Technical Report

IRI-TR-06-01

Generalization in Reinforcement Learning with a Task-Related World Description using Rules

Alejandro Agostini
Enric Celaya

June 2006

Institut de Robòtica i Informàtica Industrial

Generalization in Reinforcement Learning with a Task-Related World Description using Rules

Alejandro Agostini and Enric Celaya

Abstract. A Reinforcement Learning problem is formulated as trying to find the action policy that maximizes the accumulated reward received by the agent through time. One of the most popular algorithms used in RL is Q-Learning which uses an action-value function q(s,a) to evaluate the expectation of the maximum future cumulative reward that will be obtained from executing action a in situation s. Q-Learning, as well as conventional RL techniques, is defined for discrete environments with a finite set of states and actions. The action-value function is explicitly represented by storing values for each state-action (s,a) pair. In order to reach a good approximation of the value function all the (s,a) pairs must be experienced many times but in practical applications the amount of experience for learning to take place is unfeasible. Therefore, the value function must be generalized to infer in situations never experienced so far. The generalization problem has been widely treated in the field of machine learning. Supervised learning directly treats this issue and many generalization techniques have been developed in this field. Any of the representations used in supervised learning could, in principle, be applied to RL. But there are some important issues to take into account that make good generalization in RL very hard to achieve. One of the most remarkable is that the value function is learned while represented. In this work we propose a RL approach that uses a new function representation of the Q function that allows good generalization by capturing function regularities into decision rules. The representation is a kind of Decision List where each rule configures a subspace of the state-action space and provides an approximation of the Q function in its covered region. Rule selection for action evaluation is given by the rule with both, good accuracy in the estimation and high confidence in the related statistics.

1. Introduction

Task-achieving autonomous behaviour in an AI agent able to perceive its environment through sensors and execute actions requires it to be able to select an appropriate action in each situation it faces. A function that maps the situations perceived by the agent to a probability of selecting an action is called a *policy*.

Specifying a policy for an agent to achieve a complex task may become very hard or almost impossible if it has to be done by the user. In these cases, a better approach may be to let the agent find a policy by itself by only receiving from the environment a reward signal that roughly indicates how the action results for the task, but does not indicate how good the action is with respect to the other possible actions. In order to decide which action to execute the problem is two fold; first, the agent must learn how good is each action for the task by means of the reward, and second, decide which action to execute by comparing the goodness of each action. This learning paradigm is called Reinforcement Learning [Sutton,98].

A RL problem is formulated as trying to find the policy that maximizes the accumulated reward received by the agent through time. One of the most popular algorithms used in RL is Q-Learning [Watkins,92], which uses an action-value function $q(s,a)$ to evaluate the expectation of the maximum future cumulative reward that will be obtained executing action a in situation s. The goal of Q-Learning is to incrementally approximate this action-value function through experience during the learning process. Given a good enough approximation of the action-value function, an optimal policy is obtained by selecting the action a that maximizes $q(s, a)$ in the current situation s.

Q-Learning, and conventional RL techniques, is formulated for discrete environments with a finite set of states and a finite set of actions [Sutton,98]. The action-value function is explicitly represented by storing values for each state-action (s,a) pair. In order to reach a good approximation of the value function all the (s,a) pairs must be experienced many times. But in practical application the number of (s,a) pairs is usually very large or even infinite (like in continuous environments), and learning by experience with conventional Q-learning is unfeasible. To make it applicable, the value function must be generalized to infer the values of those (s,a) never experienced using the values of the experienced (s,a) pairs.

The generalization problem has been widely treated in the field of machine learning where inferences have to be done under a large domain [Mitchel,97]. Supervised learning directly treats this issue and many generalization techniques have been developed in this field. Learning a concept, learning to classify new examples, finding a suitable hypothesis, all refer to learn a description of a function that enables to infer values for given inputs. This description should be made with a compact representation that permits good approximation with low computational cost. Different learning techniques use different representations. Some of them are neural networks [Coulom,02], [Buck,02], induction trees [Quinlan,86], [McCallum,95], feature based [Aha,91], [Bloedorn,98], [Miyamoto, 99].

