Learning Hierarchical Fuzzy Rule-based Systems for a Mobile Robot Controller

Antony Waldock
Advanced Technology Centre
BAE SYSTEMS
Filton, Bristol, England
Email: antony.waldock@baesystems.com

Brian Carse and Chris Melhuish
Intelligent Autonomous System Lab
University of the West of England
Bristol, England
Email: brian.carse,chris.melhuish@uwe.ac.uk

***Abstract*— A reinforcement learning technique for mobile robot control must be capable of coping with a high dimensional continuous state space. Fuzzy Q-Learning provides a means of coping with a continuous state space but suffers from problems of scalability. This paper proposes a new Hierarchical Fuzzy Q-Learning (HFQL) algorithm that combines a Hierarchical Fuzzy Rule Based System (HFRBS) and Fuzzy Q-Learning (FQL). The algorithm uses the variance in the approximated value function to determine the inaccurate rules to specialise. This initial work uses the mountain car problem to compare the performance of Tabular Q-Learning with Fuzzy Q-Learning for both uniform and variable state space representations.**

I. Introduction

Development of an autonomous mobile robot for a real-world environment is extremely challenging. The construction of a suitable control policy is a complex and time-consuming process. Classical control theory [1] provides techniques for generating a control policy from mathematical models of the environment, platform and control laws. Unfortunately constructing models that accurately represent the dynamics of a complex environment (both platform and external environment) can be a long and difficult process. Reinforcement learning [2] is seen as a solution to the reliance on building accurate models of the platform and environment. Learning a control policy for a mobile robot, while on the platform in the environment, bypasses the need to mathematically specify models. Fuzzy Q-Learning provides a reinforcement learning technique to construct a Fuzzy Rule Based System (FRBS). FRBSs have been demonstrated to cope well with uncertain and imprecise environments when used for mobile robot control [3]. In recent years, Hierarchical Fuzzy Rule Based Systems (HFRBSs)[4] have been demonstrated to improve scalability using variable resolution discretization.

This contribution outlines initial investigations into the construction of a HFRBS for mobile robot control using Fuzzy Q-Learning. The paper gives a brief overview of Fuzzy Q-Learning and variable resolution discretization techniques in sections II and III followed by the algorithm details in section IV. The mountain car problem is outlined in section V with the results and discussion presented in section VI.

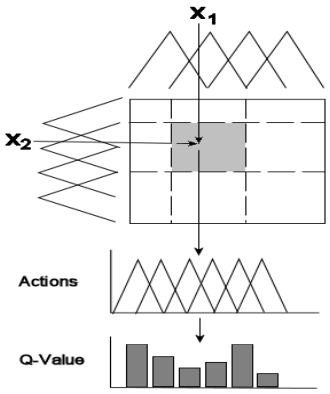
II. Fuzzy Q-Learning

Reinforcement learning [5] is concerned with learning a policy π that maps states S to actions A so as to maximize a numerical reward signal, r. Much work has been conducted on applying reinforcement learning techniques to learning a control policy for mobile robot control [6][7]. Q-Learning [8] 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, the $Q(s_t, a_t)$ of a state s_t and a_t is updated using the reward signal r_{t+1} received and the discounted future $V(s_{t+1})$ of the next state s_{t+1} (Equation 1).

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

where $V(s_{t+1}) = max_{a' \subset A} Q(s_{t+1}, a')$, α is the learning rate and γ 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 α. The best policy can be determined by selecting the action with the highest Q-Value. Fuzzy Q-Learning [9] combines Q-Learning and Fuzzy Rule Based Systems (FRBSs) to provide a means of coping with continuous inputs and outputs. FRBSs are popular for mobile robot control due to their ability to cope with uncertain environments while retaining a degree of human interpretability [10][11].

A FRBS partitions the continuous state space into a series of IF..THEN rules. Each rule consists of a set of input linguistic symbols S_i and an output linguistic symbol A_i. The layout of a FRBS with two inputs is illustrated in Figure 1.



1: Fuzzy Q-Learning

A linguistic symbol associates a fuzzy set with a natural language meaning e.g. small, medium, very large etc. A fuzzy set links an input variable x_j with a membership function, to represent its applicability with regard to the current input vector $x = (x_1, .., x_n)$ and hence, determine the rules overall influence in the decision process. For Fuzzy Q-Learning, each rule R_i has an associated Q-value q_i for each action (Equation 2).

$$R_i : \quad if \quad x_1 \, is \, S_{i,1} \, and \, ...x_n \, is \, S_{i,n} \quad (2)$$
$$then \quad y = A_i \, with \, q_i$$

where $S_{i,j}$ is a linguistic symbol on the input variable x_j and n is the number of input dimensions. The degree of activation or 'truth value' of a rule is defined as $\mu_i(x) = \prod_{j=0}^{n} S_{j,i}(x_j)$. The Match Set is defined as the set of active rules ($\mu_i(x) > 0$) for a given input vector. Centre of sums is used to infer a crisp action $a(x)$ given a set of N rules using:

$$a(x) = \frac{\sum_{i=0}^{N} \mu_i(x) \times centre(a_i)}{\sum_{i=0}^{y} \mu_i(x)} \quad (3)$$

