Approaches for the integration of a priori knowledge into an autonomously learning control architecture

Martin Spott[1], Ralf Schoknecht[2] and Martin Riedmiller[2]
[1]Institut für Programmstrukturen und Datenorganisation
[2]Institut für Logik, Komplexität und Deduktionssysteme
Universität Karlsruhe, Postfach 6980, 76128 Karlsruhe
email: [1]spott@ipd.info.uni-karlsruhe.de, [2]{schokn,riedml}@ira.uka.de

ABSTRACT: The control architecture FYNESSE learns control strategies only on basis of success and failure of former control interactions. Thus, learning can be carried out directly at the real process. For reasons of safety and design costs, the number of interactions with the real process must be kept as small as possible. Therefore FYNESSE allows the integration of a priori knowledge about the control strategy. In this paper, different techniques for the integration are proposed. In experiments it is shown that the integration of prior control knowledge reduces the number of interactions with the process dramatically. Furthermore the learning procedure is stabilized, which is indispensable for the application of FYNESSE as an adaptive system.

KEYWORDS: Reinforcement learning, real-time dynamic programming, fuzzy control

1 INTRODUCTION

Autonomous learning of control strategies has become more and more important in recent years. The reason for this fact is that the number and complexity of control applications is rising in a way that requires fast and easy design methods for controllers. Classic controller design is perfect in applications that are similar to known problems. Though, for new complex control problems experts are needed who have to develop process models and find an appropriate design method. This approach is often too time consuming or leads to suboptimal solutions, because the real process is modeled too roughly.

Approaches for autonomous learning on the other hand assume no analytical knowledge about the process. They are based on *learning by doing*; success and failure of former control interactions are used to learn a control strategy. Many experiments impressively demonstrate the capabilities of such methods. But they also reveal that many training cycles are needed, and that the learning process may be instable. That is, the controller seems to unlearn parts of the strategy it already knew.

In FYNESSE a priori knowledge is used to overcome these problems of autonomous learning. We show that rough qualitative knowledge about the control strategy is sufficient for a clear reduction of training time. The stability of learning is also considerably higher than without prior knowledge. Furthermore, the number of interactions with the real process can be reduced by an extensive exploitation of the data that is collected in the learning phase.

Altogether, controller design with FYNESSE roughly works as follows:

1. specify control problem,
2. specify optimization goal (time optimal, energy optimal, etc.),
3. if possible, collect a priori knowledge about control strategy (qualitative as e.g. fuzzy controller, linear control laws, etc.),
4. learn (optimize) control strategy.

This results in controllers of high quality without the need for an analytical model of the process. The control law is learned in direct interaction with the real process. Therefore, a FYNESSE controller is optimized with respect to the real process and not only to a simplified model. In this way, FYNESSE controllers can also adapt to changing environmental conditions.

The following section briefly explains the FYNESSE control architecture. Different possibilities to integrate a priori control knowledge are introduced in section 3. Experiments with the cart pole balancer, an instable, nonlinear process, demonstrate the benefits of the different approaches in section 4.

The control architecture FYNESSE consists of two components: a strategy element that proposes control actions and a critic element that selects the best of the proposed actions. The strategy element is fuzzy and can therefore hold vague and uncertain information about the control strategy as well as crisp control laws like control characteristics or classic linear controllers. The critic element is the learning component of the system. For each state of the system, it must learn which of the proposed actions will be the best choice with respect to the optimization goal (e.g. time optimal or energy optimal control).