Any of the representations used in supervised learning could, in principle, be applied to RL. But there are some important issues to take into account. One of the most important is that the value function is learned while represented. This leads to many convergence problems and makes good generalization in RL very hard to achieve [Kaelbling, 96].

In this work we propose a RL approach that uses a new function representation of the action-value function that allows good generalization by capturing the function regularities in a flexible and efficient way.

2. New Representation – General Outlines

The key idea of the proposed approach is to describe the function using the dependency with respect to its variables. After describing the proposed representation, we give some definitions that we later use in the explanation.

Definition. δ-irrelevant variable for a function.

Given a function f: X$\rightarrow\Re$, with X defined by the Cartesian product X= $V_1\times V_2\times\ldots\times V_n$ where the set $V_i\subseteq\Re$, i=1,..,n. We use the notation v_i to indicate any particular element of V_i, and is said to be a *variable* of the function f. Therefore, an element of the domain X is indicated as x=($v_1, v_2, ..., v_n$). We say that a variable v_i of X is *δ-irrelevant* for the description of f (from now on *δ-irrelevant* for f), if the function f has a variation less than δ (δ-invariance) with respect to that variable, i.e.,

If $\forall$ $v_1,\ldots,v_{i-1},v_{i+1},\ldots,v_n$ is $|f(v_1,\ldots,v_{i-1},v_{i1},v_{i+1},\ldots,v_n) - f(v_1,\ldots,v_{i-1},v_{i2},v_{i+1},\ldots,v_n)|\leq\delta$ for any two values v_{i1} and v_{i2} of the variable v_i.

If v_i is δ-irrelevant for f then it is possible to find a function g:X'$\rightarrow\Re$, with X'= $V_1\times\ldots\times V_{i-1}\times V_{i+1}\ldots\times V_n$, so that $\forall$ $x\in$X is $|f(x)-g(x')|\leq\delta$, where x'=($v_1,\ldots,v_{i-1},v_{i+1},...,v_n$).

Definition. δ-irrelevant variable in a region.

Given a function f: X$\rightarrow\Re$, we say that a variable v_i of X is *δ-irrelevant in a region,* if the function f is δ-invariant with respect to that variable in that region.

Definition. δ-approximation.

Given a function f: X$\rightarrow\Re$, with X $\subset\Re^n$, and a function g: X$\rightarrow\Re$ with X $\subset\Re^n$, which approximates f in X. We say that g δ-approximates f when $|f(x)-g(x)|\leq\delta$, $\forall$ $x\in$X.

Definition. Rule.

Given a set X defined by the Cartesian product X= $V_1 \times V_2 \times \ldots \times V_n$ where the set $V_i \subseteq \Re$, i=*1*,..,*n*. We define a *rule* as a function *r*: $X_r \rightarrow \Re$ where $X_r \subseteq X$. A rule is completely described by specifying X_r and the function itself. We say that X_r is the represented region of *r*.

Definition. Rule Representation.

Given a set X defined by the Cartesian product X= $V_1 \times V_2 \times \ldots \times V_n$ where the set $V_i \subseteq \Re$, i=*1*,..,*n*. We say that a set of rules $\mathbf{R}=\{r_1, r_2, \ldots, r_{|\mathbf{R}|}\}$ is a *rule-based function representation*, when the regions represented by the rules determine a covering $C=\{X_{r1}, X_{r2}, \ldots, X_{r|\mathbf{R}|}\}$ of X, and when $\forall i,j| X_{ri} \cap X_{rj} \neq \emptyset,\ x \in X_{ri} \cap X_{rj} \Rightarrow r_i(x)=r_j(x)$.

