Generalization over Environments in Reinforcement Learning

Andreas Matt and Georg Regensburger.

Institute of Mathematics .University of Innsbruck.
Technikerstr. 25/7. A-6020 Innsbruck. Austria
{andreas.matt, georg.regensburger}@uibk.ac.at

Keywords: reinforcement learning, multiple environments and objectives, generalization, obstacle avoidance.

.

We discuss the problem of reinforcement learning in one environment and applying the policy obtained to other environments. We first state a method to evaluate the utility of a policy. Then we propose a general model to apply one policy to different environments and compare them. To illustrate the theory we present examples for an obstacle avoidance behavior in various block world environments.

Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.21 (2003),pp. 47-53
ISSN: 1137-3601. © AEPIA (http://www.aepia.org/revista).

Generalization over Environments in Reinforcement Learning

Andreas Matt and Georg Regensburger
Institute of Mathematics
University of Innsbruck
Technikerstr. 25/7
A-6020 Innsbruck
Austria
{andreas.matt, georg.regensburger}@uibk.ac.at

Abstract

We discuss the problem of reinforcement learning in one environment and applying the policy obtained to other environments. We first state a method to evaluate the utility of a policy. Then we propose a general model to apply one policy to different environments and compare them. To illustrate the theory we present examples for an obstacle avoidance behavior in various block world environments.

Keywords: reinforcement learning, multiple environments and objectives, generalization, obstacle avoidance.

1 Idea

Until now reinforcement learning has been applied to abstract behavior within one environment. Several methods to find optimal or near optimal policies are known, see Bertsekas and Tsitsiklis [2], Kaelbling et al. [4], Sutton and Barto [6]. The fact that a policy learned in one environment can be successfully applied to other environments has been observed, but not investigated in detail.

In our research we focus on a general point of view of behavior that appears independently from a single environment. As an example imagine that a robot should learn to avoid obstacles. What we have in mind is a behavior suitable for any kind of environment.

We give a method to optimize single-agent behavior for several environments and reinforcement functions by learning in several environments simultaneously in [5]. Now we address the problem of learning in one and applying the policy obtained to other environments. We discuss the influence of the environment on the ability to generalize over other environments. How do good learning environments look like?

2 Preliminaries

We propose the following notation for reinforcement learning models, which allows us to treat several environments and reinforcement functions. The main difference to the standard definition of a Markov decision process, see Howard [3] or White [8], is, that we separate the reinforcement function from the environment.

An *environment* is given by

- A finite set S of *states*.

- A family $\mathbf{A} = (A(s))_{s \in S}$ of finite sets of *actions*.
  The set $A(s)$ is interpreted as the set of all allowed actions in state s.
- A family of probabilities $\mathbf{P} = P(- \mid a, s)$ on S with (a, s) such that $s \in S$ and $a \in A(s)$.
  We interpret $P(s' \mid a, s)$ as the *transition probability* that performing action a in state s leads to the successor state s'.

Let $E = (S, \mathbf{A}, \mathbf{P})$ be an environment. A *policy* for E is given by

- A family of probabilities $\pi = \pi(- \mid s)$ on $A(s)$ for $s \in S$.
  We interpret $\pi(a \mid s)$ as the probability that action a is chosen in state s.

A *Markov decision process (MDP)* is given by an environment $E = (S, \mathbf{A}, \mathbf{P})$ and

- A family $\mathbf{R} = R(s', a, s) \in \mathbb{R}$ with (s', a, s) such that s', $s \in S$ and $a \in A(s)$.
  The value $R(s', a, s)$ represents the *expected mean reward* if performing a in state s leads to the *successor state* s'. The family $\mathbf{R}$ is called a *reinforcement function.*

The goal of reinforcement learning is to find policies that maximize the sum of the expected rewards. Let $(E, \mathbf{R})$ be a MDP and $0 \leq \gamma < 1$ be a discount rate. Let π be a policy for E. The *value function* or *utility* of policy π in state s is given by

$$V^{\pi}(s) = \sum_{a} \pi(a \mid s) \cdot \left(R(a, s) + \gamma \sum_{s'} V^{\pi}(s') P(s' \mid a, s) \right),$$

where

$$R(a, s) = \sum_{s' \in S} R(s', a, s) P(s' \mid a, s)$$

is the expected reward of action a in state s. The value function describes the expected (discounted) sum of rewards following a certain policy starting in state s. The *action–value* of π of action $a \in A\,(s)$ in state s is defined by

