Fuzzy $Q$-Learning with an Adaptive Representation

A. Waldock, IEEE Student Member and B. Carse, IEEE Member

Abstract—Reinforcement Learning (RL) is learning how to map states to actions so as to maximise a numeric reward signal. Fuzzy $Q$-Learning (FQL) extends the RL technique $Q$-Learning to large or continuous problems and has been applied to a wide range of applications from data mining to robot control. Typically, FQL uses a uniform or pre-defined internal representation provided by the human designer. A uniform representation usually provides poor generalisation for control applications, and a pre-defined representation requires the designer to have an in-depth knowledge of the desired control policy. In this paper, the approach taken is to reduce the reliance on a human designer by adapting the internal representation, to improve the generalisation over the control policy, during the learning process. A Hierarchical Fuzzy Rule Based System (HFRBS) is used to improve the generalisation of the control policy through iterative refinement of an initial coarse representation on a classical RL problem called the Mountain Car problem. The process of adapting the representation is shown to significantly reduce the time taken to learn a suitable control policy.

I. BACKGROUND

In reinforcement learning problems[1], a teacher provides feedback in the form of only a reinforcement signal that indicates the desirability of a particular environmental situation. Reinforcement can be provided after every action or, for more challenging problems, at the end of a sequence of actions. The key difference is that the teacher indicates what should be achieved not how it should be achieved. Specification of the reinforcement signal is typically easier than generating training examples but learning can be extremely time-consuming especially in problems with a large or continuous state space.

For the majority of control applications, the sensor measurements and actuator demands are likely to be continuous-valued. Hence, RL algorithms applied to this domain must be able to learn a control policy that maps continuous-valued sensor measurements to continuous-valued actuator demands to achieve the desired task. Usually the only way to learn anything in such a domain is to generalise from previously seen examples to ones that have not yet been encountered. This ability to generalise is essential because most sensor measurements will never have been encountered before and are unlikely to ever be encountered exactly again.

Currently, the majority of RL algorithms for large or continuous problems (CMAC, Fuzzy $Q$-Learning, Neural Networks etc.) assume a generalisation over the state space, which is based on a uniform or pre-defined internal representation as provided by a human designer. When using a uniform representation for large or continuous problems, such as the control applications discussed in this paper, the state space quickly becomes unmanageable and the resolution of the representation is typically selected using trial-and-error. The designer could provide a non-uniform representation to improve the generalisation but this would require an in-depth knowledge of the mapping between sensor measurements and actuator demands. The approach taken in this paper is to reduce the reliance on a human designer by adapting the representation during learning to find an effective generalisation.

This paper begins with a brief review of approaches used to adapt the internal representation to improve the generalisation for RL algorithms in continuous domains. The next section introduces a method called Fuzzy $Q$-Learning, which extends $Q$-Learning to cope with continuous control problems. Hierarchical Fuzzy Rule Based Systems (HFRBSs) are then introduced as a method of adapting the internal representation with the experimental results detailed in the following section. The final section presents a conclusion and possible avenues of future work.

II. RELATED WORK

This section presents a brief review of the related work on adapting the internal representation of existing reinforcement learning algorithms. Often in robotics, the continuous control problem is simplified into a discrete one by dividing the continuous space into a finite set of bins or boxes. A number of algorithms [2], [3] have been proposed to adapt a discrete representation to improve the generalisation but these fundamentally result in a discrete partitioning of the state space, which is undesirable for a control application where smooth output responses are required.

Another popular approach is the Cerebellar Model Articulation Controller (CMAC) [4] that features spatially localised updates and in essence is a compromise between a look-up table and a weight-based approximator. CMAC has been combined with $Q$-Learning to provide a means of coping with a continuous state space[5]. Sherstov and Stone [6] explored how adjusting the number of tiles in a tiling during the learning process affected the performance of the learner. Their results “indicate that a broad generalisation is helpful at early stages of learning but detrimental in the final count”. The conclusions drawn by this work indicate that adapting the representation during the learning process could lead to improvements in performance.

Reinforcement learning algorithms, such as $Q$-Learning, have also been combined with fuzzy control [7], [8] to cope with continuous-valued sensors and actuators. Glorioso[8] provides an approach to extend a fuzzy rule based system to learn in reinforcement learning problems. In recent work,
Deng [9] and ER [10] have demonstrated how an adaptive representation can be constructed by spawning fuzzy rules when the accuracy is deemed insufficient. Another proposed approach, which uses fuzzy logic, employs a top-down approach where an effective representation is found using a Hierarchical Fuzzy Rule Based System (HFRBS)[11]. A HFRBS performs iterative refinements on inaccurate fuzzy rules in order to improve the accuracy of the approximated function. The work outlined in this paper builds on those initial experimental results.