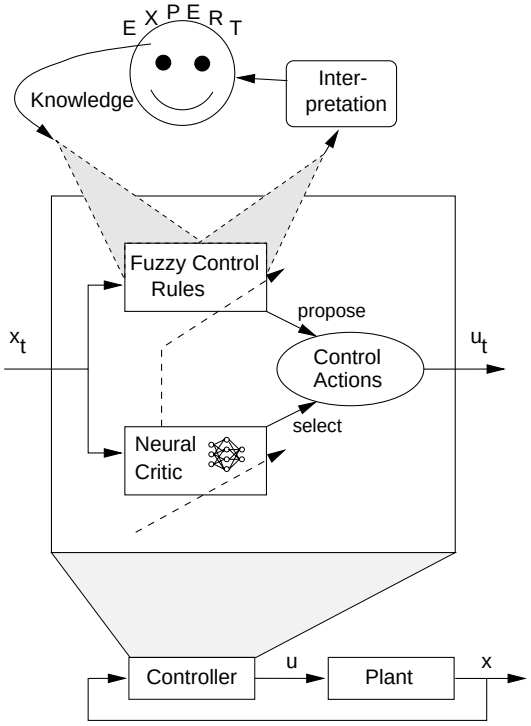


Figure 1: The FYNESSE control architecture

The idea for the learning procedure is based on *dynamic programming*, Bellman (1957), and *reinforcement learning*, Sutton and Barto (1998), i.e. learning the control strategy is mapped to an optimization problem. The objective is to iteratively improve the control strategy. This strategy corresponds to a value function that represents the future costs to go that occur, when starting in a given system state and following the control strategy. Costs can, for example, be measured in time to the goal or energy. In the value function these costs are accumulated along whole trajectories from a starting state to the goal.

The value function used here is called Q-function, Watkins (1989). It depends not only on the system state but also on the control action taken in that state. Thus, it can be used for model-free control. As $Q(x,u)$ can be interpreted in terms of the utility of action u in state x the Q-function implicitly defines a control strategy, i.e. $\pi(x) = argmin_{u \in \mathcal{U}(x)}\{Q(x,u)\}$. This dual semantic of the Q-function, on the one hand as future costs and on the other hand as strategy, is the key for the application in this reinforcement learning framework.

In order to be able to learn the Q-function a suitable representation has to be found. In continuous state spaces that are typically encountered when controlling real processes infinitely many states occur. Even if a discretization is used as in classic dynamic programming the number of states is still to large to be efficiently represented in a lookup table. Thus it is widely accepted that for such problems generalizing function approximators, e.g. neural networks, are needed, Bertsekas and Tsitsiklis (1996).

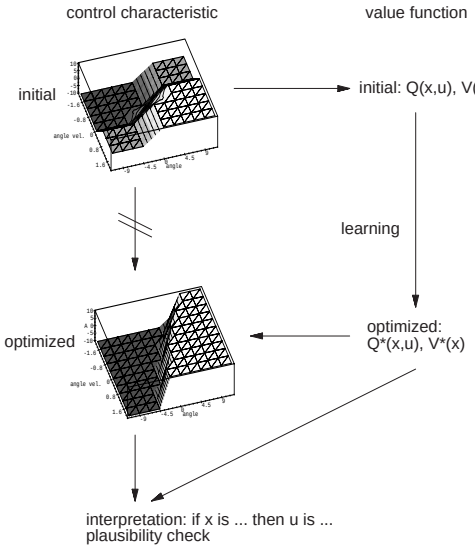


Figure 2: Learning by optimization

Fig. 2 shows the flow of information in FYNESSE. On the left hand side the *explicit* control strategy (strategy element) is depicted and on the right hand side the value function $Q(x,a)$ (critic element) is shown. In contrast to most other learning systems, learning in FYNESSE is not realized by adapting directly the strategy element. Instead, the value function $Q(x,a)$ is trained in a reinforcement learning framework. While repeatedly controlling the process, the value function is adapted on real control trajectories. This method is known as *real-time dynamic programming (RTDP)*, Barto et al. (1995). The update of the Q-function is carried out using *Q-learning* which is based on *temporal difference (TD) learning*, Sutton (1988). As the Q-function implicitly defines a control strategy an adaptation of the Q-values also causes an adaptation of the strategy. This process provably converges if rather strict conditions are satisfied. But experience shows that very good results can be obtained in a relaxed framework.

