SACJ 57 December 2015
Research Article

Upper bounds on the performance of discretisation in reinforcement learning

Michael Mitchley

School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg

ABSTRACT
Reinforcement learning is a machine learning framework whereby an agent learns to perform a task by maximising its total reward received for selecting actions in each state. The policy mapping states to actions that the agent learns is either represented explicitly, or implicitly through a value function. It is common in reinforcement learning to discretise a continuous state space using tile coding or binary features. We prove an upper bound on the performance of discretisation for direct policy representation or value function approximation.

Keywords: Reinforcement learning, tile coding, performance bounds, average case analysis

Categories: **Mathematics of computing ~ Discretization**, *Mathematics of computing ~ Probability and statistics*, *Computing methodologies ~ Markov decision processes*, *Computing methodologies ~ Sequential decision making*

Email:
Michael Mitchley michael.mitchley@wits.ac.za (CORRESPONDING)

Article history:
Received: 4 Nov 2014
Accepted: 8 Dec 2014
Available online: 10 Dec 2014

1 INTRODUCTION

Reinforcement learning is a branch of machine learning wherein an agent is not given an explicit task to perform. Instead, an agent must learn optimal behaviour from knowledge of its current state, and rewards or punishments given for performing actions, and achieving new states. This optimal behaviour is encoded in a *policy*, a mapping from states to actions that seeks to maximize the reward obtained over time. Commonly, this is both found and represented using a *value function*, storing the expected reward of each state or state-action pair. When the state space is large, or continuous, we can no longer define values or policies per state. Instead, we must approximate the policy or value function. It is common to simply discretise the state space in this case, using a tile coding basis scheme, binary features or similar. Within a tile (or binary feature—we will use the term tile to indicate a set of states treated as the same through a piecewise constant function, regardless of how that tile is derived) an agent cannot distinguish between neighbouring states. If the optimal action to select varies across a tile, an incorrect action will be selected by the agent for a portion of the tile. This paper aims to quantify that error rate, and provide upper bounds on the performance of tile coding and discretisation schemes in reinforcement learning.

Mitchley, M. (2015). Upper bounds on the performance of discretisation in reinforcement learning. *South African Computer Journal 57*, 24–31. http://dx.doi.org/10.18489/sacj.v0i57.284

SACJ is a publication of the South African Institute of Computer Scientists and Information Technologists. ISSN 1015-7999 (print) ISSN 2313-7835 (online).

2 BACKGROUND

Reinforcement learning is a machine learning framework in which agents are given rewards or punishments based on their behaviours. The agents seek to maximise their rewards subject to a possible discount of future rewards against immediate gains. The advantage of this learning framework becomes apparent when one considers tasks wherein good behaviour (or equivalently, bad behaviour) can be recognised, but it is difficult to codify that behaviour.

If reinforcement learning were applied to the game of checkers, for example, good behaviour may be taking an opponent's pieces, or simply winning games. Bad behaviour could be losing pieces, or losing games. It is, however, difficult to codify what a good game of checkers looks like as a series of rules.

If an agent has a way of maximising its own rewards through repeatedly playing games of checkers, it will eventually learn to play well. Checkers is a Markov game, since knowing the current *state* is sufficient for choosing the next move; the state would be the position of each piece on the board, and the available actions at each state would be the legal moves that could be made. Checkers has a discrete state space, although it is very large.

Reinforcement learning problems are typically modelled as Markov decision processes described by the tuple $(S, A, P, \mathscr{R}, \gamma)$, where $S \subseteq \mathbb{R}^n$ is a state space, A is a finite set of actions, $P(s'|s,a)$ is the probability density of transitioning to state s' after having performed action a in state s, $\mathscr{R}(s',a)$ is the expected reward received for executing such a transition, and $\gamma \in (0,1]$ is the discount factor. The agent is required to learn a policy π mapping states to actions, that maximizes the *return* (discounted future sum of rewards) at time t: $R_t = \sum_{i=0}^{\infty} \gamma^{i+t} \mathscr{R}_{i+t}$.