III. FUZZY Q-LEARNING

Reinforcement learning is concerned with learning a policy π that maps states S to actions A so as to maximize a numerical reward signal, r. Q-Learning [12] is a popular reinforcement learning technique where the learner incrementally builds a Q-function that attempts to estimate the discounted future reward for taking actions from given states (a state-action pair). On every time step t, Q(s_t, a_t) of the state-action pair s_t and a_t is updated using the reward signal r_{t+1} received and the value V(s_{t+1}) of the next state s_{t+1} (Equation 1).

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - Q(s_t, a_t)) \]  

(1)

where \( V(s_{t+1}) = \max_{a' \in A} Q(s_{t+1}, a') \), \( \alpha \) is the learning rate and \( \gamma \) is the discount rate. On each update, the current estimate of \( Q(s_t, a_t) \) is shifted towards the observed value \( r_{t+1} + \gamma V(s_{t+1}) \) by the learning rate \( \alpha \). The control policy is determined by selecting the action with the highest Q-Value.

Fuzzy Q-Learning combines Q-Learning and a Fuzzy Rule Based Systems (FRBSs) to provide a means of coping with continuous-valued sensor measurements and actuator demands to learn the most appropriate consequent of a fuzzy rule from a reinforcement signal. As with a traditional FRBS the continuous input space is partitioned into a series of IF...THEN rules. For Fuzzy Q-Learning, each rule \( R_j \) has an associated Q-value \( q_{j,a} \) for each output linguistic symbol \( B_{j,a} \) (Equation 2).

\[ R_{j,a}: \quad IF \quad x_1 IS A_{1j} AND \ldots x_n IS A_{nj} \quad THEN \quad y = B_{j} \quad WITH \quad q_j \]  

(2)

where \( j \) is a vector \( j_1, j_2, \ldots j_n \) that represents the fuzzy subspace for this fuzzy rule where \( A_{ij} \) is the linguistic symbol on the dimension \( i \), \( B_j \) is the consequent linguistic symbol selected for rule \( j \) and \( n \) is the number of input dimensions. The Q-value, for each fuzzy rule, can be represented in a tabular form \( q_{j,a} \) where \( j \) is the rule index and \( a \) is the linguistic symbol chosen from a set of \( M \) possible linguistic symbols in \( B \). Using the degree of activation as a weighting factor, the Q-value for an inferred action for input vector \( x \) is:

\[ Q(x, a) = \frac{\sum_{j=0}^{N} \mu_j(x) \times q_{j,a}}{\sum_{j=0}^{N} \mu_j(x)} \]  

(3)

where \( N \) is the number of fuzzy rules and the degree of activation in this paper is calculated using the product of the degree of membership for all fuzzy sets \( \prod_{i=0}^{n} \mu_i(x_i) \).

The value can be calculated in the same manner:

\[ V(x) = \frac{\sum_{j=0}^{N} \mu_j(x) \times \max_{k \geq M} q_{j,k}}{\sum_{j=0}^{N} \mu_j(x)} \]  

(4)

When the inferred action \( a \) is applied to the state \( x_t \), the environment transitions to state \( x_{t+1} \) with reward \( r_{t+1} \). The Q-value is updated using equation 1 but the learning rate \( \alpha \) is combined with the rule’s degree of activation:

\[ q_{j,a} = q_{j,a} + \alpha \frac{\mu_j(x_t)}{\sum_{j=0}^{N} \mu_j(x_t)} \Delta Q \]  

(5)

where \( \Delta Q \) is \( (r_{t+1} + \gamma V(x_{t+1}) - Q(x_t, a)) \). The action to perform is selected using an appropriate exploration/exploitation policy (EEP) where \( a^* \) represents the greedy action and is given by \( \arg\max_{a} q_{j,a} \). The EEP used in the experiments in this paper is an \( \epsilon \)-greedy policy, where most of the time the action that has the maximal estimated state-action value (\( a^* \)) is selected but with probability \( \epsilon \) a random action from \( B \) is selected. The Fuzzy Q-Learning algorithm is outlined in Algorithm 1.  

**Algorithm 1 Fuzzy Q-Learning algorithm**

1: repeat
2: \quad Observe the input vector \( x_t \)
3: \quad Select actions using an Exploration/Exploitation Policy (EEP)
4: \quad Compute the global consequence \( y(x_t) \) and the Q-value \( Q(x_t, a) \)
5: \quad Apply the action \( y(x_t) \)
6: \quad Receive the reinforcement \( r_{t+1} \)
7: \quad Observe the input vector \( x_{t+1} \)
8: \quad Update the Q-Value using (5)
9: until End

Typically, an evenly distributed set of membership functions is defined over the entire universe of discourse. The next section details an approach to adapt the representation while maintaining a degree of human interpretability using a Hierarchical Fuzzy Rule Based System (HFRBS).