Whereas it is simple to obtain the control strategy from the value function, the reverse is not directly possible. This can be easily understood by considering that the Q-function contains more information than the control strategy itself, i.e. the estimated costs to the goal. This is exactly the problem of integrating a priori knowledge about the control strategy.

3 INTEGRATION OF A PRIORI KNOWLEDGE

Autonomous learning is a very powerful technique in the sense that it is possible to learn control strategies from scratch, i.e. without any prior knowledge about the strategy. This is not for free, of course: the number of training cycles is generally very high and the learning procedure is often not very stable. Sometimes the system even locally unlearns strategies. Especially when learning on a real process, such behavior is not tolerable. The costs for learning would be too high and imperfect control strategies may drive the process into dangerous states.

In many control applications linear process models are available and therefore linear controllers can be developed. If not, the designer has at least partial or vague (qualitative) knowledge about the problem. Therefore FYNESSE allows to incorporate prior knowledge into the learning process, Schoknecht et al. (1999). Two different approaches are described in the next two sections. Using these, the number of necessary interactions with the real process is dramatically reduced. The stability of the learning procedure is significantly higher and the quality of the resulting controllers is better, as well.

3.1 SPECIFICATION OF THE ACTION SET

Generally, control actions can take any value in a bounded continuum. In the example with the cart pole balancer, actions (forces) in the interval $\mathcal{U} = [-10N, +10N]$ are allowed to be applied to the cart. Since learning of the cost function on the continuous set of possible actions is too complex, it is restricted to a finite subset of $[-10N, +10N]$. Typical choices are $\mathcal{U} = \{-10N, +10N\}$ (bang-bang control) or $\mathcal{U} = \{-10N, 0N, +10N\}$.

The action sets above are not dependent on the state of the system. If we have knowledge about the control strategy, we can use it for a more specific definition of $\mathcal{U}$. For example, a control characteristic $u_{a\_priori}(x)$ can be added to the bang-bang controller. We obtain an action set $\mathcal{U}_{a\_priori}(x) := \{-10N, u_{a\_priori}(x), +10N\}$, i.e. in each state the FYNESSE controller can choose between the a priori controller and the extremal actions. This yields a hybrid controller. After the optimization procedure, the resulting controller should be at least as good as the a priori controller. The experiments in section 4 indeed show that the learned controller is much better than the initial strategy. Furthermore, the stability of the learning procedure is considerably improved. This is due to the fact that it is less likely to choose a bad action with respect to the costs, since the set of possible actions is chosen more carefully.

Even more specific sets of possible actions can be obtained by excluding actions in parts of the state space. If, for example, the pole falls to the right und the cart goes to the left, a negative force should not be applied to the cart. Such an exclusion of actions as well as an adaption of the action sets during learning are subject to our current research.

3.2 INITIALIZING THE VALUE FUNCTION

In this section we describe the integration of prior knowledge into the learning process via an appropriate initialization of the value function. Given an initial control strategy π_0 a corresponding value function is learned that represents the costs to go from a starting state to the target region when using π_0 This is done by laying a coarse grid over the state space. The resulting grid points G are used as starting states for controlling the plant with the initial controller. Thus, for each starting state one obtains the corresponding cost-to-go value of the initial controller. These cost values are used to appropriately initialize the Q-function.

$Q^{\pi_0}(x, u)$ denotes the cost-to-go that arises when action u ist taken in state x and the initial control law is followed from then on. Thus the Q-values can be computed by generating a trajectory for each state-action pair (x, u) where $x \in G$ and $u \in \mathcal{U}_{a\_priori}(x)$. This results in a total number of $|G| \cdot |\mathcal{U}_{a\_priori}(x)|$ trajectories. The state-value pairs enable a supervised training of the Q-function in the critic element. After the training is completed the value function can be used for controlling the system. In each state the action u with minimal Q-value is selected. Provided that the Q-function has been accurately approximated this will improve control performance as it can be interpreted as the policy improvement step of the *policy iteration algorithm*, Bertsekas and Tsitsiklis (1996). Thus the Q-function is appropriately initialized and can be used for further learning in