The Q-value can be represented in a tabular form $q[i, j]$ where i is the rule index and j is the action chosen from a set of J actions. Let i' represent the selected action for rule i. The Q-value for an inferred action for input vector x is:

$$Q(x, a) = \frac{\sum_{i=0}^{N} \mu_i(x) \times q[i, i']}{\sum_{i=0}^{N} \mu_i(x)} \quad (4)$$

and the value is:

$$V(x) = \frac{\sum_{i=0}^{N} \mu_i(x) \times max_{j \geq J}(q[i, j])}{\sum_{i=0}^{N} \mu_i(x)} \quad (5)$$

When the inferred action a is applied to the state x, the environment transitions to state y with reward r. The Q-value is updated using equation 1 but the learning rate α is combined with the rule's degree of activation:

$$q[i, i'] = q[i, i'] + \alpha \frac{\mu_i(x)}{\sum_{i=0}^{N} \mu_i(x)} \Delta Q \quad (6)$$

where ΔQ is $(r_{t+1} + \gamma V(y) - Q(x, a))$. The Fuzzy Q-Learning algorithm is outlined in Algorithm 1.

Algorithm 1 Fuzzy Q-Learning algorithm

1: **repeat**
2: Observe the input vector x
3: Select actions using an Exploration/Exploitation Policy (EEP)
4: Compute the global consequence $a(x)$ and the Q-value $Q(x, a)$
5: Apply the action $a(x)$
6: Receive the reinforcement r
7: Observe the input vector y
8: Update the Q-Value using (6)
9: **until** End

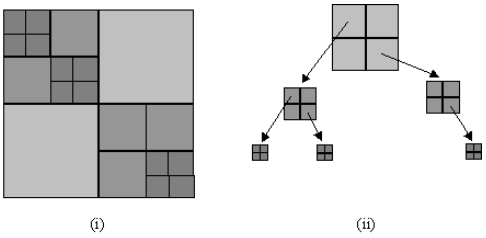
The representation selected for the linguistic symbols directly affects the accuracy of the approximated function [12]. If a greater degree of accuracy is demanded then the structure (i.e. size, shape and position) of the membership functions, which represent the linguistic symbols, must be altered. Increasing the granularity of the linguistic symbols can facilitate improvements in accuracy but interpretability is reduced. For mobile robot control, the level of granularity is also determined by the resources available (i.e. physical memory size). Hence, a trade-off exists between accuracy and resources/interpretability. A method of balancing this trade-off is termed variable resolution discretization.

III. Variable Resolution Discretization

The majority of function approximation techniques uniformly partition the state space. Variable resolution discretization varies the size of the partitions in an attempt to minimise the approximation error. Techniques move from a 'general to specific' representation by successively refining areas of the state space using a splitting-criterion or expansion policy. A large number of variable resolution discretization techniques exist and have been applied to a multitude of research areas, such as classifer systems [13], discrete reinforcement learning [14][15] and Decision Trees in the form of ID3 and C4.5 [16][17]. In this contribution, variable resolution discretization is used to approximate the value function and hence learn the control policy.

A Hierarchical Fuzzy Rule Based System (HFRBS) is used to perform variable resolution discretization. A HFRBS initially divides the state space into a fixed number of linguistic symbols. Fuzzy Q-Learning is used to learn the value function as in a standard FRBS. The HFRBS then employs an expansion policy to determine inaccurate areas of the state space and the corresponding rules. When an inaccurate area is identified, the rule representing that portion of the state space is specialised into a set of more specific rules. This process of specialisation continues until a desired level of accuracy is achieved. Figure 2 shows an example of a partitioned decision space (i) and its corresponding hierarchical representation (ii).

Current expansion policies require either access



2: Hierarchical Fuzzy Rule Base

to the entire training set [18], [19] which is not possible for a mobile robot controller or are only applicable to supervised learning [20].

IV. HFRBS for Mobile Robot Control

To perform variable resolution discretization using a HFRBS, an expansion policy to determine inaccurate areas of the state space must be defined. As a HFRBS can be viewed as a FRBS (see Figure 2), Fuzzy Q-Learning can be used to learn the optimal policy for a given task by approximating the value function. Reducing the approximation error of the value function can therefore improve the control policy. Within discrete reinforcement learning, a variety of expansion policies have been explored including using the variance in the value function, policy disagreement and a state's influence in the overall approximation. Munos and Moore [14] demonstrate that the variance of the value function can be used as an expansion policy within discrete reinforcement learning.