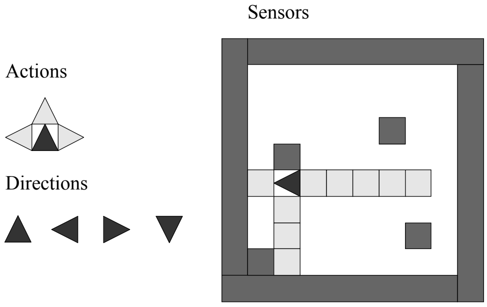

Figure 1: States and actions used in the simulator. The set of actions $A(s) = \{$forward, left, right$\}$ and the sensor values in this example are $s = (4, 2, 5, 0)$

$$Q^{\pi}\,(a, s) = R(a, s) + \gamma \sum_{s'} V^{\pi}(s') P(s' \mid a, s).$$

The action value is the expected (discounted) sum of rewards if action a is chosen in state s and afterwards policy π is followed. A policy π^* is *optimal* if $V^{\pi}(s) \leq V^{\pi^*}(s)$ for all policies π and all $s \in S$. There exists at least one deterministic optimal policy and all optimal policies share the same value function, which we denote by V^*.

3 The Blockworld Simulator Robo

To illustrate the theory developed in this paper we use the simple blockworld simulator Robo[1] to learn an obstacle avoidance behavior. The simulated robot acts in a 10x10 blockworld and has four sensors, forward, left, right, and back, with a range of 5 blocks each. The sensor values are encoded in a vector $s = (s_1, ..., s_4)$. The values vary between 0 and 5, where 5 means that there is a block right in front and 0 means that there is no block in the next 5 fields. There are three actions in each state, move forward, left and right one block (see Fig. 1).

The robot gets rewarded if it is far from obstacles or moves away from obstacles, it gets punished if it bumps into an obstacle or it moves

[1] More information on the blockworld simulator Robo is available at `http://mathematik.uibk.ac.at/users/rl`.

towards obstacles. Let $s = (s_1, \ldots, s_4)$ and $s' = (s'_1, \ldots, s'_4)$ be the sensor values for the state and the successor state. The reinforcement function is defined as follows:

$$R(s', a, s) = \begin{cases} +1 & \text{if } s'_i \le 3 \text{ for } i = 1 \ldots 4 \\ & \text{or } \sum s'_i - s_i \le -1, \\ -1 & \text{if it bumps into the wall} \\ & \text{or } \sum s'_i - s_i \ge 0.0, \\ 0 & \text{else.} \end{cases}$$

The optimal value function is calculated using value iteration, see Bellman [1] or Sutton and Barto [6]. In all examples and experiments we use the discount rate $\gamma = 0.95$ and accuracy 10^{-6}.

4 Utility of a Policy

Let $(E, \mathbf{R})$ be a MDP and γ a discount rate. Let π be a policy for E. We define the *utility* of π, denoted by $V(\pi)$, by

$$V(\pi) = \frac{1}{|S|} \sum_{s \in S} V^{\pi}(s).$$

It is the average utility of all states.

Let π^* be an optimal policy. Then $V(\pi) \le V(\pi^*)$ for all policies π. A policy is optimal if and only if its utility is maximal. The discounted utility allows us to describe a policy with one number and thus to easily compare all policies.

Example 1 *We calculate the utilities of the random policy and the optimal policy in a sample blockworld (see Fig. 2) with the reinforcement function of Sect. 3. The discounted utility of the random policy* π^{rand}*, that is all actions are chosen with the same probability,* $V(\pi^{\text{rand}}) = -4.712$ *and the discounted utility of the optimal policy* $V(\pi^*) = 18.511$.

5 The State Action Space

To apply a policy to different environments we need some new notions, which are motivated by the following example.

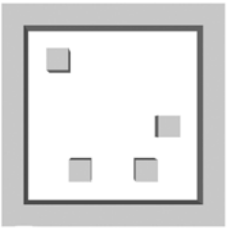

Figure 2: A sample blockworld

Consider a robot with its sensors to perceive the world. All possible sensor values together represent all possible states for the robot. In each of these states the robot can perform some actions. We call all possible states and actions the state action space.

A *state action space* $\mathbb{E} = (S, \mathbf{A})$ is given by

- A finite set S of states.
- A family $\mathbf{A} = (A(s))_{s \in S}$ of finite sets of actions.

