-based feature extraction and other bottom-up approaches is that our system also tunes the effectors according to the top-down reinforcement feedback. This causes the effector which dominates the competition for a particular input pattern $\vec{x}$ to be the one whose actions have resulted in the best feedback for the state represented by $\vec{x}$. Therefore, each effector comes to recognize the features which indicate that its action will benefit the system in terms of its reinforcement feedback. In contrast, the other bottom-up approaches simply tune feature detectors toward the probability density function of the inputs.

This distinction reflects a difference in the types of problems that the systems are designed to solve. Most bottom-up learning systems do not consider top-down feedback because they are designed to model their input space. For example, Kohonen's system is meant to produce feature maps in which topologically close nodes are sensitive to similar input vectors. In contrast, importance-based feature extraction aims to guide the system's behavior according to the reinforcement feedback given by the environment.

5.3 Other Approaches

Another relevant study is Jürgen Schmidhuber's Neural Bucket Brigade (NBB) (Schmidhuber, 1989), which is a neural net version of John Holland's Bucket Brigade algorithm (Holland, 1986) with link weights updated according to the weights of their successors. The NBB employs winner-take-all subsets as its basic hidden units, in order to implement the bidding system by which the Bucket Brigade classifiers compete for the right to fire; the competition between nodes in a subset selects the node whose links will be updated according to the feedback from above. Unfortunately, the NBB's use of competitive learning has no mechanism like our monitor circuit to tune the "losing" detectors; as a result, detectors can become "lost" and never contribute to the system. Schmidhuber describes the action of his algorithm in terms of the shifting of "weight substance" from outgoing to incoming links of nodes; he contrasts this with normal competitive learning, which shifts weight substance from the weakly contributing incoming links to the strongly contributing incoming links to a node. In contrast to both approaches, importance-based feature extraction acts to shift weight substance to incoming links which activate an effector node in contexts where that effector's action is useful.

Holdaway's work (Holdaway, 1989) is also relevant to ours, because it combines competitive learning with top-down feedback; however, his system is designed for supervised learning. It uses a Kohonen-style self-organizing map as the input to a back-propagation credit assignment layer, but the feature extraction is done off-line before the rest of the training. Consequently, the top-down feedback cannot be used to guide feature extraction. In contrast, importance-based feature extraction exploits the top-down feedback in order to favor emerging features which give the system a performance advantage.

6 Future Work

We are working on extending our system to deal with learning problems having more complex features and more temporal complexity. First, our simple model has no hidden layer of feature detectors, so that each effector node functions as its own feature detector. Therefore, features learned by one effector cannot be used by any of the others. If the patterns detected by different effectors have similarities, they might better be described in terms of more primitive features, some of which may be shared by several effectors. We are addressing these issues by adding a hidden layer of feature detectors which then feeds into the effector layer. The additional layer of nodes may also allow for more powerful feature extraction. Second, the effectors in our simple model do not have local receptive fields; internal feedback may tune all effectors away, even if they are already tuned to very different areas of the input space. To reduce this sort of cross-talk, we are using feature detectors with Gaussian response functions in the hidden layer.

Much of the work currently being done in reinforcement learning involves some form of temporal-difference learning; temporal-difference learning takes the difference between temporally-successive predictions of an outcome as the error signal for learning, instead of comparing the predicted outcome with the actual outcome (Sutton, 1988). In our simulations the temporal credit-assignment problem has not been as important as other issues involving representation and feature detection. Presumably this is because board games such as tic-tac-toe and go-moku require much the same strategy in the end-game as they do in the beginning, unlike games such as chess. But tasks with greater temporal complexity will require an additional temporal credit-assignment mechanism to determine the goodness of moves in-between the feedback provided at terminal moves.

We have defined importance-based feature extraction in terms of the values for particular combinations of system state and action: "Q-values" in the reinforcement learning literature (see Watkins, 1989). Therefore, our current work includes a temporal-difference learning component based on one-step, non-discounted Q-learning, and relates the importance of features to their Q-values. Watkins defined Q-learning in terms of a state representation equivalent to a look-up table, with an entry for each possible state. We hope to show that importance-based feature extraction can be used on-line to construct more efficient state representations for Q-learning, with little sacrifice in performance.