One method of finding this policy is to define a *value function* Q_π which is the value of each state-action pair defined recursively as

$$Q_\pi(s,a) = \mathbb{E}[R(s,a,s') + \gamma Q_\pi(s', \pi(s'))],$$

which can be understood as the expected reward from executing action a in state s plus the decayed value of the successor state s'. Intuitively, the value function measures the 'goodness' of performing each action in each state. In checkers, a state where several pieces could be taken might have a high value associated with it. The policy maximising the return can then be found as

$$\pi(s) = \arg\max_a Q(s,a).$$

Domains like the game of checkers have discrete state spaces, where a value could, in principle, be defined uniquely for each state. When the state space is *continuous*, state variables take on real values, with the number of state variables being called the *dimensionality* of the domain. An example of this is a robot with multiple actuated joints—each joint position (and perhaps joint velocity) would be a separate continuous state variable, making a state s a real-valued vector in $S \subset \mathbb{R}^d$ (most typically scaled onto the unit hypercube $[0, 1]^d$). If the state space S is not scaled, and can take on any real value, we can easily consider a nonlinear folding of the real numbers onto the unit interval.

When the state space is continuous, we cannot directly define a value for each state, and so we must approximate Q. The most common form of approximation is linear value function approximation,

http://dx.doi.org/10.18489/sacj.v0i57.284

where each $Q(s,a) = V_a(s)$ is represented as a weighted sum of a collection of m basis functions Φ:

$$\overline{V}(s) = \mathbf{w} \cdot \Phi(s) = \sum_{i=1}^{m} w_i \phi_i(s).$$

This approximation is linear in the components of the parameter (or weight) vector, $\mathbf{w}$, which results in simple update rules and a quadratic error surface, but can represent complex value functions because the basis functions themselves can be arbitrarily complex.

A common basis function scheme is tile coding, a coarse coding technique (Sutton & Barto, 1998) where a set of piecewise constant functions is used to approximate a value function. In their simplest form, these functions are a discretisation of the state space, acting as indicator functions for an exhaustive, disjoint partitioning of the states. The tiles themselves are binary, activating if a state is within that tile. By multiplying each tile ϕ_i by a weight w_i, and summing, a linear value function is obtained. The tiles can be of any shape, but typically one divides the state space in each dimension into right angled tiles of equal width. If one wishes to use a single tiling with n partitions per dimension, this results in n^d tiles of equal size across the state space.

Multiple tilings may be used, with different discretisation levels or tile shapes for each tiling, producing an overlapping set of basis functions. Typically, the different tilings are at the same discretisation level, but are offset from each other (Whiteson, Taylor & Stone, 2007). The value functions that can be represented using a multiple tiling can always be represented using a single tiling, although the speed of learning for the single tiling may be different. The number of tilings needed to achieve good performance grows exponentially with the number of dimensions (Wu & Meleis, 2009). Other basis schemes include the polynomial basis (Lagoudakis & Parr, 2003), radial basis functions, and the Fourier basis (Konidaris, Osentoski & Thomas, 2011).

The major drawback of fixed bases is that the number of basis functions grows exponentially with the dimension of the state space; an order n polynomial basis has $(n+1)^d$ basis functions (similarly for an order n Fourier basis, or $k = n+1$ radial basis functions per dimension). This has led to *feature selection* approaches (Kolter & Ng, 2009; Johns, Painter-Wakefield & Parr, 2010; Painter-Wakefield & Parr, 2012) that use a fixed basis as a feature dictionary but select only a small subset of basis functions to compactly represent the value function, and thereby accelerate learning.

Adaptive schemes also mitigate the curse of dimensionality. Early work by Moore (1994) dealt with finding a path to a defined goal region in a high dimensional continuous space, achieved via an adaptive resolution approach to discretisation, wherein an agent responds to a coarsely discretised tile with high error by splitting that tile up. Whiteson et al. (2007) also use an adaptive tile coding technique, splitting tiles adaptively and thus reducing the level of generalization over time. Geramifard, Doshi-Velez, Redding, Roy and How (2011) introduced an adaptive feature selection procedure for binary features. Here, the potential worth of each feature combination is computed, and those feature combinations with a sufficiently high potential are added to the feature set. Nonlinear approaches to discretisation adaptation also exist (Uther & Veloso, 1998; Lin & Wright, 2010).

Through this adaptive work, there is an intrinsic assumption that too-coarse discretisation results in poor performance due to a lack of representational ability, particularly in areas of the state space

http://dx.doi.org/10.18489/sacj.v0i57.284

where the agent must choose from a set of actions. We thus find an upper bound on the performance of discretisation in continuous state spaces, to quantify when a discretisation is too coarse.

3 UPPER BOUND ON PERFORMANCE

We define the decision set $D(a)$ for action a as the set of all states s such that the optimal policy $\pi^*(s) = a$, i.e. $D(a) = \{s|\pi^*(s) = a\}$. We define the decision set of a specific tile with domain ω as $D_\omega(a) = \{s|s \in \omega, \pi^*(s) = a\}$, where the domain of a tile is the set of all states for which the tile may be nonzero. We note that domain ω is not assumed to be any specific shape, or assumed to be contiguous, and so the results presented here are generalisable for any form of discretisation. For deterministic policies and actions, $D_\omega(a) \cap D_\omega(b) = D(a) \cap D(b) = \emptyset$.

3.1 Assumptions

We will assume for now an agent capable of two actions, a and b (with extension to m actions). We also assume a deterministic environment.

3.2 Results

Lemma 1. *A tile can only encode one action across its domain ω regardless of whether the tile is used to encode the policy directly, or a state-action value function.*

Proof. Suppose an agent encounters state $s \in \omega$. If the tile encodes a policy directly in some way, then the policy must be constant across all states in ω. If a state-action value function $Q(s,a)$ is used with one tile per action across domain ω, then the policy must select the action that maximises the value, i.e. $\arg\max_a Q(s,a)$. However, $\max_a Q(s,a)$ is constant for any $s \in \omega$, since each tile is constant for any $s \in \omega$. Thus, the policy across all states in ω is constant. □

Lemma 2. *If a tile with domain ω and there exists an action a such that $D_\omega(a) = \omega$, then the tile can perfectly encode optimal policy π^* on that domain.*

Proof. Every state in the tile is within the decision set of action a for that tile's domain, and so a tile can perfectly encode the value function necessary to generate policy π^*. □

Lemma 3. *If a tile with domain ω has decision sets $D_\omega(a) \cup D_\omega(b) = \omega$ with $D_\omega(a) \neq \emptyset$ and $D_\omega(b) \neq \emptyset$. If the agent enters a state within the domain of the tile, it has a probability β of landing in $D_\omega(a)$ and a probability $(1-\beta)$ of landing in $D_\omega(b)$. If we assume the decision boundaries are distributed such that all possible values of the probability β are equally likely, i.e. the decision sets are, on average, uniformly but not necessarily contiguously distributed within a tile, then $\mathbb{E}(\beta) = 0.5$.*

Proof. The expected probability of a uniform distribution is 0.5. We make no assumption here about the distribution or coverage of the sets $D_\omega(a)$ and $D_\omega(b)$. □

http://dx.doi.org/10.18489/sacj.v0i57.284

Theorem 1. *Suppose that a tile with domain ω has decision sets $D_\omega(a) \cup D_\omega(b) = \omega$ with $D_\omega(a) \neq \emptyset$ and $D_\omega(b) \neq \emptyset$, with probability β of landing in $D_\omega(a)$ and a probability $(1-\beta)$ of landing in $D_\omega(b)$. Suppose, without loss of generality, that $\beta \geq 0.5$. Then, the agent will select a suboptimal action with probability at least $(1-\beta)$.*

Proof. From Lemma 1, a tile may only encode a single action. Suppose that action a is the optimal action to encode with the tile. Then, regardless of whether state $s \in D_\omega(b)$ or $s \in D_\omega(a)$, a greedy agent will select action a. In particular, for $s \in D_\omega(b)$ occurring with probability $(1-\beta)$, the optimal action is b, yet a is selected.

Suppose b is the optimal action to encode with the tile (this can occur if the return associated with action b is greater than that for action a across ω). Then, the incorrect action according to optimal policy π^* is selected across $D_\omega(a)$ with probability $\beta \geq (1-\beta)$. Thus, a suboptimal action is selected with probability at least $(1-\beta)$. □

Corollary 1. *If a wrong decision is made with probability $\min(\beta, (1-\beta))$, and β is uniformly distributed, then $\mathbb{E}(\min(\beta, (1-\beta))) = 0.25$.*

Proof. Integrating $\min(\beta, (1-\beta))$ across all values of β gives the result above. □