3. Generalization using Rule Representation

This section introduces an approach that uses the rule representation of a function to perform generalization taking benefits of the irrelevancies of the variables.

We begin by considering functions that are stationary and deterministic with finite discrete domains, and we suppose that we have complete knowledge of the function. We consider a function *f*: $X \rightarrow \Re$, with X= $V_1 \times V_2 \times \ldots \times V_n$ where each set V_i, has a finite number of elements v_{ij}, $j=1,..,|V_i|$, that we call features. We use the notation v_i to indicate any particular element of V_i, and is said to be a *variable* of the function *f*.

3.1. Rule description

In this approach we define a *rule r*: $X_r \rightarrow \Re$ as a constant value function. The description of the region X_r represented by a rule *r* is done using a subset of features $X_r=(v_{ij},\ldots,v_{kl})$, where there are no two features of the same variable.

In Figure 1 there is an illustrative example of three regions described using a subset of features for a simple *f*. Note that, a variable is omitted in the description of a region when the region involves all the values for that variable.

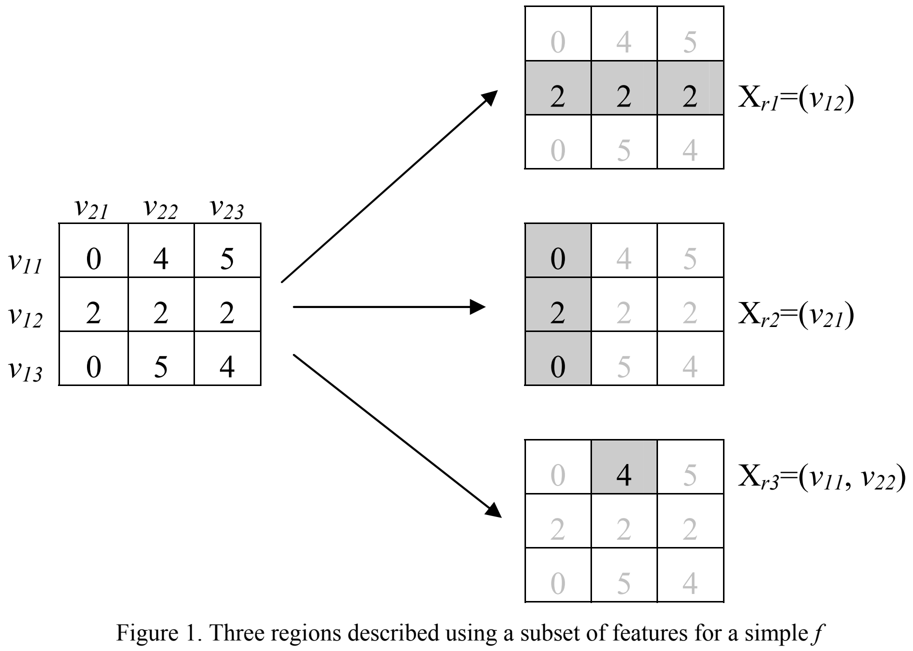


Figure 1. Three regions described using a subset of features for a simple *f*

3.2. Rule Representation

A good approximation of a function f: $X \rightarrow \Re$ is a necessary condition to perform generalization but it is not sufficient. For instance, we can make an accurate approximation by representing each point of the function with a rule, where the region represented is specified by the coordinates of the point and the constant value approximation is the actual value of the function for that point. This is the same case as the tabular representation of a function, where no generalization at all takes place. In order to perform generalization we also have to describe f using a reduced description.

Using rules a generalization of f could be done if the rule representation δ-approximates f in X and the information needed to specify the regions represented by the rules is lesser than considering a rule representation where each element of X is represented by a different rule. In this first approach a rule r is the mean value of f in X_r.

Table 1A shows a rule representation of the example function that permits generalization. For instance, the function could be represented by r_4 in three points by the region described by X_{r4}.

