A FUZZY LEARNING ROBOT FORCE CONTROLLER

RUI ARAÚJO, URBANO NUNES, A. T. DE ALMEIDA
Institute of Systems and Robotics (ISR), and
Electrical Engineering Department, University of Coimbra;
Largo Marquês de Pombal, 3000 Coimbra - PORTUGAL;
Phone:+351(39)34876; Fax:+351(39)35672; email: rui@isr.uc.pt

ABSTRACT

In this paper we describe an approach for the generation of skills for control of multidegree of freedom robotic systems. In the method the acquisition of skills is done on-line by self learning. Instead of generating skills by explicit programming of a perception to action mapping they are generated by trial and error learning, guided by a performance evaluation feedback function. The structure of the controller consists of two fuzzy subsystems both implemented by feedforward multilayer neural networks. The action fuzzy subsystem has the purpose of generating command actions for the system under control. The evaluation-prediction fuzzy subsystem which predicts the future value a function that evaluates the performance of the controller. The approach is applied in learning a robot manipulator force control skill.

KEYWORDS: Fuzzy system, reinforcement learning, force control, robot.

INTRODUCTION

There have been developed control theories to successfully deal with a large class of control problems by mathematically modelling the system/plant to be controlled. These analytical models are used in the controller to generate control actions. Traditional robot programming and control techniques have faced difficulties in contributing to the extension of application domains of robots. In fact, programming based on trajectory interpolation along a prespecified sequence of points has a limited ability to cope with complex tasks. The robot programming and control architectures must be equipped to face unstructured environments, which may be partially or totally unknown at programming time, or environments which are time varying and unpredictable. The use of sensory information feedback has been identified as a requirement for achieving flexibility in robotics. However, sensorial data is not enough. It is required a set of reflexive skills or fine-motion control strategies (e.g. [1]) that perform a real-time mapping between the current sensory situation and an appropriate action or actuation to the system. A skill represents, and has associated, knowledge to perform a certain task or operation of the robot. In this article, skills are viewed as
an activity for control of some feature characterising the interactions between the a robot manipulator and its environment.

However, the problem arises as of how to generate this skills. Supervised learning methods, the most commonly used in neural networks, require a training data set composed of input vectors and corresponding desired output vectors. If the desired output of the network is not known, these methods can not be applied. An interesting possibility, is learning to act over the world by using information received from the sensors for autonomous self-improvement of controller performance. In this article we use a reinforcement learning fuzzy logic control system to generate control actions for the robot manipulator. A reinforcement-learning robot (or system) learns by experimentation and does not require a teacher for proposing correct actions for all possible situations the robot may find itself in. The robot searches and tries different actions for every situation it encounters and selects the most useful ones. This search/selection process is guided by a reinforcement signal that is a performance evaluation feedback function. Thus, unlike supervised learning, the reinforcement learning problem has only very simple “evaluative” or “critic” information instead of “instructive” information available for learning. Reinforcement learning [2] brings some benefits. First, it’s based on a concept of self programming in which control of a complex system in principle does not need extensive analysis and modelling by human experts. Instead the system discovers it’s own abilities. Second the system can cope with disorder. For example, the ability to deal with noisy input data is important for any robot dealing with information close to the raw sensory data. Also the ability to cope with non-stationarity is important for the robot to adapt itself new or time-varying environments.

In this article we suggest the application of reinforcement learning techniques to train a fuzzy controller to perform a force control operation. The structure of the paper is as follows. In the next section we present the controller architecture. In the last two sections of the paper we formulate the force control skill where the controller will be applied, and make some concluding remarks.

**LEARNING CONTROLLER ARCHITECTURE**

Figure 1 presents the overall architecture of the reinforcement learning control system. The controller is based in reinforcement learning methods [2], [3]. It receives from the environment a time-varying vector of input states, $X(k)$, and a time-varying scalar performance evaluation function, $r(k)$, called the (external) reinforcement (signal) and is generated in an evaluation module inside the environment (fig. 1). The objective of learning is for the controller to try to maximise some function of this reinforcement signal, such as the expectation of its value on the upcoming time step or the expectation of some form of accumulation of it’s future values.