Since the actions are given by the state action space it is clear what is meant by a policy for $\mathbb{E}$. Now we imagine the robot in a physical environment, where we can observe all possible states for this environment, a subset $S_E \subset S$ of all possible states in general, and the transition probabilities $\mathbf{P}_E$. We call an environment $E = (S_E, \mathbf{A}_E, \mathbf{P}_E)$ a *realization* of a state action space $\mathbb{E}$ if

$$S_E \subset S \text{ and } \mathbf{A}_E(s) = \mathbf{A}(s) \text{ for all } s \in S.$$

We call a MDP $(E, \mathbf{R})$ a *realization* of $\mathbb{E}$ if the environment E is a realization of $\mathbb{E}$.

Let $\mathbb{E} = (S, \mathbf{A})$ be a state action space and $E = (S_E, \mathbf{A}_E, \mathbf{P}_E)$ be a realization of $\mathbb{E}$. In order to apply the policy π for E to another realization of $\mathbb{E}$ we (randomly) extend the policy π to a policy π^e for $\mathbb{E}$ in the following way:

$$\pi^e = \begin{cases} \pi^e(a \mid s) = \pi(a \mid s) & \text{for } s \in S_E, \\ \pi^e(a \mid s) = \frac{1}{|A(s)|} & \text{for } s \in S \setminus S_E, \end{cases}$$

and $a \in A(s)$. This means that in unknown states the actions are chosen randomly. Once extended, the policy π^e can be applied to all realizations of $\mathbb{E}$. We simply write π for π^e.

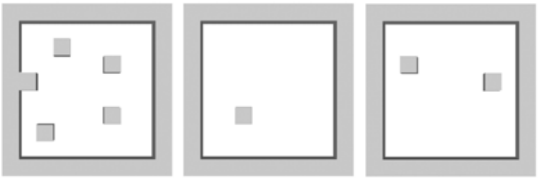

Figure 3: Sample blockworlds for the realizations B_1, B_2 and B_3

Example 2 *In the blockworld simulator the state action space is given by all possible sensor values $S = \{(s_1, \ldots, s_4)$ with $0 \leq s_i \leq 5\}$ and in each state the actions forward, left, right. We consider three different blockworlds (see Fig. 3). By observing the possible states and transition probabilities in each blockworld we obtain realizations B_1, B_2 and B_3 with the reinforcement function of Sect. 3.*
We extend an optimal policy π_1^* for B_1 and apply it to the realizations B_2 and B_3. Evaluating the utilities of π_1^* in all realizations, and the optimal and random policies, π_i^* resp. π_i^{rand}, for each realization B_i, we obtain:

Results	π_1^*	π_i^*	π_i^{rand}
B_1	18.448	18.448	−6.060
B_2	15.190	18.877	−4.873
B_3	3.081	18.951	−3.678

(1)

Note that π_1 performs relatively well in B_2 and bad in B_3.

Example 3 *Let the state action space and B_2 be as in Example 2. We consider the empty blockworld and obtain realization B_0. Comparing the utilities of their extended optimal policies π_i^* we obtain:*

Results	π_2^*	π_0^*	π_i^{rand}
B_2	18.877	9.360	−4.873
B_0	19.264	19.264	−4.473

Observe that the optimal policy for B_2 remains optimal in B_0, but not viceversa.

6 Optimal Actions

Let $\mathbb{E} = (S, \mathbf{A})$ be a state action space and $(E, \mathbf{R})$ be a realization of $\mathbb{E}$. There can be several optimal policies for this realization. While they are equally optimal in $(E, \mathbf{R})$ they can perform better or worse in other realizations of $\mathbb{E}$. Since we do not know which of the optimal policies performs best in other realizations we propose a random choice between all optimal actions.

Let $(E, \mathbf{R})$ be a MDP and γ a discount rate. Let $s \in S$. Let V^* be the optimal value function and Q^* be the optimal action values. We define the *set of optimal actions* in s by

$$A^*(s) = \{a^* \in A(s) \text{ with } Q^*(a^*, s) = V^*(s)\}\,.$$

We define the *random optimal policy* $\pi^{\mathrm{rand}\,*}$ by

$$\pi^{\mathrm{rand}\,*} = \begin{cases} \pi^*(a \mid s) = \frac{1}{|A^*(s)|} & \text{for } s \in S_E \\ & \text{and } a \in A^*(s), \\ \pi^*(- \mid s) = 0 & \text{else.} \end{cases}$$

Example 4 *We repeat the Example 2 with the random optimal policy $\pi_1^{\mathrm{rand}\,*}$ for B_1 and obtain:*

Results	$\pi_1^{\mathrm{rand}\,*}$	π_i^*	π_i^{rand}
B_1	18.448	18.448	−6.060
B_2	15.166	18.877	−4.873
B_3	3.202	18.951	−3.678

(2)

Comparing (1) and (2) we find that $\pi_1^{\mathrm{rand}\,*}$ performs better in B_3 and slightly worse in B_2.