IV. HIERARCHICAL FUZZY Q-LEARNING

An approach pioneered by Cordón, Herrera and Zwick [13] termed a Hierarchical Fuzzy Rule Based System (HFRBS) is seen as a means of balancing the trade-off between accuracy and interpretability by incrementally refining linguistic symbols instead of altering the fuzzy sets directly.

A HFRBS initially divides the input space into a fixed number of linguistic symbols each corresponding to a natural...
language meaning e.g. very small, small, large etc. The set of initial linguistic symbols for each input dimension could be uniformly defined over the universe of discourse or a pre-defined set if the designer has prior knowledge of the desired control policy. Training data is then used to automatically learn the consequent of the fuzzy rules as in a standard FRBS. The HFRBS then employs an expansion policy to determine inaccurate areas of the input space and the corresponding rules. When an inaccurate area is identified, the rule representing that portion of the input space is specialised into a set of smaller specific fuzzy rules. Specialisation continues until a desired level of accuracy is satisfied or the limit of fuzzy rules has been reached.

Figure 1 shows an example of a partitioned decision space 1(a) and the corresponding hierarchical representation 1(b). The proposed HFRBS algorithm is presented in Algorithm 2.

Algorithm 2 HFRBS Iterative Refinement Process

1: Initialise the fuzzy rules
2: repeat
3: repeat
4: Fuzzification, Learning, Inference and Defuzzification (Algorithm 1)
5: Evaluating the accuracy of a fuzzy rule
6: until End of Trial
7: Extract the leaf fuzzy rules
8: Sort using some weighted splitting criterion
9: if sufficiently evaluated then
10: Select the top scoring fuzzy rule for refinement
11: end if
12: until End

The fuzzification, learning, inference and defuzzification is performed in the same manner as a traditional FRBS except only on the leaf fuzzy rules in the hierarchy. In the experiments presented, the membership functions are assumed to be triangular with the fuzzification performed using a logical product, the defuzzification is performed using the centre of sums and learning is performed using Fuzzy Q-Learning as described in the previous section. The following two sections cover the evaluation and refinement of fuzzy rules to improve the initial coarse representation and hence improve the generalisation.

A. Evaluating the Accuracy of a Fuzzy Rule

The power of a HFRBS is in the ability to identify and refine inaccurate fuzzy rules. As the HFRBS is trained, the accuracy of the fuzzy rules is evaluated or scored using a splitting criteria. Before the splitting criteria can be evaluated, the state-action value must have been sufficiently updated to estimated the true value.

Recall, that the value of a state-action pair \( q[j, a] \) for each linguistic symbol of a fuzzy rule is defined as the immediate reward plus the expected value when the optimal control policy is followed. The state-action value \( q[j, a] \) for each fuzzy rule \( j \) may not have been updated sufficiently to approximate the true state-action value because of the dynamic nature of the Bellman equation. A typical approach, used in Value Iteration, is to assume that the state-action value function has converged (i.e. accurately approximated) when the error between the current state-action value and the expected state-action value is small (\( Q(s, a) - (r(s) + \gamma v(s')) \)). Unfortunately, this is only possible when the continuous state space is sufficiently finely partitioned otherwise the error may never converge to a small value. Accurately estimating the convergence of the value function is still an open research question when combined with function approximation for a continuous state space. In this paper, a heuristic is used that assumes that the value for a fuzzy rule has converged when a pre-defined number of \( Q \)-Learning updates has been performed.

Given that the state-action value has been sufficiently updated, the proposed splitting criterion can now be evaluated. The splitting criteria is based on estimating the variance of the value of each fuzzy rule. After an initial period to converge (as discussed in the previous paragraph), the variance in the value of each fuzzy rule is used to indicate the accuracy of the fuzzy rule in approximating this region of the value function. This splitting criterion was inspired by the research conducted by Munos[14], which indicates that when the variance is high the resolution is too coarse to accurately approximate the value function. Munos shows that as the resolution of the partitioning is increased, the variance and approximation error is reduced. In this paper, the variance in the value of a fuzzy rule is estimated and the resolution of the representation improved only in those regions with a high variance.