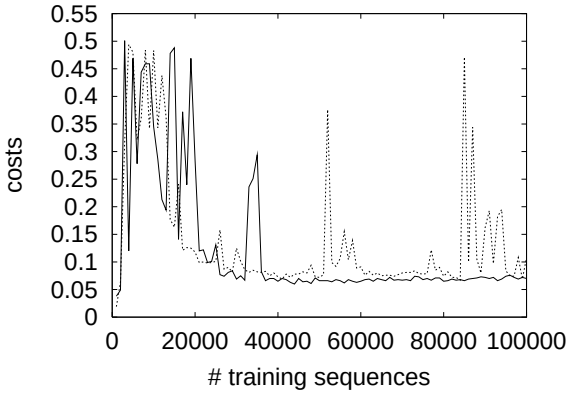


Figure 3: Learning process in case of using a priori control knowledge in form of a simple fuzzy control law (solid line) and in case using no a priori knowledge (dashed line).

the FYNESSE framework. Using this method for integrating prior knowledge the number of interactions with the real system needed to learn a good control strategy can be drastically reduced. This is indispensable when learning to control a technical process incurs costs, e.g. in terms of operation and wear.

4 EXPERIMENTS

In the following we present results for the different ways of integrating prior knowledge into the process of learning a control law for the cart pole balancer. The objective is to move the pole in upright position and the cart in the middle of the track. In principle the approaches described here can be used to develop a controller in direct interaction with a real cart pole balancer. As a robust physical system was not available we used a simulation which models the input-output behaviour of the system.

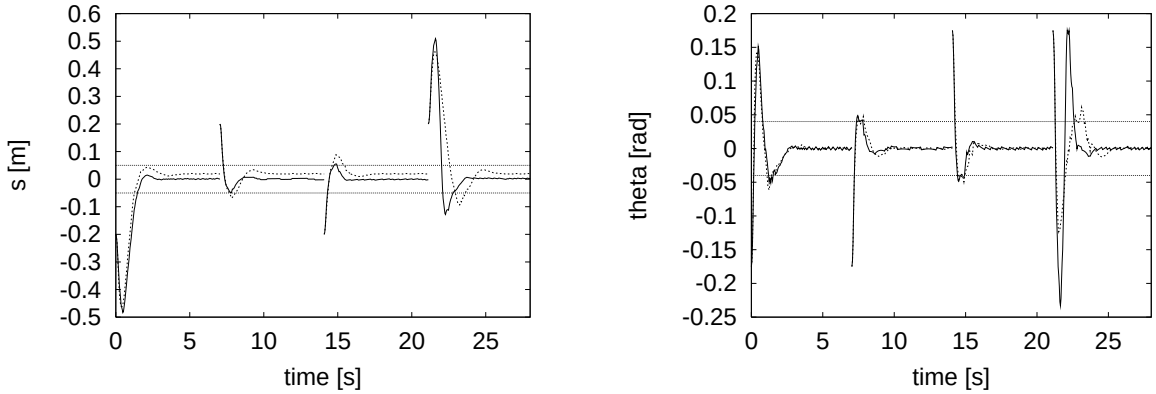
4.1 SPECIFICATION OF THE ACTION SET

As described, one possibility of integrating prior knowledge into the learning process is a modification of the action set. The basis action set $\mathcal{U} = \{\pm 10N\}$ is extended by the a priori action which yields $\mathcal{U}_{a\_priori}(x) = \{u_{a\_priori}(x), \pm 10N\}$. As prior knowledge a simple fuzzy controller is used that was developed in approximately 2 hours. A great deal of improvement in control behaviour is still possible. Mainly the response time is too long and the controlled variables keep oscillating around the target. Nevertheless, the system is approximately controlled into the target region and the pole does not fall down any more. Thus the initial fuzzy controller fulfils the basic control requirements and it can be used as starting point for further controller design. The goal is to stabilize the learning behaviour of the autonomously learning control architecture FYNESSE by using the fuzzy prior knowledge. Moreover, in the resulting hybrid controller the disadvantageous characteristics of the simple fuzzy controller should be eliminated.