7 Utility of a Policy for a Family of Realizations

Let $\mathbb{E} = (S, \mathbf{A})$ be a state action space. Let $\mathbf{E} = (E_i, \mathbf{R}_i)_{i=1\ldots n}$ be a finite family of realizations of $\mathbb{E}$ and $(\gamma_i)_{i=1\ldots n}$ discount rates. Note that the reinforcement functions and discount rates can be different for each realization. Let π be a policy for $\mathbb{E}$. We calculate for each $(E_i, \mathbf{R}_i)$ and γ_i the utility of π, which we denote by $V_i(\pi)$.

We define the utility of π for a finite family of realizations $\mathbf{E}$ by

$$V_{\mathbf{E}}(\pi) = \frac{1}{n} \sum_{i=1\ldots n} V_i(\pi),$$

the average utility of π in all realizations. This utility is used to measure the performance of policies for a family of realizations. We say that a policy π performs better in $\mathbf{E}$ than policy π' if $V_{\mathbf{E}}(\pi) > V_{\mathbf{E}}(\pi')$.

8 Generalizing over Environments

Recall our main idea to learn a policy in one environment and apply it to other environments. We say that an environment generalizes over other environments if a policy learned in this environment performs well in the others.

In the framework defined above we can formulate this more precisely using random optimal policies: Let $\mathbb{E} = (S, \mathbf{A})$ be a state action space and $\mathbf{E} = (E_i, \mathbf{R}_i)_{i=1...n}$ be a finite family of realizations of $\mathbb{E}$. We say that $(E_i, \mathbf{R}_i)$ *generalizes* better over $\mathbf{E}$ than $(E_j, \mathbf{R}_j)$ if $V_{\mathbf{E}}(\pi_i^{\text{rand}\,*}) > V_{\mathbf{E}}(\pi_j^{\text{rand}\,*})$.

This general approach includes generalization over reinforcement functions. In our context generalization is established without using methods such as neural networks.

9 Experiments

We observed in the above examples that the environment influences the utility of its optimal policies for other realizations. Our question now is, how to choose an environment that generalizes well over other environments. We conduct three experiments.

In all experiments we use the blockworld simulator as described in Sect. 3. We choose a family $\mathbf{E}$ of realizations of the state action space in Example 2. We calculate the random optimal policies (Sect. 6) for all realizations of $\mathbf{E}$ and compare their utilities for $\mathbf{E}$ (Sect. 7).

9.1 How Many Blocks?

In the first experiment we discuss how many blocks we should distribute in an empty blockworld. We generate environments with one to ten blocks distributed randomly. We generate ten environments for each number of blocks and obtain a family $\mathbf{E}$ of 100 realizations. The utility of the random optimal policies of the n-block environments for all realizations of $\mathbf{E}$ are then averaged for $n = 1 \ldots 10$. The results of two experiments are shown in Fig. 4.

We observe that the average utility first increases with the number of blocks and then decreases if there are more than three blocks. This confirms the intuitive idea that a “good” environment to generalize should be neither too simple nor too complex, more on the simple side though.

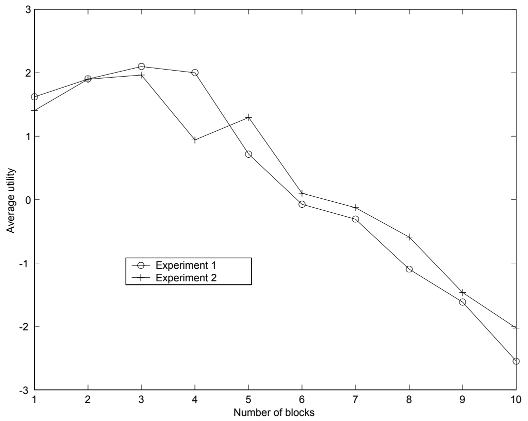

Figure 4: Averaged utility of the random optimal policies of the n-block environments for all realizations

9.2 One-Block Environments

We consider all 64 possible one-block environments. The question is now, on which position we put the block in order to have a good environment to generalize over all one-block environments. Before you read on, you can think by yourself where to put the block.

The checkerboard in Fig. 5 represents the utilities. The brighter the color of the block the higher is the utility of the random optimal policy for all one-block environments of the environment with a block on this position. The utilities vary between 13.584 and 14.873. Note the symmetry of the environments and the eight best ones.

The checkerboard in Fig. 6 shows the number of different states in each one-block environment. Again the brighter the color of the block the higher is the number. The number of states lies between 65 and 104. The best one-block environment has 100 different states.

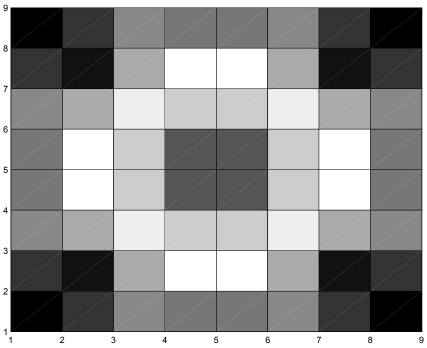

Figure 5: The checkerboard shows the utility for all 64 one-block environments

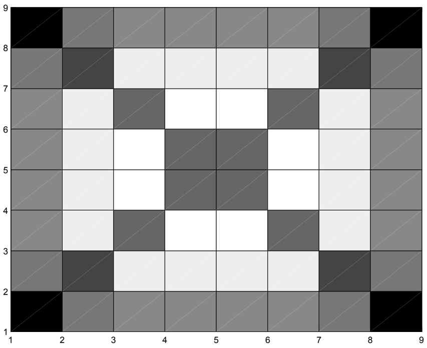

Figure 6: The checkerboard represents the number of different states for all 64 one-block environments

9.3 Two-Block Environments

We extend the above experiment to all two-block environments. There are 2016 different environments to compare. Using symmetries we end up with 632 which still results in a fair amount of calculations. The best and worst environment are shown in Fig. 7.

The utility of the random optimal policy of the best environment is 11.195, the utility of the worst environment 3.404. The number of states ranges from 51 to 143. The best environment has 126 different states, the worst 60. We observe that "good" environments to generalize have a high number of different states.

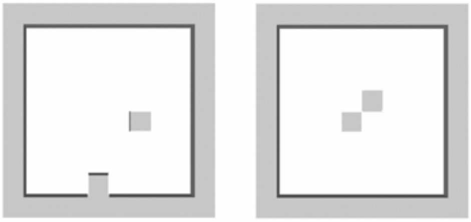

Figure 7: The *left* blockworld is the best, the *right* one the worst environment generalizing over all two-block environments

10 Discussion

We give a method to apply policies to different environments and compare them. The results in Sect. 9 emphasize the influence of the realization, that is the environment with its states and transition probabilities and the reinforcement function, on the ability to generalize over other realizations. Example 3 even shows that an environment may include all the necessary information to learn an optimal policy for other environments.

To obtain one policy suitable for many realizations it is important to choose an appropriate environment to learn in. Our future work consists in finding a priori criteria, such as the number of states, complexity of transitions, rewards, etc., to predict the ability of an environment to generalize.

In Sect. 6 we extend the policy randomly for unknown states. Applying neural networks to choose similar actions for similar unknown states as in Touzet [7] could improve the method.

Acknowledgements

This research was partially supported by the Austrian Science Fund (FWF), Project Y-123 INF.

References

[1] Bellman, R. E.: Dynamic Programming. Princeton University Press. Princeton (1957)

[2] Bertsekas, D. P., Tsitsiklis, J. N.: Neuro-dynamic programming. Athena Scientific. Belmont, MA (1996)

[3] Howard, R. A.: Dynamic Programming and Markov Processes. MIT Press. Cambridge, MA (1960)

[4] Kaelbling, L. P., Littman, M. L., Moore, A. W.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research **4** (1996) 237–285

[5] Matt, A., Regensburger, G.: Policy Improvement for Several Environments. Proceedings of the 5th European Workshop on Reinforcement Learning (EWRL-5) (2001) 30–32

[6] Sutton, R. S., Barto, A. G.: Reinforcement Learning: An Introduction. MIT Press. Cambridge, MA (1998)

[7] Touzet, C.: Neural Reinforcement Learning for Behavior Synthesis. Robotics and Autonomous Systems **22** (1997) 251–281

[8] White, D. J.: Markov Decision Processes. John Wiley & Sons Ltd. Chichester (1993)