The raw score method is used to estimate the variance in the value function approximation due to this fuzzy rule. The raw score method is a convenient computation alternative for calculating the standard deviation from a set of observations. The method only requires three variables \( (\sum_{t=0}^{n} x_t^2, \sum_{t=0}^{n} x_t, n) \) where \( x_t \) is the observation at time \( t \) and \( n \) is the number of observations. The variance of fuzzy rule \( j \) is calculated using \( v_{j,t} \) at each time step \( t \):

\[
Variance_j = \sqrt{\frac{\sum_{t=0}^{n} (v_{j,t}^2) - (\sum_{t=0}^{n} v_{j,t})^2}{n}}
\]

where \( n \) is the number of updates performed and \( v_{j,t} \) is the value of fuzzy rule \( j \) at time \( t \) (i.e. \( max_k(q[j, k]) \)).

The splitting criteria is assumed to be zero until a sufficient number of observations have been received to make a reliable estimate. To prevent the refinement process concentrating on a single discontinuity while other coarse regions of the representation remain unrefined, the splitting criteria is weighted by the volume of the input space covered by the fuzzy rule.

This proposed approach balances the trade-off between accurately modelling discontinuities and generating a smooth and stable approximation of the value function. By weighting the splitting criteria by the volume of the fuzzy rule in the
input space, the refinement process will evenly refine all the discontinuities and not focus on a single region. The volume of the input space occupied by a fuzzy rule is calculated using:

$$\text{volume}_j = \prod_{i} \text{width}_{A_{ij}}$$  \hspace{1cm} (7)

In equation 7, $A_{ij}$ is the fuzzy set for input $i$ for fuzzy rule $j$ and $n$ is the number of input dimensions. The next section outlines how fuzzy rules are refined given a score by the proposed splitting criteria defined in this section.

B. Refining Inaccurate Fuzzy Rules

For a reinforcement learning problem, refinement may not be possible at the end of each trial because a trial contains an element of exploration. Exploration of the state space is necessary in a reinforcement learning problem to accurately approximate the true state-action value function, but it may prevent the splitting criteria from gaining sufficient experience of the relevant regions of the control policy on a single trial so as to accurately score the fuzzy rules used to approximate it. For example, the approximated value for a fuzzy rule may not have been updated sufficiently for the variance to be evaluated. Hence, a percentage of the fuzzy rulebase is required to be sufficiently evaluated before refinement can be performed. Once the fuzzy rules have been sufficiently evaluated by the splitting criteria, the refinement of inaccurate fuzzy rules can commence.

Once identified for refinement, a fuzzy rule is decomposed into a set of smaller and more specific fuzzy rules. The refinement of an inaccurate fuzzy rule is achieved by first refining each of the linguistic symbols in the antecedents of the fuzzy rule. A linguistic symbol is refined by increasing the resolution of the corresponding fuzzy set and then selecting the new linguistic label for the fuzzy set. Firstly, the fuzzy set associated with the linguistic symbol is equally divided by selecting the new linguistic label for the fuzzy set from the next layer in the hierarchy. It is worth noting that the hierarchy of fuzzy sets is defined without any prior knowledge of the control problem (i.e. fuzzy sets are typically defined using an odd number of fuzzy sets for control problems). With the first layer containing a single fuzzy set over the entire discourse, the second containing two fuzzy sets that evenly divide the discourse and so on. After the fuzzy set has been refined, the linguistic label associated with the corresponding fuzzy set is then selected from a pre-defined hierarchy of labels depending on the sensor type this input variable represents.

Once all the linguistic symbols have been divided, the specific fuzzy rules are created by combining all the combinations of the linguistic symbols for all the input dimensions. The current estimate of the state-action values is also propagated from the general fuzzy rule to each of the specific fuzzy rules as an initial approximation of the value function in this region of the state space.

V. EXPERIMENTAL RESULTS

A. Problem Definition

The example problem used to demonstrate this approach is the Mountain Car problem as defined by Munos[15]. The car is initially parked in a valley between two hills. The aim is to reach the goal at the top of the right hill without hitting the wall at the top of the left hill. Unfortunately, the car is underpowered and must, in some instances, gain sufficient inertia from the left slope to reach the goal. The reinforcement signal is defined as follows:

$$R_t = \begin{cases} 
-1 & \text{if the car hits the wall} \\
1 & \text{if the car reaches the goal} \\
0 & \text{otherwise}
\end{cases}$$

In all the experiments, the fuzzy rules are initialised with a $Q$-value of zero. In order to produce a fair comparison between the algorithms, they were all run with the discrete actions $(-1, +1)$ and the performance evaluated using 100 test points that are evenly distributed over the state space. For a test point, the mountain car was initialised with the corresponding parameters (position and velocity) and run until the car reached the goal or at the time limit of 200 time steps.