There is another way of specifying the region represented by a rule that permits an important reduction in the description of f. It consists in specifying the points represented by a rule using other rules region description.

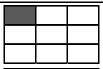
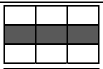
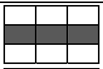
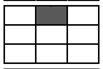
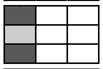
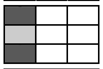
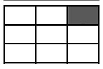
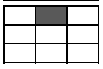
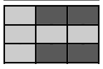
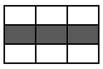
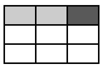
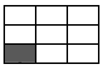
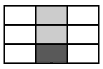
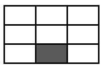
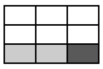
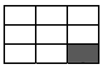
Suppose that we have a region X_L that involves many points of the domain, where f presents a dispersion greater than δ for some points $X_{high} \subset X_L$ with respect to the mean value of f in X_L. Then, a rule can not make a δ-approximation of the function in X_L and many rules should be considered. But, we can reduce the number of rules needed and still use the description of X_L using a layered structure that permits an efficient specification of the region represented by a rule. Each layer L contains a description of a region of the domain using a subset of features. Each layer L has a rule r associated. The region X_r represented by a rule r is then determined by the description of the region in the layer X_L and the descriptions of the regions of the upper layers in the structure. This is done by making that for a given point x the function $f(x)$ is approximated using the value of the rule r associated to the upper layer that contains x. Therefore, given a set of rules in a layered structure where each layer position is specified using natural numbers beginning with 1 from the upper layer, the points X_{rj} represented by a rule r_j associated to layer j are,

$$X_{rj} = X_{Lj} - X_{L(j+1)} - \ldots - X_{L1}$$

In table 1B there is an example of rule representation with a layered regions description for the exampled function. Note the reduction in the number of rules and in the rules region description compared with a non layered description (Table 1A).

Finally, we can generalize even more with the rule description by specifying some δ>0. Table 1C illustrates how a good generalization is reached by setting δ=0.5 in the exampled function.

Table 1. Examples of a rule representation for the discrete function in Figure 1.

A- Not layered description $\delta=0$			B- Layer with $\delta=0$			C- Layer with $\delta=0.5$		
r	X_r	Points	r	X_L	Points	r	X_L	Points
$r_1(x)=0$	(v_{11}, v_{21})		$r_1(x)=2$	(v_{12})		$rB_1(x)=2$	(v_{12})	
$r_2(x)=4$	(v_{11}, v_{22})		$r_2(x)=0$	(v_{21})		$r_2(x)=0$	(v_{21})	
$r_3(x)=5$	(v_{11}, v_{23})		$r_3(x)=4$	(v_{11}, v_{21})		$r_3(x)=4.5$	$(\emptyset)$	
$r_4(x)=2$	(v_{12})		$r_4(x)=5$	(v_{11})				
$r_5(x)=0$	(v_{13}, v_{21})		$r_5(x)=5$	(v_{22})				
$r_6(x)=5$	(v_{13}, v_{22})		$r_6(x)=4$	(v_{13})				
$r_7(x)=4$	(v_{13}, v_{23})							

4. Generating a Rule Representation by Sampling

We have already described a rule representation that permits generalization of a finite discrete function assuming we have complete knowledge of it. However, RL is a paradigm that learns by experience, learning the optimal value function using consecutive observations of its approximation without considering any previous knowledge. Therefore, we will now make another little step toward the applicability of RL. We present in this section an approach to build a rule representation of a function f using consecutive sampling trying to reach a δ-approximation with a reduced description of f, and without any previous knowledge.

The function considered is again deterministic and stationary with discrete finite domain f: $X \rightarrow \Re$, with $X = V_1 \times V_2 \times \ldots \times V_n$ where each V_i has elements v_{ij}, $j=1,..,|V_i|$.