Theorem 2. *If the agent has a choice of $m = |A|$ actions, with the decision sets and tile domains defined as before. From Lemma 1, only one action can be encoded per tile. Suppose that action a is the optimal action to encode for that domain of the m actions, with the agent landing in $D_\omega(a)$ with probability β. Then, in extension of Corollary 1, the agent will select a suboptimal action with expected probability $1 - \frac{H_m}{m}$ where $H_m = \sum_{i=1}^{m} \frac{1}{i}$ is the m-th harmonic number.*

Proof. As all points in the tile are equally likely, this is equivalent to finding one minus the expected maximum of an m dimensional symmetric Dirichlet distribution, i.e. considering x sampled from $\{X | X \in [0, 1]^m, ||X||_1 = 1\}$ and finding $1 - \mathbb{E}(||x||_\infty)$, which give the result above. □

Corollary 2. *If some proportion α of the n^d tiles in a tile coding for $m = |\mathscr{A}|$ actions are uniformly randomly divided by a decision boundary, and each tile covers an area of $\frac{1}{n^d}$ units, the resulting policy across the domain must select a nonoptimal action with probability $1 - \frac{H_m}{m}$ for αn^d of the tiles.*

Proof. Proof follows from Theorem 2. □

3.3 Discussion

As the number of tiles per dimension n grows, the tile coding will get more accurate, and fewer tiles will straddle decision boundaries. There is a relationship between α and n, which can be characterised if some assumptions are made about the nature of the decision boundaries. If we assume the decision boundaries are $d-1$ dimensional manifolds embedded in the d dimensional state space, then the number of tiles on this manifold will grow in $O(n^{d-1})$, whereas the total number of tiles will be n^d. The proportion of tiles on a decision boundary α is then in $O(\frac{1}{n})$. Under this assumption, an incorrect decision is made with probability $O(\frac{1-\frac{H_m}{m}}{n})$ assuming uniform distribution across the state space. As $n \to \infty$, this error decreases to zero.

http://dx.doi.org/10.18489/sacj.v0i57.284

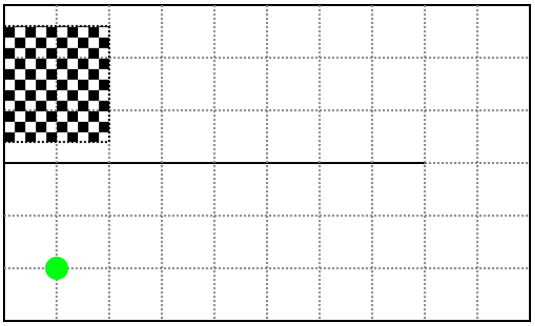

Figure 1: The Discontinuous Room domain, with start state shown in green

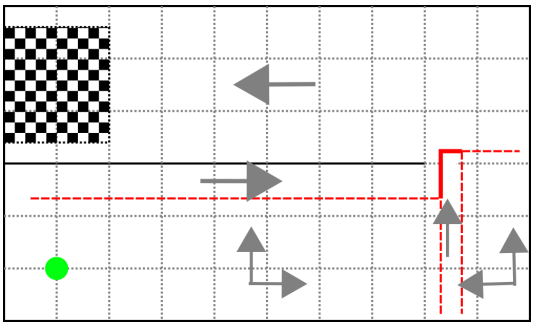

Figure 2: Optimal policy for Discontinuous Room, with decision boundaries in red

3.4 Case Study

Discontinuous room is a continuous state gridworld with a narrow gap to the goal room (Konidaris et al., 2011), as shown in Figure 1.

The room is 6 units high (we will call this y) and 10 units wide (x), with a wall from $[0, 3]$ to $[8, 3]$. The agent may move half a unit north, south, east or west. The start position is $(1, 1)$, and the goal position is $(1, 4.5)$, although the agent need only be within one unit of this goal. The agent is rewarded -1 for every time step taken, and 1000 upon reaching the goal state. The agent's position is represented using a two dimensional state, rescaled to be between zero and one.

We can thus determine the optimal policy for the agent. This policy is shown in Figure 2, with the decision boundaries shown in red. The dotted red lines indicate boundaries with overlapping optimal actions, as more than one action is optimal.

For this example however, we will assume the actions and decision boundaries shown in Figure 3, and compute the number of tiles along the decision boundaries, α and the expected number of incorrect decisions, for a number of tiling resolutions.