In Figure 3 the learning process of the hybrid fuzzy controller in comparison to a controller learned without prior knowledge is depicted. It shows the average costs of the control trajectories in the course of training where each control step outside the target region incurs local costs of 0.002. As can be easily noticed the costs for the best hybrid controller (training sequence 44000) are less than for the best controller learned from scratch (training sequence 41000). Moreover, the learning process of the hybrid controller is much more stable than for the controller without prior knowledge which keeps forgetting successful control strategies. A stable learning process is indispensable for an adaptive autonomously learning controller because every instability would cause a substantial decline in control performance while adapting to the system behaviour.

In Figure 4 the performance of the best controllers is shown by plotting the time histories for the controlled variables starting from four representative initial conditions. In comparison to the original fuzzy controller the transient response of the hybrid controller is much faster. Moreover, wave-like oscillations of the cart position that occured with the fuzzy controller could be eliminated during learning. Now the target region is reached quickly and permanently.

The controller without prior knowledge as well solves the control task but the transient response is slower which causes higher costs. Since the control task is fulfilled when the system reaches the target region slight



(a) (b)

Figure 4: Time histories for (a) cart position and (b) pole angle when using the hybrid controller (solid line) and the controller learned from scratch (dashed line). Starting from the four initial conditions control sequences of $7s$ each are depicted. The corresponding target region is dotted.

stationary inaccuracies may occur as can be seen in the case of the controller without prior knowledge. As the initial fuzzy controller represented a symmetric control law the hybrid controller could more easily develop an implicit symmetric switching curve between the positive and negative action. This leads to considerably higher stationary accuracy.

4.2 INITIALIZING THE VALUE FUNCTION

In the following prior knowledge about the control law is integrated into the learning process via an initialization of the Q-function. Here the prior knowledge consists of a linear controller that was developed using *linear quadratic regulator (LQR) design* based on a linearized model of the plant. Such a linearized model of the plant is often easy to obtain. Given a nonlinear system, however, this linear model is only valid in tight bounds around the working point. This is the motivation for using a Q-function for controlling the plant, which does not need a system model. As described in section 3.2, the Q-function for the linear control law is computed on a grid laid over the state space[1]. Cart position s and pole angle θ were discretized using 7 and 5 values with a spacing of $0.1m$ and $0.1rad$ respectively. The $\dot{s}$-$\dot{\theta}$ plane was discretized using 9 points that were equally distributed over the range where the system is operated. This yields 315 grid points altogether. As $|\mathcal{U}_{a\_priori}(x)| = 3$ a total number of 945 trajectories has to be generated in direct interaction with the system. This involves about 80000 interactions with the system. Compared to the training in the last section where in over 40000 trajectories about $3.8 \cdot 10^6$ interactions were needed this is a reduction by a factor of 50.