The experiment is conducted 50 times with the final performance presented as the mean, lower and upper confidence
intervals (95%). The start position is randomly selected for each of the trials and the sequence is different for the 50 runs. The sequence of the randomly generated start positions was the same for all the algorithms tested. The confidence intervals are presented (as dashed lines on the plots) to indicate the reliability of the learning algorithms as they have some stochastic elements. The closer the confidence intervals to the mean the more reliable the algorithms operation.

B. Baseline and Benchmark

This section provides the baseline and benchmark results for the mountain car problem. As a baseline, a fixed control policy that performs a constant action throughout is used. For a benchmark or “optimal” result, a finely discretized tabular Q-Learning algorithm is used which is extensively trained for 100 million trials. Each of the continuous inputs is divided into 20 discrete bins while the continuous output is divided into only 2 actions: -1 and +1. These baseline and benchmark are presented because they are used to explain the results achieved with an adaptive representation later in this paper. Table I shows that the control problem can be partially solved using a controller with a fixed action. Applying a constant left force (-1) results in 12% of the test points reaching the goal and surprising applying no force at all has the effect of achieving the goal state for 25% of the test points. The highest percentage completed is when a constant right force (+1) is applied with 80% completed. The Tabular Q-Learning, after extensive training, completes all the test runs and achieves the maximum mean reward received (only 0.80 is achievable because from some starting positions the car will always hit the left wall).

C. Results: Uniform Representation

This section presents the experimental results of using a FQL with a uniform representation as the resolution is increased. The results are graphically illustrated in Figure 2 where the mean reward received is shown during the training process for FQL with 49, 225 and 400 fuzzy rules.

Figure 2 shows that as the resolution is increased the task accuracy is slower to improve. The increase in the number of time steps stems from the time required for the value function to be correctly approximated i.e. the value to propagated back across the state space. As the resolution of the uniform generalisation is increased, a larger number of trials are required for the value function to be approximated accurately, which therefore effects the time taken to learn the control policy. These results demonstrate that as the density of the uniform representation is increased, the time taken to learn the control policy is increased.

D. Results: Adaptive Representation

The HFQL is initialised with a single fuzzy rule covering the entire state space. At the end of every trial, inaccurate fuzzy rules are identified and refined. A single fuzzy rule is selected for refinement but only once enough fuzzy rules have been evaluated. In these experiments, a fuzzy rule is assumed to have converged after 50 updates and been evaluated sufficiently after another 50 updates. The splitting criteria is assumed to have a sufficient estimate once 10% of the fuzzy rules have been evaluated.

Figure 3 shows the performance for every 10 trials during the learning process of 500 trials and reveals how this difference between a FQL and a HFQL approach arises. The results show no significant improvement in the reward received at the end of all the trials for both FQL and HFQL. Although, HFQL achieves a higher level of performance in
far fewer trials. Using a coarse generalisation in the initial stages of the learning process allows the controller to quickly compute an approximation of the value function and hence, a coarse control policy to follow. In fact, HFQL jumps to 80% completed after only 10 trials because the initial coarse representation discovers an effective coarse control policy i.e. using the fixed action +1 for the entire state space. As the controller experiences the state space, inaccurate areas of the state space are identified and the fuzzy rules refined. Therefore, the control policy is only improved in those areas of the state space that require an improvement to the existing control policy with the remainder using the default coarse control policy +1. Where a FQL approach must learn the best action through approximation of the value function for all the fuzzy rules independently.

The improvement in the generalisation is illustrated in Figure 4 by observing the approximated value functions for FQL and HFQL with 225 fuzzy rules. The FQL approximation in (a) has not experienced enough trials to optimise actions for the fuzzy rules at the top the state space where the HFQL approximation in (b) defaults to the value used in the coarse value function and was generalised without requiring this region to be experienced individually. By using the coarse value function as a default value, the performance is significantly reduced as shown in the previous comparison between the approaches.

VI. CONCLUSION

In conclusion, the performance of HFQL at the end of 500 trials does not show a significant difference from FQL, but HFQL does achieve a higher performance in far fewer trials. The initial coarse representation allows a coarse approximation of the value function and hence, an initial control policy to be computed quickly. The coarse approximation of the value function is then refined to improve only those areas that are required to accurately model the desired control policy. Building an adaptive representation is shown to circumvent the relationship between increasing the resolution of a uniform representation and the corresponding increase in the number of trials required. These initial results have demonstrated that adapting the internal representation has not only the promise of improving performance, but has the benefit of significantly speeding up the learning time for reinforcement learning algorithms.

REFERENCES


Fig. 4. Comparison of the FQL and HFQL Value Function