Within this contribution, a new expansion policy is introduced that uses the Raw Score Method to calculate the variance and hence estimate the approximation error. The raw score method is a convenient computation alternative for calculating the standard deviation from a set of observations. The raw score method only requires three variables ($\sum_{t=0}^{n} x_t^2$, $\sum_{t=0}^{n} x_t$ and n) where x_t is the observation at time t and n is the number of observations. The variance of an action j on rule i can be calculated using $q_t[i,j]$ at each time step t as an observation:

$$SD_{i,j} = \sqrt{\frac{\sum_{t=0}^{n} q_t[i,j]^2 - \frac{(\sum_{t=0}^{n} q_t[i,j])^2}{n}}{n}} \quad (7)$$

Equation 7 calculates the standard deviation of the Q-Value for each rule's action. The variance of

a rule is determined, in the same way as the value, by using the variance of the currently selected action i'. Due to the dynamic programming nature of Q-Learning, $q[i,j]$ requires a period of time, p, to converge. If the raw score method was calculated from all observations of $q[i,j]$ the variance would be artificially high. An *estimate* of the time to converge can be derived from the learning rate and discount rate. The estimated convergence time can be calculated by solving the geometric series:

$$\frac{1}{1-\gamma} - \epsilon \geq \sum_{t=0}^{p} (\alpha - t\alpha^t) \quad (8)$$

where ϵ determines the accuracy required. For Fuzzy Q-Learning, the learning rate α varies depending on the rule activation. The convergence $c[i,j]$ of an action j on rule i can be estimated using a dynamic programming approach [15].

$$c[i,j] \leftarrow c[i,j] + \alpha\mu(x)(Q_{max} - c[i,j]) \quad (9)$$

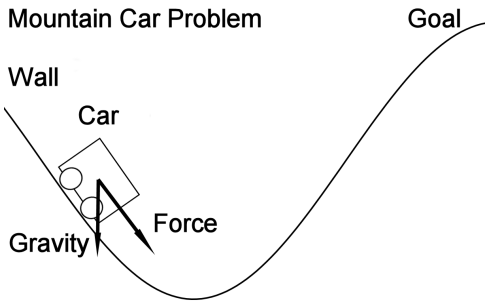
where $Q_{max} = \frac{1}{1-\gamma}$ and $\mu(x)$ is $\frac{\mu_i(x)}{\sum_{i=0}^{N} \mu_i(x)}$. $q[i,j]$ is deemed to have converged when $c[i,j]$ is greater than ϵ. The variance of a rule can only be updated when all the rules in the current match set and previous match set have converged. The overall HFRBS algorithm is outlined in Algorithm 2. In the current implementation, specialisation is only performed at the beginning of every episode but it is envisaged that specialisation could be done after every update.

Algorithm 2 Hierarchical Fuzzy Q-Learning

1: Specialise inaccurate rules
2: **repeat**
3: Observe the input vector x (Compute the match set M_1)
4: Select actions using an Exploration/Exploitation Policy (EEP)
5: Compute the global consequence $a(x)$ and the Q-value $Q(x,a)$
6: Apply the action $a(x)$
7: Receive the reinforcement r
8: Observe the input vector y (Compute the match set M_2)
9: Update the Q-Value of M_1 using (6)
10: Update the Convergence of M_1 using (9)
11: **if** M_1 and M_2 have converged **then**
12: Update the variance with M_1 using (7)
13: **end if**
14: **until** (goal reached) OR (time out)

V. Mountain Car Experiment

The experiments were conducted on a classic reinforcement learning problem, the mountain car as defined by Sutton and Singh [21]. The mountain car problem provides a continuous state space but with only 2 dimensions visualisation of the variable resolution discretization is still possible. The aim of the task is to park a car with zero velocity at the top of a steep mountain road. The car is underpowered and must therefore gain sufficient inertia from the opposite slope to reach the goal. The layout of the problem is depicted in Figure 3.



3: The Mountain Car Problem

The position and velocity are updated according to a simplified physics model as defined by

$$\begin{aligned} x_{t+1} &= x_t + dx_{t+1} \\ dx_{t+1} &= dx_t + 0.001a_t - 0.0025cos(3x_t) \end{aligned} \quad (10)$$

The position is bounded between $[-1.2 \leq x_t \geq 0.5]$ and the velocity between $[-0.07 \leq dx_t \geq 0.07]$. The actions range between $[-1 \leq a_t \geq 1]$ and control the acceleration of the car. At the beginning of each episode, the car is positioned randomly between -1.2 and 0.5 on the slope with a random velocity between -0.07 and 0.07. An episode lasts for 10,000 time steps or until the car reaches the goal at the top of the slope ($x_t \geq 0.5$). If the car hits the left hand side (wall) the velocity is reset to 0. The reward function is defined as: