Reinforcement learning without an explicit terminal state

Martin Riedmiller
Institut für Logik, Komplexität und Deduktionssysteme
Universität Karlsruhe, D-76128 Karlsruhe, Germany
e-mail: riedml@ira.uka.de

Abstract—
The article introduces a reinforcement learning framework based on dynamic programming for a class of control problems, where no explicit terminal state exists. This situation especially occurs in the context of technical process control: the control task is not terminated once a predefined target value is reached, but instead the controller has to continue to control the system in order to avoid the system's output drifting away from its target value again. We propose a set of assumptions and give a proof for the convergence of the value iteration method. From this a new algorithm, which we call the fixed horizon algorithm, is derived. The performance of the proposed algorithm is compared to an approach, that assumes the existence of an explicit terminal state. The application to a cart/double pole-system finally shows the application to a difficult practical control task.

I. Introduction

A typical task in process control is to control the outputs of a system to permanently equal some externally given target values. Consider for example the control of a chemical reactor where an externally specified concentration of a chemical substance must be permanently kept independently of external distortions by an appropriate choice of heating and cooling decisions.

In the following, we focus on a neural network based reinforcement learning controller, where the only training information is given in terms of the control target [6]. Learning is based on the application of approximate dynamic programming methods which is realized by an iterative approximation of an optimal cost function during control of the process [2], [1], [4].

A typical way to ensure convergence within this framework is to assume an absorbing terminal state [7]. However, in many technical process control applications, such an absorbing terminal state does not exist. Instead, control is an ongoing task: once the output has reached its target value the task is *not* terminated, but the controller has to *actively* keep the output at the desired level. This means, that in order to tackle such problems in an appropriate manner, we have to look for a framework that allows to formulate the control problem without assuming an explicit terminal state.

In the following we present a framework, that makes no such assumption. We proof convergence of the value iteration method and show, that a certain region in state space is absorbing, when the resulting optimal policy is applied. The implementation of the framework leads to a special form of a *Real Time Dynamic Programming* [1] learning algorithm which is called fixed horizon algorithm here.

II. Problem Formulation

In the following, we consider a dynamic system which is observed at discrete time steps

$$\begin{aligned} x_{t+1} &= f(x_t, u_t, w_t) \\ y_t &= g(x_t) \end{aligned} \qquad (1)$$

where x_t denotes the current state of the system, u_t is the action or control signal applied to the plant, w_t denotes random noise and f describes the system's transfer function. The output y_t of the system is computed depending on state x_t by the output function g. The control signal u_t that is applied to the plant is determined by the control policy $\pi : \mathcal{X} \mapsto \mathcal{U}, u_t := \pi(x_t)$, which selects one control signal out of a finite set of available actions $\mathcal{U} = \{u_1, \ldots, u_n\}$. The idea of a self-learning controller is to embed the learning problem in a dynamical optimization problem [2], [1], [6]. The goal then is to find an optimal control strategy π^* that minimizes the accumulated costs over a complete control trajectory, when at the beginning the system's state is x_0:

$$J^*(x_0) = \min_{\pi \in \Pi} E\{\sum_{t=0}^{\infty} r(x_t, \pi(x_t))\}. \qquad (2)$$

Here, the function $r : (x, u) \mapsto \mathbb{R}$ denotes the immediate costs that arise, when action u is applied in state x. Thus by choosing r, the quality of the control strategy can be specified. A method to solve the dynamic optimization problem above is the *value iteration* technique that approximates the optimal accumulated cost function J^* by iteratively improving estimates for the expected optimal costs for the states [2]:

$$J_{k+1}(x) := \min_u E\{r(x,u) + J_k(f(x,u,w))\}. \quad (3)$$

Under certain assumptions, which will be discussed in section III, the sequence J_k converges against J^*. Once J^* has been determined, the optimal control strategy is also known by greedily evaluating J^*

$$\pi^*(x) := \arg\min_u E\{r(x,u) + J^*(f(x,u,w))\}. \quad (4)$$

III. The fixed horizon framework

Convergence of the value iteration method can be guaranteed under several assumptions [3], [1]. As discussed in the above section, the broadly used framework of assuming an explicit terminal state often does not allow the appropriate formulation of process control problems. In the following, we proof convergence under assumptions, that do not assume an absorbing terminal state and therefore will turn out to provide an adequate formulation for many technical control tasks.