At the beginning we are optimistic and we suppose that the function admits a rule representation with δ-approximation by using a few layers with very simple region descriptions. Therefore, we begin by considering $k = \sum_{i=1}^{n} |V_i|$ layers where each layer L describes a region represented by one different feature. As there is no previous knowledge of f the layers are initially arranged in an arbitrary order.

For every point $x \in X$ there is a set of layers A_L that involve x in their region description. We denote A_L as the *active* set of layers in x.

Each L has a rule associated r_j with an estimation of the mean value m_{rj} of f in $x \in X_{rj}$. In order to permit a better evaluation of the approximation made, each rule has also associated an estimation of the variance e^2_{rL} of f in X_{rL}. The number of samples n_{rL} used to feed the estimations is also stored. With these statistics it is possible to estimate the probability distribution of the value of f in X_{rL}. This probability is estimated using a normal probability distribution with mean m_r and variance e^2_r.

4.1. Learning Approach

The information available to find a reduced rule representation with δ-approximation of f is only given by the sampled values. The learning approach should try to refine the approximation for the current rule representation using the value $f(x_s)$ of the sampled point x_s to update the statistics associated to the rules, and evaluate if the rule structure would permit a δ-approximation. If it is not possible, new layers should be generated.

4.1.1. Statistics Update

Every new sample $f(x_s)$, should be used to feed the upper rule in A_L that represents f in x_s. Nevertheless, the initial set of layers is arbitrarily arranged and the approach attempts to find the arrangement that permits the best approximation. It is expected that the rule that better approximates the function in x_s would finally represent f in x_s. Then, the updating is performed taking into account the approximation made by a rule rather than its position in the layered arrangement.

The approximation made by a rule r in x_s is calculated using the estimation of the probability distribution of f in X_r. If the distribution is concentrated around $f(x_s)$ with a little dispersion, then we consider that r represents f in x_s with a good approximation. As the value δ determines the desired accuracy in the approximation, the approximation made by the rule r is calculated using the probability of $r(x)$ in the interval of length 2δ centred in $f(x_s)$. This probability is denoted as $P_r[f(x_s)]$. The updating is then distributed among the rules in A_L in a proportion to $P_r[f(x_s)]$.

We adopt the criterion that, if a rule has P_r~1 then it should take almost all the updating because it is expected to finally represent f in x_s. One way of implementing this is by updating each rule r in a proportion given by P_r'=1/(1 - P_r). Then, the proportion of updating considered for each rule β_r is,

$$\beta_r = \frac{P_r'[x_s]}{\sum_{\forall ri \in A_L} P_{ri}'[x_s]}$$

The indeterminacy of P_r' when P_r=1 is solved by making β_r=1 for the corresponding rule.

4.1.2. Updating Formula

The update formula we use for the estimations of the mean and variance is the cumulative weighted formula,

$$m_r = (1-\alpha_r\beta_r)m_r + \alpha_r\beta_r f(x_s)$$

$$e_r^2 = (1-\alpha_r\beta_r)e_r^2 + \alpha_r\beta_r (f(x_s) - m_r)^2$$

where α_r is the updating coefficient. This formula requires an updating coefficient that decays with the number of samples in order to converge to the actual estimated values. We use a simple decay equation that fulfils this requirement: an inverse proportion with the number of samples n_r.

$$\alpha_r = \frac{1}{1+\min(n_c, n_r)}$$

where n_c is a critic number of samples. The value n_c prevents the updating coefficient from being too small and permits a better adaptation when new layers generation affect point representation changing the distribution of f in the X_r's.

4.1.3. Layer Management

There is no clear criterion about how layers rearrangement and generation would favour the learning algorithm to finally reach the desired convergence. For instance, layer management changes the rule representation leading, perhaps, to a better approximation of the function for the sampled point, but the resulting structure could make difficult the algorithm convergence. There are many other unsolved questions about layer management. Layer management must be carefully studied before defining a method that permits efficient learning. In this section it is described a method that constitutes a first attempt to perform layer management in the learning process.