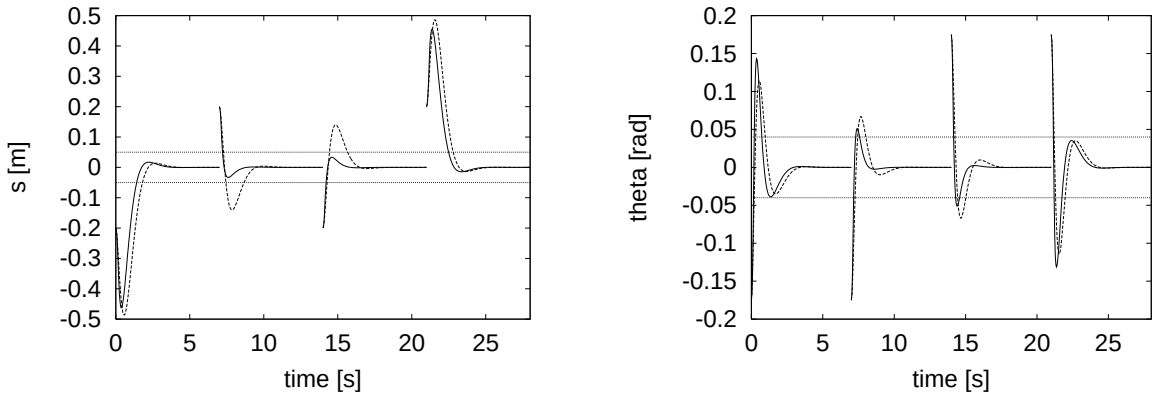
With the resulting patterns (state-action)$\rightarrow$value a neural network (5-10-10-1 architecture) is trained using fast gradient techniques such as Rprop, Riedmiller and Braun (1993). After 10000 epochs a good approximation of Q has been learned. Note, once the patterns are determined no further interactions with the system are required for the supervised learning task. Figure 5 depicts the control behaviour obtained with this learned Q-function in comparison to the initial linear controller. The improvement is obvious: The average time spent outside the target region is reduced from $1.66s$ to $1s$ for the four trajectories. This result is so good that further training within the FYNESSE framework only results in a slight improvement.

5 CONCLUSIONS

In this paper we presented the autonomously learning control architecture FYNESSE that is suited for model-free adaptive controller design. There are two important objectives for such a learning approach:

- The learning behaviour should be stable so that good control strategies are preserved because a substantial decline in control performance is not tolerable for an adaptive controller.

[1] The state of the cart pole balancer is uniquely defined by the vector $(s, \theta, \dot{s}, \dot{\theta})$, where s denotes the cart position and θ denotes the pole angle.



(a) (b)

Figure 5: Time histories for (a) cart position and (b) pole angle when using the controller initialized with linear prior knowledge (solid line) and the pure linear controller (dashed line). Starting from the four initial conditions control sequences of $7s$ each are depicted. The corresponding target region is dotted.

- The number of interactions with the real system should be kept low in order to reduce operation costs and wear.

Integration of prior knowledge is the key mechanism for achieving these objectives. Using prior knowledge to modify the action set has a stabilizing effect on the learning behaviour. As we showed, a quite imperfect initial fuzzy controller is already sufficient as prior knowledge. The learned hybrid controller exploits the strenghts of the initial controller and compensates its weaknesses.

The number of interactions with the system needed to learn a good control law can be drastically reduced by appropriately initializing the Q-function. On the basis of a linear control law and a coarse discretization of the state space the corresponding Q-function was very well approximated. The resulting control performance was clearly superior to the initial linear controller.

These results show that autonomously learning a control law with the FYNESSE architecture is a powerful and flexible method of designing controllers for dynamic systems.

REFERENCES

Barto, A. G., Bradtke, S. J. and Singh, S. P. (1995): Learning to Act using Real-Time Dynamic Programming. *Artificial Intelligence*, (72): 81–138.

Bellman, R. E. (1957): *Dynamic Programming.* Princeton University Press, Princeton, NJ.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996): *Neuro Dynamic Programming.* Athena Scientific, Belmont, Massachusetts.

Riedmiller, M. and Braun, H. (1993): A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, edited by Ruspini, H., pp. 586 – 591, San Francisco.

Schoknecht, R., Spott, M. and Riedmiller, M. (1999): Design of self-learning controllers using FYNESSE. In *Deep Fusion of Computational and Symbolic Processing*, edited by Furuhashi, T., Tano, S. and Jacobsen, H. Physica. to appear.

Sutton, R. S. (1988): Learning to predict by the methods of temporal differences. *Machine Learning*, (3): 9–44.

Sutton, R. S. and Barto, A. G. (1998): *Reinforcement Learning.* MIT Press, Cambridge, MA.

Watkins, C. J. (1989): *Learning from Delayed Rewards.* Phd thesis, Cambridge University.