A. Theoretical foundations

Proposition 1: We consider a stochastic discrete state dynamic system with a finite state set $\mathcal{X} = \{1 \ldots, n\}$ with transition probabilities $p_{ij}(u) = P\{x_{t+1} = j | x_t = i \wedge u_t = u\}$. Further we assume a finite set of available actions $\mathcal{U}(x)$ and $J_0(x) \leq J^*(x), \quad \forall x \in \mathcal{X}$. Under the assumptions:

1. Exists a nonempty subset $\mathcal{X}^0 \subseteq \mathcal{X}$ where
   (a) $\forall x \in \mathcal{X}^0 \, \forall u \in \mathcal{U}(x) : r(x,u) = 0$
   (b) $\forall x \notin \mathcal{X}^0 \, \forall u \in \mathcal{U}(x) : r(x,u) > 0$
2. Exists an (unknown) policy π' and a nonempty subset of cost-free states $\mathcal{X}' \subseteq \mathcal{X}^0$
   (a) $P(x_{t+1} \in \mathcal{X}' | x_t \in \mathcal{X}', u_t = \pi'(x_t)) = 1$
   (X' is 'absorbing' when policy π' is applied)
   (b) π' is proper with respect to $\mathcal{X}'$, i.e. there exists an integer $m \leq n$ after which a state $x \in \mathcal{X}'$ is reached, when policy π' is applied

then, value iteration (equation 3) converges against the optimal value function J^* (equation 2) and J^* is bounded

$$\forall x \in \mathcal{X} : J^*(x) = E\{\sum_{t=0}^{\infty} r(x_t, \pi^*(x_t))\} \leq c. \quad (5)$$

Proof: Under assumption 1, the principle convergence of the value iteration procedure against the optimal value function can be shown. A detailed proof can be found in [3][1]. Since in this general scenario J^* may have an infinite value for some states, it remains to show, that under the additional assumptions, J^* only takes finite values. The boundedness of J^* (equation 5) results when assumptions 2(a) and (b) are fulfilled: Assume that policy π' is applied to the system. Then, with 2(a) for all states $x \in \mathcal{X}'$:

$$x \in \mathcal{X}' \Rightarrow J^{\pi'}(x) = \sum_{t=0}^{\infty} \underbrace{r(x_t, \pi'(x_t))}_{0, \text{ since } x_t \in \mathcal{X}' \subseteq \mathcal{X}^0} = 0.$$

On the other hand, by assumption 2(b) when applying π', the probability of not reaching a state within $\mathcal{X}'$ in at most m steps is smaller than one, i.e. $\rho := \max_{x \in \mathcal{X}} P\{x_m \notin \mathcal{X}', x_0 = x\} < 1$. Thus, the probability of not reaching $\mathcal{X}'$ after v times m steps is $P\{x_{v\,m} \notin \mathcal{X}'\} \leq \rho^v$. If $\bar{r} := \max_{x,u}\{r(x,u)\}$ denotes the maximum of immediate costs (exists, because $\mathcal{X}$ and $\mathcal{U}$ are finite sets) then the maximum costs for a sequence of m steps are bounded by $G := m\,\bar{r}$. Partitioning an infinite sequence into v sequences of length m yields

$$\begin{aligned} J^{\pi'}(x) &= E\{\sum_{v=0}^{\infty} \underbrace{\sum_{t=v\,m}^{(v+1)\,m-1} r(x_t, \pi'(x_t))\}}_{\leq G} \\ &\leq \sum_{v=0}^{\infty} \rho^v\, G = \frac{G}{1-\rho}. \end{aligned}$$

This means that the value function under policy π', $J^{\pi'}$ is bounded above. Considering the optimal value function J^* which generally fulfills $\forall \pi \in \Pi : J^* \leq J^{\pi}$, and therefore the following equation holds:

$$\forall x \in \mathcal{X} : J^*(x) \leq J^{\pi'}(x) \leq \frac{G}{1-\rho}.$$

■

Corollary 1: When the optimal policy π^* (equation 4) is applied to the system, then there exists a nonempty subset $\mathcal{X}^* \subseteq \mathcal{X}^0, \mathcal{X}^* := \{x | J^*(x) = 0\}$ for which the following is true:

1. $P(x_{t+1} \in \mathcal{X}^* | x_t \in \mathcal{X}^*, u_t = \pi^*(x_t)) = 1$
   (X^* is absorbing when policy π^* is applied)
2. π^* is proper with respect to $\mathcal{X}^*$. (the application of π^* guarantees, that independently of

[1]to proof this, assumption 1 can be relaxed to $r(x,u) \geq 0$

the initial state, the process finally reaches a state within $\mathcal{X}^*$)

Proof: The nonemptyness of $\mathcal{X}^*$ follows from the existence of such states for the policy π' and the optimality of J^*.

1. Assume $\mathcal{X}^*$ is not absorbing when π^* is applied. Then there exist states i, j where $i \in \mathcal{X}^*, j \notin \mathcal{X}^*$ and $p_{ij}(\pi^*(i)) > 0$. Then $J^{\pi^*}(j) > 0$, since $j \notin \mathcal{X}^*$. According to the Principle of Optimality

$$J^*(i) \overset{!}{=} \{r(i, \pi^*(i)) + \sum_{j=1}^{n} \underbrace{p_{ij}(\pi^*(i))\, J^*(j)}_{>0}\} > 0$$

This contradicts the assumption that $i \in \mathcal{X}^*$.

2. Assume there is a sequence which does not reach $\mathcal{X}^*$ under the optimal policy. Then there are infinitely many states for which $r(x_t, \pi^*(x_t)) \geq \min_{(x,u)}\{r(x,u) | r(x,u) > 0\}$. (Otherwise there exists an N where $t \geq N \Rightarrow r(x_t, \pi^*(x_t)) = 0$ and therefore $x_N \in T(\pi^*)$). Thus the costs for this sequence are unbounded, which contradicts $J^* \leq \frac{G}{1-\rho}$. ■

B. *Discussion of the results*

A crucial point of the above framework is, that no absorbing terminal state is assumed. Instead, as stated in corollary 1 the property of absorption is *policy-depending*: the optimal policy π^* that is automatically acquired during iteration, controls the system from any starting situation to a certain region $\mathcal{X}^*$ in state space. This region is absorbing *if the optimal policy is applied*, which means, that *then* the system's state is permanently kept within that region. Furthermore, an important fact about the location of this region is known: It lies within a region of states, for which the immediate costs $r(x, u)$ are zero. Thus in order to use this framework for a self-learning controller, we simply have to choose the immediate costs to be zero, if the output of the system equals the target. In all other cases, positive costs arise (assumption 1). The latter is a quite natural demand since every single operation applied to a physical system means positive costs in terms of resources, e.g. energy or time. Assumption 2 of the proposition requires the existence of a policy, that is able to control the system such that after a certain time the current immediate costs are equal to zero. Again, in the application framework considered here, this is a reasonable demand, meaning that it must be in principle possible for the controller to control the system's output close to its target value. Note, however, that this policy is not assumed to be known in advance and therefore does not conflict with the demand of not assuming any a priori knowledge.

C. *Requirements*

Consider a control task, where the system's output (vector) $y(t)$ should equal a given target value after some time, i.e. $\forall t > t_0 : |y(t) - y^{\text{target}}| < \epsilon$. Then the above framework requires to

- select

$$r(x,u) := \begin{cases} 0\,, \text{if } |g(x) - y^{\text{target}}| < \epsilon \\ > 0\,, \text{(arbitrarily), else} \end{cases}$$

- provide control signals $\mathcal{U}$, such that the task in principle can be solved

Then by corollary 1 it is assured, that the resulting optimal policy controls the system permanently within the set of cost-free states, which by the above choice of $r(x, u)$ means, that the output equals to its target value.

D. *The fixed horizon algorithm*

The implementation of the value iteration method 3 from dynamic programming is the foundation of a neural control system that is able to learn complex policies, when only the control goal is specified [6], [5]. The idea of applying the iteration scheme during the control of the system, allows to extend the basic value iteration method to systems with unknown dynamic behaviour and/ or very large state spaces. This idea is known as the *Real Time Dynamic Programming* approach [1]. A control trial is started in an arbitrary initial state and the system is controlled according to the current policy, while the value function is permanently updated due to the experience collected during the control trial. If one assumes an absorbing terminal state, a control trial is naturally stopped, if such a terminal state is reached and the algorithm is formulated accordingly (see figure 1). To distinguish this type of algorithm from the following, we call this approach the terminal state algorithm.

```
Start system in $x_0 \in \mathcal{X}^{\text{start}}$
While ($\mathbf{x}_t \notin \mathcal{X}^+$) {
  select & apply action
  update $J$ }
$J_{k+1}(x_t) := 0$
```

Fig. 1. The terminal state approach. Each trial stops, if a certain target state is reached.

In the fixed horizon framework proposed in this paper, no terminal state is given and therefore each control trial theoretically has infinite length - hence an alternative stopping criterion has to be found. The idea here is to partition the trial into infinitely many parts of finite length N, which leads to

$$\begin{aligned} J^*(x) &= E\{\sum_{t=0}^{\infty} r(x_t, \pi^*(x_t))\} \\ &= E\{\sum_{v=0}^{\infty} \sum_{t=v\,N}^{(v+1)\,N-1} r(x_t, \pi^*(x_t))\} \end{aligned}$$

and to approximate the value function on the finite parts. Therefore, each trial ends after a predefined number of time steps N. For the last state, no special terminal costs arise; instead we might do no assignment at all and consider the last state as a potential starting state for a future trial. Due to the fact that each trial lasts the same number of time steps, we named this approach the fixed horizon algorithm (see figure 2).

```
*Start* system in $x_0 \in \mathcal{X}^{\text{start}}$
*While* ($\mathbf{t} < \mathbf{N}$) {
    *select & apply action*
    *update* $J$ }
$\mathcal{X}^{\mathbf{start}} := \mathcal{X}^{\mathbf{start}} \cup \{x_N\}$
```

Fig. 2. The fixed horizon algorithm. Each trial runs for a predefined number of time steps. The control target is expressed in terms of the immediate cost function $r(x, u)$.

IV. Practical Comparison

In the following, the proposed fixed horizon algorithm is compared to an approach, that assumes a terminal state. The task is to control the output of a low dampened second order dynamical system from random starting situations to equal to the target value $y^{\text{target}} = 0$ and to keep it there. The controller can use two different control signals $\mathcal{U} = \{-1, 1\}$. A multi-layer perceptron with 2 input and 5 hidden units is used to learn and represent the value function [5].

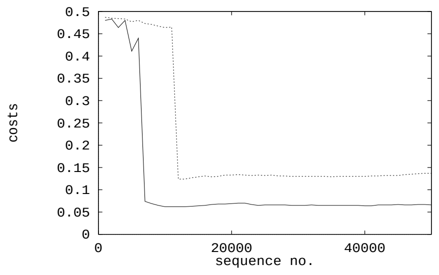
A. The terminal state approach

First, we consider the terminal state approach, where we have to choose a set of terminal states $\mathcal{X}^+ := \{x | y = g(x) < 0.05\}$. Each trial starts with the system being in a random initial state and is terminated, if a state within $\mathcal{X}^+$ is entered. For this terminal state, the value function is set to zero (see algorithm 1). The immediate cost function is chosen such that each action implies equal costs independent of the current state or the action selected; this expresses our desire to reach the target in a minimum number of time steps [6].

B. The fixed horizon approach

Within the proposed fixed horizon framework, *no* absorbing terminal state is assumed. Instead, the control target is expressed in terms of the immediate cost function $r(x, u)$. With respect to the target value $y^{\text{target}} = 0$, we choose $r(x, u) = r(y) = 0 :\Leftrightarrow |y| < 0.05$ according to the proposal in section III-C. In all other situations, we consider equal positive costs independent of state and actions. In contrast to the above terminal state approach, here this means, that we try to minimize the amount of time steps *from now to infinity* where the output of the system deviates from its target value by more than 0.05. During the learning phase, each control trial lasts a fixed number of time steps (here: $N = 100$), then a new trial is started (figure 2(a)).

C. Results



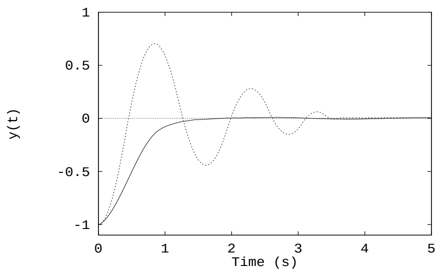


Fig. 3. Comparison between terminal state approach (dotted line) and fixed horizon approach (solid line). (a) cumulated costs over a period of 5 s. Figure (b): Output of the controlled system. The terminal state approach controller very quickly reaches the target value but then overshoots and shows strongly oscillating behavior. The fixed horizon approach reaches the target value later but then manages to keep the output at the target value with high accuracy.

Figure 3(a) shows the cumulated immediate costs that arise during a control period of 5 s averaged over 100 control trials. At the beginning of training, both algorithms start with random policies, that lead to high control costs. This initial bad behaviour is due to the fact, that no a priori control knowledge is provided to the learning system. After about 8,000 trials, the costs for the fixed horizon system begin to decrease rapidly and constantly reach a low level - the controller has learned to fulfill its task (solid line). The terminal state controller also learns, but notably never reaches the same good performance as the fixed horizon controller. This is also demonstrated, when one looks at the output of the controlled system (figure 3(b)). The solid line shows the output of the system controlled by the fixed horizon controller. The target value is reached quickly and with a good stationary accuracy leading to low overall control costs. In contrast to that, the terminal state controller (dotted line) reaches the target value very quickly, but fails to keep the output at this value and thus overshoots the target. This behaviour leads to high overall control costs.

Both algorithms perfectly learn their task, but the different underlying optimization goals lead to substantially different results in the application phase. Optimizing the cumulated costs to an assumed terminal state which is the notion of the terminal state approach, results in a strongly oscillating control behavior during application. On the other side, by considering the complete control trial, the fixed horizon approach has proven to be much more appropriate to deal with the considered type of control problem. It has shown to be able to automatically find a reasonable balance between quickly approaching the target value and avoiding to overshoot.

V. Application example: A cart pole system with two poles

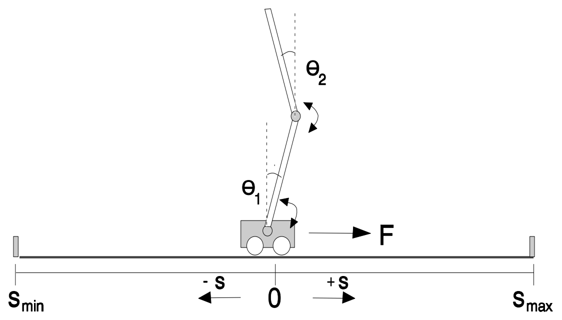


Fig. 4. Cart with two poles

From a technical point of view, the double inverted pendulum is an instable single input, multiple output system, with the acceleration of the cart being the control input and the position of the cart and the angles of the two poles being the three outputs to be controlled. The control task considered here is to control the system from an initial disturbance to a stable state, where the two poles stand upright ($\theta_1 = \theta_2 = 0$) and additionally, the cart is moved to a certain position, which is set arbitrarily to the middle of the track here ($s = 0$). We allow the controller only to use three control signals $\mathcal{U} = \{-15N, 15N, 0\}$ corresponding to an acceleration to the left, to the right and no acceleration at all. No a priori knowledge is provided, the neural controller has to learn to control the instable system completely from scratch.

A. Self-learning neural control

The above situation is a perfect example for the application of the proposed fixed horizon algorithm. The double inverted pendulum is an instable dynamical system. Therefore we cannot assume to control the system to a terminal state where no further control is needed. But we may hope that the controller is able to learn to control the process finally within a region of the state space, where all of the interesting output variables permanently fulfill the specified conditions.

Within the fixed horizon framework this means, that the desired control target has to be expressed in terms of the immediate cost function $r(x, u)$: The immediate costs are equal to zero, if all the three output values are *temporarily* close to their target value (i.e. the cart is in the middle and the two poles are upright). In all other cases, constant immediate costs occur. In order to minimize the *cumulated* costs, the controller thus has to find a policy that manages to control all three outputs to their target values and permanently keep them there. Neither a strategy that only temporarily fulfills the goal, nor a strategy that considers only a part of the outputs is a successful one.

The neural network that was used to learn the task consists of 6 input, 30 hidden units and a single output unit. Training was started with randomly initialized weights, which means that the controller starts with a random initial policy. Each trial lasts for 100 time steps, then a new trial with a random initial position is started. Additional constraints have to be considered: one is to always keep the poles balanced ($|\theta_{1,2}| < 90^o$) and the second is to keep the cart within the boundaries of the track ($|s| < 1.0m$). If one of the constraints is violated, the trial is immediately stopped, and the target value of the neural cost function is set to a maximum value.

After 49 000 trials (about an hour of computer time) the controller has learned a successful policy from scratch. As expressed by the choice of our immediate cost function $r(x, u)$, the controller not only has to balance the system *somehow*, but has to learn a *time optimal* policy. In the control problem at hand this means, that the time until *all three* outputs are zero has to be minimized - thus an appropriate control policy has to consider its impact on all three output variables *simultaneously*. For example a modular control strategy that firstly will control the upper angle, then the lower angle and finally will move the cart towards the middle will be very likely to be *not* the optimal policy in this case.

Figure 5 shows the sophisticated policy that the controller learned to fulfill this difficult control task: First, the cart is shortly accelerated towards the middle, resulting in an increase of the lower angle θ_1. Then after about 0.2 seconds it is immediately accelerated in the opposite direction, until both angles have approximately the same declination towards the middle of the track. Finally, the cart is again accelerated towards the middle position, and therewith eventually reaches its goal of two upright poles and the cart being placed in the middle of the track.

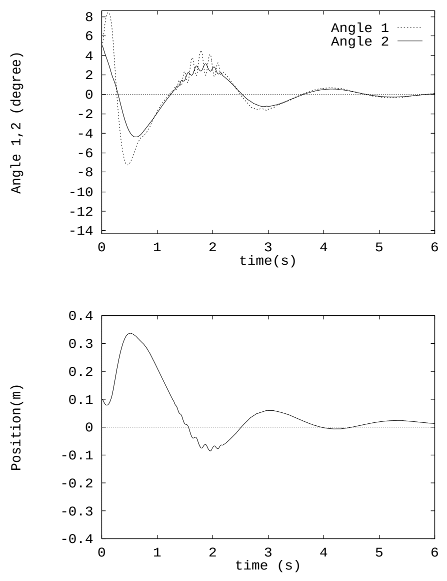


Fig. 5. Neural control of a double inverted pendulum: temporal behaviour of the three system variables θ_1, θ_2 and position s of the cart.

VI. Conclusion

The article gives a convergence proof for the value iteration method for an important class of technical control problems, where no absorbing terminal state can be assumed. The derived fixed horizon algorithm has been compared to the alternative terminal state approach, that assumes a terminal state. It has been shown on a typical control task, that if an assumed absorbing terminal state does not *actually* exists then the proposed fixed horizon approach is the more appropriate framework to formulate the learning task. Finally, its practical applicability has been successfully shown on a instable multi output system.

References

[1] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, (72):81–138, 1995.

[2] A. G. Barto, R. S. Sutton, and C. J. C. H. Watkins. Learning and sequential decision making. Technical Report COINS TR 89-95, Department of Computer and Information Science, University of Massachusetts, Amherst, September 1989.

[3] D. P. Bertsekas. *Dynamic Programming*. Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

[4] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro Dynamic Programming*. Athena Scientific, Belmont, Massachusetts, 1996.

[5] M. Riedmiller. Application of sequential reinforcement learning to control dynamic systems. In *IEEE Intenational Conference on Neural Networks (ICNN '96)*, Washington, 1996.

[6] M. Riedmiller. Learning to control dynamic systems. In Robert Trappl, editor, *Proceedings of the 13th. European Meeting on Cybernetics and Systems Research - 1996 (EMCSR '96)*, Vienna, 1996.

[7] M. Riedmiller and B. Janusz. Self learning control of a mobile robot. In *Proceedings of the IEEE ICNN '95*, Perth, Australia, 1995.