As mentioned before, the upper rule r_{up} in A_L gives the approximation of the function for a given sampled point x_s. If r_{up} has a probability $P_{rup}[f(x_s)]$ over a predefined value P_c then we consider that the rule representation makes a good approximation of f in x_s, and we say that the approximation is a δ-approximation with probability P_c ($P_c\delta$-approximation). On the other hand, when $P_{rup}<P_c$ we consider that the approximation made is bad when there is high confidence in the statistics associated to r_{up}, and that the approximation is weak when there is low confidence in those estimations. The confidence in the estimations is given by the number of samples n_r used to update them. Therefore, we consider a bad approximation when $P_{rup}<P_c$ and $n_r>n_c$, and a weak approximation when $P_{rup}<P_c$ and $n_r<n_c$.

If the approximation made by the rule representation is bad or weak, a first attempt to improve the approximation is made by trying to arrange the layers by the $P_r[f(x_s)]$ of the associated rules, trying to prevent a damage of other well represented regions. We suppose that a potential damage could occur when the reordering of a layer modifies the description of a region X_r for a rule r that has low dispersion of the represented values of f. It is considered that a rule r has low dispersion when the probability calculated in an interval of 2δ centred in m_r is greater than P_c.

4.1.3.1. Layer Generation

If the approximation made by r_{up} is bad and there is no possible reordering of the active layers a new layer must be generated and placed above the upper active layer. This new layer should permit to reach a $P_c\delta$-approximation of the function with the shortest possible description.

In the proposed approach a new layer is generated by combining the features of the region descriptions of two active layers randomly selected from A_L. This selection is made preferring those whose combination leads to a covered region with the smallest intersection with those regions represented by rules with low dispersion placed in lower layers. As in the rearrangement case, this is done to prevent potential damage of the other well represented regions.

4.1.3.2. Layer Elimination

During the learning process some rules could be generated that finally may result useless for the representation and could be eliminated. These rules are those whose regions are completely represented by the above rules.

5. References

[Aha,91] Aha D. Incremental constructive induction: An instance-based approach. *Proceedings of the Eighth International Workshop on Machine Learning*. Evanston, IL. In Lawrence Birnbaum and Gregg Collins, editors. pp. 117-121. 1991.

[Bloedorn,98] Bloedorn E, Michalski R. Data-Driven Constructive Induction. *IEEE Intelligent Systems* 13(2): 30-37. 1998.

[Buck,02] Buck S, Beetz M, and Schmitt T. Approximating the Value Function for Continuous Space Reinforcement Learning in Robot Control. *Proceedings of the IEEE/RSJ IROS 2002*. Lausanne, Switzerland. 2002.

[Coulom,02] Coulom R. Feedforward Neural Networks in Reinforcement Learning Applied to High-Dimensional Motor Control. *ALT 2002:* 403-414.
[Kaelbling,96] Kaelbling L, Littman M. and Moore A. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237–285, 1996.

[McCallum,95] McCallum A. Reinforcement Learning with Selective Perception and Hidden State. *Phd. thesis*, Department of Computer Science, University of Rochester, Rochester, NY, 1995.

[Mitchel,97] Mitchel T. Machine Learning. McGraw-Hill, 1997.

[Miyamoto,99] Miyamoto Y. and Uehara K. Improving the Effectiveness of Q-Learning by Using Feature Construction. *IPSJ Transactions on Mathematical Modeling and Its Applications*, Vol. 40, No. SIG9 (TOM2), pp. 62-71. 1999.

[Quinlan,86] Quinlan J.R. Induction of Decision Trees. *Machine Learning*. 1(1), 81-106. 1986.
[Sutton,98] Sutton R, Barto A. Reinforcement Learning: An Introduction, "A Bradford Book", *MIT Press*, 1998.
[Watkins,92] Watkins C, Dayan P. Q-Learning. *Machine Learning*, 8:279-292, 1992.