The precise method for the generation of the reinforcement must be appropriate to both the particular problem being solved, and the type of sensory information available and it’s relation to the problem. The generation of the reinforcement is assumed to be done in the environment, and the method used for this is assumed to
be unknown to the learning system. Generally the reinforcement is a function of both the state of the environment and the action command generated by the controller (and possibly some other internal state).

The overall controller is composed of a fuzzy logic controller, a learning module, and stochastic search module. The fuzzy logic controller [4] has the role of mapping input state data read from sensors. It has four major components: fuzzifier, knowledge base, inference engine, and defuzzification. The fuzzification module converts input real-valued vector of data, \( \mathbf{X}(k) \), from an observed input space into linguistic fuzzy variables through a predefined number of membership functions. The rule base consists of a set of fuzzy logic “IF-THEN” rules to describe the control policy. The purpose of the inference engine is to match the output of the fuzzifier with the fuzzy logic rules and perform fuzzy implication and approximate reasoning to generate a fuzzy control action. The defuzzifier transforms the inferred fuzzy control action, represented by a combination of output membership functions, into a non-fuzzy output control action. Finally, the data base contains various parameters that characterise the fuzzification, inference, and defuzzification processes.

The defuzzification module has mainly two types of outputs. The first type is a scalar prediction, \( p(k) \), of the future value of the reinforcement signal. The second is an expected action vector \( \mathbf{y}(k) \). This vector is not applied directly to the environment. Instead, it is treated as a mean (expected) action. The actual action, \( \hat{\mathbf{y}}(k) \), is chosen by exploring a neighbourhood around this mean point. The range of exploration is controlled by the variance of a probability density function. In our work, assuming that the predicted reinforcement is \( x = p(k) \in [x_0, x_1] \), the amount...
of exploration is determined by the following standard deviation, $\sigma(k)$:

$$\sigma(k) = K \left| \frac{x - x_1}{x_0 - x_1} \right|^{2\alpha}$$  \hspace{1cm} (1)

where $K = \sigma(x_0)$ is a constant that determines the maximum search range that occurs at minimum predicted evaluation. Equation (1) represents a monotone decreasing function between $K$ and 0, and $\sigma(k)$ can be seen as the extent to which the output node searches for a better action. Since $p(k)$ is the expected reinforcement signal, if $p(k)$ is small, according to (1), the exploratory range will be large, such that the actual action $\hat{y}(k)$ has high probability of being quite different from the mean action $\bar{y}(k)$. This can be seen as an attempt to discover a better action, since the expected action for the current state, $\hat{y}(k)$, is predicted to give a not very good evaluation. On the contrary, if $p(k)$ is large, the exploratory range will be small, such that the actual action $\hat{y}(k)$ has high probability of being very close to mean action $\bar{y}(k)$ since it is expected that the mean action is very close to the best action for the current input vector. The amount of search is thus a function of the current predicted quality of the expected action vector.

In this work we used a Gaussian probability density function to generate the actual action vector, $\hat{y}(k)$, from the expected action vector, $\bar{y}(k)$:

$$\hat{y}(k) = N(\bar{y}(k), \sigma(k))$$

Thus, $\hat{y}(k)$ is a normal random vector where the density function of each component is given by:

$$f(\hat{y}_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\|\hat{y}_i - \bar{y}_i\|^2}{2\sigma^2}}$$  \hspace{1cm} (2)

The transformation from $\bar{y}(k)$ to $\hat{y}(k)$ is performed by the Stochastic Search module of figure 1.