We see from Table 1 that the number of tiles on the boundary does appear to grow linearly with the number of tiles along each dimension. Furthermore, the upper bound on the probability of an incorrect decision decreases in inverse proportion to the number of tiles per dimension. When one uses at least five tiles per dimension, the expected error probability is approximately 10% across the whole state space. While the error decreasing with the number of tiles used may be obvious, we

http://dx.doi.org/10.18489/sacj.v0i57.284

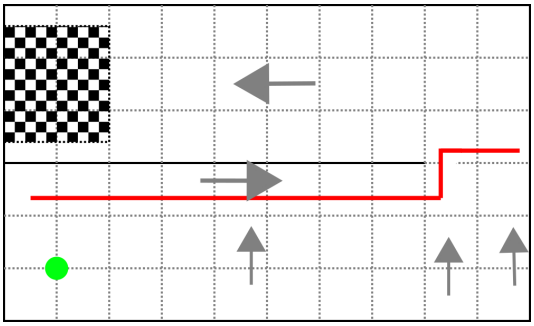

Figure 3: Simplified policy, with decision boundaries in red

Tiles per dim.	1	2	3	4	5
Total tiles	1	4	9	16	25
Tiles on boundary	1	3	3	5	5
α	1	0.75	0.33	0.31	0.2
$\frac{1-\frac{H_m}{m}}{n}$	0.48	0.24	0.16	0.12	0.1

Table 1: Fraction tiles encoding incorrect actions for Discontinuous Room

have previously been unable to quantify the amount of error decrease.

4 CONCLUSION

When discretisation or tile coding are used in reinforcement learning in continuous state space domains, there is an intrinsic loss of detail due to the representation of the policy or value function as being piecewise constant. If the optimal policy π^* changes over the domain of a tile, a suboptimal action will be selected for a portion of the tile. If we assume these decision boundaries are uniformly distributed within the affected tile, a suboptimal action is selected with probability $1-\frac{H_m}{m}$ within the tile, where $H_m = \sum_{i=1}^{m} \frac{1}{i}$ is the m-th harmonic number. If we further assume the decision boundaries are $d-1$ dimensional manifolds embedded in the d dimensional state space, and the proportion of affected tiles α is in $O(\frac{1}{n})$ where n is the number of tiles per dimension, value function or policy representation with tile coding will have an expected error probability of $O(\frac{1-\frac{H_m}{m}}{n})$, regardless of the learning technique used. This indicates that the performance of tile coding and discretisation schemes is bounded above.

ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation grant number TTK14052667930.

http://dx.doi.org/10.18489/sacj.v0i57.284

References

Geramifard, A., Doshi-Velez, F., Redding, J., Roy, N. & How, J. (2011, June). Online discovery of feature dependencies. In *Proceedings of the 28th International Conference on Machine Learning* (pp. 881–888). New York, NY, USA.

Johns, J., Painter-Wakefield, C. & Parr, R. (2010). Linear complementarity for regularized policy evaluation and improvement. In *Advances in neural information processing systems 23*.

Kolter, J. & Ng, A. (2009). Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th International Conference on Machine Learning* (pp. 521–528). http://dx.doi.org/10.1145/1553374.1553442

Konidaris, G., Osentoski, S. & Thomas, P. (2011). Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence* (pp. 380–385).

Lagoudakis, M. & Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, *4*, 1107–1149.

Lin, S. & Wright, R. (2010). Evolutionary tile coding: an automated state abstraction algorithm for reinforcement learning. In *Proceedings of the AAI Workshop on Abstraction, Reformulation, and Approximation*.

Moore, A. (1994). The parti-Game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In *Advances in neural information processing systems 6* (pp. 711–718).

Painter-Wakefield, C. & Parr, R. (2012). Greedy algorithms for sparse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*.

Sutton, R. & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Uther, W. T. & Veloso, M. M. (1998). Tree based discretization for continuous state space reinforcement learning. In *Proceedings of AAAI-98* (pp. 769–774).

Whiteson, S., Taylor, M. & Stone, P. (2007). *Adaptive tile coding for value function approximation*. Computer Science Department, University of Texas at Austin AI-TR-07-339.

Wu, C. & Meleis, W. (2009). Function approximation using tile and Kanerva coding for multi-agent systems. In *Proceedings of Adaptive Learning Agents Workshop (ALA) in AAMAS*.

http://dx.doi.org/10.18489/sacj.v0i57.284