7 Conclusion

We feel that constructing good input representations is one of the important problems in neural network learning. A common approach is to use top-down feedback with gradient-descent to produce feature detectors in a hidden layer of nodes. This strategy works poorly for reinforcement learning problems because of their sparse, delayed feedback signals. Our work suggests that combining top-down feedback with the bottom-up influence of a monitor circuit may be a more effective technique for feature extraction in reinforcement learning.

The effect is to guide feature extraction by the *importance* of features, rather than by their frequency. The simplicity of our model shows that such mechanisms need not be complex in order to be effective. We conclude that importance-based feature extraction is a promising basis for further studies in reinforcement learning.

8 Acknowledgements

We would like to thank Rich Sutton for constructive criticism, encouragement, and pointers to relevant literature.

A Analysis of the Update Algorithm

By recursively applying the update rule we can write the weight $w_{i,j}$ as the sum of its updates. Let $FB_j(t)$ denote the sum of external and internal feedback seen by effector j at time t; then

$$FB_j(t) = \begin{cases} r(t) + m(t) & \text{if a terminal move occurs at } t \text{ and } j \text{ is the chosen effector} \\ m(t) & \text{otherwise} \end{cases}$$

Rewriting the update rule in terms of FB_j we have

$$w_{i,j}(t) = \begin{cases} w_{i,j}(t-1) + \alpha x_i(t)(FB_j(t) - w_{i,j}(t-1)) & \text{if } FB_j(t) \neq 0 \\ w_{i,j}(t-1) & \text{otherwise} \end{cases}$$

If $FB_j(t) \neq 0$,

$$\begin{aligned} w_{i,j}(t) &= (1 - \alpha x_i(t))w_{i,j}(t-1) + \alpha x_i(t)FB_j(t) \\ &= (1 - \alpha x_i(t))\{(1 - \alpha x_i(t'))w_{i,j}(t'-1) + \alpha x_i(t')FB_j(t')\} + \alpha x_i(t)FB_j(t), \end{aligned}$$

where t' is the time of the most recent update before time t for which $FB_j(t') \neq 0$.

Since $w_{i,j}$ is not updated when $FB_j(t) = 0$, let us label the sequence of nonzero $FB_j(t)$ as $\{FB_j^1, FB_j^2, \ldots, FB_j^n\}$. For $FB_j^k = FB_j(\tau)$, let x_i^k denote $x_i(\tau)$ and $w_{i,j}^k$ denote $w_{i,j}(\tau)$. Let $w_{i,j}^0$ denote $w_{i,j}(0)$.

Then we can write

$$\begin{aligned} w_{i,j}^n &= (1 - \alpha x_i^n)w_{i,j}^{n-1} + \alpha x_i^n FB_j^n \\ &= (1 - \alpha x_i^n)\{(1 - \alpha x_i^{n-1})w_{i,j}^{n-2} + \alpha x_i^{n-1} FB_j^{n-1}\} + \alpha x_i^n FB_j^n \\ &\vdots \\ &= w_{i,j}^0 \prod_{r=1}^{n}(1 - \alpha x_i^r) + \sum_{k=1}^{n-1} \alpha x_i^k FB_j^k \prod_{r=k+1}^{n}(1 - \alpha x_i^r) + \alpha x_i^n FB_j^n \end{aligned}$$

Essentially this means that $\mathbf{w}_j$ is a time-averaged trace of terms $\vec{x}(t)FB_j(t)$, where $\mathbf{w}_j^k = (w_{1j}^k, w_{2j}^k, \ldots, w_{mj}^k)$ is the vector of weights from receptor nodes into the effector node j, and $\vec{x}^k = (x_1^k, x_2^k, \ldots, x_m^k)$is the vector of receptor values.

The presence of x_i^r in the terms $(1 - \alpha x_i^r)$ is unimportant, since it tends to get averaged out when a succession of such terms are multiplied. We verified this by running a simulation with the modified update rule

$$w_{i,j}(t) = (1 - \alpha) w_{i,j}(t-1) + \alpha x_i(t) FB_j(t)$$

Without changing any other aspects of the system, this resulted in quicker convergence, although the system was more sensitive to initial conditions, and some trials did not converge.