The fuzzy controller part of the system is implemented by a five-layer neural network as can be seen in figure 2. This was inspired by the work of Jang and Sun [5]. The nodes in layer 1 just transmit the input values to the nodes of the next layer. Layer 2 constitutes the fuzzyfication module, and is composed of term nodes corresponding to membership functions. In layer 3 we have rule nodes that perform the precondition matching of fuzzy logic rules. This nodes perform the fuzzy “AND” operation. There are two separate sets of rule nodes, one associated with the reinforcement prediction output and the other associated with the mean action outputs. Therefore we may identify a fuzzy prediction network, that generates the reinforcement prediction signal $p(k)$, and a fuzzy action network, for generation of the expected action $\bar{y}(k)$. However, all the rule nodes share the same set of term nodes of layer 2. The links at layer 4 perform the fuzzy “OR” operation to integrate the fired rules which have the same consequent. The nodes at layer 5 and the links to them act as a defuzzyfier and transmit the output signals of the network.

Learning in this system is performed by error back-propagation [6]. In the prediction network we have the desired output - the external reinforcement signal -
Figure 2: Neural-network-based implementation of the fuzzy logic controller.

so that the output gradient can be calculated and the problem reduces to a back-
proagation supervised learning problem. For the action network we must take into
account the stochastic search unit. In this case we use a method closely similar to
the REINFORCE algorithm [2]. For this purpose we first estimate the gradient of
the reinforcement signal $r(k)$ with respect to the mean output action. Supposing
the function we are wishing to maximise is $r(k)$ itself then, by closely following the
REINFORCE approach we have the following estimate for the gradient information:

$$\frac{\partial r}{\partial y} \approx [r(k) - p(k)] \left[ \frac{\hat{y} - y}{\sigma} \right]_{k-1} \quad (3)$$

The time indexes in equation (3) take into account the assumption that the re-
forcement signal at time step $k$ depends on the input actions at time step $k - 1$. In
equation (3), the term $(\hat{y} - y) / \sigma$ is the normalised difference between the actual
and expected actions, $r(k)$ is the real reinforcement signal provided by the evaluation
module in response to the actual action $\hat{y}(k - 1)$, and $p(k)$ is the predicted rein-
forcement for the expected action $y(k - 1)$. Equation (3) has the following intuitive
foundation. If $r(k) > p(k)$, then $\hat{y}(k - 1)$ is a better action than the expected one
$y(k - 1)$. In this case $y(k - 1)$ should be moved closer to $\hat{y}(k - 1)$. If $r(k) < p(k)$,
then $\hat{y}(k - 1)$ is a worse action than the expected one $y(k - 1)$. In this case $y(k - 1)$
should be moved farther away from $\hat{y}(k - 1)$. After the gradient information for the
action network is estimated using equation (3), we have transformed the reinforce-
ment learning problem in a supervised learning problem. At this point we can apply
the error back-propagation algorithm [6] to adjust the action network weights.
FORCE CONTROL SKILL

We applied the method of the previous section to problem of learning a “force control skill”. This skill consists of taking the robot tool-tip from a free space situation to a contact state with a compliant surface and having a constant interaction force. In this skill the control system generates a unidimensional “change of position” command to the environment (thus \( \hat{\mathbf{f}}(k) \) reduces to a scalar). Due to the compliance of the environment, this action will generate a change in the force state. The PUMA 560 robot manipulator working in “Real-time Path Control” mode was simulated along the Cartesian Z-translation degree of freedom. The contact was against a spring-like compliant surface whose model can be approximated by a stiffness of \( K_m = 37dN/mm \). The reinforcement function, \( r(k) \), used to represent the skill in the learning algorithm, is zero in non-contact situation. When in contact, \( r(k) \) decreases linearly from a maximum value to zero as the absolute value of the error force increases.

CONCLUSION

In this paper we presented an approach for a reinforcement learning fuzzy logic control system. The fuzzy system is implemented by a five layer neural network. Simulations of learning a robot force control task, were performed using the described method. Other techniques can be used for designing force controllers [7]. However, this problem is a starting point to explore the application of learning to typical robot tasks. Learning approaches may prove itself useful in other more complex tasks where modelling and the designing of the controller is more difficult.

REFERENCES